November 2024

On the Role of Prediction in Streaming Hierarchical Learning

Ramy Mounir
University of South Florida

On the Role of Prediction in Streaming Hierarchical Learning

by

Ramy Mounir

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Date of Approval:
October 27, 2024

Keywords: Online Learning, Compositional Perceptual Processing, Predictive Learning

Acknowledgments

First and foremost, I would like to thank my advisor, Dr. Sudeep Sarkar, for his unwavering support and guidance. The countless hours of discussions we've had over the years challenged me to think critically and deeply, helping me to seek the bigger picture of my research. I am also incredibly fortunate to have had the mentorship of Dr. Anuj Srivastava, as well as Sathyanarayanan Aakur, Maurício Pamplona Segundo, and Fillipe DM de Souza throughout my PhD. Their expertise and advice helped steer me through many complexities of research and academia, and for that, I will always be thankful.

This dissertation would not have been possible without the generous financial support from the National Science Foundation (NSF) [1] and the USF Dissertation Completion Fellowship. Their funding made my research a reality, and for that, I am deeply grateful. A special thank you to Dr. Ruth Bahr for her incredible support. Her kindness and encouragement during the most stressful moments of my final semester, especially when the pressure of dissertation writing was at its peak, made all the difference.

To my collaborators and incredible members of the lab—Sujal Vijayaraghavan, Aditi Basu Bal, Cole Hill, Caio Da Silva, Ahmed Shahabaz, Gilbert Rotich, and Daniel Sawyer—I owe a great deal of gratitude. Our conversations, both scientific and personal, have enriched my PhD journey and made the challenges easier to bear. Specifically, I want to thank Sujal for always encouraging me and pushing me forward. Our collaborations has been remarkably fruitful, but more importantly, his friendship is something I will treasure for years to come.

[1]This research was supported by the US National Science Foundation grants CNS 1513126 and IIS 1956050. The bird video dataset used in Chapter 3 was made possible through funding from the Polish National Science Centre (grant NCN 2011/01/M/NZ8/03344 and 2018/29/B/NZ8/02312). Province Sud (New Caledonia) issued all permits required for data collection.

I am also grateful to Numenta for the wonderful internship experience. A special thank you to Niels Leadholm and Viviane Clay for their mentorship and supervision during my time there. Subutai Ahmad, thank you for your constant support and guidance! And to Jeff Hawkins, thank you for the inspiring discussions that profoundly shaped my thinking. Your work and vision continue to be a source of inspiration for me.

No words can adequately express my gratitude to my family. To my parents and my brother—thank you for believing in me and for being my anchor throughout this entire process. I owe an immense debt of gratitude to my father, whose encouragement to pursue a PhD set me on this path. I could not have reached this point without his wisdom and belief in my potential.

Lastly, to my wife, Margrate—thank you for standing by my side through every disappointment, every rejected paper, and every small victory. Your love, patience, and unwavering support have been my greatest source of strength. You have shared in every struggle and triumph, and for that, I am forever grateful. This dissertation is as much yours as it is mine.

Table of Contents

List of Tables

List of Figures

Abstract

In today's world, AI systems need to make sense of large amounts of data as it unfolds in real-time, whether it's a video from surveillance and monitoring cameras, streams of egocentric footage, or sequences in other domains such as text or audio. The ability to break these continuous data streams into meaningful events, discover nested structures, and predict what might happen next at different levels of abstraction is crucial for applications ranging from passive surveillance systems to sensory-motor autonomous learning. However, most existing models rely heavily on large, annotated datasets with fixed data distributions and offline epoch-based training, which makes them impractical for handling the unpredictability and scale of dynamic real-world environments. This dissertation tackles these challenges by introducing a set of predictive models designed to process streaming data efficiently, segment events, and build sequential memory models without supervision or data storage.

First, we present a single-layer predictive model that segments long, unstructured video streams by detecting temporal events and spatially localizing objects in each frame. The model is applied to wildlife monitoring footage, where it processes continuous, high-frame-rate video and successfully detects and tracks events without supervision. It operates in an online streaming manner to perform simultaneous training and inference without storing or revisiting the processed data. This approach alleviates the need for manual labeling, making it ideal for handling long-duration, real-world video footage. Building on this, we introduce STREAMER, a multi-layered architecture that extends the single-layer model into a hierarchical predictive framework. STREAMER segments events at different levels of abstraction, capturing the compositional structure of activities in egocentric videos. By dynamically adapting to various timescales, it creates a hierarchy of nested events and forms more complex and abstract representations of the input data. Finally, we propose the Predictive Attractor Model (PAM), which builds biologically plausible memory models of

sequential data. Inspired by neuroscience, PAM uses sparse distributed representations and local learning rules to avoid catastrophic forgetting, allowing it to continually learn and make predictions without overwriting previous knowledge. Unlike many traditional models, PAM can generate multiple potential future outcomes conditioned on the same context, which allows for handling uncertainty in generative tasks.

Together, these models form a unified framework of predictive learning that addresses multiple challenges in event understanding and temporal data analyses. By using *prediction* as the core mechanism, they segment continuous data streams into events, discover hierarchical structures across multiple levels of abstraction, learn semantic event representations, and model sequences without catastrophic forgetting.

Chapter 1: Introduction

"I want AI to do my laundry and dishes so that I can do art and writing, not for AI to do my art and writing so that I can do my laundry and dishes."

— Joanna Maciejewska

The human brain is continuously bombarded with an immense amount of sensory information from the world around us. Every second, we receive multimodal inputs from our environment through our eyes, ears, skin, and other sensory organs, each transmitting a myriad of signals that need to be processed and interpreted in real-time. Additionally, these perceptual inputs are usually redundant, incomplete, and ambiguous; the brain continuously filters out redundant information, detects and summarizes events, as well as *fills-in* [132, 177] information based on our current beliefs about the world (e.g., physics laws, expected behaviors). This raises fundamental questions about cognition: *How does the brain manage to process such overwhelming sensory input efficiently?* And, more intriguingly, *how do we perceive the world with all its richness and complexity, turning this chaotic stream of data into coherent and meaningful experiences?*

Consider, for example, the simple act of walking through a bustling city street. As we navigate the crowded sidewalks, our eyes and ears receive an enormous amount of perceptual input - people moving in different directions, flashing signs, the sound of many distant conversations, and honking cars rushing by. Despite this overwhelming sensory input, we effortlessly and efficiently navigate to our destinations while avoiding obstacles and recognizing familiar faces and landmarks on the way. While the brain excels at real-time processing of the sensory stream, it is challenging to reconstruct specific details about what has been perceived. For example, we may remember the scene of a busy street with many cars but not the models or colors of each one of these cars. This suggests that the brain operates at multiple levels of abstraction to efficiently deal with the overwhelming complexity of the world. Higher levels of the hierarchy represent more stable and abstracted events (e.g., a scene of a busy street), whereas lower levels process more detailed aspects of a scene (e.g., individual cars). On the other hand, we can effortlessly retrieve the memory of a vintage car and recall the unique sound of its bulb horn. *What determines which events are more important (i.e., vintage car) and which ones can be summarized (i.e., busy street)?*

A growing body of research, from cognitive neuroscience [34, 62, 193, 194] and cognitive psychology [128, 278], suggests that the key lies in the brain's ability to continuously predict incoming sensory stimuli. Instead of passively receiving and processing sensory input, the brain actively builds models of the world that can anticipate the incoming perceptual inputs. Prediction errors result in a learning signal that refines the world models for more accurate predictions. In addition to building world models and representations, the transient increase in prediction errors marks segmentation boundaries [279], separating events (i.e., reaching and grasping) from each other. These segmentation boundaries form a spatio-temporal compositional structure of events at multiple levels of abstraction. *The notion of the brain as a dynamic predictive system, where continuous predictions about sensory input and environmental interactions form the basis for perceptual learning and inference, is fundamental to this dissertation and forms its core thesis.*

This thesis proposes novel solutions to long-standing challenges in artificial intelligence, particularly in the field of computer vision, by integrating insights from other cognitive disciplines, such as neuroscience and psychology. We leverage the notion of perceptual prediction to develop learning rules for (1) discovering compositional (i.e., part-whole) structures of events and enhancing their representation, (2) supporting streaming online learning and inference, (3) mitigating catastrophic forgetting, and (4) generating multiple possibilities in a non-deterministic sequence modeling framework. The following sections begin with a brief history of AI and an overview of the fundamentals of deep learning, as well as their current limitations, before delving into our proposed solutions, which are detailed in the subsequent chapters.

1.1 Brief History of Artificial Intelligence

Artificial intelligence (AI) has significantly evolved since its inception; the groundwork for AI was laid in the 1940s and 1950s. In 1943, the seminal work of Warren McCulloch and Walter Pitts was published [159]; they showed that neurons can be used to perform binary logic. After which, Alan Turing, in his 1950 paper “Computing Machinery and Intelligence” [250], asked whether machines could think, introducing what we now call the Turing Test — a way to determine if a machine’s behavior is indistinguishable from that of a human. During the same period, John von Neumann’s work on digital computer architecture provided the necessary foundation for programmable machines. His design allowed computers to store both data and instructions in memory, which was a critical step toward creating machines capable of performing complex tasks.

The term “Artificial Intelligence” was coined at the 1956 Dartmouth conference organized by John McCarthy, Marvin Minsky, Nathaniel Rochester, and Claude Shannon. The conference brought together researchers who were optimistic about the possibilities of creating machines that could mimic human intelligence. Two years later, in 1958, Frank Rosenblatt developed “the Perceptron” [205] as one of the earliest models for neural networks. The Perceptron demonstrated how a neural network could learn to classify by adapting its weights. While the McCulloch and Pitts Neuron only showed how neurons can collectively represent symbolic logic gates without any mention of how the weights could be learned, the Perceptron, on the other hand, showed how neurons can *learn* to classify from data.

The 1980s saw significant developments in neural networks, including the introduction of the Hopfield Network by John Hopfield in 1982 [103]. The Hopfield Network is a type of recurrent neural network that functions as an associative memory system. It can store and retrieve patterns, even when presented with noisy or incomplete inputs. This capability was a key advancement in understanding how memory could be modeled in machines and provided insights into how networks of neurons might function in the brain. In 1980, Kunihiko Fukushima developed the Neocognitron [66], a neural network model inspired by the visual cortex and the findings of Hubel and Wiesel [106]. The Neocognitron introduced a hierarchical, layered structure that allowed it to recognize patterns in images, such as characters, regardless of their position or scale. This hierarchy represented different levels of visual stimuli and features such as edges or movement. The Neocognitron was a precursor to the convolutional neural networks (CNNs [133]) that would use backpropagation [206] to later become central to computer vision tasks in AI.

AI research saw a resurgence in the 1990s, largely driven by the advent of machine learning. Unlike earlier rule-based approaches (e.g., expert systems), machine learning algorithms were designed to learn from data and improve over time. This period saw the development of key learning algorithms such as Support Vector Machines [37], decision trees [99], and a renewed interest in neural networks. A pivotal moment in the development of AI, particularly in computer vision, came with the creation of the ImageNet dataset and the corresponding ImageNet Large Scale Visual Recognition Challenge (ILSVRC). In 2009, Fei-Fei Li and her team at Stanford University introduced ImageNet [43], a large dataset of millions of labeled images organized into thousands of categories. The dataset not only provided an unprecedented resource for training models but also set the stage for deep learning breakthroughs in visual recognition.

One such breakthrough came in 2012 with the success of AlexNet [125], a deep convolutional neural network (CNN) developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. AlexNet significantly outperformed previous approaches in the ILSVRC competition by stacking multiple layers of convolutional filters to form a deeper architecture. The success of AlexNet demonstrated that stacking more CNN layers and using powerful hardware (GPUs) could lead to significant improvements in performance, marking a turning point for computer vision and fueling the interest in AI. The success of ImageNet and AlexNet spurred interest in CNNs, leading to the development of even deeper architectures, such as VGGNet [228] and ResNet [89], which continued to push the boundaries of computer vision performance. The most recent breakthrough in AI has been the development of transformer architecture [254], which has revolutionized natural language processing [188], with advancements extending to computer vision [49].

1.2 Deep Learning Fundamentals

Artificial Neural Networks (ANNs), in the context of deep learning, refers to a model's ability to approximate a function $\mathbf{x}$ $\mathbf{y}$ given a dataset $\mathcal{D}$. These data points can be anything. For example, can be a sentence in English, and is its translation in French, or can be an image while is a caption describing the image. This approximation is achieved by applying a series of linear projections, separated by non-linear activation functions, on the input. The last projection outputs a prediction of the label (i.e.,). The error between the predicted label and the true label constructs a learning signal to adjust the weights of each projection layer. Learning this function approximator enables the model to generalize to novel data points (i.e., not seen during training) from the *same* data distribution.

1.2.1 Single and Multi-Layer Perceptron

Deep learning can be traced back to Rosenblatt's Perceptron [205], where a single-layer neural network was used to classify inputs into one of two categories. A single perceptron computes the weighted average of its inputs and applies a step function as an activation. These weights are tuned to minimize the output error. Rosenblatt's single-layer architecture could only approximate linear functions; however, it laid the foundation for more complex neural network architectures with multiple layers of projections and non-linear activations. The Multi-Layer Perceptron (MLP) stacks multiple layers of perceptrons and backpropatates the gradient of the error with respect to every weight in every projection layer using the chain rule [206]. The training can be stabilized by averaging the gradient over multiple data points (i.e., a batch) in an optimization procedure called Stochastic Gradient Descent (SGD) [200].

1.2.2 Types of Objective Functions

The objective function is crucial in guiding the training process by quantifying the error between the model's prediction and the target values. The choice of objective function depends on the nature of the task, with *regression* and *classification* being two of the most common types. In regression-based prediction, the model outputs a continuous value and often uses *mean squared error (or L1)* to minimize the error in prediction. For example, a model that is trained to reconstruct images can apply a pixel-wise regression loss to minimize the difference in raw pixel values between the prediction and the actual image. A classification-based prediction assumes a finite set of categories and learns to minimize the error in predicting the correct class. By leveraging contrastive *cross-entropy loss*, the model is not only trained to predict the correct class but also to differentiate between multiple classes, improving its overall discriminative power. The usefulness of the model depends on the quality and granularity of the categories. For example, an object detection model tasked to detect the existence of the class "animal" in an image will learn to inhibit discriminative features between different subclasses of animals (e.g., dog, cat).

1.2.3 Architectures and Inductive Biases

The simple idea of a perceptron has inspired the creation of various neural network architectures with specific designs that are more tailored to the type of data being processed. While the perceptron laid the groundwork by introducing the concept of learnable weights and activation functions, modern architectures have built upon these ideas by embedding more complex assumptions about data structure and relationships. An *Inductive Bias* refers to the assumptions a model makes about the data it processes; these assumptions can be incorporated into the architecture design to improve its learning, sampling efficiency, and performance. We will only discuss some inductive biases that exist in the design of Convolutional Neural Networks (CNNs [133]) and Recurrent Neural Networks (RNNs [100]); however, many other architectures (e.g., Graph Neural Networks [118]) have incorporated more tailored biases for their own targeted applications.

1.2.3.1 Convolutional Neural Networks

CNNs are designed with an inductive bias towards spatial hierarchies of features, local receptive fields, and spatial weight sharing.

- The *hierarchical feature learning* refers to the progressive growth of the receptive field with each stacked layer. Deeper layers have higher receptive fields allowing for the processing of higher-level features (e.g., shapes and objects), whereas earlier layers can only process simple low-level features (e.g., edges).
- *Local receptive fields* is a design choice embedded in the shape of the CNN filter kernel. CNNs assume that nearby pixels in an image are more closely related than distant pixels. Therefore, a grid-shaped square kernel is applied to images to detect local features.
- *Weight sharing* is an inductive bias in that it assumes translation invariance. Applying the convolution operation of a CNN filter on an image forces each filter to become a detector of features that can exist anywhere in the image (i.e., translation invariance).

1.2.3.2 Recurrent Neural Networks

RNNs introduce an inductive bias towards sequential data, making them well-suited for order-specific tasks such as Natural Language Processing (NLP) and processing frames in videos. RNNs assume that the current input is dependent on previous inputs (i.e., *Temporal Dependency*); therefore, the inductive bias is implemented through recurrent connections, where the hidden state from the previous time step is fed back into the network along with the current input (maintaining memory and context). Unlike the spatial weight sharing used in CNNs, the learnable weights in RNNs are shared temporally. This allows an RNN block to learn general feature extractors from different inputs and integrate the information over time.

1.3 Biological Inspiration and Lack Thereof

Neural networks, both biological and artificial, are designed to process information in a manner that resembles the functioning of the human brain. However, despite their similar goals, biological neural networks (BNNs) and artificial neural networks (ANNs) differ significantly in their structure, connectivity, and learning mechanisms.

1.3.1 The Neuron Model

Although the perceptron model [205] of Rosenblatt was originally inspired by the biological neuron model [159] and was later extended to build deep neural networks architectures [125], it did not model the full complexity of a biological neuron; the perceptron is a highly simplified version of the neuron. For example, an artificial neuron implements a global summation function over all its synaptic inputs, whereas biological neurons are characterized by dendritic branches exhibiting more complex local computations [148]. The spatial proximity of synaptic inputs on the dendrites controls the summation properties and, therefore, spiking. For example, nearby inputs on the same branch are summed sigmoidally, as opposed to the linear summation of separated inputs [184]. The spatial and temporal proximity of synaptic input are often required to generate a dendritic spike. On the other hand, a perceptron performs a simplified global summation over all presynaptic neurons, eliminating the compartmentalization effect of dendritic branches and spatial proximity. Active dendrites [107] not only receive and integrate synaptic input but also actively modify and amplify them, which adds to the complexity of neural processing. This complexity allows for non-linear dendritic computations [153], which cannot be easily predicted, even with multiple layers of perceptrons [21, 183].

1.3.2 Activation and Weight Sparsity

In biological neural networks, only a small subset of neurons are engaged in processing a given stimulus at any one time while most neurons remain inactive [175]. This property of activation sparsity also extends to weight sparsity, where only a fraction of possible synapses are formed between populations of active neurons [59]. This sparsity property can have various advantages in terms of memory capacity [83, 247] and speed of learning [221]. On the other hand, ANNs are formed of densely-activated perceptrons with fully-connected weights between the layers.

1.3.3 The Learning and Plasticity

ANNs adjust the synaptic weights by calculating and backpropagating the gradient of the error with respect to every learnable parameter [206]. This weight update process requires information that is not locally available, whereas learning in the brain utilizes only locally available information [233]. While some research has shown that backpropagation can be approximated using more local methods [143, 265, 266], some aspects of backpropagation are still regarded as biologically implausible [39]. For example, in ANNs, there is an assumption of symmetry of forward and backward weights; the same weights are used to back-propagate errors and forward-propagate information during prediction [143, 212]. Other theories of local plasticity in the brain are based on Hebbian learning [90], where correlation in the firing of neurons strengthens the synapse between them. More recent theories have also factored in the timing of the neuron spikes; spike timing-dependent plasticity (STDP [232]) adjusts the synaptic strength based on the relative timing of pre and postsynaptic action potential firing.

1.4 The Unresolved Challenges of ANNs

Despite recent significant advancements [26, 109, 226] in artificial intelligence research, a paradoxical gap remains between the capabilities of artificial and biological intelligence. As Moravec noted, “It is comparatively easy to make computers exhibit adult-level performance on intelligence tests or playing checkers, and difficult or impossible to give them the skills of a one-year-old when it comes to perceiving and manipulating objects.” This disparity, known as Moravec’s paradox [164], highlights the surprising difficulty of replicating human-like intelligence in machines despite their ability to excel in complex tasks such as playing chess or recognizing faces. In this section, we highlight some of the challenges faced by deep learning when attempting to replicate human-like intelligence.

1.4.1 Continual and Stream Learning

Continual and Stream learning address the need for models to adapt to non-static, evolving data streams, moving beyond the limitations of traditional, epoch-based training setups. In continual learning, models are trained on a sequence of tasks and must learn the tasks incrementally while retaining knowledge from previous tasks. Stream learning introduces a more complex training setup where models must learn continuously from a non-stop flow of data. In this section, we explore the key challenges of continual learning, present current approaches to overcoming issues like concept drift and catastrophic forgetting, and introduce stream learning as a more challenging framework where models must adapt to continuously evolving data streams.

1.4.1.1 Incremental Continual Learning

1.4.1.1.1 The I.I.D Assumption and Concept Drift

Traditional deep learning approaches rely heavily on the i.i.d (independent and identically distributed) assumption, which states that the training data is statistically independent and is randomly sampled from a fixed distribution. This allows the models to be trained in batches of data points, where every batch becomes a representative sample of the fixed data distribution, and the stochastic gradient becomes an unbiased estimate of the full gradient [25, 68, 191]. A larger batch size, combined with a small optimization step size (i.e., learning rate) and many training iterations, slowly adjusts the model's parameters such that it aligns with the fixed data distribution. However, this i.i.d assumption is often violated in real-world applications, where data is collected as a stream of sequential data points, leading to correlations and temporal dependencies between the distribution samples. In real-world applications, the training distribution is continuously changing to introduce new concepts and model the everchanging true distribution of data in a dynamic environment. This is referred to as concept drift [67, 260].

1.4.1.1.2 Catastrophic Forgetting

Consider a dataset of handwritten digits (e.g., MNIST [44]); the sampling of data points for training must follow the i.i.d assumption such that a training batch contains data points that are uniformly and randomly sampled from the full distribution of data (i.e., digits classes). However, in an incremental learning setting [198, 255], the model is trained on a single class at a time (e.g., digit "0", then digit "1"). In this case, a deep learning model will overwrite the previously-learned digit with each new digit [48, 244]. This inability to process incrementally available information from non-stationary data distributions generally leads to the *catastrophic forgetting* [60, 158, 178] of older knowledge. Finding an effective solution for catastrophic forgetting remains a long-standing challenge in deep learning [137, 196], as well as some biologically-inspired models, such as Hopfield Networks [27, 172].

1.4.1.2 Current Solutions

Current solutions to continual learning predominantly focus on task-segmented continual learning [198], where the boundaries between tasks are explicitly defined. However, a few works [12, 81] explore the use of change point detectors to automatically identify task transitions. The ability to identify task boundaries and distinguish between current and new tasks enables targeted strategies to preserve knowledge and adapt to changing tasks.

1.4.1.2.1 Regularization-Based Approaches

Regularization-based methods impose constraints on the model's parameters. These constraints typically prevent parameter changes that would result in overwriting previous knowledge. Elastic Weight Consolidation (EWC [119]) penalizes changes to important weights, effectively consolidating knowledge from previous tasks while adapting to new ones. Another approach, Learning without Forgetting (LwF [142]), uses a distillation loss [95] to retain knowledge from earlier tasks, forcing the model to produce similar outputs for previous tasks while learning new ones.

1.4.1.2.2 Dynamic Architectures

Dynamic architectures accommodate novel neural resources to account for the learning of new tasks. These approaches typically expand the model size by adding new parameters to train on subsequent tasks. For instance, Progressive Network [208] trains a separate neural network for each incremental task while learning lateral connections with the existing tasks. This approach continually learns with minimal forgetting at the expense of linear growth of the model size with the number of tasks. In contrast, Dynamically Expandable Networks (DEN [276]) strike a balance by selectively retraining the old parameters while still expanding the model parameters to accommodate new knowledge from subsequent tasks. This adaptive approach reduces the number of added parameters, enabling the model to dynamically determine its optimal capacity as new tasks are introduced.

1.4.1.2.3 Replay-Based Approaches

Replay-based methods use stored or generated data from previously learned tasks to interleave old and new tasks data, thereby maintaining performance on both and mitigating catastrophic forgetting. Gradient Episodic Memory (GEM [149]) stores a subset of examples from previously learned tasks and ensures that the loss on these examples does not increase during the training of new tasks. This requires the model to store data, which increases the memory requirements. In contrast, generative replay approaches [224] recreate pseudo-data from previous tasks using a generative architecture (e.g., GAN [71]). The generated data is then mixed with the data from the current task. These generative approaches store the generative model weights for each task and require additional computations to generate the pseudo-data during training.

1.4.1.3 A More Challenging Streaming Setup

In addition to continual learning, it is important to distinguish a more challenging type of learning that separates artificial from biological intelligence. Deep learning approaches mark a clear distinction between training and testing phases [48, 101]. In the deep learning setup, the model has access to the full training data and is allowed to iterate over the data until learning stabilizes and the loss converges to a minimum value then the model is evaluated on the test set in an offline mode where the weights are not allowed to change. This setup does not support real-world applications (e.g., surveillance and monitoring) where the information is continuously being streamed and processed and may not be stored [20]. In such applications, the model should only see each data point a single time while performing training and inference simultaneously. *Stream learning* presents a more realistic and challenging problem compared to continual learning, as it encompasses all the difficulties associated with continual learning while introducing additional constraints. Specifically, stream learning inherits the challenges of catastrophic forgetting, concept drift, and distribution shift from continual learning and imposes further restrictions of simultaneous training and inference from a continuous, non-stationary data stream, without data storage. A comparison between traditional,

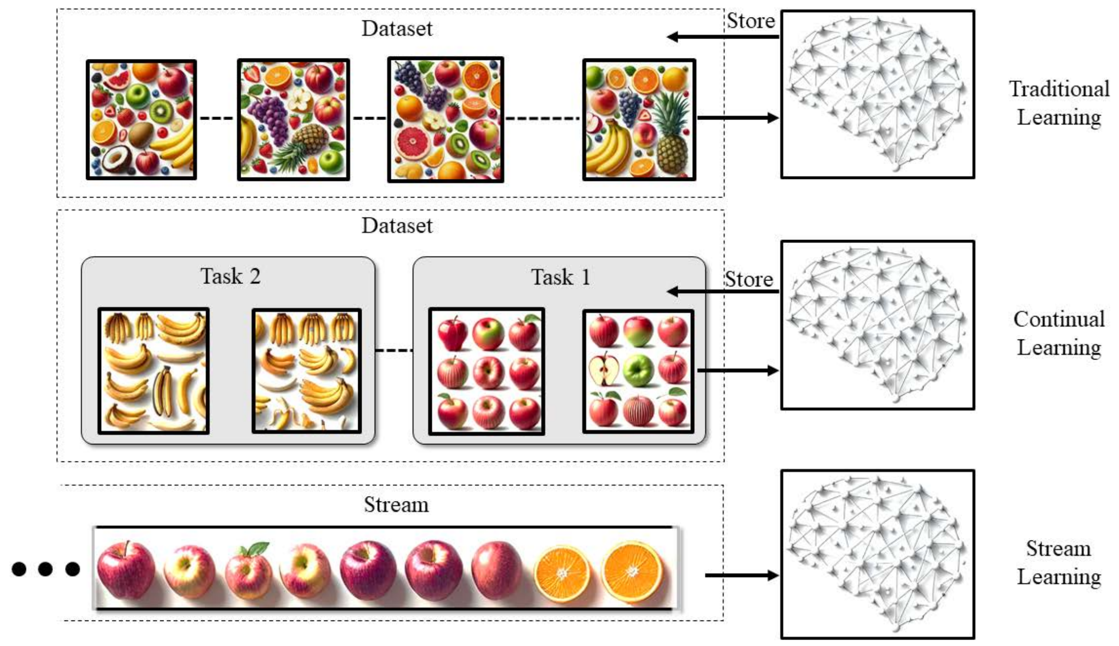


Figure 1.1: A comparison between traditional, continual and stream learning. The figure shows a simple fruit classification example. In stream learning, the data is not stored in a dataset and used for epoch-based training, instead each sample is processed a single time as part of a stream. Stream learning performs training and inference simultaneously.

continual and stream learning is shown in Figure 1.1. *This dissertation targets the stream learning setup.*

1.4.2 Compositional Structures and Interpretability

Humans perceive a structural representation of the world. A long sequential data stream is effortlessly parsed into chunks of coherent units, forming a part-whole hierarchical structure. This nested compositional structure can be seen in visual scenes [96, 97], temporal events [268, 281], or even natural language [157, 181]. This structured representation extends to hierarchical action planning [8, 204]; tasks are usually composed of subtasks in a nested hierarchy of action goals at different levels of complexity and detail [193]. Consider the seemingly simple event of making a sandwich as shown in Figure 1.2, the higher-level event “Make Sandwich” can be composed of multiple lower-level events, i.e., Cut Bun, Smear Butter, and Put Topping. The multiple levels perceive the world at different levels of abstraction, trading off detail for receptive field.

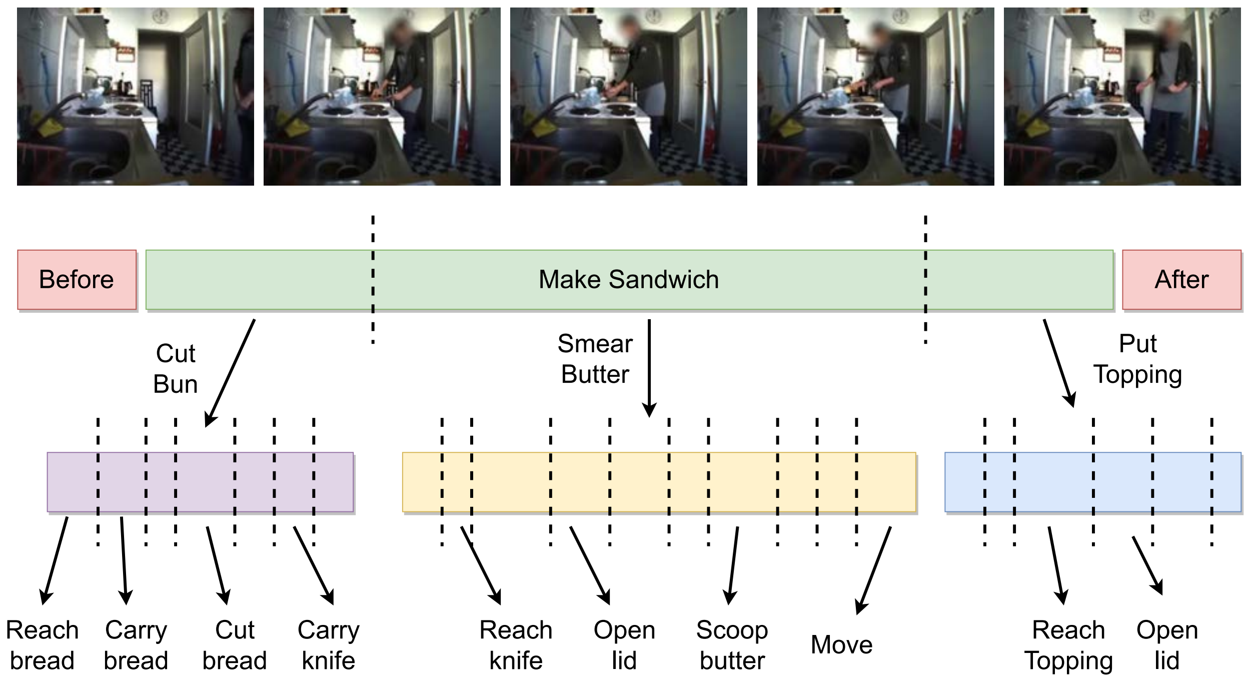


Figure 1.2: An example of nested events structure. The events hierarchy consists of multiple levels of segmentation. The higher-level event "Make Sandwich" can be segmented into lower-level events "Cut Bun", "Smear Butter", and "Put Topping" Each lower-level event can be further segmented into its constituent events at a lower level. The event "Make Sandwich" can be part of an even higher level event, "Have Breakfast". This compositional relationship between events forms "the events hierarchy". Images are from the Breakfast Actions Dataset [127]. Figure reused with permission in Appendix D.

While ANNs are often perceived as hierarchical due to the layered arrangement of convolutional filters in CNNs or the stacking of perceptron layers in MLPs, this hierarchy is limited to learning a progression of feature complexity rather than discovering the inherent part-whole compositional structure in the data [96]. The inability to represent the part-whole hierarchy motivated a series of works on "capsules" [94, 209], as well as the hierarchical JEPA architecture [132], discussed later in Chapter 2. Perceiving a structural representation of the world can significantly enhance the explainability and interpretability of deep learning models. By explicitly representing the relationships between parts and wholes, these models provide a clear understanding of how they arrive at their decisions. Moreover, learning such compositional representations can significantly facilitate communication between intelligent agents, enabling them to share knowledge and coordinate actions more effectively and efficiently [220]. In Chapter 4, we propose a solution to discovering the nested structure of events using a simple predictive objective.

1.4.3 Multiple Future Possibilities

ANNs are inherently deterministic; given a specific input and a set of weights, they always produce the same output. This limits the ability of the model to represent multiple equally probable possibilities in regression-based prediction tasks [42, 132]. Large Language Models (LLMs) have mitigated this issue by transforming the regression task into a classification task, predicting a probability distribution over all the possible tokens, and sampling the next token id from this distribution [45, 188]. However, this approach relies on a priori knowledge of all the possible tokens, which may not be feasible for other tasks and modalities, e.g., video frame prediction [259]. Inspired by the success of LLMs, some approaches [53, 252] have implemented a two-step method to learn and quantize a bottleneck representation into a codebook (i.e., dictionary of embeddings), then perform classification in an autoregressive manner on the codebook entries. While this approach has shown promise, it is not without limitations. The use of fixed embeddings as tokens would not adapt to concept drift or data distribution shift. Additionally, the generation performance depends on the quality of the quantization and the number of entries in the codebook.

To model the conditional probability of future inputs given the observed context, generative models offer a promising solution. Theoretically, these models can capture the complex conditional distributions required for future predictions. However, existing generative models, such as Variational Autoencoders (VAEs [117]), Generative Adversarial Networks (GANs [71]), and diffusion models [98], exhibit limitations that restrict their ability to predict longer sequences of future predictions. Diffusion models, for instance, rely on a slow iterative denoising process, resulting in limited generation durations. GANs, while capable of single-pass generation, employ an adversarial objective that can be challenging to optimize, resulting in unstable training. VAEs, on the other hand, often suffer from mode collapse, failing to capture the full data distribution, and typically yield lower-quality generations compared to diffusion and GAN models. In Chapter 5, we introduce a novel predictive framework that learns through local plasticity rules, specifically Hebbian-based rules, in conjunction with competitive learning and fixed-point attractors. This approach enables stochastic sampling and prediction of multiple future possibilities while learning sequences in a streaming manner and avoiding catastrophic forgetting.

1.4.4 Noise Robustness and Pattern Completion

Despite the significant advancements in deep learning, current models exhibit a critical weakness in their robustness to noisy and incomplete inputs. Even small perturbations to the input data can cause these models to produce drastically incorrect outputs with high confidence [70, 230, 239]. Another key shortcoming of deep learning models is their inability to apply common sense or higher-order reasoning to fill in missing information. These models lack the capacity to perform pattern completion, which involves leveraging prior knowledge and beliefs about the world (i.e., world model) to reconstruct incomplete or corrupted inputs.

We argue that the issues of noise robustness and pattern completion are not separate challenges but are intrinsically linked. For a model to be robust to noise, it must be capable of using high-level predictive priors to infer and complete missing or ambiguous information from sensory data. This form of top-down inference is a key feature of human cognition [177] but is notably absent from current artificial neural network architectures [38, 42, 132]. In Chapter 5, we introduce a predictive attractor model that uses higher-level predictive priors to build fixed-point attractors of observations. We show that our proposed approach is robust to a large amount of visual perturbations.

1.5 Layout and Contributions

In this thesis, we propose novel solutions to several key challenges in deep learning, specifically focusing on self-supervised streaming learning and hierarchical event segmentation. These contributions aim to address the limitations of existing models (Section 1.4), particularly in processing streaming data, handling dynamic data distributions, and avoiding catastrophic forgetting. We lay out the contributions in three main chapters, each building on the core idea of how predictive learning allows for building event models to solve a diverse set of tasks. In Chapter 2, we start by providing a high-level overview of prior works and theories of cognition that rely on prediction as a core learning signal. In subsequent chapters, we present our predictive framework.

- We propose a *single layer predictive model* [168, 169] (Chapter 3) for continual processing of streaming video data. This method is used to efficiently process streaming wildlife monitoring video (over ten days of footage) and successfully detect temporal events. An attention mechanism is used to spatially localize the event without using any labels to guide the learning process.
- We extend the single-layer model to a *multi-layered predictive architecture* [167] (Chapter 4) for hierarchical processing of egocentric streaming video. This approach uses a stack of predictive layers operating at different dynamic timescales to discover the compositional structure of events in egocentric videos without any supervision. The learned event representations are used for event retrieval across videos.
- We propose a *Predictive Attractor Model* [166] (Chapter 5) which operates within the constraints of a biological system to model long sequences of data (e.g., video frames, protein sequences, text). PAM uses sparse representations (i.e., SDR) and biologically plausible learning rules (i.e., Hebbian plasticity) to continually learn sequences without catastrophic forgetting. Additionally, PAM can regress multiple future predictions by learning fixed-point attractors in a contrastive online manner.

Chapter 2: Prediction in Prior Works

"If I have seen further than others, it is by standing upon the shoulders of giants."

— Isaac Newton

Our brains are constantly making predictions of what features our senses expect to see next. When we observe someone cooking, we continuously make predictions. These predictions are made at different temporal granularities. We predict the trajectory of motion at fine temporal scales to anticipate hand location in the next frame. And, at coarse scales, anticipate what utensils could be used next. We make predictions at multiple partonomic hierarchy levels about the event being observed.

Indeed, there is a definition of artificial intelligence that puts the ability to make predictions as the central feature [85, 91, 132, 194]. The extent to which an agent can be considered intelligent is determined by the temporal window and spatial extent to which it can predict with precision. Thus, a small insect can make predictions of its immediate surroundings and its near future. Humans, armed with high-level scientific reasoning and logic, can predict events over much larger space and longer temporal windows. The content of the prediction depends on the task at hand. For example, an agent can predict the existence of an occluded logo on a coffee mug [84], the next note in a familiar melody [83], or even learn to make predictions in abstract spaces such as traversing a family tree [267].

The ability of an agent to make accurate predictions is indicative of the quality of learned representations. Many cognitive theories of prediction converge on the idea that in order to make precise long-term predictions, the agent must build a world model constructed from the data it receives. A world model summarizes a single event at a specific level of abstraction and, therefore, can only make precise predictions of future observations *within its scope*. Furthermore, there is a general consensus among these theories that these world models are organized compositionally in a nested hierarchical structure, such that lower-level models serve as constituent components of higher-level world models [268]. Consequently, the relationships between the models can be described in terms of transformations between coordinate reference frames [96, 97, 101]. In this chapter, we cover a few influential cognitive theories that place prediction at the core of learning structured representations of the world.

2.1 Zacks's Theory of Event Segmentation

Event Segmentation Theory (EST), introduced by Zacks *et al.* [278, 279], explains how the human brain processes continuous streams of sensory information by breaking them down into discrete, manageable events. According to EST, the brain organizes perception by maintaining a model of the current event, referred to as the *working event model*, which acts as a prior during predicting future sensory inputs. When these predictions are incorrect, the brain updates the model, effectively marking the transition from one event to another. This predictive process of segmenting events provides a valid theory for explaining how humans structure their experience of the world, allowing for efficient processing and comprehension of complex sequences of actions, interactions, and occurrences.

At its core, EST suggests that perception is not a passive process but an active one, where the brain constantly anticipates what will happen next. The theory applies across various domains of cognitive function, from understanding visual sequences like someone walking down the street to more abstract mental processes like solving a complex mathematical problem. The segmentation of continuous sensory input into events allows for more efficient processing, memory consolidation and retrieval, and comprehension of the broader context in which these events occur.

2.1.1 The Working Event Model

The working event model is the central concept within EST. It represents the brain's internal model (i.e., world model) of the current event and is constantly updated to reflect changes in the environment. The working event model acts as a dynamic representation, guiding the brain's interpretation of incoming sensory data by anticipating future states based on what is currently happening.

This model allows the brain to process perceptual input by focusing on relevant information and filtering out noise or minor variations that are unlikely to indicate a significant change. For instance, small deviations in someone's walking speed or facial expressions are accounted for by the working event model, ensuring that perception remains stable and coherent over time. Therefore, the event model is a summary of the observations in an event and can explain all possible observations within the scope of such event.

The brain uses this model to predict what sensory features will be encountered next. When sensory input aligns with these predictions, the event model is reinforced, allowing for smooth and uninterrupted processing of information. However, the working event model is not static; it is continuously updated as new information comes in, ensuring that the brain's predictions remain accurate. The more accurate the model, the better the brain is at maintaining a stable representation of the current event.

2.1.2 Prediction Error and Event Boundaries

Figure 2.1 shows an overview of EST. The perceptual processing unit receives, filters, and encodes sensory features into higher-level representations conditioned on the working event model. This conditioning ensures that feature representations are robust to small variations from one instant to another. The perceptual processing unit sends the extracted features to a prediction unit, which anticipates future perceptual features based on the working event model.

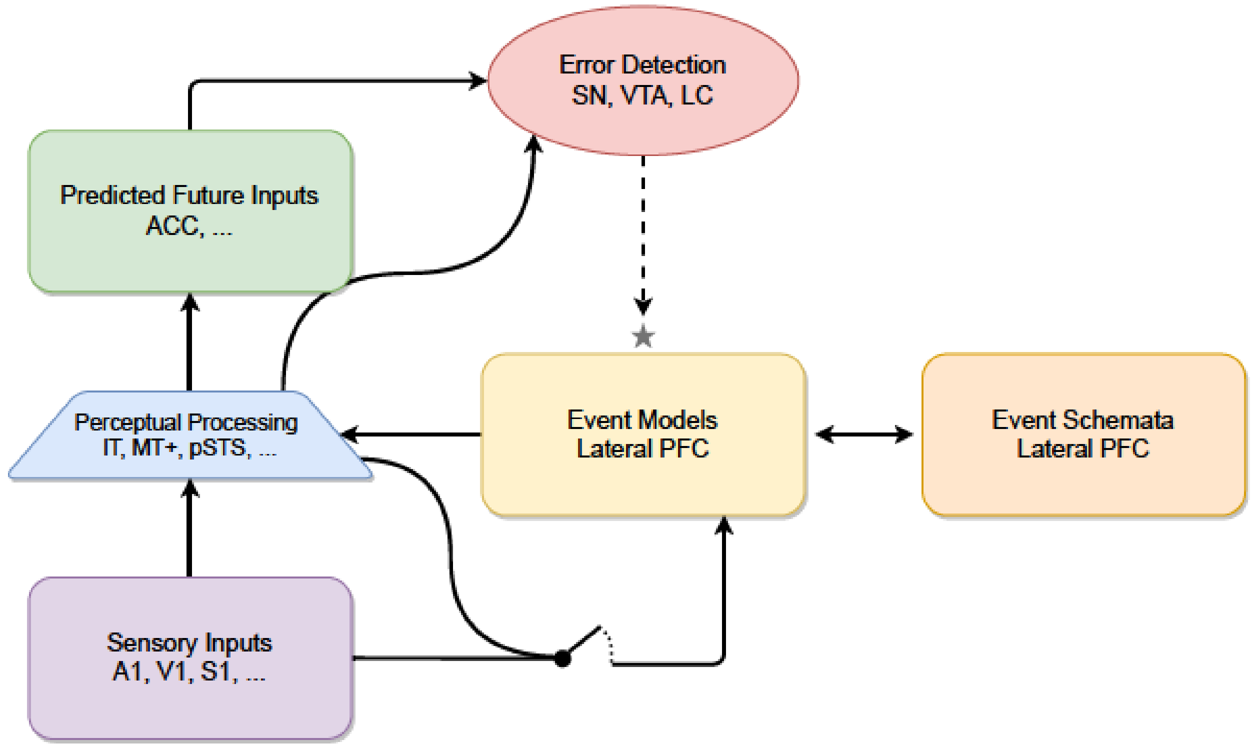


Figure 2.1: Overview of Event Segmentation Theory (EST). The information flow is adapted from [281]. The letter acronyms refer to the different brain regions where these activities are happening, as discovered by cognitive neuroscience experiments. The "Perceptual Processing" unit uses "Sensory Inputs" to extract higher-level features relevant to the predictive task. The extracted features are used to predict future perceptual features and compare them to the actual future perceptual features. The "Error Detection" module computes the prediction error, which generates a reset signal (denoted by $\star$) at high prediction errors. The reset signal updates the "Event Model" by accepting input from the "Sensory Inputs" and "Perceptual Processing" blocks. Figure reused with permission in Appendix D.

Prediction error is the mismatch between the brain's expectations and the actual sensory input. Such errors form a learning signal, indicating that the working event model requires updating to maintain high prediction accuracy. When the model's predictions fail to align with the actual sensory input features, a significant change in the environment may be occurring. This change marks the end of one event and the beginning of another, referred to as an event boundary.

Transient spikes in prediction error indicate that the current event has concluded, prompting the model to revise its working event model. For example, while watching someone cook, the model might predict that the next action will involve stirring a pot. If the person suddenly stops and answers the phone instead, the model experiences a prediction error. The shift in action is flagged as an event boundary, and the working event model is adjusted to reflect this new state of affairs. Note that the event segmentation process in EST (i.e., detecting event boundaries) occurs as a side effect of perceptual processing and does not require additional computations.

2.1.3 Event Schemata and Long-Term Memory

While the working event model operates dynamically in real-time, there is also a need for more stable, long-term representations of event models. The event schemata serve as a repository for a wide range of event models that have been stored in memory over time. These models include information about objects, actions, goals, and even social interactions, all of which are derived from past experiences.

At the boundary between events, when prediction errors signal that a change has occurred, the working event model is stored in the schemata, and a more suitable model is retrieved to explain the new sensory input. For instance, if the current event model no longer aligns with the input — such as shifting from a walking scenario to a conversation — the event schemata are accessed to retrieve a more suitable model for understanding the new context (i.e., conversation event), while the updated event model for walking is stored in the schemata.

Event schemata are updated more slowly compared to the working event model. This slower rate of updating allows the schemata to integrate new information while retaining stable representations of frequently encountered events. This ensures that the model can adapt to new situations without constantly rewriting its long-term memory models. The combination of the flexible, short-term working event model and the more stable, long-term event schemata provides a comprehensive framework for perceiving and processing a continuous stream of sensory information.

2.2 Hawkins's Theory of Intelligence

Hawkins's theory of intelligence [82] provides a comprehensive model that redefines our understanding of how the cortex might function. This section begins by revisiting Mountcastle's observations of the cortex, emphasizing the columnar organization of the Neocortex and the role of cortical columns in sensory processing. We then discuss the Hierarchical Temporal Memory (HTM), a framework that highlights the brain's ability to learn temporal patterns and make predictions using sparse distributed representations. Finally, we discuss the Thousand Brains Theory (TBT) and voting across cortical columns.

2.2.1 Mountcastle's Observations

2.2.1.1 Columnar Organization

Vernon Mountcastle described a modular, laminar structure of the cortex [170], where the neurons are organized into vertical structures called cortical columns. These columns are repeated throughout the cortex and vary from 300 µm to 600 µm in diameter [57, 84]. These columns extend through the six layers of the cortex and represent the basic functional units of cortical processing. A cortical column - spanning from the surface of the cortex (layer 1) to the white matter below layer 6 - comprises smaller units called minicolumns, which consist of 80-100 neurons each. Mountcastle focused on the similarities of the neuronal structure across the Neocortex rather than the structural differences between cortical regions.

2.2.1.2 Receptive Field

Receptive fields are zones in which external stimuli can activate neurons within a column. In somatosensory regions, for example, each neuron in a minicolumn shares a common receptive field, meaning they respond to stimuli from the same area of the body. Thus, the columnar organization allows for precise, localized sensory processing. In Mountcastle's research [170], based on nerve regeneration experiments [110], microelectrode penetrations perpendicular to the cortical surface revealed that neurons within a column respond to the same type of stimulus, with their receptive fields nearly superimposed. Conversely, tangential penetrations across the cortical surface encountered distinct columns with differing receptive field properties, showing abrupt shifts in response properties from one column to another.

2.2.2 Hierarchical Temporal Memory

2.2.2.1 Dendritic Specialization

Unlike artificial neurons, which assume uniform dendritic depolarization and synaptic input integration, biological neurons exhibit dendritic specialization [83], where distinct types of dendrites serve different functional roles based on their proximity to the soma. Excitatory pyramidal cells, for instance, are characterized by proximal, basal, and apical dendrites, each with unique contributions to cellular activity. Proximal dendrites receive driving synapses, capable of directly inducing action potentials in the cell body. In contrast, basal and apical dendrites have modulatory roles, depolarizing the cell body without triggering any spikes.

2.2.2.2 Sparse Distributed Representations

Sparse Distributed Representations (SDRs) are high-dimensional binary vectors that capture the spiking state of neurons, where each bit corresponds to the on or off state of a neuron. The SDR is characterized by being sparse where the number of off-bits are much more than the on-bits, mirroring the sparse cellular activity observed in the neocortex [7, 83]. The use of SDRs in learning has been shown to significantly increase capacity while minimizing the risk of representation collision and false positives during matching tasks [7]. Sparsity also has other desirable properties, such as unions of representations, which can represent multiple possibilities simultaneously in the same encoding.

2.2.2.3 Online Learning and Lateral Inhibition

According to the Hierarchical Temporal Memory (HTM) framework [83], depolarization of neurons without triggering action potentials (caused by distal synapses) can still carry crucial information, indicating that the cell has entered a predictive state. In HTM, cells in the same minicolumn (i.e., sharing the same receptive field) exhibit lateral inhibition, such that a cell that fires inhibits all other cells in the same minicolumn from firing. When a cell is in a predictive state (depolarized by basal input), it is primed to fire before other cells. Therefore, when a predictive cell receives driving proximal input from the Lateral Geniculate Nucleus (LGN) of the Thalamus, it spikes and inhibits the other cells - sharing the same receptive field - from firing, resulting in a unique context encoding. This unique context prevents the overwriting of previously learned knowledge.

2.2.3 Thousand Brains Theory of Intelligence

Hierarchical Temporal Memory describes a sequential memory framework that takes place in layer 4 of a cortical column. It can learn to predict into the future and detect anomalies in the data. However, In 2021 [82], Hawkins extended HTM by proposing the thousand brains theory of intelligence, which proposed that “cortical columns are more powerful than currently believed” [86]. Each cortical column learns models of complete objects, and the columns vote [84] on which object is being perceived during recognition.

2.2.3.1 Reference Frames and Allocentric Locations

In the Thousand Brains Theory, Hawkins draws parallels between grid cells, which are known for their role in spatial navigation in the hippocampus [73, 77], and how the Neocortex might use similar mechanisms for defining locations of features within an object model. He proposes that each cortical column contains an internal reference frame that works similarly to how grid cells track positions in space. This reference frame helps the column map features of objects in relation to each other, allowing for allocentric (object-centered) perception [138]. Just as grid cells help animals navigate an environment by representing the environment's layout, the neocortical columns learn the structure of objects using locations relative to the object itself. Recent research has demonstrated the effectiveness of a two-layered model (i.e., GridCellNet) that combines a Hierarchical Temporal Memory (HTM) layer with grid cell modules for path integration in a single framework, to successfully perform visual object recognition [136].

2.2.3.2 Voting

One of the key ideas in the Thousand Brains Theory is that individual cortical columns work together by voting on what object is being perceived [84]. Each column independently forms its own hypothesis about the object and communicates with each other a stable representation of the object and its pose information. The correct object is determined when a consensus is reached across multiple columns, integrating diverse sensory inputs into a unified perception. This collective voting system allows the brain to be resilient to noise and ambiguity, as even if one column receives imperfect or partial information, the correct hypothesis can still emerge from the broader agreement among columns, as shown in Figure 2.2.

2.3 Heeger's Theory of Cortical Function

Traditional models often describe sensory processing as a sequential, hierarchical operation where each stage processes inputs from the previous one in a unidirectional flow. These feedforward architectures, particularly in artificial neural networks, have shown success in modeling various neurophysiological and psychophysical phenomena [69]. However, Heeger's theory [91] introduces an alternative computational model that is based on observations of information flow in the neocortex. Heeger proposes a framework where information flow is not strictly feedforward but is instead driven by a combination of feedforward inputs (bottom-up sensory information), feedback signals (top-down contextual information), and prior expectations (learned predictions of sensory inputs).

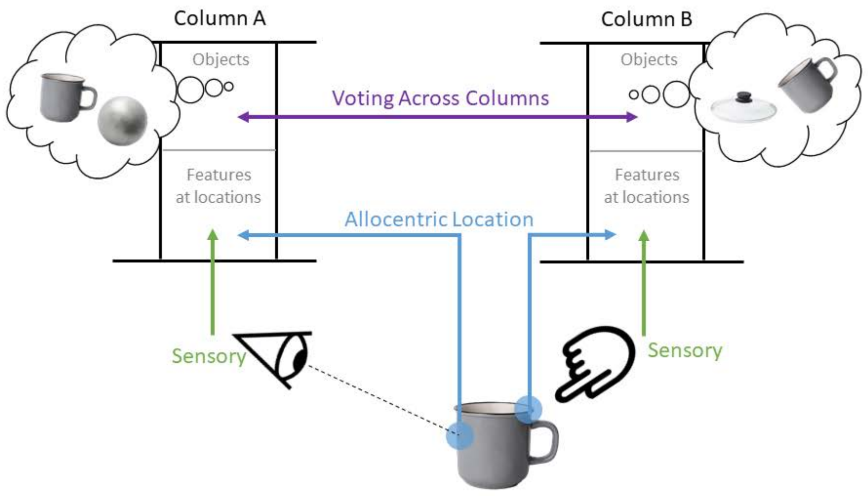


Figure 2.2: Overview of Thousand Brains Theory (TBT). An illustrative figure showing two cortical columns voting on the perceived object. Each column can be attached to a different sensory modality (e.g., touch, vision) and forms its own representation of the same object. Every column receives the sensory feature at an allocentric location, i.e., with respect to the object.

Heeger's theory suggests that neuromodulators play a key role in controlling brain states, which determine how neural processing shifts between different types of information flow. According to Heeger, brain states can range from being dominated by feedforward drive (a bottom-up process where information flows from lower to higher processing areas) to being controlled by inference-based top-down processes (where prior knowledge or expectations influence perception) or a hybrid of both. These state parameters are not static and can be modulated by specific neuromodulators (ACh) and brain oscillatory activity. By adjusting the state parameters, the brain can transition between different modes of operation — whether purely sensory-driven, inference-driven, or a combination of the two, enabling it to adapt flexibly to different perceptual or cognitive demands.

Heeger defines a single energy function which computes a feedforward drive and a prior drive. Minimizing this energy function by taking its derivative adds an additional term for feedback drive. Having both feedforward and feedback terms in the energy minimization function has an effect of balancing both terms. As a result, a neuron's response is driven towards the value of the feedforward processing of its inputs, and vice versa. The prior term accounts for sensory predictions, where minimizing the prior drive in the energy formulation minimizes the mismatch between predicted and observed values. Balancing the three drives with a single energy formulation provides an elegant approach to balancing perception, inference, and prediction in the brain.

2.4 Rao's Theory of Active Predictive Coding

2.4.1 Predictive Coding

Predictive coding (PC) proposes a framework for the hierarchical processing of information. It was initially formulated as a time series compression algorithm to create a more efficient coding system [52, 174]. A few decades later, PC was used to model visual processing in the Retina [104, 235] as an inference model. In the seminal work of Rao and Ballard [194], PC was reformulated as a general computational model of the cortex. The main intuition is that the brain continuously predicts all perceptual inputs, resulting in a quantity of prediction error, which can be minimized by adjusting its neural activities and synaptic strengths. In-depth variational free energy derivation is provided in Appendix C.3.1.

PC defines two subgroups of neurons: value and error. Each neuron contains a value node sending its prediction to the lower level through learnable function , and error node propagating its computed error to the higher level. The total prediction error is computed as , which is minimized by first running the network value nodes to equilibrium through optimizing the value nodes . At equilibrium, the value nodes are fixed, and inferential optimization is performed by optimizing the functions . Both optimizations aim to minimize the same prediction error over all neurons. This propagation of error to equilibrium is shown to be equivalent to backpropagation but using only local computations [265]. The PC formulation has shown success in training on static and i.i.d data [79, 213, 265, 275]. More recently [241], Temporal Predictive Coding (tPC) has also shown some success in sequential memory tasks by modifying error formulation to account for a one-step synaptic delay through interneurons, thus modeling temporal associations between sequence inputs. A simplified illustration of PC is provided in Figure 2.3 (A).

2.4.2 Active Predictive Coding

Recently, Rao proposed an extension to Predictive Coding (PC) termed *Active Predictive Coding (APC)*, which takes into account the interaction between sensory and motor processes within the neocortex [193, 195]. As shown in Figure 2.3 (B), the APC framework introduces a hierarchical architecture consisting of two key components: a state-transition function ( ) and an action-policy function ( ). The state-transition function models the evolution of the system's state over time by predicting the transition from a previous state to the current state , conditioned on the prior state and action ( ). The action-policy function generates the next action based on the current state and prior action , effectively closing the sensory-motor loop.

This recurrent interaction between state and action networks offers a framework that aligns with the cortical laminar structure. Rao hypothesizes that the state-transition computations are primarily carried out in cortical layers 2/3, responsible for sensory input integration, while the action-policy computations occur in layer 5, which is traditionally associated with motor output [115]. Furthermore, layer 6 is postulated to compute predictions, which are then propagated to lower regions in the cortical hierarchy. At the lowest level, layer 6 computes predictions of sensory inputs. APC presents working examples of compositional processing of visual inputs (i.e., images), where hierarchical, top-down processes facilitate the recognition of complex objects by breaking them down into constituent parts (i.e., part-whole hierarchy).

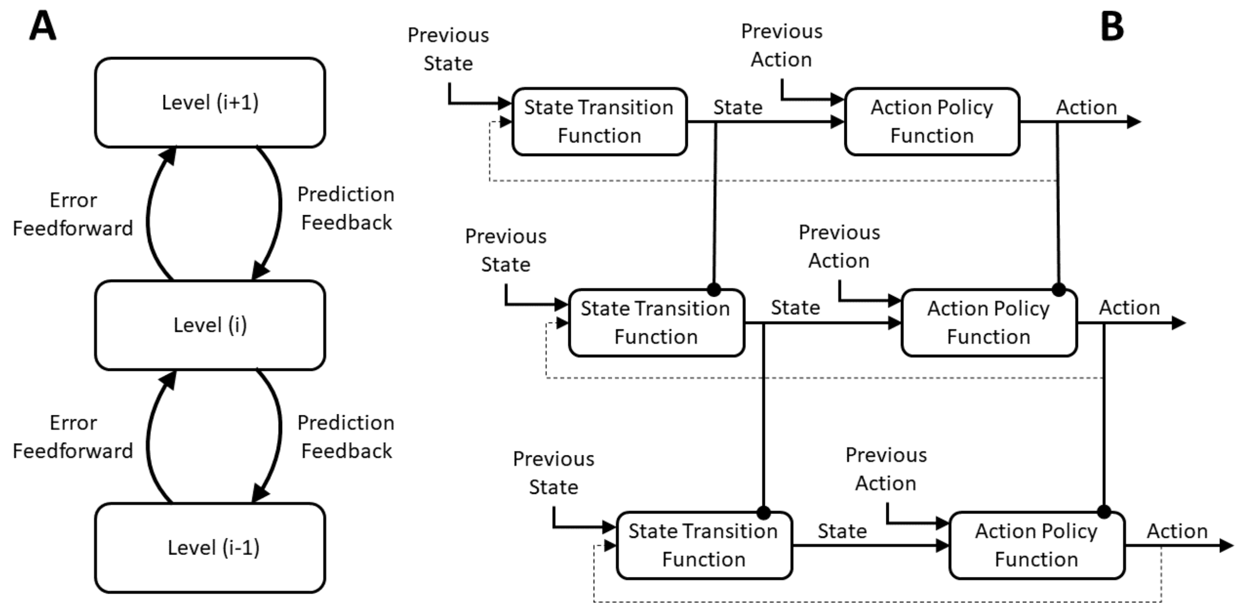


Figure 2.3: Overview of Predictive Coding (PC) and Active Predictive Coding (APC). (A) In Predictive Coding, errors in prediction are fed forward to the higher levels, whereas a prediction of the sensory information is fed backward down the stack of layers. (B) In Active Predictive Coding, each layer forms a sensory-motor loop of states and actions. The higher-level state-action pairs become priors in lower-level predictions. Dashed lines represent a time-step delay, and lines with a circular termination represent learnable transformation functions.

2.4.3 Deep Predictive Coding

Other PC-inspired models have diverged from the biologically plausible constraints by training through backpropagation through time as opposed to local learning rules (e.g., Hebbian-based plasticity). Some of these models still adhere to general principles of predictive coding and show compelling results despite using biologically-implausible deep recurrent models. One example is PredNet [152], and subsequent models [186], which builds a hierarchical model of multiple stackable layers with prediction error being sent up the hierarchy and predictions from recurrent models are sent down the hierarchy.

2.5 Hopfield's Model of Pattern Completion

Attractor dynamics refer to mathematical models that describe the behavior of dynamical systems. In our review, we focus on fixed point attractors, specifically Hopfield Networks [103] (HN), which is an instance of associative memory models [111, 113, 122]. Hopfield Networks are capable of storing and retrieving patterns even when presented with an incomplete or noisy query. The network operates by minimizing an *energy function* that guides it toward stable configurations (i.e., the stored patterns). Each pattern in the Hopfield network is encoded as a stable state, known as an attractor, in the system's energy landscape. When the network is given a partial or distorted version of a pattern, it iteratively updates its internal states to converge toward the nearest attractor, eventually recalling the complete pattern.

Consider, an ordered sequence of consecutive patterns , where . We refer to the Universal Hopfield Networks (UHN) framework [162] to describe all variants of HN architecture using a similarity (sim) function and a separation (sep) function, as shown in equation 2.1. This family of models can be viewed as a memory recall function, where a query (i.e., noisy or incomplete pattern) is compared to the existing patterns to compute similarity scores using the "sim" function. These scores are then used to weight the projection patterns after applying a "sep" function to increase the separation between similarity scores. The classical HN uses symmetric weights to store the patterns; therefore, it cannot be used to model temporal associations in a sequence. The asymmetric Hopfield Network (AHN) [231] uses asymmetric weights to recall the next pattern in the sequence for a given query .

$$\underbrace{\ }_{\text{Projection}}\ \underbrace{\text{sep}}_{\text{Separation}}\ \underbrace{\text{sim}\ \overline{\ \ }\ \overline{\ \ }}_{\text{Similarity}} \qquad \begin{array}{ll} \text{sep}\ \ \text{sim} & \text{Asymmetric Weights} \\ \text{sep}\ \ \text{sim} & \text{Symmetric Weights} \end{array} \tag{2.1}$$

When a dot product “sim” function and identity “sep” function are used, we get the classical HN [103] and AHN [231]. A few variants have been proposed to increase the capacity of the model. Recently, [30] has extended AHN by using a polynomial (with degree) or a softmax function (with temperature) as the “sep” function. HN can also be applied to continuous dense patterns [30, 126, 192].

2.6 The Tollman-Eichenbaum Machine

The Tolman-Eichenbaum Machine (TEM) [267] is a computational framework designed to model how the hippocampal-entorhinal system supports both spatial and relational memory tasks. It builds upon cognitive map theory (originally proposed by Tolman in 1948 [246]) to create a unified mechanism for representing abstract structural knowledge. At the core of the TEM model is the *factorization* of sensory and relational knowledge into separate representations, as shown in Figure 2.4. More specifically, TEM separates relational memory (i.e., relationships between experiences) from the content of those experiences to learn generalizable structural knowledge that can be applied to different sensory experiences. For example, a cognitive map of the family tree structure can be learned and transferred to different instances of family trees. This allows TEM to make novel predictions of sensory inputs based on the inferred relational representation. In line with the work of Manns and Eichenbaum [155], TEM assumes that the *medial entorhinal cortex (MEC)* encodes spatial structural knowledge, while the *lateral entorhinal cortex (LEC)* encodes sensory information.

While TEM demonstrates compelling similarities between the learned representations and neural activity in both the hippocampus and entorhinal cortex (e.g., grid and object-vector cells), some aspects of TEM’s implementation are not biologically plausible. Notably, the learning of structural representations in TEM relies on the backpropagation of gradients through a deep recurrent network. This process resembles slow learning of the general relational structure (“outer loop” process). In contrast, the learning of specific contents occurs through fast Hebbian learning ("inner loop" process). The binding between specific sensory inputs and a structural cognitive map is implemented using an associative attractor network [103].

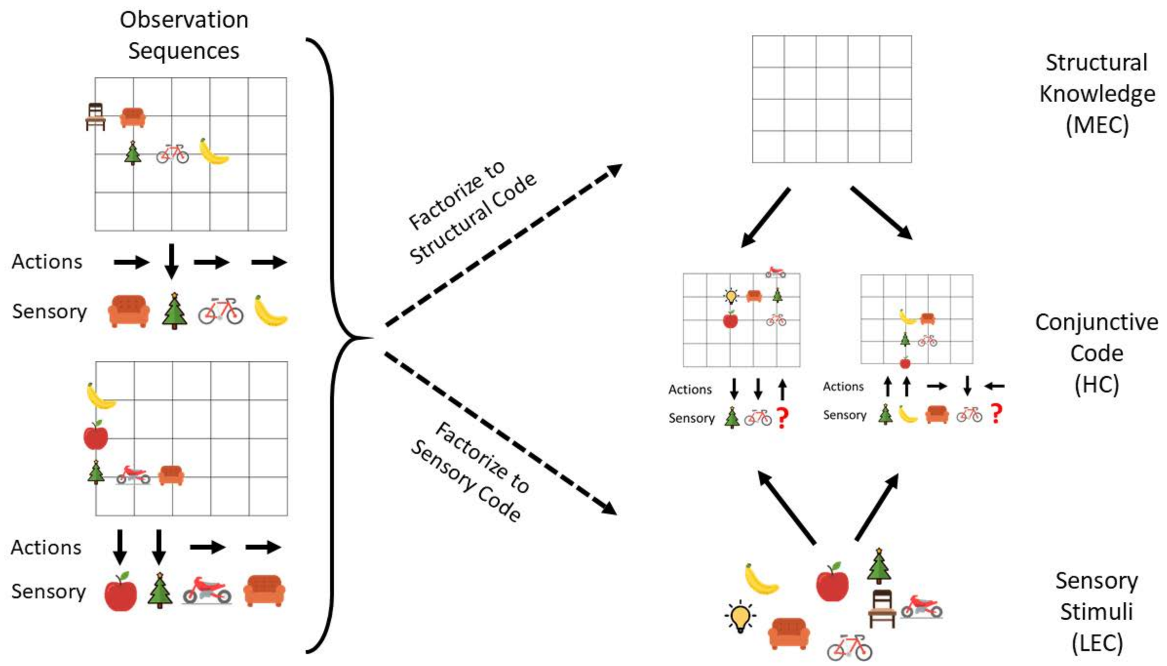


Figure 2.4: Overview of the Tolman-Eichenbaum Machine (TEM). The learning in TEM factorizes sequences of sensory features and actions into structural knowledge and sensory stimuli. This factorization allows for building associative maps (i.e., conjunctive code) for new environments using the same structural priors. MEC, LEC, and HC denote Medial Entorhinal Cortex, Lateral Entorhinal Cortex, and Hippocampus, respectively.

2.7 Hinton's Theory of Learning Agreement Islands

2.7.1 Capsules and Routing

2.7.1.1 Dynamic Allocation of Neurons

Capsules, as introduced Hinton [93, 209], aim to solve the problem of how neurons represent objects and their parts in a hierarchical manner, ideally constructing a parse tree of objects. In standard neural networks, neurons cannot rapidly change their weights to represent different parts dynamically. Instead, groups of neurons, known as capsules, are allocated to represent specific entities, such as an object or a part of an object. These capsules generate activity vectors that encode the object's properties, such as pose, position, and orientation. The length of this activity vector represents the probability that the object exists, while the orientation describes the instantiation parameters (i.e., object properties).

2.7.1.2 Iterative Routing Algorithm

To address the challenge of connecting capsules across layers, Sabour, Frosst, and Hinton [209] proposed an iterative routing algorithm known as "routing by agreement". In this algorithm, lower-level capsules predict the instantiation parameters of higher-level capsules through transformation matrices. In practice, each capsule is encouraged to align with a capsule in the layer above it to become its parent in the parse tree. If multiple predictions align, the higher-level capsule becomes active. The routing process iteratively adjusts the connection strengths between capsules, ensuring that parts are correctly assigned to wholes. This method provides a more effective alternative to max-pooling.

2.7.2 The GLOM Architecture

2.7.2.1 GLOM Structure

The GLOM architecture [96] is inspired by the organization of cortical columns, also known as hypercolumns, in the brain's visual cortex [106]. Each GLOM column represents different hierarchical levels of visual processing within a fixed spatial region. These columns consist of multiple layers of autoencoders, each responsible for transforming the embedding at one level into an embedding at an adjacent level through a combination of bottom-up and top-down signals. Each GLOM column has its own receptive field, which determines the region of the image it processes. This receptive field expands as the information moves up the hierarchy, similar to how receptive fields in the visual cortex become larger and more complex at higher levels of abstraction.

2.7.2.2 Embedding Computation

In GLOM, part-whole representations are handled through embeddings that reflect the relationship between parts and wholes at various levels of a hierarchy. At each level, interactions occur not only vertically between layers in a column but also horizontally with nearby columns. Each embedding at a particular level is updated iteratively, considering four key factors: (1) bottom-up prediction from the level below, (2) top-down prediction from the level above, (3) the embedding from the previous timestep, and (4) attention-weighted contributions from neighboring columns. These contributions are combined together to form a weighted average that refines the representation over time.

2.7.2.3 Emergence of Agreement Islands

As attention-weighted contributions encourage similar vectors to cluster together, agreement islands emerge, where nearby embeddings converge toward similar values. At higher levels in the hierarchy, these islands grow larger, forming nodes of a parse tree. For example, if the image is of a face, one column might initially represent a small feature, such as a nostril. As information moves up the levels of the column, the receptive field grows, allowing the column to represent larger parts of the face, such as the nose or eyes, in the same column.

2.8 Lecun's Theory of Predictive Architectures

2.8.1 Energy-Based Models

Energy-Based Models (EBMs) serve as the theoretical foundation for Lecun's predictive architectures [132]. At their core, EBMs assign a scalar "energy" value to measure the compatibility between two inputs. For instance, a model can learn to evaluate the energy of video frames, where consecutive frames have low energy and non-consecutive frames have high energy due to incompatibility. This enables the model to learn multiple valid possibilities for any input (i.e., low energy values for all compatible pairs), modeling stochastic and ambiguous solutions. The primary objective is to minimize this energy by optimizing the model's parameters. Unlike generative models, EBMs prioritize learning abstract representations and useful world models over generating accurate predictions. The energy formulation allows for representing multiple compatible possibilities, but generating a single possibility requires navigating the energy landscape to find a low-energy solution.

2.8.2 Joint-Embedding Predictive Architecture

The Joint-Embedding Predictive Architecture (JEPA) builds on the principles of EBMs, using two separate encoders to process input data. The architecture allows independent encoders to handle different modalities. For example, one encoder might process visual inputs while another processes audio inputs. The system then learns to predict one modality from the representation of another, thus supporting multi-modal learning. By embedding both inputs into a shared latent space, JEPA enables the model to learn relationships between different types of data in an abstract space.

JEPA can be trained using contrastive or non-contrastive approaches. Contrastive approaches train the model to output low energy for positive pairs and high energy for incompatible negative pairs. Such contrastive objectives require negative pairs for training. In contrast, non-contrastive approaches rely on regularization techniques to "measure the volume of space that can take low energy values" [132]. This can be mainly achieved through maximizing the information content of the encoder embeddings, as well as learning embeddings that are easily predictable from each other in the representation space, i.e., minimizing the energy term. VICReg [18] and Barlow Twins [282] are two examples of such architectures. More recently, JEPA-based architectures [15, 17, 19] have shown success in learning useful and transferable representations.

2.8.3 Hierarchical JEPA

Hierarchical JEPA (H-JEPA) builds on top of the JEPA framework by adding multiple layers of abstraction, allowing for multi-level predictions. Each layer of the hierarchy represents increasingly abstract representations, enabling the model to predict not just immediate future states but also longer-term, more complex representations. At the lower levels, the system might predict basic sensory information, such as object positions or movement trajectories. At higher levels, it can predict more abstract representations, such as actions, strategies, or goals. This hierarchical approach allows H-JEPA to perform predictions across different time scales, from immediate reactions to long-term planning. However, H-JEPA remains a theoretical concept that has yet to be implemented and proven effective in practice.

Chapter 3: Single-Layered Predictive Model

"It is difficult to make predictions, especially about the future."

— Niels Bohr

Advances in visual perceptual tasks have been mainly driven by the amount and types of annotations of large-scale datasets. Researchers have focused on fully-supervised settings to train models using offline epoch-based schemes. Despite the evident advancements, limitations and cost of manually annotated datasets have hindered further development for event perceptual tasks, such as detection and localization of objects and events in videos. The problem is more apparent in zoological applications due to the scarcity of annotations and length of videos - most videos are at most ten minutes long. Inspired by cognitive theories, we present a self-supervised perceptual prediction framework[2] to tackle the problem of temporal event segmentation by building a stable representation of event-related objects. The approach is simple but effective. We rely on LSTM predictions of high-level features computed by a standard deep learning backbone. For spatial segmentation, the stable representation of the object is used by an attention mechanism to filter the input features before the prediction step. The self-learned attention maps effectively localize the object as a side effect of perceptual prediction. We demonstrate our approach on long videos from continuous wildlife video monitoring, spanning multiple days at 25 FPS. We aim to facilitate automated ethogramming by detecting and localizing events without the need for labels. Our approach is trained in an online manner on streaming input and requires only a single pass through the video, with no separate training set. Given the lack of long and realistic (includes real-world challenges) datasets, we introduce a new wildlife video dataset - nest monitoring of the Kagu (a flightless bird from New Caledonia) - to benchmark our approach. Our dataset features a

[2]This chapter was published in the International Journal of Computer Vision. Permission is included in Appendix D

video from 10 days (over 23 million frames) of continuous monitoring of the Kagu in its natural habitat. We annotate every frame with bounding boxes and event labels. Additionally, each frame is annotated with time-of-day and illumination conditions. We find that the approach significantly outperforms other self-supervised, traditional (e.g., Optical Flow, Background Subtraction) and NN-based (e.g., PA-DPC, DINO, iBOT) baselines and performs on par with supervised boundary detection approaches (i.e., PC). At a recall rate of 80%, our best-performing model detects one false positive activity every 50 minutes of training. On average, we at least double the performance of self-supervised approaches for spatial segmentation. Additionally, we show that our approach is robust to various environmental conditions (e.g., moving shadows). We also benchmark the framework on other datasets (i.e., Kinetics-GEBD, TAPOS) from different domains to demonstrate its generalizability.

3.1 Introduction

Much effort in computer vision science has been devoted toward tracking, modeling, and understanding the behavior of humans [51, 92, 120, 182]; however, fewer works explore the same tasks for other species [88, 243]. The scarcity of annotation and difficulty of collecting data has a significant contribution to the slow progress of the field [88]. Recent developments in digital video and recording technology opened new opportunities to study animal behavior in the wild. Not only do these recent developments increase the volume of data, but also the speed at which data is generated. Monitoring video systems can generate data at faster rates than those that can be handled by AI systems, leading to the notion of fast data [129]. Batch (offline) learning systems fail to process fast data, where - possibly infinite - items arrive in a temporal sequence [210], due to the need to access old data. Revisiting old data not only wastes computational resources but also requires that the data be stored, which is impractical for large volumes of data and hinders the scalability of the learning systems. For streaming input applications (e.g., monitoring and surveillance), storage costs can exceed computational costs. The goal is to process and summarize streaming input data to store only useful high-level information, such as detected events and their objects' attributes while ignoring spatial and temporal background clutter.

Stream Learning (SL) aims to build adaptive models of the streaming data [20]. These models must be updated after every data point without access to any past data. Additionally, the SL systems need to account for distribution shifts by adapting to the changes in the streaming data. Our training scheme completely disregards data points after being processed by the network. Training and inference are done simultaneously, alleviating the need for epoch-based training in order to appeal to practical applications and reduce training time. We address the distribution shift problem by building adaptable *event models* which contain a feature representation of the event and its objects (i.e., bird), allowing us to effectively segment the object from its surrounding at every frame - despite changes in illumination and environmental conditions.

Ethology is the scientific and objective study of animal behavior, from which stems the term *Ethogram*. Ethogramming defines and categorizes animal behaviors in a completely objective manner. It is important to avoid subjectivity and bias when describing animal behavior in ethograms; behavior definitions must be based on "mechanical actions that are observed rather than on any intentionality, motivating the expression of that behavior" [203]. The required objectivity in the behavior descriptions makes computer vision algorithms a suitable candidate for automating ethograms. Ethogramming can ideally be automated by correlating low-level attributes (e.g., location, motion patterns, time-of-day) to high-level behaviors (e.g., incubation) [88]. Our goal is to detect the low-level attributes used in describing the behavior of animals in ethograms, such that these attributes can then be used to generate and localize objective behavior labels automatically.

Events are ubiquitous; they are the building blocks of videos. Events can be present at various time scales, whereas actions are a type of event that occurs at a smaller temporal scale. For example, the event of a bird feeding its chick can be composed of multiple feeding actions, where *feeding* here refers to the activity. To truly understand videos, algorithms must be capable of detecting and segmenting significant events from background noise. The detected events can then be encoded in an embedding space for representation learning [146, 176], annotated and

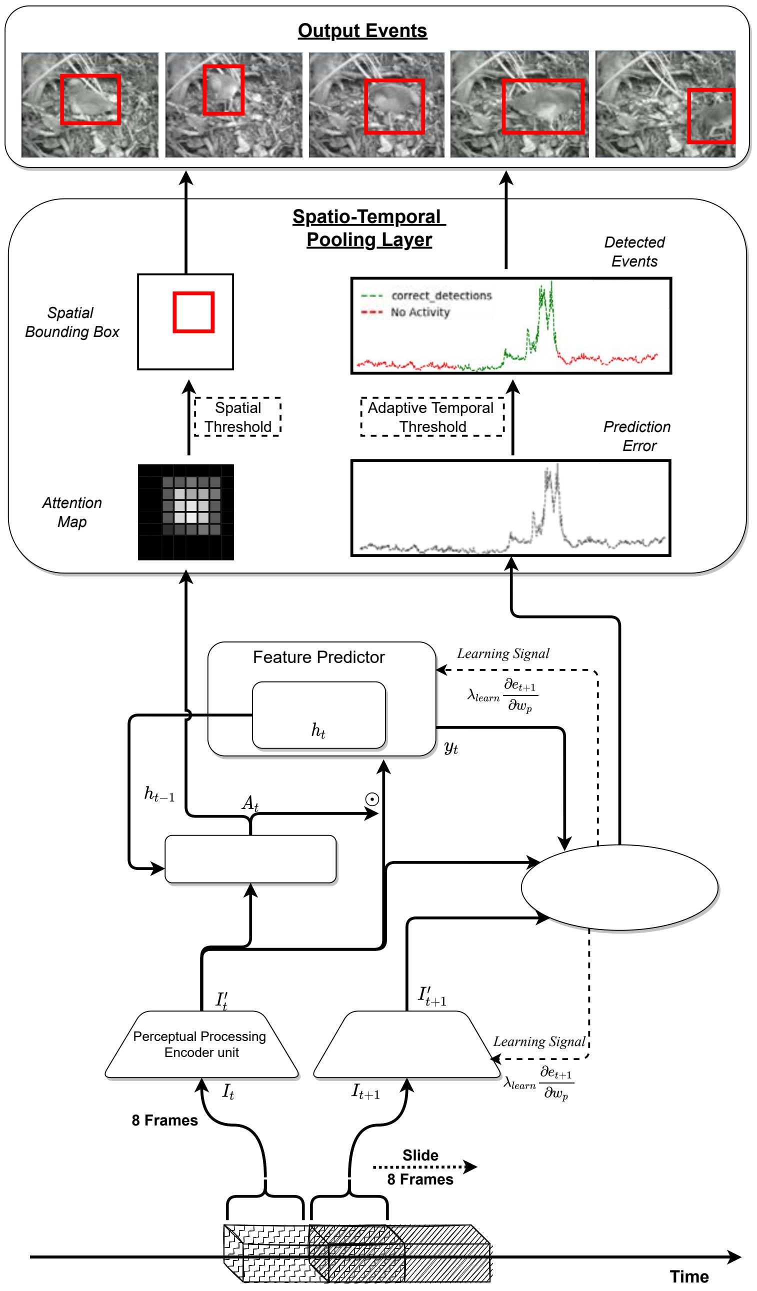


Figure 3.1: Overview of the single-layer predictive architecture. The perceptual processing unit encodes current frames and future frames into grid feature representations. An attention operation is applied to the current features to spatially segment the event objects. The predictor combines the event model representation with the current features to predict future features. Error in the prediction is used as a learning signal for the trainable weights. The spatio-temporal pooling layer receives as input spatial localization map and prediction error signal and outputs the detected events.

classified [14, 141, 147, 270], or even used for video summarization tasks [13, 55, 108, 286]. The task of detecting prominent events becomes even more important when processing long videos for wildlife monitoring or even video monitoring of other contexts [36]. To perceive events, one must not rely on the noisy low-level features but instead, build algorithms that detect the dynamics and patterns of high-level features. These high-level features should be sufficient to capture object-level representations; the designed architectures should ideally capture these features' temporal evolution over time.

Very few publications show event detection and localization performance on video spanning several days, mainly due to the challenges arising from storing, processing, and evaluating on large datasets. Researchers have mainly focused on event-centric datasets by using trimmed videos of actions, thus eliminating the need to segment events. This is analogous to removing background from images and using only masks of objects to train an object classifier. Although this makes the task easier on the learning model, it becomes harder to generalize to real-world applications. Real-world datasets contain more challenging scenarios of untrimmed events (e.g., empty segments) and noisy low-level changes in the foreground (e.g., occlusions, shadows). Most wildlife monitoring studies use camera traps with motion triggers to store images of animals or record short videos of animals in controlled environments - eliminating real-world lighting variations and environmental conditions. We address the issues mentioned above by collecting and annotating a ten-day continuous monitoring dataset of a nest of the Kagu; a flightless bird of New Caledonia. The dataset is sampled at 25 FPS, offering more than 23 million frames and more than 253 hours of video footage. Events are not trimmed to allow researchers to evaluate the models' performances in more realistic and challenging scenarios. Ideally, algorithms will have to deal with overfitting, catastrophic forgetting, and sparsity of events to maintain high performance in spatial and temporal event segmentation tasks.

Our framework follows key ideas from the perceptual prediction line of work in cognitive psychology [151, 278, 279, 280]. Research has shown that "event segmentation is an ongoing process in human perception, which helps form the basis of memory and learning" [190, 281].

Humans can identify event boundaries, in a purely bottom-up fashion, using a perceptual predictive model that predicts future states based on the current perceived sensory information. Experiments have shown that the human perceptual system identifies event boundaries based on the appearance and motion cues in the video [189, 234, 277]. Our model implements this perceptual predictive framework and introduces a motion-weighted loss function to allow for the localization and processing of motion cues.

As shown in Figure 3.1, our approach uses a feature encoding network to transform low-level perceptual information into a higher-level feature representation. The model is trained to predict the future perceptual encoded input and signal an event if the prediction is significantly different from the future perceived features (i.e., unpredictable features). The prediction error signal is used to flag events and train the model to predict better features. Prediction occurs after aggregating higher-level features with a recurrent cell; the hidden state incorporates a higher-level representation of the movement cues within frames. We utilize an attention mechanism on the backbone features to provide a more focused prediction error signal and, more importantly, spatially segment the frames. The error signal is used for temporal event segmentation, while an attention map segments a frame spatially. Our approach takes a significant step toward an eventual system that can detect adverse behavioral events without requiring a dataset of such behaviors. We aim to facilitate the goal of automated ethogramming, which is detecting and characterizing the behaviors of animals objectively without prior domain information or low-level annotations.

Our *key contributions* can be summarized as:

- Introducing a stream learning framework capable of detecting and localizing prominent events without prior knowledge of the target events.
- Collecting and annotating a remarkably long wildlife video monitoring dataset (over 23M frames) with frame-level bounding box annotation and behavioral categories. For all frames containing the bird (i.e., moving or stationary), a single frame-level bounding box is provided to highlight its location.

- Providing extensive evaluation of our framework's performance with respect to other traditional and state-of-the-art approaches on multiple domains and datasets.

3.2 Predictive Model

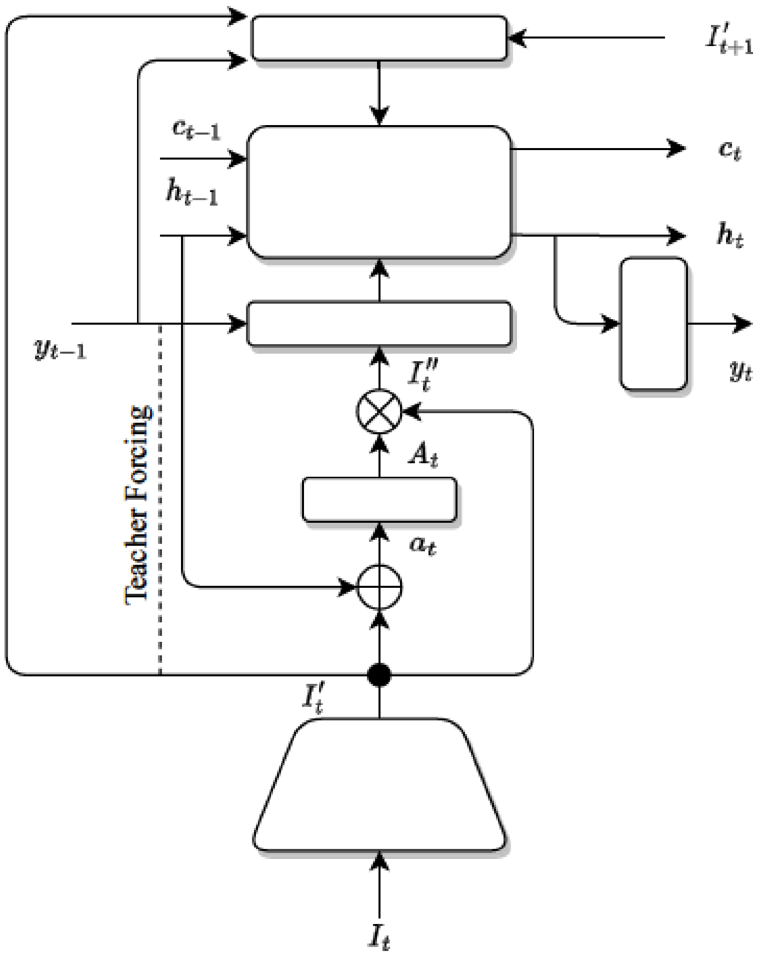


Figure 3.2: Detailed architecture of the perceptual prediction algorithm. Input frames from each time instant are encoded into high-level features using a deep-learning stack, followed by an attention overlay that is based on inputs from the previous time instant, which is input to an LSTM. The training loss is composed of the predicted and computed features from the current and next frames. $\oplus$ and $\otimes$ denote element-wise addition and multiplication operations, whereas I, y, h, c, and A represent input image, prediction, hidden state, cell state, and attention map, respectively. The teacher-forcing connection is discussed in the implementation details section.

Algorithm 1 Temporal Event Segmentation Model with Attention-based Spatial Event Localization. The input is an untrimmed/streaming video , which is a set of frame blocks , each of the size frames. The output is a boolean set of event predictions .

```
Input: Video frames
Output: Event prediction values
 1: procedure ATTENTION( )
 2:
 3:
 4:
 5:     return
 6: end procedure

 7: procedure SEGMENT( )
 8:         ENCODER
 9:           ENCODER
10:          ATTENTION
11:          LSTM
12:         DECODER
13:
14:         GATE
15:     return
16: end procedure

17:
18: for  do
19:                SEGMENT
20:
21:
22: end for
```

This section introduces the technical details of our cognitively-inspired stream learning framework. We first discuss the inspiration from cognitive theories in Section 3.2.1. The proposed architecture, summarised in Figure 3.2, can be divided into several individual components. We explain the role of each component, starting with the encoder network and attention unit in Section 3.2.2 & Section 3.2.3, followed by a discussion on the recurrent predictive layer in Section 3.2.4. We conclude by introducing the loss functions (Section 3.2.5) used for self-supervised learning as well as the output pooling layer (Section 3.2.6). Full pseudocode is provided in Algorithm 1.

3.2.1 Cognitive Inspiration

Our proposed framework is heavily inspired by cognitive psychology theories, more specifically, the Event Segmentation Theory (EST). The theory is developed by Zacks *el al.* [278] based on findings from cognitive neuroscience experiments. EST introduces a cognitive learning theory, where making predictions of future inputs plays a central role in both, learning and event segmentation. The theory posits that humans maintain a stable representation of the current event called the "event model". The event model and the current perceptual inputs are used to predict the future perceptual input. Failing to predict the next input is an indicator that the current event has changed, and the event model cannot be used to predict inputs received from a different event, which results in perceiving an event boundary. Thus, event segmentation does not require conscious attention; instead, it emerges as a side effect of the ongoing perceptual prediction process. The predictive learning approach allows for detecting generic events without providing a definition or a description of the target event.

The mismatch between the predicted future inputs and the actual future inputs drives the segmentation process to group events into discrete time intervals. EST is a continual perceptual process that segments a continuous stream of multimodal sensory input into a discrete and coherent set of events. The prediction error is also used as a training signal for the predictor to finetune the predictions within the same event. A more detailed explanation of EST is provided in Section 2.1.

Our approach uses an LSTM unit as a predictor where the hidden state builds an event representation over time. We also use a CNN backbone to extract features, resembling the perceptual processing unit in EST. We study the prediction error to detect and localize events in a stream of RGB images. Additionally, we utilize the event model representation inside the LSTM to spatially localize the objects in the input feature representation.

3.2.2 Input Encoding

The raw input images are transformed from pixel space into a higher-level feature space by utilizing an encoder (CNN) model. This encoded feature representation allows the network to extract features of higher importance to the task being learned. We denote the output of the CNN layers by where is the learnable weights and biases parameters and is the input image at time . The encoder network transforms an input image with dimensions to output features with dimensions , where is the spatial dimensions and is the feature vector length.

3.2.3 Attention Unit

In this framework, we utilize Bahdanau attention [16] to spatially localize the event in each processed frame. The attention unit receives as an input the encoded features and outputs a set of attention weights ( ) with dimensions . The hidden feature vectors ( ) from the prediction layer of the previous time step are used to calculate the output set of weights using Equation (3.1), expressed visually in Figure 3.2.

(3.1)

where represents hyperbolic tangent ( ) function, and represents a softmax function. The weights ( ) are then multiplied by the encoded input feature vectors ( ) to generate the masked feature vectors ( ). The attention function uses the object representation embedded in to filter the features and remove the noisy background features. In other words, the attention function ensures that the LSTM model receives only the object (i.e., bird) features when making its prediction. Interestingly, as we will see in the experimental section, this change does not have a significant quantifiable effect on the overall temporal segmentation performance due to the sparsity of events in the proposed dataset; however, it does enable spatial segmentation. We have experimented with other datasets (i.e., Kinetics) in different domains where multiple objects can be present. Results (Figure 3.18) show that the model can localize multiple objects simultaneously.

3.2.4 Prediction Layer

The process of future prediction requires a layer capable of storing a flexible internal state (event model) of the previous frames. For this purpose, we use a recurrent layer, specifically long-short term memory cell (LSTM) [100], which is designed to output a future prediction based on the current input and a feature representation of the internal state. More formally, the LSTM cell can be described using the function , where and are the output hidden state and previous hidden state respectively, is the attention-masked input features at time step and is a set of weights and biases vectors controlling the internal state of the LSTM. The input to the LSTM can be formulated as:

(3.2)

where is the masked encoded input feature vector and is defined as . The notation [.] represents vector concatenation.

3.2.5 Loss Function

The perceptual prediction model aims to train a model capable of predicting the feature vectors of the next time step. We define two different loss functions; prediction loss and motion-weighted loss.

3.2.5.1 Prediction Loss

This function is defined as the L2 Euclidean distance loss between the output prediction and the next frame encoded feature vectors .

(3.3)

3.2.5.2 Motion Weighted Loss

This function aims to extract the motion-related feature vectors from two consecutive frames to generate a motion-dependent mask, which is applied to the prediction loss. The motion-weighted loss function allows the network to benefit from motion information in higher-level feature space rather than pixel space. This function is formally defined as:

(3.4)

where denotes an element-wise multiplication operation.

3.2.6 Spatio-Temporal Pooling Layer

The spatio-temporal pooling layer is a decision-based function that receives as input a spatial attention map and continuous prediction error signal. The pooling layer generates a bounding box from the attention map and detects events from the error signal. The output of the layer is the detected events segmented spatially and temporally. We define pooling, in this context, as going from noisy frame-level representation of prediction error to event-level classification. Ideally, more layers of prediction and pooling can be built on top of this pooled event-level representation to create a hierarchy of event models [83, 84, 278].

3.2.6.1 Bounding Box Generation

We use the attention map output from Equation (3.1) to spatially localize the objects in the frames. However, it can be further processed to generate bounding boxes, which are more important to ethogramming. As shown in Figure 3.1, the model outputs an attention map, which we resize to the original image size using bilinear interpolation. Then, we apply MinMax scaling to the attention map and threshold it. The result is a binary map for which we calculate the bounding rectangle. We evaluate the performance of the attention maps and the generated bounding boxes in Section 3.4.2.2.

3.2.6.2 Activity Gating

The Activity gating function receives, as an input, the error signal defined in Section 3.2.5 and applies a thresholding function to classify each frame. The threshold function applies a threshold ( ) to each loss value, resulting in a binary signal (i.e.,). Groups of frames with loss above the threshold value are detected as events. Additionally, we define the parameter as the gap (in frames) between two consecutive events, below which the two events will be merged as one. The performance of our approach with different thresholds and model ablations are provided in Section 3.4.1.2.

3.2.6.2.1 Simple Threshold

The simple threshold function applies a constant threshold value to the prediction error signal **e** . Each loss value is compared against the threshold as shown in Equation (3.6).

$$\mathbf{e} \tag{3.5}$$

$$\begin{cases} & \text{if} \\ & \text{otherwise} \end{cases} \tag{3.6}$$

3.2.6.2.2 Adaptive Threshold

A simple threshold does not adapt to changes in the loss values over time; it is expected for the loss values to decrease as the model learns to predict better future features. Therefore, we propose to use an adaptive threshold function that defines the threshold value as an offset above the smoothed error signal **e** . First, we compute the smoothed values **e** as shown in Equation (3.7). The smoothing function acts as a low-pass filter to attenuate high-frequency noise in the raw prediction signal **e**. Then, we use Equation (3.8) to calculate the difference between the raw prediction signal and the smoothed signal. Finally, we apply a threshold to each value in **e** using Equation (3.9).

$$\begin{gathered} \mathbf{e} \\ \mathbf{e} \quad - \quad - \end{gathered} \tag{3.7}$$

$$\mathbf{e} \quad \mathbf{e} \quad \mathbf{e} \tag{3.8}$$

$$\begin{cases} & \text{if} \\ & \text{otherwise} \end{cases} \tag{3.9}$$

where represents a 1D convolution operation.

3.3 Nest Monitoring of the Kagu

3.3.1 Dataset Overview

We used a dataset of videos from nest monitoring of the Kagu [75]. The dataset consists of around ten days (253 hours) of continuous monitoring sampled at 25 frames per second. We fully annotated the entire dataset (23M frames) with spatial localization labels in the form of a tight bounding box. Additionally, we provide temporal event segmentation labels as five unique bird activities: Feeding, Pushing leaves, Throwing leaves, Walk-In, Walk-Out . The feeding event represents the period of time when the birds feed the chick. The nest-building events (pushing/throwing leaves) occur when the birds work on the nest during incubation. Pushing leaves is a nest-building behavior during which the birds form a crater by pushing leaves with their legs toward the edges of the nest while sitting on the nest. Throwing leaves is another nest-building behavior during which the birds throw leaves with the bill towards the nest while being, most of the time, outside the nest. Walk-in and walk-out events represent the transitioning events from an empty nest to incubation or brooding, and vice versa. We also provide five additional labels that are based on time-of-day and lighting conditions: Day, Night, Sunrise, Sunset, Shadows . While our approach ignores the lighting conditions and focuses only on object-centric (bird) activities, we include these labels to evaluate and contrast the different performances under different conditions. Ideally, a model with a robust representation of the bird should perform consistently well during all lighting conditions. A good model learns to extract useful features and inhibit background features. Figure 3.3 shows a sample of images from the dataset.

3.3.2 Dataset Statistics

We present the distributions of event counts and their durations in Figure 3.4. Both event counts and event durations are further categorized by Day and Night conditions. As can be seen from the stacked bar charts, the Walk-in, Walk-out, and Feeding contribute the highest number of events, whereas the Throwing event has the highest total duration. Most events have a high day-to-night ratio, except for the Throwing event, where around 40% of the event duration occurs during twilight periods. Feeding event occurs only in the daytime. It can be seen that both count and duration plots show the same day-to-night ratio for each event type, which means that the duration of each event at night equals its duration at day. In other words, the average duration of each event does not change based on the time of day.

Figure 3.5 shows the durations of different event types and bird states (stationary inside frame, moving inside frame, outside frame). As shown in the right subfigure, event duration varies

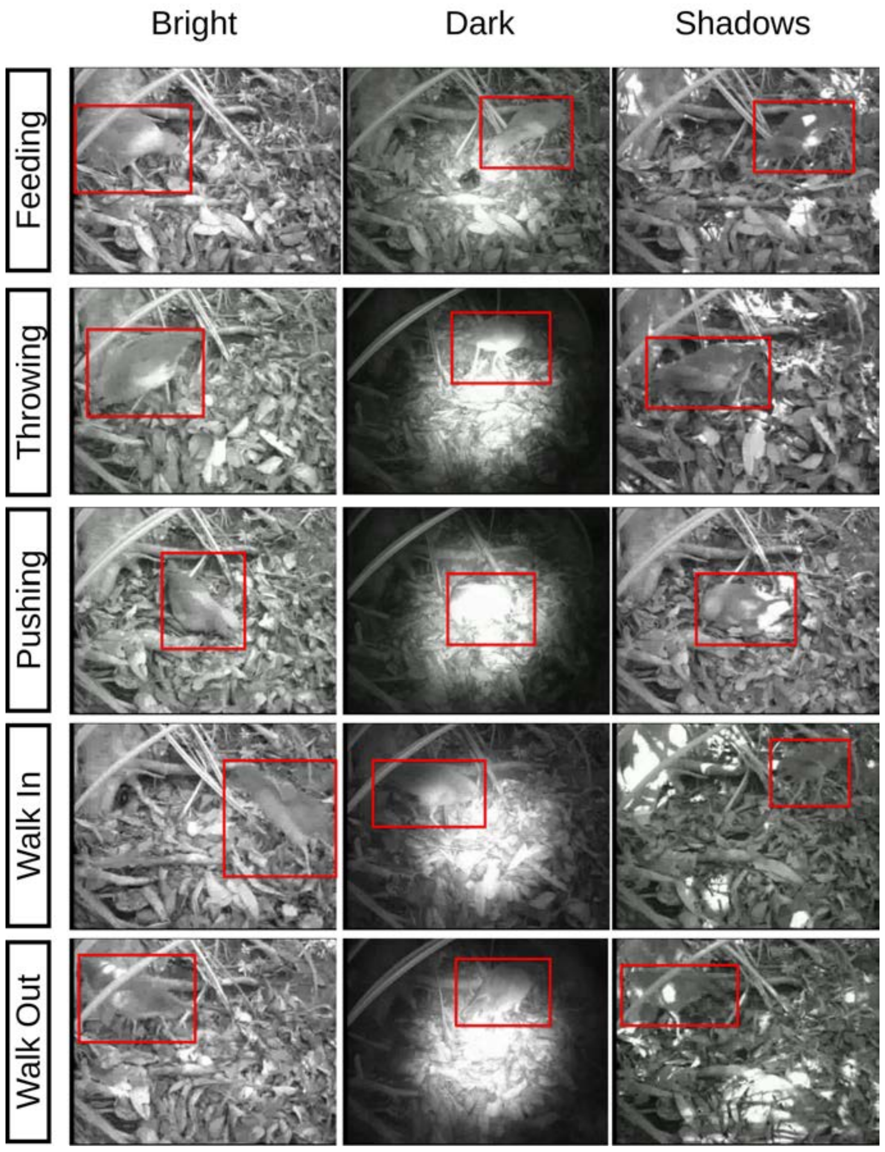


Figure 3.3: Samples of images from the nest of the Kagu dataset. Samples presented at different behavioral categories and lighting conditions. The red bounding box shows the ground truth label of the bird's location.

significantly based on its type. Throwing events can span over two minutes, whereas short events (e.g., Walk-In/Out) only take a few seconds. This variability in durations makes the dataset more challenging for activity detection algorithms. The left subfigure introduces a different kind of challenge - sparsity of events. We show that the dataset contains a total duration of around one hour of motion/activities (for the whole dataset), while for the rest of the dataset, the bird is either

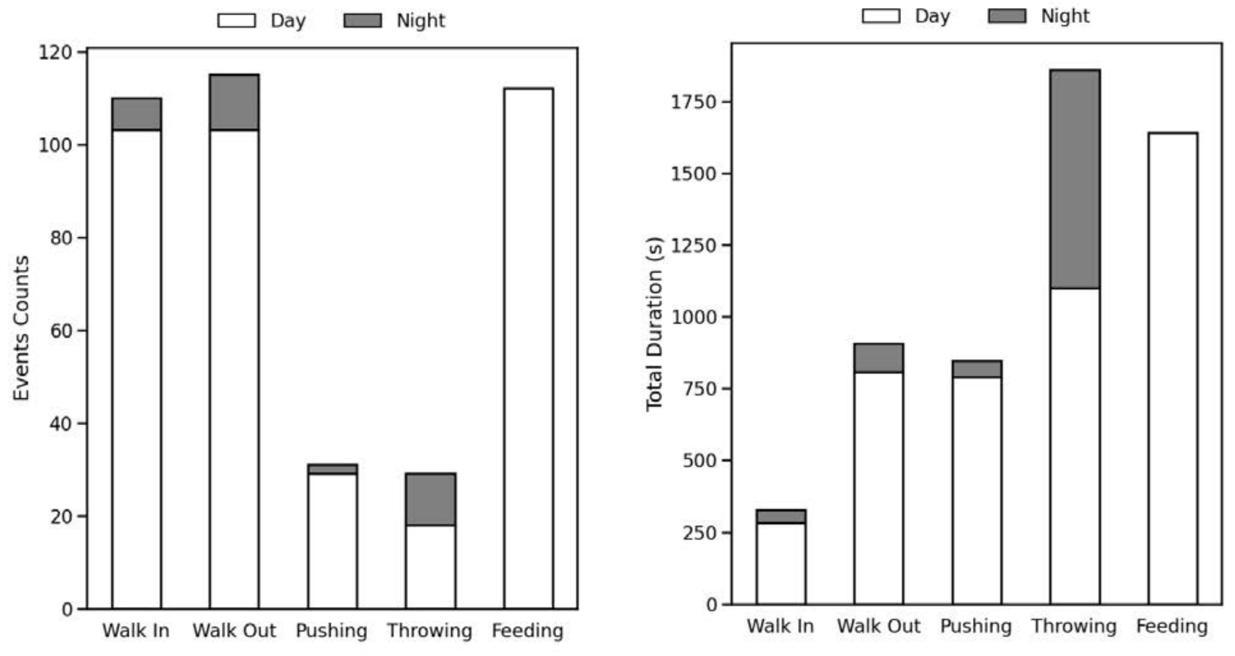


Figure 3.4: Statistics of behavioral categories for the nest of the Kagu dataset. Information is subcategorized by day and night and presented for event counts *(left)* and event durations *(right)*

stationary in the nest or outside of the image frame. The sparsity of events simulates real-life conditions; online algorithms should be robust against the resulting challenges - unbalanced dataset, catastrophic forgetting, etc.

3.3.3 Annotation Protocol

We have manually annotated the dataset with temporal events, time-of-day/lighting conditions, and spatial bounding boxes without relying on any object detection/tracking algorithms. The temporal annotations were initially created by experts who study the behavior of the Kagu bird and later refined to improve the precision of the temporal boundaries. Additional labels, such as lighting conditions, were added during the refinement process. The spatial bounding box annotations of 23M frames were created manually using professional video editing software (Davinci Resolve). We attempted to use available data annotation software tools, but they did not work for the scale of our video (10 days of continuous monitoring). We resorted to video editing software, which

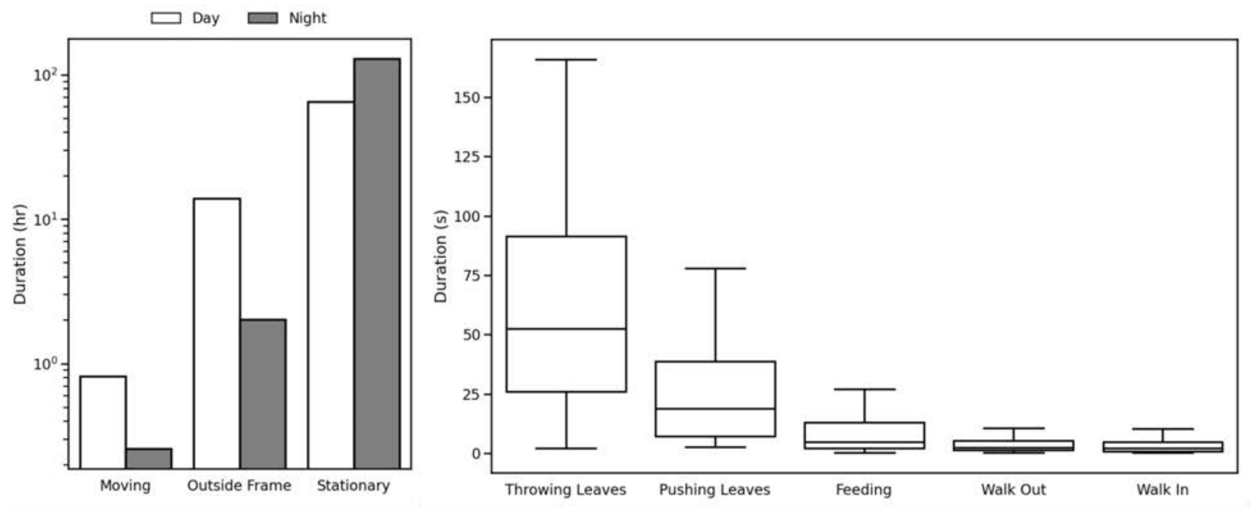


Figure 3.5: Event statistics for the nest of the Kagu dataset. *(Left)*: Total durations of bird states plotted in log scale and categorized by Day and Night conditions. *(Right)*: Box-and-Whisker-Plots for the duration of behavioral categories.

helped us annotate and export bounding box masks as videos. The masks were then post-processed to convert annotations from binary mask frames to bounding box coordinates for storage. It is worth noting that the video editing software allowed us to linearly interpolate between keyframes of the bounding boxes annotations, which helped save time and effort when the bird's motion is linear. Both temporal and spatial annotations were verified by two volunteer graduate students. The process of creating spatial and temporal annotations took approximately two months.

3.3.4 Validation and Test Splits

We split the full 10-day dataset into a validation and test set. The validation set is chosen as the first two days of the dataset (20% of the total frames), while the remaining eight days are used to create the test set. The validation set can be used to tune hyperparameters, such as segmentation thresholds, for adjusting segmentation granularity. The tuned parameters are then used to evaluate the test set. As shown in Figure 3.6, less than 10% of the total number of events occur in the first two days. Additionally, the feeding event occurs only after the chick has hatched on day 8, which also affects the frequency of walking in and out of the nest. The significant changes in event statistics

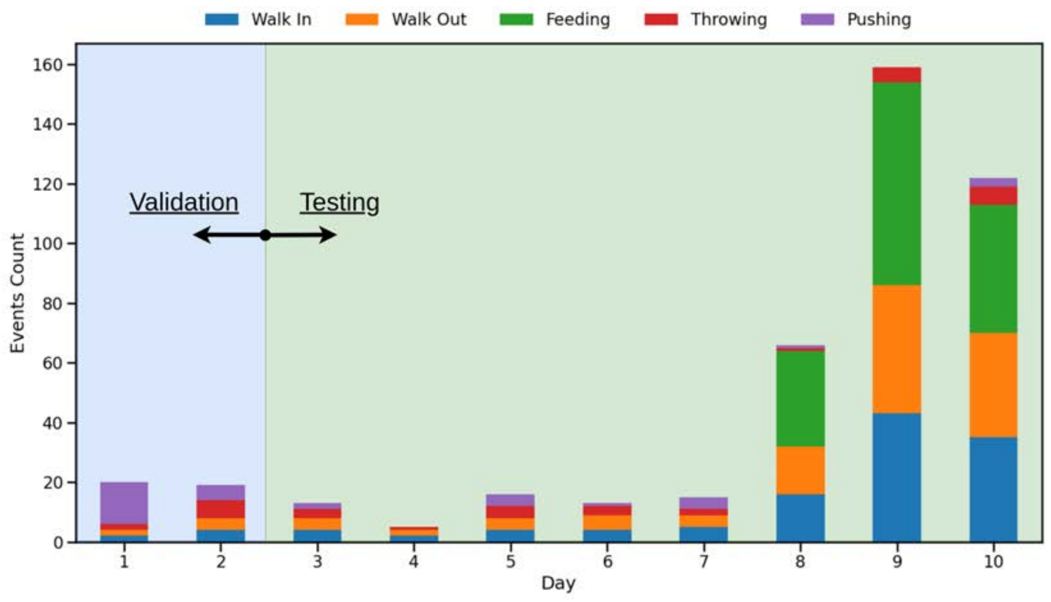


Figure 3.6: Training-validation splits for the nest of the Kagu dataset. Stack plot of event counts for each day of the proposed 10-day dataset categorized by the event type. The validation split is chosen as the first two days.

(counts and durations) between the validation and test splits provide a challenging setup for learning algorithms.

3.4 Experimental Evaluation

In this section, we present the results of our experiments for our approach defined in Section 3.2. We divide the results section into temporal segmentation and spatial segmentation. In both sections, we explain the evaluation metrics used to quantify the performance, discuss model variations, and conclude by presenting quantitative and qualitative results on the Kagu monitoring dataset. We also provide comparisons with state-of-the-art approaches on other datasets in different domains in Section 3.4.3

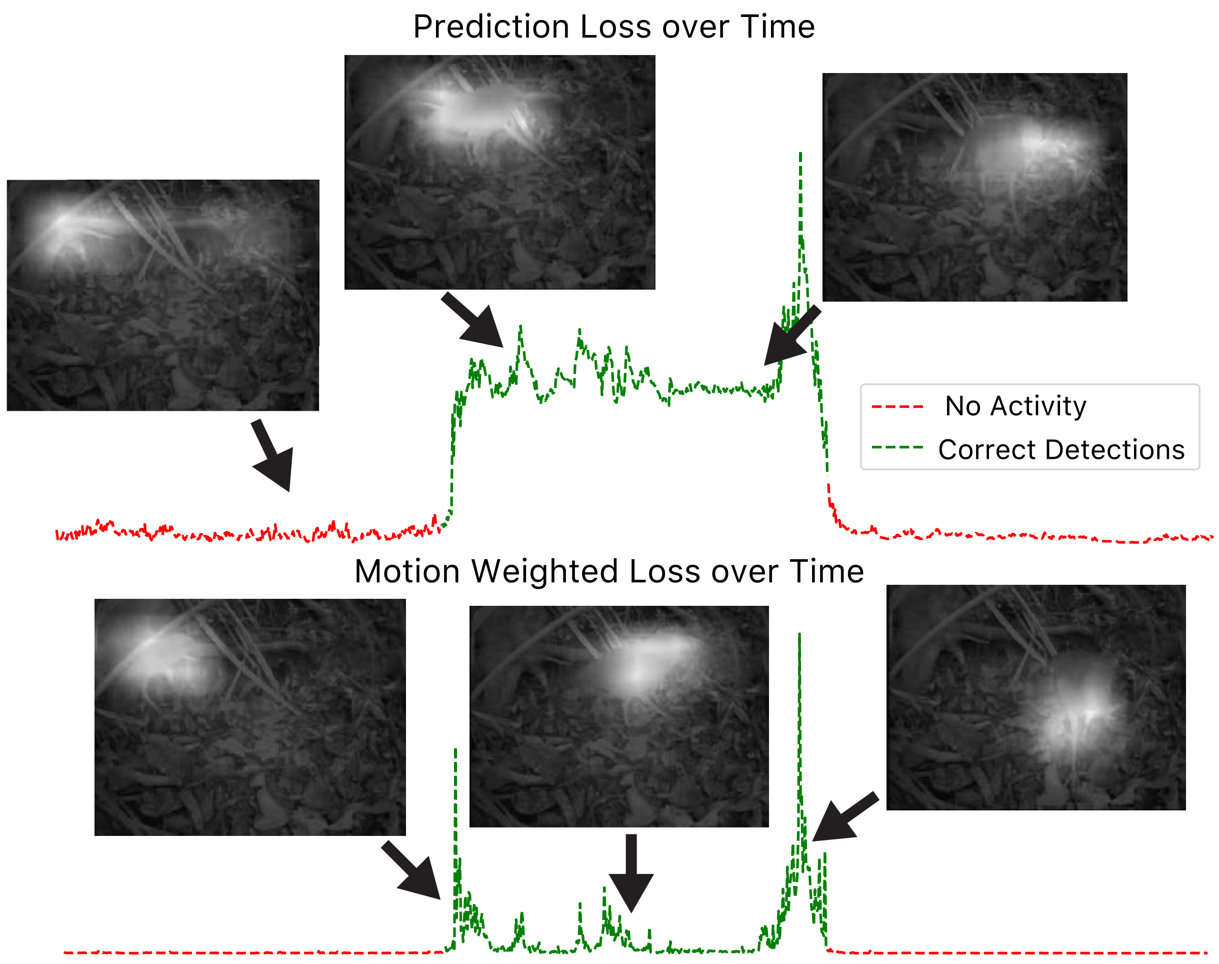


Figure 3.7: Illustration of prediction and motion-weighted loss. Plots present both kinds of errors before, during, and after an activity: *(top):* feature prediction loss over the frames, *(bottom):* motion weighted feature prediction loss over the frames. Some selected frames are shown and overlaid with corresponding attention maps (after bilinear interpolation resizing).

3.4.1 Temporal Segmentation

Temporal segmentation is the task of segmenting events in untrimmed videos. The goal is to detect and localize events such that important activities (e.g., bird walking out) are accurately trimmed to their start and end boundaries. We present two different evaluation approaches to temporal event segmentation; activity detection and boundary detection. Activity detection targets the detection of the full event and distinguishes between the start and end boundaries. Boundary detection only considers the detection of event boundaries as a separation between events. As discussed in Section 3.4.1.1, the evaluation of activity detection is based on one-to-one mapping, with IoU maximization objective, of the predicted events and the ground truth events, whereas the boundary detection is evaluated based on the distance between the predicted boundaries and the ground truth boundaries. Figure 3.7 shows an example of temporal segmentation for an event; the event is detected by thresholding the error signal.

3.4.1.1 Evaluation Metrics

Our evaluation metrics are tied directly to the quality of ethogramming. Accurately localizing the events and their boundaries can lead to higher ethogramming quality since the behavior attributes can only be found within the boundaries of the event. It is important to increase the recall rate (reduce missed detections) and decrease false positives to ensure all important events are detected and analyzed while erroneous and irrelevant activities are disregarded. We use three different evaluation metrics to assess the temporal segmentation performance of our approach. Conceptually, frame-level evaluation captures the ability of a model to correctly classify whether a frame belongs to an event. Unlike frame-level, activity-level evaluation focuses on the ability to capture the existence and location of full events within an untrimmed video. The boundary distance metric measures the temporal precision of the predicted boundaries to the ground truth boundaries. We discuss the evaluation metrics in more detail in the following paragraphs.

3.4.1.1.1 Frame Level

The frame-level evaluation of temporal segmentation measures the ability of an algorithm to classify whether each frame belongs to an event. The recall value in frame-level ROC is calculated as the ratio of true positive frames (event present) to the number of positive frames in the annotations dataset, while the false positive rate is expressed as the ratio of the false positive frames to the total number of negative frames (event not present) in the annotation dataset. Frame window size ( ) is defined as the maximum joining window size between events; a high value can cause separate detected events to merge, which decreases the overall performance. The threshold value ( ) is varied to obtain a single ROC line while varying the frame window size ( ) results in a different ROC line.

3.4.1.1.2 Activity Level

The Activity level evaluation measures the ability of an algorithm to detect events. We utilize the Hungarian matching (Munkres assignment) algorithm to achieve one-to-one mapping between the ground truth labeled events and the detected events. The recall is defined as the ratio of the number of correctly detected events (overlapping frames) to the total number of ground truth events. For the activity level ROC chart, the recall values are plotted against the false positive rate per minute, defined as the ratio of the total number of false-positive detected events to the total duration of the dataset in minutes. The false-positive rate per minute evaluation metric is also used in the ActEV TRECVID challenge [3]. Frame window size value ( ) is varied to obtain a single ROC line while varying the threshold value ( ) results in a different ROC line.

3.4.1.1.3 Boundary Distance

In addition to activity detection evaluation, we also evaluate the ability of our approach to detect generic boundaries of events. We quantify the performance of boundary detection by applying one-to-one (Hungarian) matching between the detected boundaries and the annotated boundaries. The distances, in seconds, between the resulting matches are calculated and thresholded. If the distance for a boundary is lower than the specified threshold value, it is considered a true positive (TP). As the threshold increases, more TPs occur, resulting in higher recall and precision values and vice versa. By using Hungarian matching, we make sure that the algorithm is penalized for duplicated detections by reducing its precision value.

3.4.1.2 Activity Detection Results

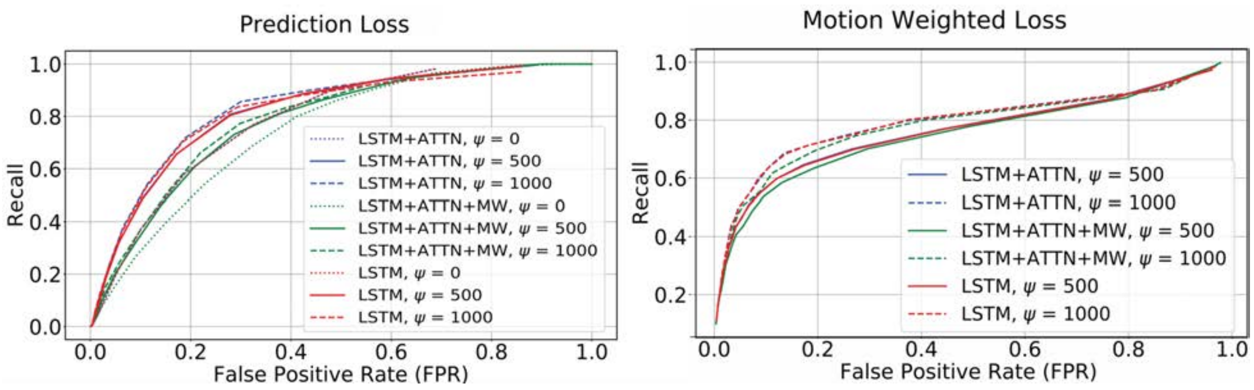


Figure 3.8: Frame-level event segmentation ROC plots. Results are shown for activities detected based on simple thresholding of the prediction and motion-weighted loss signals. Plots are shown for different ablation studies.

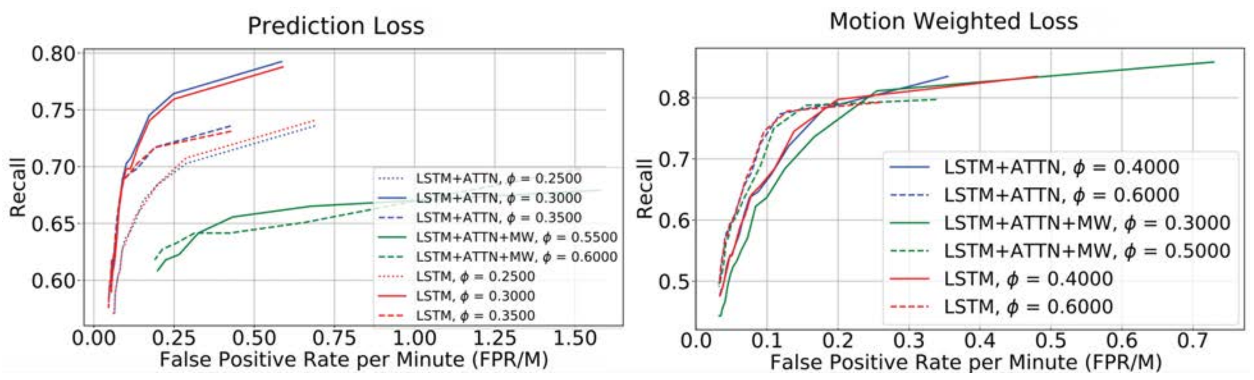


Figure 3.9: Activity-level event segmentation ROC plots for simple thresholding. Results are shown for simple thresholding of the prediction and motion-weighted loss signals. Plots are shown for different ablation studies.

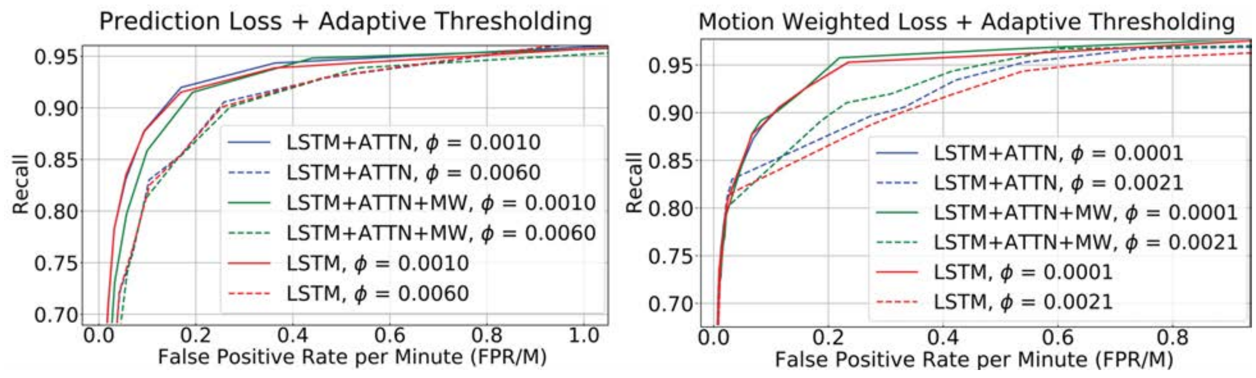


Figure 3.10: Activity-level event segmentation ROC plots for adaptive thresholding. Results are shown for adaptive thresholding of the prediction and motion-weighted loss signals. Plots are shown for different ablation studies.

3.4.1.2.1 Model Variations

Different variations of our framework (Section 3.2) have been evaluated to quantify the effects of individual components on the overall performance. In our experiments, we tested the base model, which trains the perceptual prediction framework - including the attention unit - using the prediction loss function for backpropagation of the error signal. We refer to the base model as *LSTM+ATTN*. We also experimented with the effect of removing the attention unit, from the model architecture, on the overall segmentation performance; results of this variation are reported under the model name *LSTM*. Further testing includes using the motion-weighted loss for backpropagation of the error signal. We refer to the motion-weighted model as *LSTM+ATTN+MW*. Each of the models has been tested extensively; frame-level segmentation results are shown in Figure 3.8, while activity-level results are provided in Figure 3.9 and Figure 3.10. Ablations are evaluated on the full ten-day dataset.

Comparing the results shown in Figure 3.9 & Figure 3.10 indicates a significant increase in overall performance when using an adaptive threshold for loss signal gating. The efficacy of adaptive thresholding is evident when applied to activity-level event segmentation. Comparing the results (LSTM & LSTM+ATTN) shows that the model can effectively generate attention maps for spatial segmentation without impacting the temporal segmentation performance.

3.4.1.2.2 Results Discussion

We tested three different models, *LSTM*, *LSTM+ATTN*, and *LSTM+ATTN+MW*, for frame level and activity level event segmentation. Simple and adaptive gating functions (Section 3.2.6.2) were applied to prediction and motion-weighted loss signals (Section 3.2.5) for frame-level and activity-level experiments. For each model, we vary parameters such as the threshold value and the frame window size to achieve the ROC charts presented in Figure 3.8, Figure 3.9 & Figure 3.10.

It is to be noted that thresholding a loss signal does not necessarily imply that the model was trained to minimize this particular signal. In other words, loss functions used for backpropagating the error to the models' learnable parameters are identified only in the model name; however, thresholding experiments have been conducted on different types of loss signals, regardless of the backpropagating loss function used for training.

The best-performing model, for frame level segmentation, (*LSTM + ATTN,*) is capable of achieving 40%, 60%, 70% frame recall value at 5%, 10%, 20% frame false positive rate respectively. Activity level segmentation can recall 80%, 90%, 95% of the activities at 0.02, 0.1, 0.2 activity false positive rate per minute, respectively, for the model (*LSTM + ATTN + MW,*) as presented in Figure 3.10. A 0.02 false positive activity rate per minute can also be interpreted as one false activity detection every 50 minutes of training (for detecting 80% of the ground truth activities). While the attention mechanism does not offer a clear improvement in temporal segmentation, it allows us to localize the object in every frame. Spatial segmentation was not possible without the attention mechanism added to the LSTM baseline. Other modifications, such as motion-weighted loss and adaptive thresholding, contribute more to a significant improvement in temporal segmentation performance.

We further inspect the performance of the best-performing activity-level model by presenting the IoU between the detected events and the ground truth events - categorized by event type in Figure 3.11. Based on the results, it can be seen that there is a correlation between the event

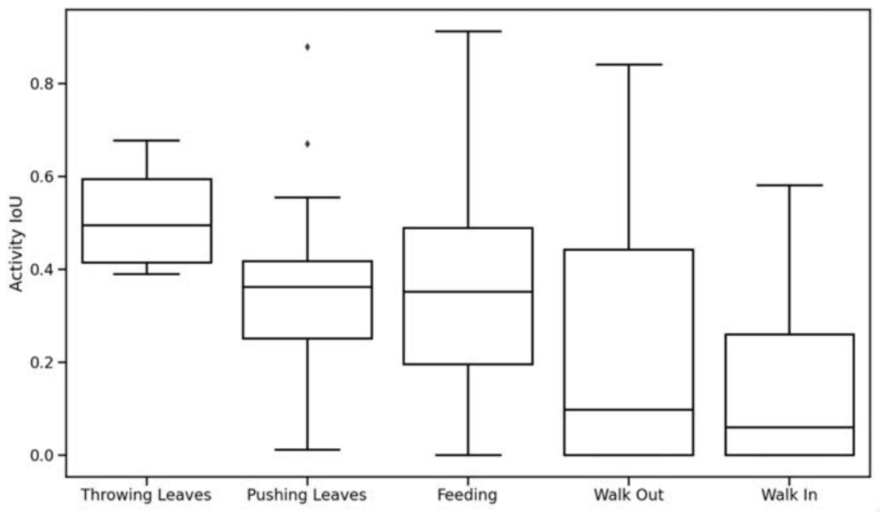


Figure 3.11: Activity IoU for the best performing activity-level temporal segmentation model. Results are categorized by event type

durations and the activity overlap. Long events (e.g., Throwing) result in higher IoU, with the detected event, than short events (e.g., Walk-In).

3.4.1.3 Boundary Detection Results

Even though our approach targets the detection of full activities, we can evaluate the performance of our approach on the boundary detection task by converting each detected activity into two boundaries. Using boundaries, we can quantify the performance of our approach by calculating the distance of each detected boundary to the ground truth boundary. We evaluate the performance at varying distance thresholds and report the results (with baseline comparisons) in Table 3.1. All segmentation thresholds are tuned on the validation set and used for evaluation on the test set.

Table 3.1: Temporal segmentation results on Kagu dataset. Results are shown for unsupervised event boundary detection methods at different distance thresholds. FT denotes fine-tuned on the Kagu dataset. SceneDetect's threshold parameter is tuned on the full dataset, otherwise the performance is zero.

Metric	Approach	Distance threshold (seconds)									
		10	50	100	150	200	250	300	350	400	450
Precision	SceneDetect [219]	0.095	0.095	0.143	0.143	0.143	0.143	0.238	0.238	0.238	0.238
	Uniform	0.005	0.019	0.030	0.044	0.055	0.069	0.078	0.088	0.098	0.103
	PA-DPC [225]	0.005	0.016	0.026	0.035	0.049	0.058	0.063	0.067	0.075	0.087
	PA-DPC-FT [225]	0.002	0.009	0.012	0.012	0.012	0.014	0.014	0.014	0.014	0.014
	LSTM+AL [1]	0.030	0.042	0.052	0.059	0.062	0.067	0.069	0.073	0.079	0.088
	KNN	0.120	0.201	0.233	0.254	0.265	0.273	0.281	0.283	0.289	0.294
	Ours	**0.167**	**0.304**	**0.361**	**0.369**	**0.386**	**0.411**	**0.416**	**0.439**	**0.449**	**0.460**
Recall	SceneDetect [219]	0.003	0.003	0.004	0.004	0.004	0.004	0.007	0.007	0.007	0.007
	Uniform	0.013	0.054	0.084	0.121	0.152	0.192	0.218	0.244	0.271	0.287
	PA-DPC [225]	0.006	0.018	0.029	0.040	0.056	0.066	0.072	0.077	0.075	0.087
	PA-DPC-FT [225]	0.001	0.006	0.009	0.009	0.009	0.010	0.010	0.010	0.010	0.010
	LSTM+AL [1]	0.042	0.057	0.072	0.081	0.086	0.092	0.095	0.100	0.109	0.121
	KNN	0.055	0.092	0.106	0.116	0.121	0.125	0.128	0.130	0.132	0.134
	Ours	**0.108**	**0.196**	**0.232**	**0.237**	**0.248**	**0.264**	**0.268**	**0.282**	**0.289**	**0.296**
F1	SceneDetect [219]	0.005	0.005	0.008	0.008	0.008	0.008	0.013	0.013	0.013	0.013
	Uniform	0.007	0.028	0.045	0.064	0.080	0.102	0.115	0.129	0.144	0.152
	PA-DPC [225]	0.006	0.017	0.027	0.038	0.053	0.062	0.067	0.072	0.080	0.092
	PA-DPC-FT [225]	0.001	0.007	0.010	0.010	0.010	0.011	0.011	0.011	0.011	0.011
	LSTM+AL [1]	0.035	0.048	0.061	0.068	0.072	0.077	0.080	0.084	0.092	0.102
	KNN	0.076	0.126	0.146	0.159	0.166	0.171	0.176	0.178	0.181	0.185
	Ours	**0.131**	**0.238**	**0.283**	**0.289**	**0.302**	**0.321**	**0.326**	**0.344**	**0.351**	**0.360**

3.4.1.3.1 Baselines

Scene detect [219] is a popular online tool for shot boundary detection. Scenedetect analyzes the video for changes in average frame brightness/intensity and applies a threshold to detect boundaries. Lower threshold values result in more boundaries, which increases recall and decreases precision.

Uniform is a simple baseline created by using equally separated boundaries as predictions. Varying the frequency of predicted boundaries results in moving to different positions on the

Precision-Recall line. We use the frequency that results in the best F1 score on the validation set for evaluation on the test set.

PredictAbility (PA-DPC) [225] is a baseline created by using the ability of the model to predict future frames. PA uses the Dense Predictive Coding (DPC) model [80] as a backbone and provides state-of-the-art results for self-supervised generic event boundary detection. This baseline calculates the prediction loss for a given frame by computing the difference between past and future context embeddings. The context is defined as five frames before and after the target frame. DPC uses a 3D ResNet architecture as the backbone, which is used to calculate the context embeddings. The PA model proposes event boundaries at the local maxima of the error signal. The Laplacian of Gaussian (LoG) is applied to the 1D temporal error signal. Peaks are detected at the negative to positive zero crossings of the derivative of the LoG signal. In our experiments, we found that thresholding the error signal and applying a 1D non-max suppression operation provides better results (reported) on our proposed dataset. *PA-DPC-FT* [225] is a finetuned version of PA-DPC, where the DPC model [80] is finetuned on 1000 short videos (seconds each) extracted from the Kagu dataset.

LSTM+AL [1] is a predictive learning baseline that tackles the problem of event boundary detection by learning a future prediction function on high-level frame features. LSTM+AL detects event boundaries as peaks in the prediction loss signal. Similar to our proposed approach, LSTM+AL is heavily inspired by the "Event Segmentation Theory" introduced by Zacks *et al.* [281]. However, in addition to the added attention module, there exist several important distinctions between the two approaches. We summarize the main differences in the following points:

- Our approach processes a sliding window of 16 frames (8 frames predicting 8 frames) with a stride of 8 frames, allowing the representation to be more robust by predicting farther into the future. LSTM+AL [1] uses a single frame to predict a single future frame, limiting the representation and prediction capacity of the network.
- Unlike LSTM+AL [1], which only uses the prediction loss peaks to detect boundaries, our approach uses motion-weighted loss to learn events from motion cues

- Our approach freezes the encoder weights and only allows the prediction network weights to be modified. According to recent findings [253], “an extreme form of knowledge preservation—freezing the classifier-initialized backbone— consistently improves many different detection models, and leads to considerable resource savings.” We also find that freezing the encoder network leads to better performance and prevents model collapse.
- Our approach does not use the adaptive learning trick introduced in LSTM+AL [1] to keep the prediction loss in a specific range. Instead, we use adaptive thresholding to detect events.
- LSTM+AL [1] focuses on detecting boundaries by detecting peaks in the prediction loss, whereas our approach detects full events by thresholding the motion-weighted prediction loss signal.

KNN is an unsupervised online learning baseline that aims to adapt to changes in the data distributions. We design an unsupervised KNN algorithm where every frame in the dataset is compared to the five nearest neighbors in a moving window of the past four seconds (100 frames). The average (over the nearest five neighbors) cosine similarity score is stored for every frame. We threshold the similarity scores and apply 1D non-max suppression to remove duplicate boundary detections in a window of 1 second.

3.4.1.3.2 Results Discussion

We report the evaluation results of boundary detection in Table 3.1. The F1 score metric is used to compare the overall performance of our method to other baselines. We additionally provide precision and recall values to analyze and compare the behavior of each detector with varying distance thresholds. Based on the F1 results, it can be seen that we at least double the performance of all the other methods at all thresholds.

SceneDetect [219] is designed to detect shot boundaries based on illumination changes; therefore, it fails when applied to a dataset of continuous monitoring. It results in higher precision values than the other baselines because there is a higher chance of what is perceived as a shot boundary to be also classified as an activity boundary. However, the precision values are matched with a significantly lower recall rate, demonstrating its inability to detect most of the boundaries.

Uniform results in high recall rates and low precision values, which demonstrate the ability to detect a high percentage of the ground truth boundaries, but only when sampling predicted boundaries at high frequency (shown by the low precision values). This baseline is implemented to demonstrate the irregular distribution of activities over time, causing simple, equally spaced boundary predictions to fail - especially at lower distance thresholds.

PA-DPC uses a pretrained backbone to detect boundaries based on the difference between context embeddings. Surprisingly, the PA baseline fails to detect boundaries and results in average performance with respect to the other baselines. The F1 scores are slightly lower than the Uniform baseline. The low performance of this baseline can be attributed to the difference between the pretraining domain (kinetics 400) and the animal monitoring domain (Kagu dataset). A substantial difference between our approach and PA-DPC lies in the continuous predictive learning and adaptation on the target dataset, which cannot be done with PA-DPC by merely comparing the context embeddings. The results highlight the importance of the proposed online stream learning approach. Other results comparing our method to PA-DPC on in-domain datasets (e.g., GEBD-Kinetics) are discussed in Section 3.4.3.

PA-DPC-FT attempts to bridge the gap in domains between the Kinetics dataset and the Kagu dataset by finetuning the DPC architecture on a subset of our proposed Kagu dataset. Results show that training on the Kagu dataset did not increase performance; our approach still outperforms it by a significant margin. It is important to note that contrastive approaches such as DPC rely heavily on the diversity of labels and features within and across video samples, making the Kagu videos a challenging dataset for learning useful representation in a contrastive manner.

LSTM-AL shows a slight improvement in performance over PA-DPC-FT; however, the performance is low compared to our approach. The low performance can be attributed to the differences discussed earlier in the baselines section.

KNN reports approximately double the performance of LSTM-AL. The results highlight the importance of online training and adaptation for processing streaming data. However, the KNN approach is still significantly outperformed by our predictive framework indicating that the similarity between frame features does not contain enough information to detect event boundaries.

3.4.2 Spatial Segmentation

Spatial segmentation is the task of detecting and localizing the object of interest in each frame; the foreground (i.e., bird) is segmented from the background (i.e., nest). We use the Bhadanau attention map (Section 3.2.3) to spatially localize the bird.

3.4.2.1 Evaluation Metrics

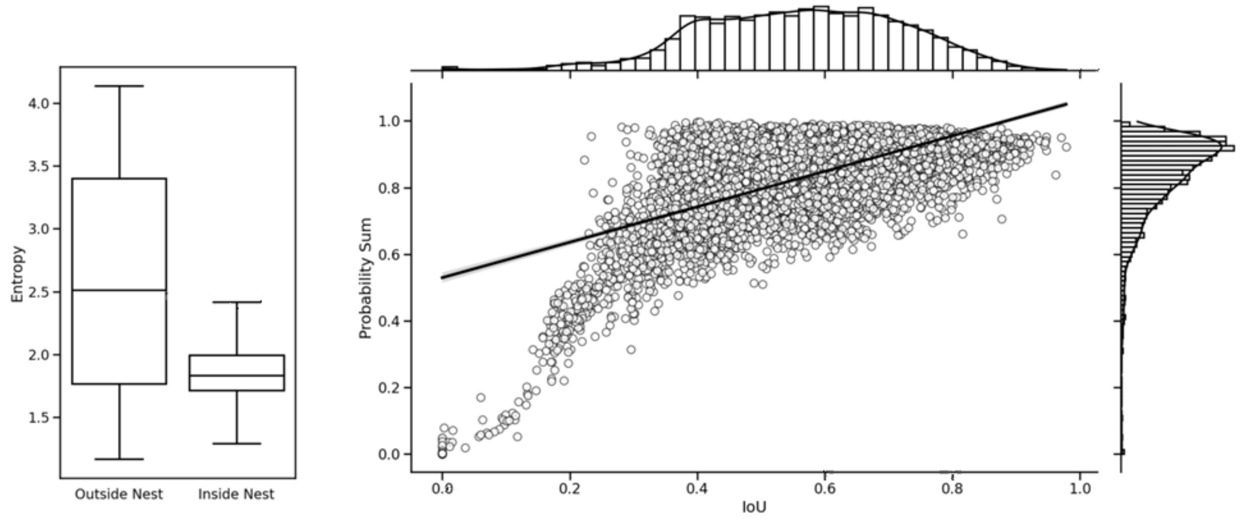


Figure 3.12: Spatial segmentation evaluation metrics. *(Left):* Entropy distributions for when the bird is inside and outside of the nest. *(Right):* Joint distributions of Probability Sum and IoU metrics.

Successful spatial segmentation of the events not only depends on the ability of the algorithm to spatially locate the event-causing objects but also detect their presence inside the frame. To generate high quality ethograms, the algorithm must be evaluated on event detection and localization tasks. The bird's location, velocity, and motion patterns in the nest can be correlated with specific behavior. Therefore, the accurate localization of events is essential for automating ethograms.

3.4.2.1.1 Evaluating Attention Maps

Although the ground truth annotations are given in a bounding box format, most of our baselines output a heat map - or a mask - indicating the predicted location of the bird in each frame. The *probability sum* is a metric designed to evaluate the success of the mask in predicting the location of the object. First, the predicted mask is converted into a probability distribution using the Softmax function. The probability sum is then calculated by adding all the probability values located inside the ground truth bounding box. Applying the Softmax function provides a means for penalizing the mask for predicting high values outside of the ground truth bounding box. A high probability sum value indicates a high percentage of mask values inside the bounding box; however, it does not guarantee a high overlap.

3.4.2.1.2 Generating Bounding Boxes

We generate bounding boxes by thresholding the predicted mask to detect the biggest contour for which we find the bounding rectangle. The *IoU metric* measures the overlap between the two bounding boxes normalized by the union of their areas. We use the Jaccard Index (IoU) to quantify the overlap of the predicted bounding box with the ground truth bounding box. As can be seen in Figure 3.12 (right), there is a correlation between the Probability Sum values and the calculated IoU values; however, we also expect to see some values with high Probability Sum and relatively lower IoU representing detections of parts of the bird (e.g., bill) inside a larger ground truth bounding box.

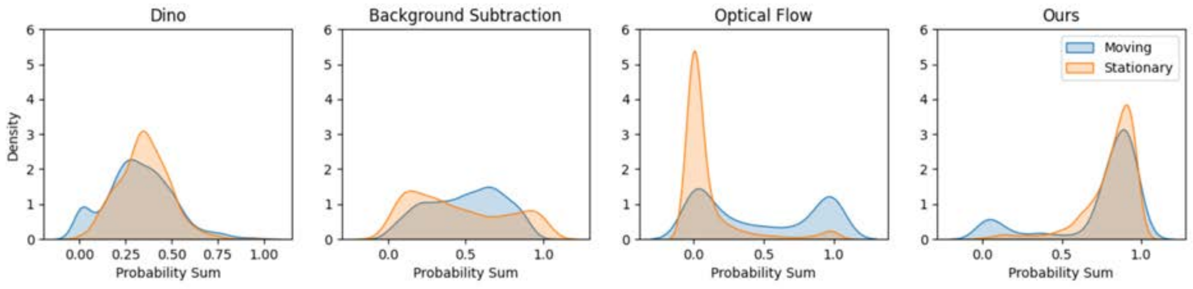


Figure 3.13: The density function of the probability sum for several spatial segmentation approaches. Each plot is further categorized by the motion state of the bird.

3.4.2.1.3 Bird Detection

Both, Probability Sum and IoU, metrics merely quantify the accuracy of the true positive detections (i.e., when the bird is in the frame); however, we can also quantify the ability of the approaches to detect when the bird is outside of the frame. The average precision (AP) uses the detector's confidence as an indicator of whether an object is present. In our approach, we use the entropy of the probability distribution of the attention map to indicate the presence of the bird. As shown in Figure 3.12 (left), when the bird is in the nest, entropy values are low, indicating that the model is more confident in its predictions; however, when the bird is not in the nest, attention becomes more spatially distributed resulting in higher entropy values. A Precision-Recall chart can be generated at different Probability Sum/IoU thresholds by calculating the true positives, false positives, and false negatives at each entropy threshold. The AP is calculated by taking the average of 11-recall points [54] on the precision-recall chart. We report the AP values at different Probability Sum/IoU thresholds in Table 3.3. The entropy $\mathcal{H}$ is calculated using the Equation (3.10).

$$\mathcal{H} \tag{3.10}$$

Table 3.2: Probability sum evaluation for different approaches at various illumination conditions.

Approach	Probability Sum at different conditions					
	Day	Night	Shadows	Sunrise	Sunset	Avg
Optical Flow	0.147	0.047	0.190	0.181	0.153	0.144
Background Subtraction [287]	0.339	0.593	0.163	0.330	0.432	0.371
Pretrained Inception [240]	0.131	0.139	0.129	0.131	0.140	0.134
DINO [29]	0.361	0.354	0.273	0.411	0.472	0.374
Ours	**0.858**	**0.779**	**0.784**	**0.708**	**0.864**	**0.799**

3.4.2.2 Spatial Localization Results

3.4.2.2.1 Baselines

We evaluated the performance of five different baselines on the spatial segmentation task. Some of the baselines are traditional computer vision approaches (e.g., Optical Flow) that do not require any training data, while others (i.e., DINO) are trained on ImageNet in a self-supervised manner.

Optical Flow outputs the magnitude and direction of the pixels' motion patterns. We only use the magnitude of the distance traveled by each pixel to predict the bird's location. In addition to using the magnitude for localization, we also use the maximum magnitude as a confidence score to detect if the bird is not inside the nest, which is necessary for Average Precision computation.

Background Subtraction [287] shares the same goal (i.e., motion detection) with Optical Flow, but instead of comparing two consecutive frames, it builds a background model of the scene over time and compares it to the current frame. The background model is constantly updated with new images to ensure continuous adaptation to the changing background features. The resulting foreground maps consisted of many disconnected contours. We applied ellipse kernel convolutions to dilate the foreground map and join the disconnected - but close - contours before finding the largest contour for bounding box calculation.

Fixed Boundingbox is based on the premise that the bird stays most of the time sitting in the nest. We design a baseline that assumes a stationary bounding box placed at the center of the nest at all times. This simple baseline uses priors about the behavior of the bird in the dataset and cannot be blindly applied to other datasets.

Pretrained Inception is the same backbone that we use for our approach (Section 3.2.2). The Inception-V3 [240] backbone is pretrained on ImageNet to extract useful features. We compare the feature grids extracted from every two consecutive images to calculate a feature difference grid. The difference in pretrained features should, in theory, inhibit background features (e.g., shadows) and highlight important bird features. This baseline is designed to test whether a representation of the bird is embedded in the pretrained weights of the backbone or is trained in an online learning manner in the LSTM and attention weights of our approach.

DINO-ImageNet [29] is a recent self-supervised vision transformer model (ViT [49]), designed specifically for representation learning. The DINO model is trained on ImageNet from scratch, and the output representations achieve competitive classification results on ImageNet. DINO also utilizes the ViT model to visualize the attention as a heat map. We use the attention maps from DINO as a baseline for spatial segmentation. *DINO-Landmarks v2* uses weights from the ViT model pretrained on the Google Landmarks v2 dataset [264]. *DINO-FT-* N *E* refers to a DINO model pretrained on ImageNet and finetuned on 1000 randomly sampled images from the Kagu dataset for N epochs.

iBOT [285] is another self-supervised approach to representation learning in images. It uses the masked language model pretraining paradigm to perform masked prediction with an online tokenizer. Similar to DINO, we extract the output attention from the last layer of the ViT architecture [49] and use it as a spatial segmentation baseline on the Kagu dataset.

3.4.2.2.2 Results Discussion

We evaluate our results against the baselines and show that our approach outperforms all the baselines. Table 3.2 shows the Probability Sum performances at different conditions (e.g., day, night). Table 3.3 presents the Average Precision performances at different probability sum and IoU thresholds. We also provide additional results to further categorize the performance by the bird's state and the type of activity in Figure 3.13 and Figure 3.14, respectively.

Table 3.2 only considers the frames where the bird is present inside the frame and presents the average Probability Sum for each of the baselines. Based on the results, it can be seen that the shadows pose the biggest challenge to many of the baselines, except for Optical Flow, which is more affected by the low illumination - and less movement - conditions at night. The sharp shadows present in the Kagu dataset can become problematic to object detectors and traditional motion trackers because the shadows can cause severe changes to the look and texture of both the object and the background. A robust and adaptable representation of the bird is a must to overcome these challenges; our approach learns a robust representation by attending to the object and its motion cues and adapting to the changing background through continuous online training. On average, our approach doubles the performance of SOTA and traditional methods while providing stable and robust localization performance across all the different illumination and shadow conditions.

The Average Precision (AP) results reported in Table 3.3 consider not only the localization performance but also evaluate the detection performance and the ability of the different models to predict if the bird is not inside the nest. For a detection to be counted as true positive, both the detection score (Entropy or Magnitude) and localization score (Prob. Sum. or IoU) must exceed the specified threshold values. Varying the detection threshold results in a Precision-Recall chart, which is summarized by the AP value; however, each column reports the results at different localization thresholds. Our approach significantly outperforms the other baselines at all localization thresholds. Both Optical Flow and Background Subtraction result in a low performance, which can be attributed to the extended periods of time when the bird is stationary in the nest. Background Subtraction [287]

Table 3.3: Comparison of Average Precision (AP) performance for spatial segmentation approaches. Results shown at different localization metrics' thresholds for traditional and self-supervised spatial segmentation approaches. FT denotes fine-tuned on the Kagu dataset.

Threshold	Approach	Threshold Value								
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Probability Sum	Optical Flow	0.228	0.129	0.089	0.069	0.056	0.047	0.040	0.033	0.025
	Background Subtraction [287]	0.750	0.611	0.508	0.425	0.353	0.285	0.255	0.172	0.122
	Pretrained Inception [240]	0.753	0.109	0.092	0.091	0.091	0.000	0.000	0.000	0.000
	DINO-ImageNet [29]	0.855	0.872	0.590	0.309	0.102	0.033	0.010	0.003	0.002
	DINO-Landmarks v2 [29]	0.844	0.766	0.563	0.200	0.020	0.000	0.000	0.000	0.000
	DINO-FT-20E [29]	0.064	0.002	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	DINO-FT-100E [29]	0.038	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	iBOT [285]	0.858	0.842	0.797	0.684	0.435	0.178	0.045	0.009	0.002
	Ours	**0.876**	**0.876**	**0.875**	**0.874**	**0.869**	**0.860**	**0.821**	**0.716**	**0.392**
IoU	Optical Flow	0.121	0.085	0.054	0.033	0.018	0.009	0.003	0.001	0.000
	Background Subtraction [287]	0.389	0.198	0.132	0.093	0.065	0.046	0.030	0.013	0.002
	Fixed Boundingbox	0.838	0.835	0.826	0.610	0.281	0.098	0.023	0.001	0.000
	Pretrained Inception [240]	0.840	0.605	0.314	0.192	0.130	0.101	0.013	0.006	0.001
	DINO-ImageNet [29]	0.686	0.558	0.420	0.225	0.083	0.021	0.004	0.001	0.000
	iBOT [285]	0.787	0.753	0.666	0.467	0.224	0.078	0.013	0.001	0.000
	Ours	**0.876**	**0.874**	**0.855**	**0.758**	**0.569**	**0.376**	**0.188**	**0.053**	**0.004**

results show a big difference between Prob. Sum. and IoU, which can be interpreted as a very localized prediction (low entropy) on the part of the bird inside the ground truth bounding box. On the contrary, Pretrained Inception predicts high entropy masks, resulting in low Prob. Sum. scores and a large predicted bounding box. Outperforming the pretrained Inception weights highlights the efficacy of our approach in learning a robust representation of the bird, which is not embedded in the Inception pretrained weights. The fixed bounding box baseline is designed based on prior knowledge of the dataset and the bird's location; it performs better than the other baselines but is still outperformed by our approach. We noticed that the DINO baseline consistently attends to a tree branch in the background, which affects the qualitative results by consistently producing average localization scores. Results from DINO highlight the importance of continuous domain adaptation and online training. Using the Google Landmarks V2 [264] weights instead of ImageNet results in lower performance, which is likely due to the lower domain overlap of the Kagu dataset with the Landmarks dataset than with ImageNet. Finetuning the DINO architecture (pretrained on ImageNet) shows a significant decrease in performance, which could result from model collapse.

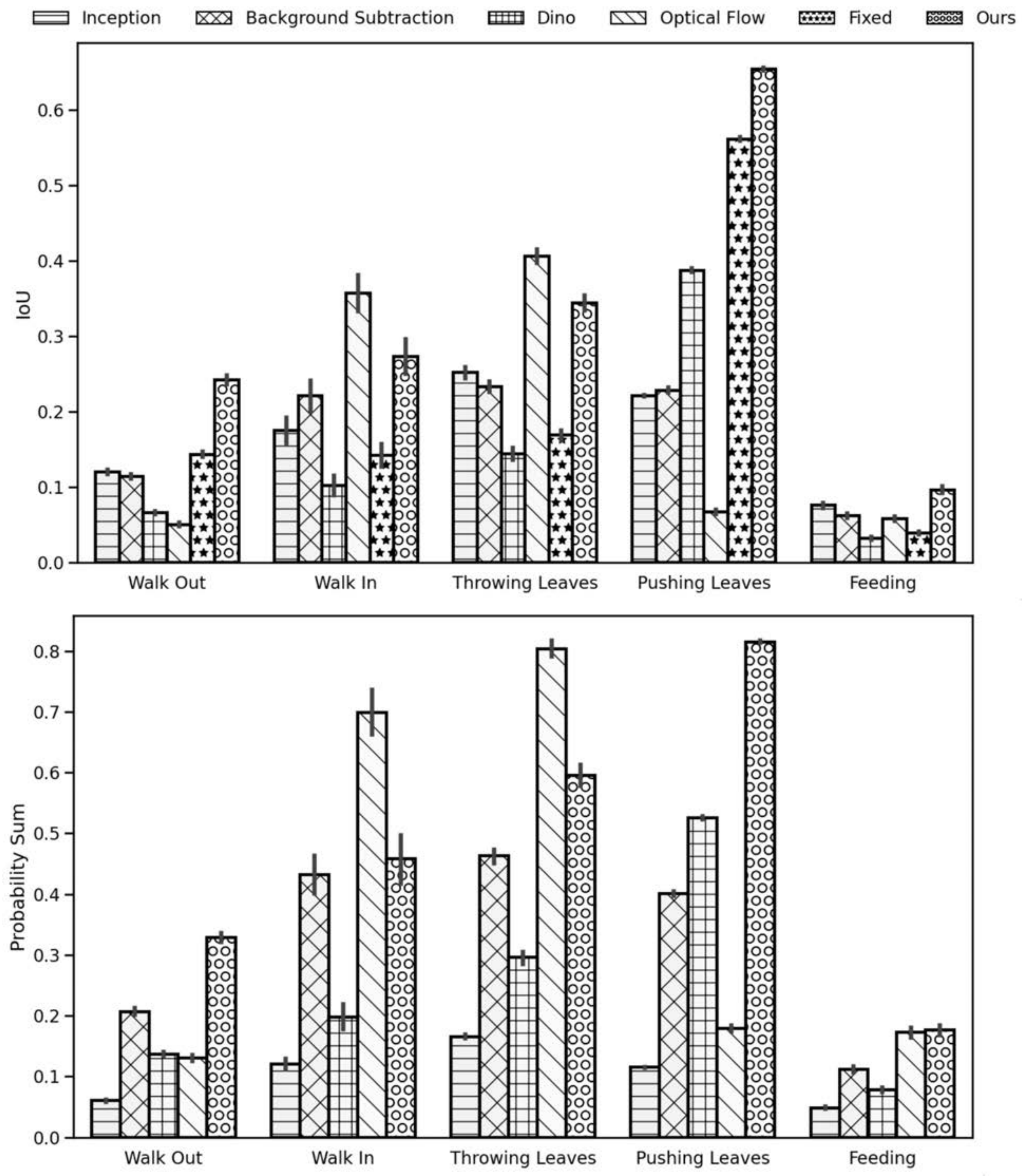


Figure 3.14: Spatial segmentation performance comparison for different methods. *(top)*: IoU metric. *(bottom)*: Probability Sum metric

DINO [29] relies on the similarity between crops of the same image and requires the main object to occupy most of the image (otherwise, the crop will not contain the object). Our dataset can prove to be challenging for crop-based contrastive approaches like BYOL [74] and DINO [29] because the object of interest (Kagu bird) occupies, on average, less than 10% of the image. iBOT [285] performs significantly better than DINO on our proposed Kagu dataset, yet it is still outperformed by our approach.

Figure 3.13 can provide better insights into the performance of each baseline and their differences. We show the density function for the Probability Sum scores over the entire dataset and further categorize the results by the motion state of the bird. It can be seen that Optical Flow performs much worse when the bird is stationary (shown as a high peak close to zero Prob. Sum score). Background Subtraction performs slightly better when the bird is moving, but the performance is almost uniformly distributed over the entire range of Prob. Sum. DINO shows both moving and stationary peaks at approximately 0.4; both distributions have average variances. Our approach shows two high peaks (low variance) close to 1.0 Prob.Sum score. Having similar moving and stationary distributions is a desirable behavior for object detectors because it shows the robustness of the detector to the location and motion state of the object of interest. Figure 3.15 provides qualitative examples to support the findings presented in Figure 3.13. We show that, despite our low resolution (8 8 grid) attention, our bird representation is more accurate and robust to different lighting and environmental conditions when compared to DINO and traditional approaches. Results show that DINO is always distracted by other features (e.g., tree branch), which explains the 0.4 Prob.Sum score presented in Figure 3.13. Traditional approaches (i.e., Optical Flow and Background Subtraction) are heavily affected by the bird's motion state and environmental conditions (e.g., shadows).

In addition to the motion state of the bird, we investigate the performance of all approaches categorized by the event type in Figure 3.14. Results show that our approach outperforms the other baselines for all event types except for "Walk-In" and "Throwing", where Optical Flow performs better due to the bird being located at the far edges of the nest. Optical Flow detects motion toward

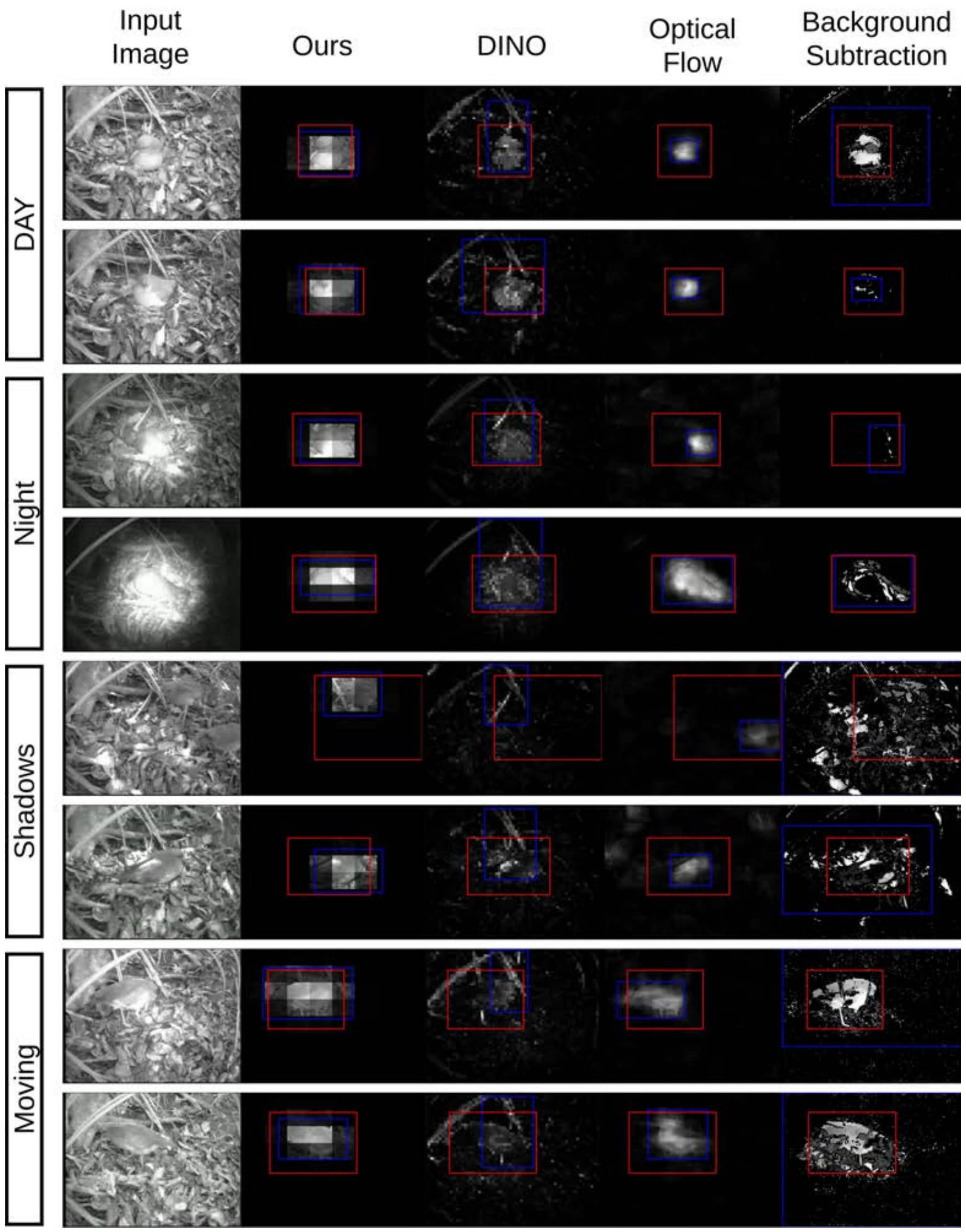


Figure 3.15: Qualitative comparison of the spatial segmentation task. Red bounding boxes indicate ground truth labels, while blue boxes indicate the predicted location of the bird.

the edges of the frame more accurately than our approach due to the scarcity of training examples with the bird moving close to the edge of the frame/nest.

It is challenging to quantify the performance of models that learn in an online manner from streaming input, mainly due to the absence of explicit training split. Therefore, it is necessary to monitor the model's performance and training progress during the initial phase of training (i.e., the first few hours of data). Ideally, the model should not be able to localize the bird at the beginning of the training; however, it should start to more accurately and confidently locate the bird as the training progresses. During the first few hours, we expect to see an increase in the localization metrics and a decrease in entropy - indicating higher performance and confidence in the predictions as training progresses. Figure 3.16 displays the IoU and Prob.Sum. localization metrics and the entropy of the attention map plotted for the first two hours of streaming data. Results show a significant increase in localization performance, as well as a decrease in entropy - indicating more localized predictions. We repeat the experiment starting at four different starting times in the dataset; the lines report the averages, while the widths of the lines (shown with faded color) report the deviations from the averages.

3.4.3 Applicability to Other Vision Domains

To test the validity and generalizability of our approach in other domains, we evaluate its performance on other datasets and report state-of-the-art results compared to both supervised and self-supervised generic event boundary detection (GEBD) approaches. In addition to evaluating the performance on a different domain, we test the robustness of our approach to model variations. In this section, we experiment with using a transformer architecture (instead of an LSTM) as the prediction layer. We use a transformer encoder architecture to both encode temporal patches of eight images and decode patch predictions of the next eight images. The transformer is a drop-in replacement to the LSTM recurrent architecture; both models are used for prediction and have the same input and output dimensions.

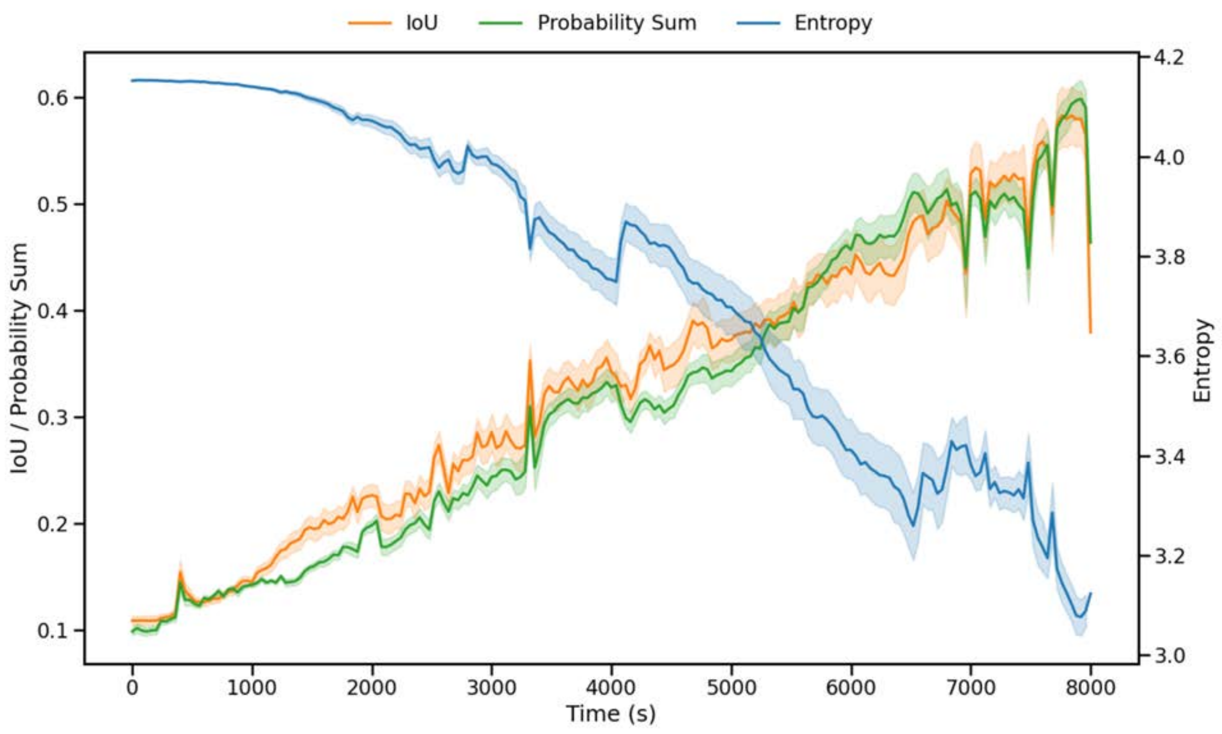


Figure 3.16: Localization metrics and entropy values during the first two hours of training. As expected, IoU and Prob.Sum. increase while the entropy of the attention map decreases, indicating more localized predictions. The experiment is repeated four times at different dataset starting points. Results are shown as average lines with widths (in faded color) indicating deviation from the averages.

3.4.3.1 Datasets

<u>3.4.3.1.1 TAPOS</u>

The TAPOS [223] dataset contains 21 action categories of Olympics sports videos collected from public resources (e.g., YouTube). 16,294 video instances are provided, with an average duration of 9.4 seconds per instance. The total number of boundaries for all instances is 33K, and the number of instances per class varies from 200 to 1,600. A single annotation is released for each video instance.

3.4.3.1.2 Kinetics-GEBD

The Kinetics-GEBD [225] dataset contains 55,351 videos with 1,498K generic boundaries collected from the Kinetics400 dataset (originally from YouTube). The average duration of each video is 10 seconds at 30 FPS (300 Frames total). The dataset is divided evenly between training, validation, and test splits. Five annotations per video are released. Unlike other datasets, Kinetics-GEBD provides *generic* boundaries, not only boundaries caused by action change. The main two boundary causes are action change and shot change, which constitute 63.6% and 19.0% of the dataset, respectively. Consistency scores between annotators are provided, and inconsistent annotations are disregarded.

3.4.3.2 Results Discussion

The performance of our approach is mainly quantified in Table 3.4. We use the evaluation protocol in [225] to test the performance of our approach compared to other baselines. The relative distance metric requires normalizing the boundary distance by the total duration of the video instance. We compare our self-supervised architecture to other supervised and self-supervised approaches. The results are reported at 5% Rel.Dis. threshold value for both TAPOS and Kinetics-GEBD datasets. It is clear from the results that our approach significantly outperforms all the other self-supervised approaches and performs competitively with the supervised state-of-the-art on both datasets. Our approach achieved a 0.733 F1 score on the Kinetics-GEBD test set.

Finally, we provide F1 scores at different thresholds for TAPOS and kinetics-GEBD datasets in Table 3.5 and Table 3.6, respectively. The tables compare the F1 scores to other approaches at different Rel.Dis. thresholds. For the TAPOS dataset (Table 3.6), the results show that our approach consistently outperforms all other approaches across all thresholds. However, for the Kinetics-GEBD (Table 3.5), our approach doubles the performance of all other self-supervised approaches and performs on par with the best-supervised approach when considering the whole range of thresholds.

Table 3.4: Quantitative results on the validation set of TAPOS and Kinetics-GEBD datasets. Results are reported as F1 scores and shown at 0.05 Rel.Dis. threshold.

Supervision	**Approach**	Rel.Dis.@5%	
		TAPOS	Kinetics
Full	ISBA [47]	0.106	-
	TCN [134, 145]	0.237	0.588
	CTM [105]	0.244	-
	TransParser [223]	0.289	-
	BMN [144]	-	0.186
	BMN-StartEnd [225]	-	0.491
	TCN-TAPOS [225]	-	0.464
	PC [225]	**0.522**	0.625
	Kang *et al.* [114]	-	**0.813**
None	SceneDetect [225]	0.035	0.275
	PA-Random [225]	0.158	0.336
	PA-DPC [225]	0.360	0.396
	Ours	**0.562**	**0.672**

Table 3.5: F1 validation results on Kinetics-GEBD dataset. Results shown for various supervised and unsupervised GEBD methods at different Rel.Dis. thresholds.

Supervision	Approach	Rel.Dis. threshold									
		0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5
Full	BMN [144]	0.186	0.204	0.213	0.220	0.226	0.230	0.233	0.237	0.239	0.241
	BMN-StartEnd [225]	0.491	0.589	0.627	0.648	0.660	0.668	0.674	0.678	0.681	0.683
	TCN-TAPOS [225]	0.464	0.560	0.602	0.628	0.645	0.659	0.669	0.676	0.682	0.687
	TCN [134, 145]	0.588	0.657	0.679	0.691	0.698	0.703	0.706	0.708	0.710	0.712
	PC [225]	**0.625**	**0.758**	**0.804**	**0.829**	**0.844**	**0.853**	**0.859**	**0.864**	**0.867**	**0.870**
None	SceneDetect [225]	0.275	0.300	0.312	0.319	0.324	0.327	0.330	0.332	0.334	0.335
	PA - Random [225]	0.336	0.435	0.484	0.512	0.529	0.541	0.548	0.554	0.558	0.561
	PA-DPC [225]	0.396	0.488	0.520	0.534	0.544	0.550	0.555	0.558	0.561	0.564
	Ours	**0.672**	**0.768**	**0.793**	**0.804**	**0.809**	**0.812**	**0.814**	**0.815**	**0.816**	**0.818**

We present qualitative results (Figure 3.17) in the form of prediction error plots annotated with ground truth and prediction timestamps. We show that the predicted boundaries (shown in green lines), detected at the peaks of the prediction error signal, appear close to the ground truth boundaries (red vertical lines), which results in high precision. It is also clear that the number of predicted boundaries is relatively close to the ground truth boundaries, resulting in a high recall

Table 3.6: F1 validation results on TAPOS dataset. Results shown for various supervised and unsupervised GEBD methods at different Rel.Dis. thresholds.

Supervision	Approach	Rel.Dis. threshold									
		0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5
Full	ISBA [47]	0.106	0.170	0.227	0.265	0.298	0.326	0.348	0.369	0.382	0.396
	TCN [134, 145]	0.237	0.312	0.331	0.339	0.342	0.344	0.347	0.348	0.348	0.348
	CTM [105]	0.244	0.312	0.336	0.351	0.361	0.369	0.374	0.381	0.383	0.385
	TransParser [223]	0.289	0.381	0.435	0.475	0.500	0.514	0.527	0.534	0.540	0.545
	PC [225]	**0.522**	**0.595**	**0.628**	**0.646**	**0.659**	**0.665**	**0.671**	**0.676**	**0.679**	**0.683**
None	SceneDetect [225]	0.035	0.045	0.047	0.051	0.053	0.054	0.055	0.056	0.057	0.058
	PA - Random [225]	0.158	0.233	0.273	0.310	0.331	0.347	0.357	0.369	0.376	0.384
	PA-DPC [225]	0.360	0.459	0.507	0.543	0.567	0.579	0.592	0.601	0.609	0.615
	Ours	**0.562**	**0.617**	**0.644**	**0.668**	**0.682**	**0.692**	**0.700**	**0.708**	**0.715**	**0.718**

rate. The figure also shows the corresponding action localization result for each plot. Action localization results are presented as alpha/transparency channels overlaid on top of the input images; the visible regions of the image highlight the most dominant action. We provide qualitative results showing videos during which the camera is moving when our approach performs reasonably well on both segmentation and localization tasks. This demonstrates the effectiveness of the predictive approach in learning to predict and ignore camera motion when calculating the total loss. More action localization results are provided in Figure 3.18.

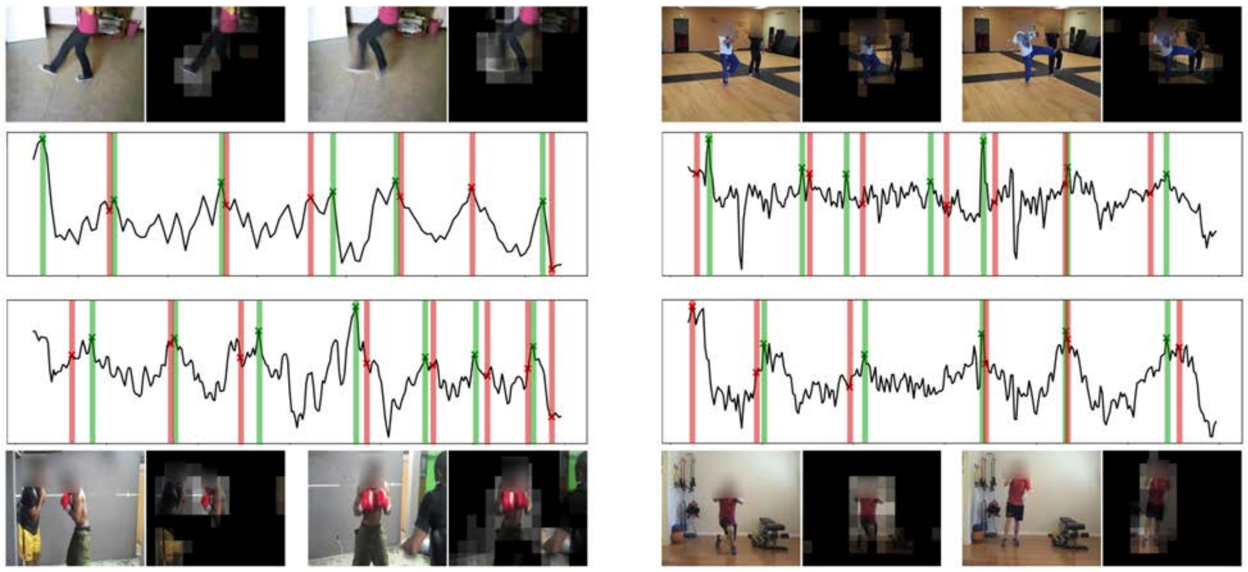


Figure 3.17: Qualitative results of spatio-temporal event boundary detection. Results demonstrate the efficacy of our approach at detecting event boundaries and localizing actions within input frames. Middle plots show the gradient of the error signal for four different examples. Green vertical lines show the location of the predicted boundaries relative to the annotated boundaries presented as red vertical lines. Action localization qualitative results are presented as an overlay of the transparency map.

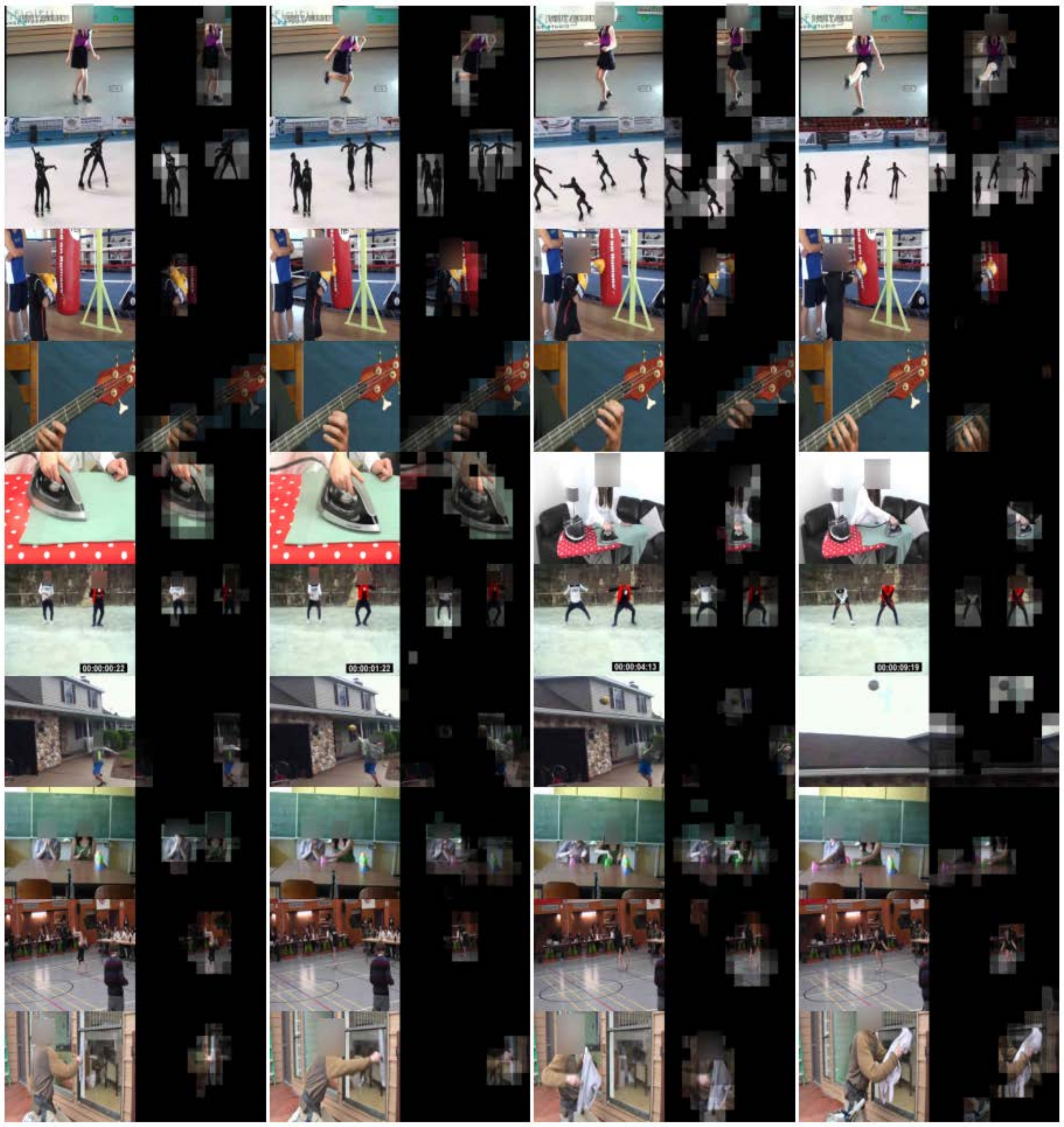


Figure 3.18: Additional qualitative results for action localization from the Kinetics-GEBD dataset. Results are displayed as transparency map overlaid on the input RGB images.

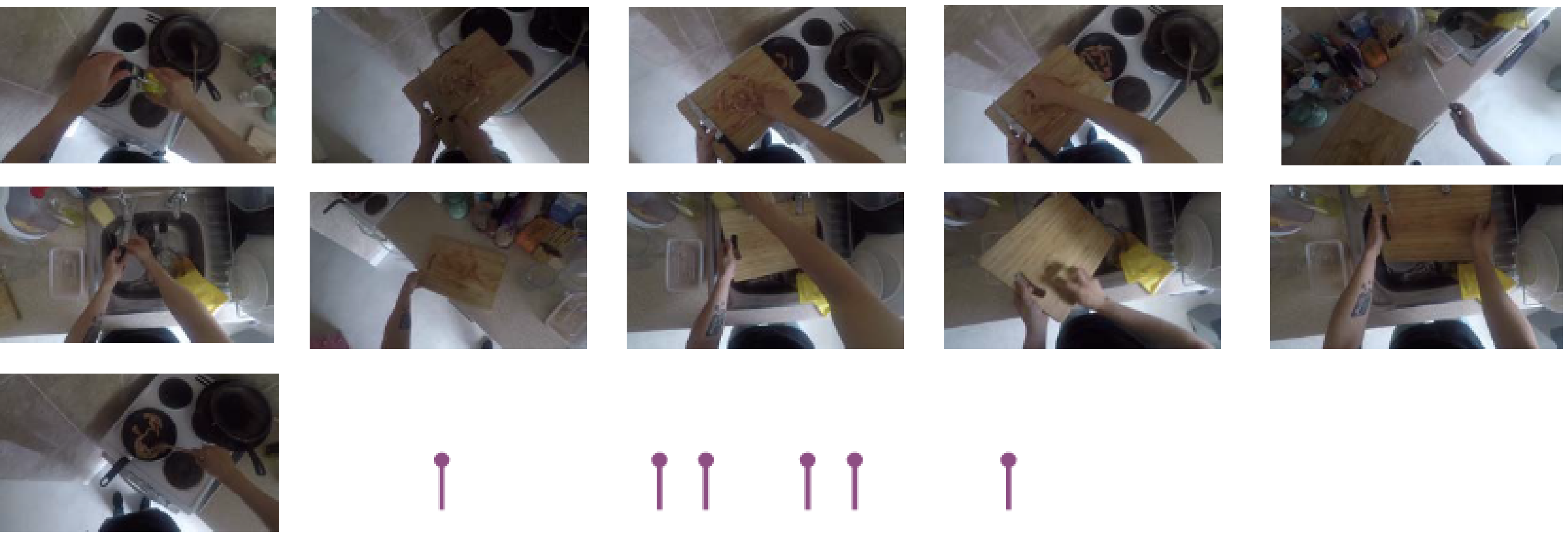
Chapter 4: Multi-Layered Hierarchical Prediction

"A man just beginning to learn radiotelegraphic code hears each dit and dah as a separate chunk. Soon he is able to organize these sounds into letters and then he can deal with the letters as chunks. Then the letters organize themselves as words, which are still larger chunks, and he begins to hear whole phrases."

— George A. Miller (1956)

We present a novel[3] self-supervised approach for hierarchical representation learning and segmentation of perceptual inputs in a streaming fashion. This chapter addresses how to semantically group streaming inputs into chunks at various levels of a hierarchy while simultaneously learning, for each chunk, robust global representations throughout the domain. To achieve this, we propose STREAMER, an architecture that is trained layer-by-layer, adapting to the complexity of the input domain. Building on the predictive learning principles introduced in Chapter 3, this chapter improves upon that foundation by expanding the architecture from a single layer to multiple layers. This extension enables the model to discover compositional part-whole structures (i.e., events) in videos. Unlike standard approaches that simply stack layers, such as CNNs or Transformer architectures, our method uses a more sophisticated hierarchical design that supports both top-down and bottom-up information flow across layers. As a result, our model can capture richer and more complex representations of the input data, further enhancing representation quality and event segmentation. In our approach, each layer is trained with two primary objectives: making accurate predictions into the future and providing necessary information to other levels for achieving the same objective. The event hierarchy is constructed by detecting prediction error peaks at different levels, where a detected boundary triggers a bottom-up information flow. At an event boundary, the encoded representation of inputs at one layer becomes the input to a higher-level layer.

[3]This chapter was published in the Neural Information Processing Systems conference. The author owns the copyright of the published material.



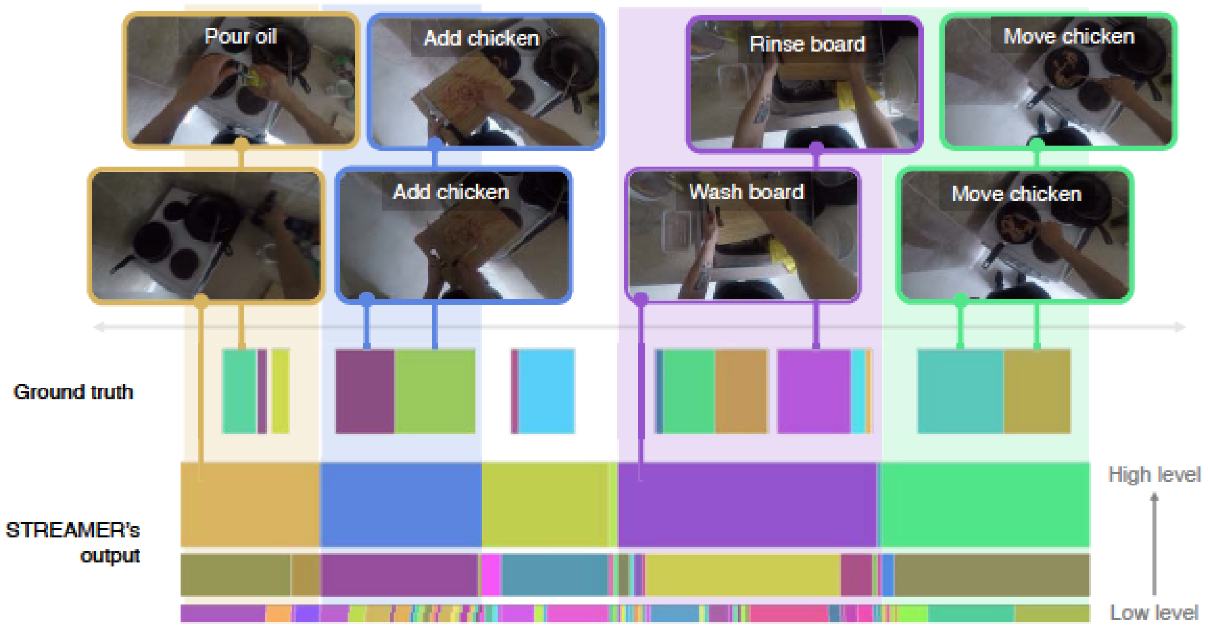


Figure 4.1: Comparison of STREAMER's hierarchical output to single-level ground truth annotations from EPIC-KITCHENS. The ground truth contains redundant narrations for successive annotations (*e.g.*, *add chicken* 🟪, 🟩); STREAMER identifies such instances as a single high level event (🟦). (Narrations from ground truth)

In temporal event analysis, an event is defined as "a segment in time that is perceived by an observer to have a *beginning* and an *end*" [281]. Events could be described by a sequence of constituent events of relatively finer detail, thus forming a hierarchical structure. The end of an event and the beginning of the next is a *segmentation boundary*, marking an event transition. Segmentation boundaries in the lower levels of the hierarchy represent event transitions at relatively granular scales, whereas boundaries in higher levels denote higher-level event transitions.

We propose a structurally self-evolving model to learn the hierarchical representation of such events in a self-supervised streaming fashion through predictive learning. Structural evolution refers to the model's capability to create learnable layers *ad hoc* during training. One may argue that existing deep learning architectures are compositional in nature, where high-level features are composed of lower-level features, forming a hierarchy of features. However, it is important to distinguish between a feature hierarchy and an event hierarchy: an event hierarchy is similar to a part/whole hierarchy in the sense that each event has clear boundaries that reflect the beginning and the end of a coherent chunk of information. One may also view the hierarchy as a redundancy pooling mechanism, where information grouped as one event is considered redundant for a higher level and can be summarized into a single representation for higher-level processing.

Our model is capable of generating a hierarchy of event segments (Figure 4.1) by learning unique semantic representations for each event type directly from video frames. This is achieved through predictive learning, which models the causal structure of events. These learned representations are expressive enough to enable video snippet retrieval across videos. Each level in the hierarchy selectively groups inputs from the level below to form coherent event representations, which are then sent to the level above. As a result, the hierarchy exhibits temporally aligned boundaries, with each level containing a subset of the boundaries detected in the lower level.

As often prescribed [83, 101], we impose the following biologically-plausible constraints on our learning algorithm:

1. The learning algorithm should be *continuous and online*. Most existing learning algorithms offer batch-based offline learning. However, the learning in the neocortex occurs continuously in a streaming fashion while seeing each datapoint only once

2. The learning should involve the ability to make *high-order predictions* by “incorporating contextual information from the past. The network needs to dynamically determine how much temporal context is needed to make the best predictions” [83] (Section 4.1.1)

3. Learning algorithms should be *self-supervised* and should not assume labels for training [132]; instead, they should be able to figure out the learning objective from patterns and causal structures within the data

4. The learning should stem from a *universal general-purpose algorithm*. This is supported by observations of the brain circuitry showing that all neocortical regions are doing the same task in a repeated structure of cells [83]. Therefore, there should be no need for a global loss function (*i.e.*, end-to-end training with high-level labels); local learning rules should suffice (Section 4.2)

4.1.1 Predictive Learning

Predictive learning refers to the brain’s ability to generate predictions about future events based on past experiences. It is a fundamental process in human cognition that guides perception, action, and thought [123, 274]. The discrepancy between the brain’s predictions and the observed perceptual inputs forms a useful training signal for optimizing cortical functions: if a model can predict into the future, it implies that it has learned the underlying causal structure of the surrounding environment. Theories of cognition hypothesize that the brain only extracts and selects features from the previous context that help in minimizing future prediction errors, thus making the sensory cortex optimized for prediction of future input [229]. A measure of intelligence can be formulated as the ability of a model to generate accurate, long-range future prediction [245].To this end, we design an architecture with the main goal of minimizing the prediction error, also referred to as maximizing the *model evidence* in Bayesian inference according to the free energy principle [63, 64].

Event segmentation theory (EST) suggests that desirable properties such as event segmentation emerge as a byproduct of minimizing the prediction loss [281]. Humans are capable of *chunking* streaming perceptual inputs into events (and *chunking* spatial regions into objects [46]) to allow for memory consolidation and event retrieval for better future predictions. EST breaks down streaming sensory input into chunks by detecting event boundaries as transient peaks in the prediction error. The detected boundaries trigger a process of transitioning (*i.e.*, shifting) to a new event model whereby the current event model is saved in the event schemata, and a different event model is retrieved, or a new one initialized to better explain the new observations. One challenge in implementing a computational model of EST is encoding long-range dependencies from the previous context to allow for contextualized representations and accurate predictions. To address this challenge, we construct a hierarchy of event models operating at different time-scales, predicting future events with varying granularity. This hierarchical structure enables the prediction function at any layer to extract context information dynamically from any other layer, enhancing prediction during inference (learning constraint 2). Recent approaches [165, 168, 169, 258] inspired by EST have focused on event boundary detection using predictive learning. However, these methods typically train a single level and do not support higher-order predictions.

4.1.2 Hierarchical Event Models

A single-level predictive model considers events that occur only at a single level of granularity rendering them unable to encode long-range, higher-order causal relationships in complex events. Conversely, a high-level representation does not contain the level of detail needed for accurately predicting low-level actions; it only encodes a high-level conceptual understanding of the action. Therefore, a hierarchy of event models is necessary to make predictions accurately at different levels of granularity [96, 132]. It is necessary to continuously predict future events at different levels of granularity, where low-level event models encode highly detailed information to perform short-term prediction and high-level event models encode conceptual low-detail features to perform long-term prediction.

EST identifies event boundaries based on transient peaks in the prediction error. To learn a hierarchical structure, we extend EST: we use event models at the boundaries in a layer as inputs to the layer above. The prediction error of each layer determines event demarcation, regulating the number of inputs pooled and sent to the layer above. This enables dynamic access to long-range context for short-term prediction, as required. This setup results in stacked predictive layers that perform the same prediction process with varying timescales subjective to their internal representations.

4.1.3 Cross-Layer Communication

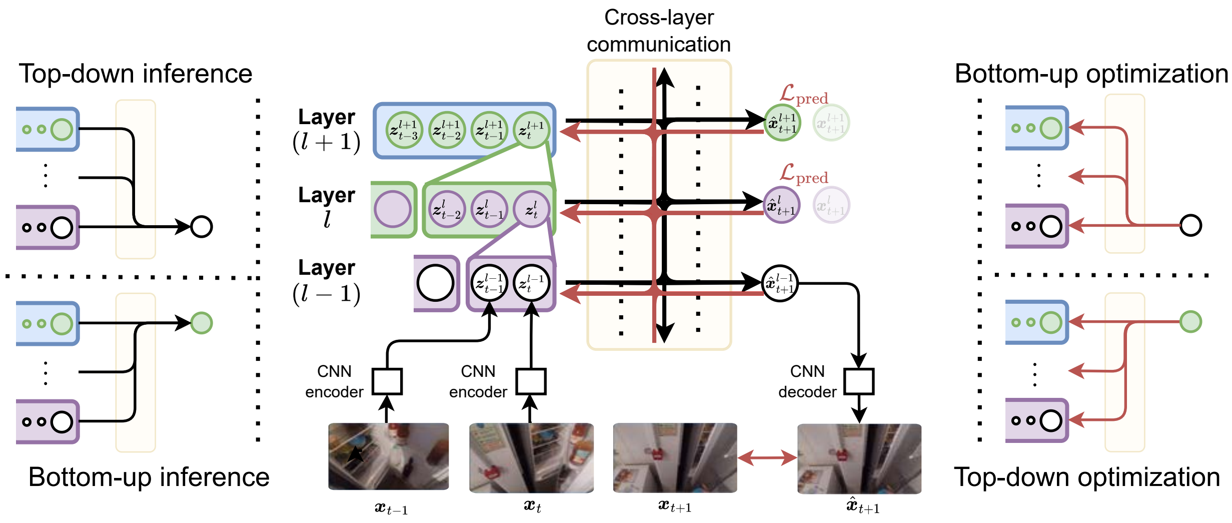


Figure 4.2: An overview of the multi-level hierarchical predictive learning approach. Given a stream of inputs at any layer, our model combines them and generates a bottleneck representation, which becomes the input to the level above it. The cross-layer communication could be broken down into top-down and bottom-up contextualized inference (left) and optimization (right).

As noted in Section 4.1.2, coarsely detailed long-range contexts come from higher layers (the top-left block of Figure 4.2), and highly detailed short-range contexts come from lower layers (the bottom-left block of Figure 4.2), both of which are crucial to predict future events accurately. Therefore, the prediction at each layer should be conditioned upon its own representation and those of the other layers (Equation (4.2)). These two types of contexts can be derived by minimizing the prediction error at different layers. Hence, making perfect predictions is not the primary goal of this model but rather continuously improving its overall predictive capability.

4.1.3.1 Contextualized Inference

A major challenge in current architectures is modeling long-range temporal dependencies between inputs. Most research has focused on modifying recurrent networks [32, 100] or extending the sequence length of transformers [40, 271] to mitigate the problem of vanishing and exploding gradients in long-range backpropagation through time [179]. Instead, we solve this problem by allowing the multi-level context representations to be shared across layers during inference. It is worth noting that this type of inference is rarely used in typical deep learning paradigms, where the top-down influence only comes from backpropagating a supervised loss signal (*i.e.*, top-down optimization). Biologically-inspired architectures such as PredNet [152] utilize top-down inference connections to improve low-level predictions (*i.e.*, frames); however, these predictive coding architectures send the prediction error signal from each low-level observation (*i.e.*, each frame) to higher levels which prevents the network from explicitly building hierarchical levels with varying degrees of context granularity.

4.1.3.2 Contextualized Optimization

Contextualized inference improves prediction, which is crucial for event boundary detection. However, we also aim to learn rich, meaningful representations. In Section 4.1, we noted that a 'parent' event could consist of multiple interchangeable low-level events. For instance, making a sandwich can involve spreading butter or adding cheese. From a high-level, using either ingredient amounts to the same parent event: "making a sandwich". Despite their visual differences, the prediction network must embed meaning and learn semantic similarities between these low-level events (*i.e.*, spreading butter and adding cheese).

We implement this through "contextualized optimization" of events (Section 4.2.2.1), where each layer aligns the input representations from the lower level to minimize its own prediction loss using its context. It must be noted that the contextualization from higher layers (Figure 4.2, bottom-right) is balanced by the predictive inference at the lower levels (Figure 4.2, top-right), which visually distinguishes the interchangeable events. This balance of optimization embeds meaningful representations into the distinct low-level representations without collapsing the model. These representations can also be utilized for event retrieval at different hierarchical levels (Figure 4.5). Unlike other representation learning frameworks that employ techniques like exponential moving average (EMA) or asymmetric branches to prevent model collapse [29, 31, 74], we ensure that higher layers remain grounded in predicting lower-level inputs through bottom-up optimization.

4.2 Hierarchical Predictive Model

Our goal is to incrementally build a stack of identical layers over the course of the learning, where each layer communicates with the layers above and below it. The layers are created as needed and are trained to function at different timescales; the output events from layer become the inputs to the layer , as illustrated in Figure 4.3. We describe the model and its connections for a single layer , but the same structure applies to all the layers in the model (learning constraint 4). In what follows, we describe the design of a mathematical model for a single predictive layer that is capable of (1) encoding temporal input into unique semantic representations (the *event model*) contextualized by previous events, (2) predicting the location of event boundaries (event demarcation), and (3) allowing for communication with other existing layers in the prediction stack to minimize its own prediction loss.

4.2.1 Temporal Encoding

Let $\mathcal{X}$ be a set of inputs to a layer at discrete time steps in the range where each input . First, we aim to generate an "*event model*" which is a single bottleneck representation of the given inputs $\mathcal{X}$. To accomplish this, we define a function with temporally shared learnable weights to evaluate the importance of each input in $\mathcal{X}^{(l)}$ for solving the prediction task at hand, as expressed in Equation (4.1).

$$z^{(l)} = f^{(l)}(\mathcal{X}^{(l)}; \Phi^{(l)}) \tag{4.1}$$

This event model will be trained to extract information from $\mathcal{X}^{(l)}$ that is helpful for hierarchical prediction. Ideally, the bottleneck representation should encode top-down semantics, which allow for event retrieval and a bottom-up interpretation of the input to minimize the prediction loss of the following input. The following subsection describes the learning objective to accomplish this encoding task.

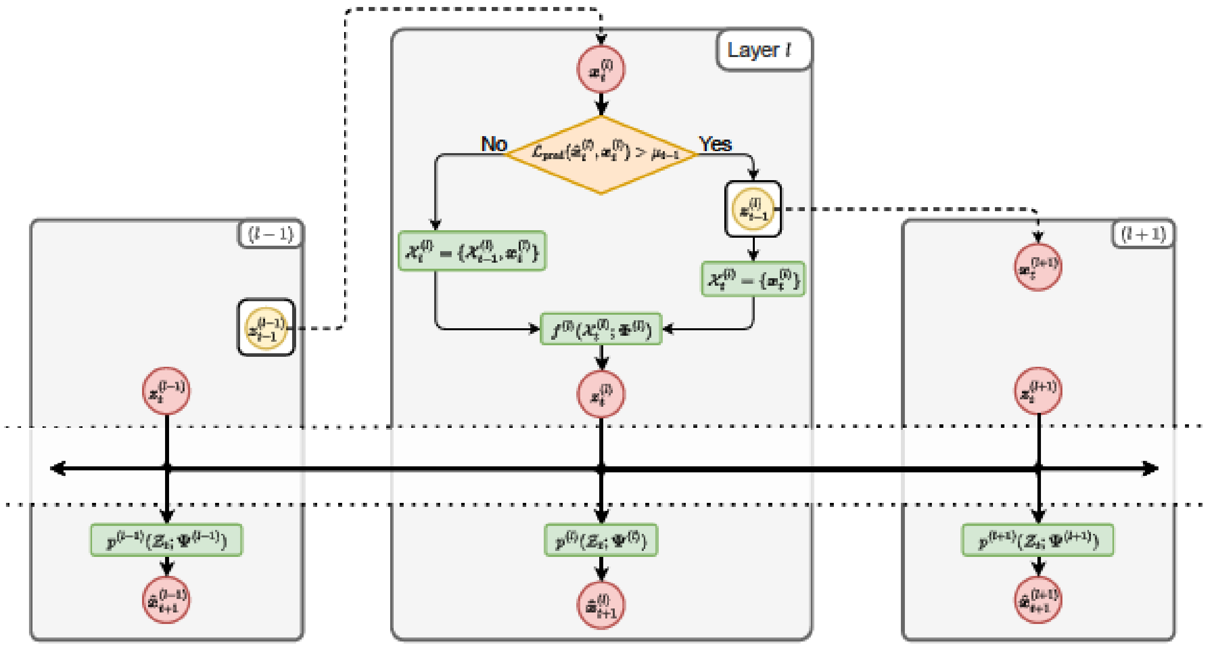

Layer l
$x_t^{(l)}$
No
$\mathcal{L}_{\text{pred}}(\hat{x}_t^{(l)}, x_t^{(l)}) > \mu_{t-1}$
Yes
$z_{t-1}^{(l)}$
$\mathcal{X}_t^{(l)} = \{\mathcal{X}_{t-1}^{(l)}, x_t^{(l)}\}$
$\mathcal{X}_t^{(l)} = \{x_t^{(l)}\}$
$f^{(l)}(\mathcal{X}_t^{(l)}; \Phi^{(l)})$
$z_t^{(l)}$
$p^{(l)}(Z_t; \Psi^{(l)})$
$\hat{x}_{t+1}^{(l)}$
$(l-1)$
$z_{t-1}^{(l-1)}$
$z_t^{(l-1)}$
$p^{(l-1)}(Z_t; \Psi^{(l-1)})$
$\hat{x}_{t+1}^{(l-1)}$
$(l+1)$
$x_t^{(l+1)}$
$z_t^{(l+1)}$
$p^{(l+1)}(Z_t; \Psi^{(l+1)})$
$\hat{x}_{t+1}^{(l+1)}$

Figure 4.3: A diagram illustrating information flow across stacked identical layers. Each layer compares its prediction $\hat{x}_t$ with the input x_t received from the layer below. If the prediction error $\mathcal{L}_{\text{pred}}$ is over a threshold μ_{t-1}, the current representation z_{t-1} becomes the input to the layer above, and the working set is reset with x_t; otherwise, x_t is appended to the working set $\mathcal{X}_t$

4.2.2 Temporal Prediction

At the core of our architecture is the prediction block, which serves two purposes: event demarcation and cross-layer communication. As previously mentioned, our architecture is built on the premise that minimizing the prediction loss is the only needed objective function for hierarchical event segmentation and representation learning.

4.2.2.1 Cross-Layer Communication

Cross-layer communication allows the representation to utilize information from higher and lower layers when predicting the next input at layer , where is the total number of layers. Let $\mathcal{Z}$ be a set of event models where each element is the output of the temporal encoding function at its corresponding layer as expressed in Equation (4.1). Note that the same time variable is used for representation across layers for simplicity; however, each layer operates in its own subjective timescale. Let be a function of $\mathcal{Z}$ to predict the next input at layer as expressed in Equation (4.2)

$$\mathcal{Z} \tag{4.2}$$

where denotes the learnable parameters of the predictor at layer . The difference between the layer's prediction and the actual input is minimized, allowing the gradients to flow back into the functions to modify each layer's representation as expressed in Equation (4.3).

$$\begin{aligned} \mathcal{L}_{\text{pred}} & \\ & \mathcal{L}_{\text{pred}} \end{aligned} \tag{4.3}$$

The symbol represents an appropriate distance measure between two vectors.

4.2.2.2 Event Demarcation

Event demarcation is the process of detecting event boundaries by using the prediction loss, $\mathcal{L}_{\text{pred}}$, from Equation (4.3). As noted earlier, according to EST, when a boundary is detected, an event model transition occurs, and a new event model is used to explain the previously unpredictable observations. Instead of saving the event model to the event schemata at boundary locations as described in EST, we use it as a detached input (denoted by) to train the predictive model of the layer above it (*i.e.*, . We compute the running average of the prediction loss with a window of size , expressed by Equation (4.5), and assume that a boundary is detected when the new prediction loss is higher than the smoothed prediction loss, as expressed by the decision function in Equation (4.4).

$$\begin{array}{l} \text{if } \mathcal{L}_{\text{pred}} \\ \text{otherwise} \end{array} \tag{4.4}$$

where the running average is given by

$$— \quad \mathcal{L}_{\text{pred}} \tag{4.5}$$

4.2.3 Hierarchical Gradient Normalization

It is necessary to scale the gradients differently from conventional gradient updates because of the hierarchical nature of the model and its learning based on dynamic temporal contexts. There are three variables influencing the amount of accumulation of gradients:

1. The *relative timescale* between each layer is determined by the number of inputs. For instance, let the event encoder in layer have seen $\mathcal{X}$ inputs, that at layer have seen $\mathcal{X}$ inputs, and that at have seen $\mathcal{X}$ inputs. Then the input to layer is

a result of seeing a total of . This term can then be used to scale up the learning at any level , expressed as $\mathcal{X}$.

2. The *reach of influence* of each level's representation on a given level's encoder is influenced by its distance from another. For instance, if the input to comes from the levels , then the weight of learning should be centered at and diminish as the distance increases. Such a weight at any level is given by . To ensure that the learning values sum up to 1 when this scaling is applied, the weights are normalized to add up to 1 as $\frac{}{}$.
3. The encoder receives *accumulated feedback* from predictors of all the layers; therefore the change in prediction loss with respect to encoder parameters in any given layer should be normalized by the total number of layers, given by $\frac{}{}$.

The temporal encoding model can be learned by scaling its gradients as expressed by the scaled Jacobian $\mathcal{L}$ in Equation (4.6).

$$\mathcal{L} \qquad \mathcal{L} \qquad \begin{matrix} \frac{\mathcal{L}}{} & & \frac{\mathcal{L}}{} \\ \vdots & \ddots & \vdots \\ \frac{\mathcal{L}}{} & & \frac{\mathcal{L}}{} \end{matrix} \tag{4.6}$$

where

$$\underbrace{\frac{}{}}_{\text{feedback}} \quad \underbrace{\frac{}{}}_{\text{reach of influence}} \quad \underbrace{\mathcal{X}}_{\text{timescale}} \tag{4.7}$$

Similarly, the temporal prediction model's gradients are controlled with scaling factors as expressed in Equation (4.8).

$$\mathcal{L} \qquad \mathcal{L} \qquad \frac{\mathcal{L}}{} \qquad \frac{\mathcal{L}}{} \tag{4.8}$$

where

$$\underbrace{\frac{\qquad}{}}_{\text{reach of influence}} \quad \underbrace{\mathcal{X}}_{\text{timescale}} \tag{4.9}$$

4.2.4 Hierarchical Level Reduction

Since the ground truth annotations are provided as a single level of event annotations, it is not possible to compare them with rich hierarchical event segmentation predicted by STREAMER. For a given video and its ground truth annotations and the predicted annotations, several one-to-one mappings between them exist; we desire to find the one with the highest average IoU. In addition, it is necessary to ensure that the resulting one-to-one mapping does not contain temporally overlapping predicted annotations.

To solve this optimization problem, we design a hierarchical level reduction (HLR) algorithm that reduces multiple layers of hierarchical events down to a single layer by selecting prediction events that maximize IoU with the ground truth while ensuring no overlap of events during reduction. We design HLR as a recursive greedy optimization strategy. At each recursive level of Algorithm 2, multiple ground truth events are competing to be assigned to a single predicted event, so HLR returns the best of the two options (line 18): (1) assigning the event to the ground truth with the highest IoU, and (2) averaging the outputs of same recursive function over all children of the predicted event.

For models generating a hierarchical structure of events, the proposed Hierarchical Level Reduction (HLR) algorithm could be applied for comparison and evaluation. On the other hand, methods that generate a single layer of event segments can directly compare to our 1-layer evaluation reported in Table 4.1.

Algorithm 2 : *Hierarchy Level Reduction*. Given a list of the highest level annotations $\mathcal{A}$ from the predicted hierarchy and the ground truth annotations $\mathcal{G}$, this algorithm finds the optimal match of the predicted annotations across the hierarchy with the ground truth while avoiding any temporal overlap between events.

Input:
$\mathcal{A}$: a list of predicted events at the highest level
$\mathcal{G}$: a list of ground truth event annotations
Output:
The overall IoU of the resulting hierarchy reduction

```
procedure FindMatches($\mathcal{A}$ $\mathcal{G}$)
    for all $\mathcal{G}$ do
        max_ious $\mathcal{A}$
        max_ious
        .matches.push( )
        .ious.push(ious )
    end for
    for all $\mathcal{A}$ do
        FindMatches ( .children, .matches)
    end for
end procedure

procedure ReduceLevels( )
    if .matches then return .ious[0]
    end if
    .ious
    if .children then return
    else
        return ReduceLevels .children
    end if
end procedure

FindMatches $\mathcal{A}$ $\mathcal{G}$
return ReduceLevels $\mathcal{A}$
```

4.3 Experimental Evaluation

4.3.1 Delayed Gradient Stepping and Distributed Learning

Unlike our proposed approach, conventional deep learning networks do not utilize high-level outputs in the intermediate-level predictions. Since our model includes a top-down inference component, such that a lower level (*e.g.*,) backpropagates its loss gradients into the temporal encoding functions of a higher level (*e.g.*,), we cannot apply the gradients immediately after loss calculation at layer . Therefore, we allow for scaled (*i.e.*, Section 4.2.3 and Equation (4.8)) gradients to accumulate at all layers, then perform a single gradient step when the highest layer backpropagates its loss.

In our streaming hierarchical learning approach, event demarcation is based on the data (*i.e.*, some events are longer than others), posing a challenge for traditional parallelization schemes. We cannot directly batch events as inputs because each layer operates on a different subjective timeline. Therefore, each model is trained separately on a single stream of video data, and the models' parameters are averaged periodically during training. We train eight parallel streams on different sets of videos and average the models' parameters every 1K frames.

4.3.2 Datasets and Comparisons

In our training and evaluation, we use two large-scale egocentric datasets: Ego4D [41] and EPIC-KITCHENS 100 [72]. Ego4D is a collection of videos with a total of 3670 hours of daily-life activity collected from 74 worldwide locations. EPIC-KITCHENS 100 contains 100 hours of egocentric video, 90K action segments, and 20K unique narrations. We train our model in a self-supervised layer-by-layer manner (using only RGB frames and inputting them exactly once) on a random 20% subset of Ego4D and 80% of EPIC-KITCHENS, then evaluate on the rest 20% of EPIC-KITCHENS. We define two evaluation protocols: Protocol 1 divides EPIC-KITCHENS such that the 20% test split comes from kitchens that have not been seen in the training set, whereas Protocol 2 ensures that the kitchens in the test set are also in the training set.

We compare our method with TW-FINCH [217], Offline ABD [50], Online ABD [50], and LSTM+AL [1]. ABD, to the best of our knowledge, is the state of the art in unsupervised event segmentation. Clustering-based event segmentation models do not evaluate on egocentric datasets due to the challenges of camera motion and noise. Most clustering based-approaches use pre-trained or optical-flow-based features, which are not effective when clustered in an egocentric setting. We re-implement ABD due to the unavailability of official code and use available official implementations for the other methods.

4.3.3 Evaluation Metrics and Protocols

4.3.3.1 Event Segmentation

The 20K unique *narrations* in EPIC-KITCHENS include different labels referring to the same actions (*e.g.*, turn tap on, turn on tap); therefore we cannot evaluate the labeling performance of the model. We follow the protocol of LSTM+AL [1] to calculate the Jaccard index (IoU) and mean over frames (MoF) of the one-to-one mapping between the ground truth and predicted segments. Unlike LSTM+AL [1], which uses Hungarian matching to find the one-to-one mapping, we design a generalized recursive algorithm called Hierarchical Level Reduction (HLR) in Algorithm 2 which finds a one-to-one mapping between the ground truth events and (a single-layer or multi-layer hierarchical) predicted events.

4.3.3.2 Representation Quality

To assess the quality of the learned representations, we use the large language model (LLM) GPT 3.5 to first create a dataset of events labels ranked by semantic similarity according to the LLM. In particular, we generate 1K data points sampled from EPIC-KITCHENS, where each data point comprises a ‘query’ narration and a set of 10 ‘key’ narrations, and each key is ranked by its similarity to the query. We then retrieve the features for each event in the comparison and compute the appropriate vector similarity measure, and accordingly rank each key event. This rank list is then compared with the LLM ranking to report the MSE and the Levenshtein edit distance between them. Examples of LLM similarity rankings are available in Supplementary Material.

4.3.4 Results

Table 4.1: Event segmentation comparison for STREAMER and other methods evaluated on EPIC-KITCHENS. None of the methods listed below requires labels. The column 'Layers' refers to the number of layers evaluated against the ground truth: 1 reports the performance of the best layer in the prediction hierarchy, whereas 3 uses the proposed Hierarchical Level Reduction algorithm for evaluation. Numbers in bold typeface indicate the highest performance; underline indicates the second highest

Method	**Backbone**	**Pretrained**	**Layers**	**Protocol 1**		**Protocol 2**	
				MoF	**IoU**	**MoF**	**IoU**
LSTM+AL [1]	VGG16 [228]	ImageNet [207]	1	0.694	0.417	0.659	0.442
TW-FINCH [218]	MTRN [284]	EPIC 50 [41]	1	<u>0.707</u>	0.443	0.692	0.442
Offline ABD [50]			1	0.704	0.438	<u>0.699</u>	0.432
Online ABD [50]			1	0.610	<u>0.496</u>	0.605	<u>0.487</u>
STREAMER	4-layer CNN	-	1	**0.759**	**0.508**	**0.754**	**0.489**
			3	**0.736**	**0.511**	**0.729**	**0.494**

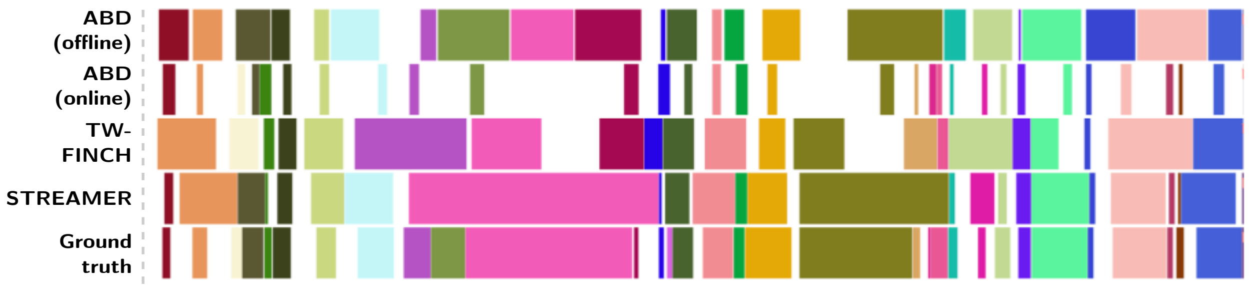


Figure 4.4: Qualitative comparisons of STREAMER and other methods on temporal event segmentation. The Gantt chart shows a more accurate alignment of STREAMER's predictions with the ground truth compared to other methods.

We evaluate STREAMER's performance of event segmentation and compare it with streaming and clustering-based SOTA methods as shown in Table 4.1. Our findings show that the performance of a single layer in STREAMER's hierarchy (the best-performing layer out of three per video) and the full 3-layer hierarchy outperform all other state of the art using IoU and MoF metrics on both testing protocols. It is worth noting that all the other methods use a large CNN backbone with supervised pre-trained weights (some on the same test dataset: EPIC-KITCHENS), whereas our model is trained from scratch using random initialization with a simple 4-layer CNN. We show comparative qualitative results in Figure 4.4. More qualitative results are provided in Supplementary Material.

Additionally, we evaluate the quality of event representation in Table 4.2. We show that self-supervising STREAMER from randomly initialized weights outperforms most clustering-based approaches with pre-trained weights; we perform on par with TW-FINCH when using supervised EPIC-KITCHENS pre-trained weights. Qualitative results of retrieval for the first three nearest neighbors on all the events in the test split are shown in Figure 4.5. More qualitative results are reported in Supplementary Material.

Table 4.2: Retrieval evaluation based on MSE and the Levenshtein edit distance (LD) of the features. All experiments are on EPIC-KITCHENS.

Method	**Weights**	**MSE**	**LD**
Supervised			
TW-FINCH	EPIC	<u>1.00</u>	**0.67**
	IN	1.018	0.710
Offline ABD	EPIC	1.02	0.71
	IN	1.005	0.708
Online ABD	EPIC	<u>1.00</u>	0.70
	IN	1.039	0.704
No supervison			
STREAMER	-	**0.967**	<u>0.695</u>

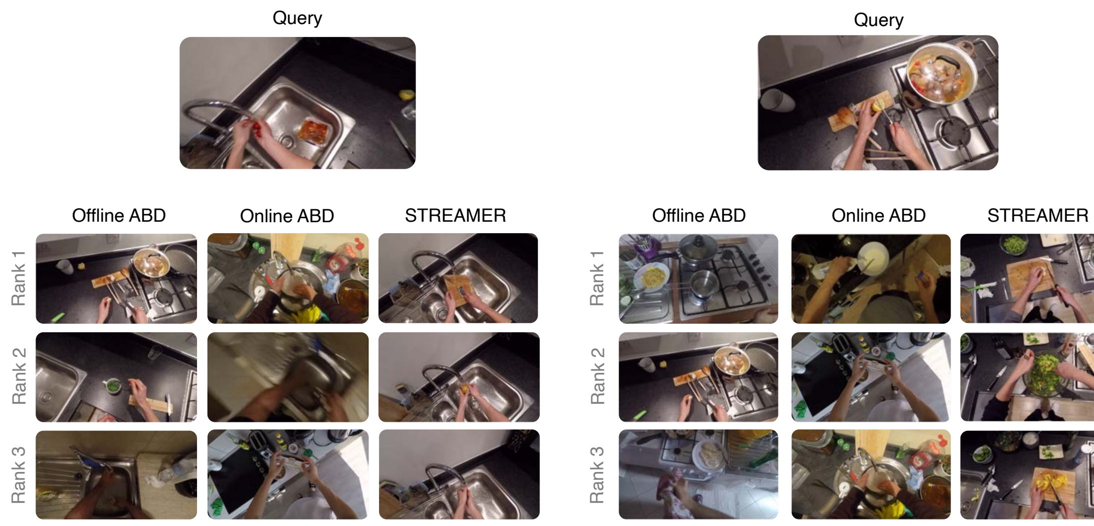


Figure 4.5: Qualitative examples of STREAMER's retrieval of relevant events compared to other methods.

4.3.4.1 Ablations

We investigate three main aspects of STREAMER (Table 4.3): (1) varying the architecture of the temporal encoding model , (2) varying the predictor function , and (3) experimenting with the 'reach of influence' parameter in Equation 4.7. Our findings suggest that STREAMER is robust to different architectural choices of . Our experiments also illustrate the importance of the cross-layer communication of : simply taking the average of $\mathcal{Z}$ as the prediction performs worse than applying a layer-specific MLP to the average; using a transformer to retrieve context from other layers dynamically performs the best. Finally, adjusting the reach of influence by gradient scaling improves the segmentation performance.

To determine the quality of the backbone features learned by STREAMER, we run ablations of using our 4-layer pretrained CNN features on SoTA clustering methods. The results, plotted in Figure 4.6, show significant improvement in the average mean over frames (MoF) performance of event segmentation on the EPIC-KITCHENS dataset. This improvement could be attributed to the robust representations learned by the encoder through hierarchical predictive learning. In particular, since these features are learned through top-down optimization, the CNN backbone is able to predict longer events at higher levels, thus improving the features and contextualization quality.

Table 4.3: STREAMER ablation study reporting performance difference with model hyperparameters.Results show the model's MoF and IoU for different values of (Equation (4.7)); different variants of the predictor and the temporal encoder . 'Best' refers to the layer with the highest performance.

	MoF	**IoU**	**MoF**	**IoU**	**MoF**	**IoU**
	GRU		**LSTM**		**Transformer**	
Best	0.759	0.503	0.761	0.506	0.759	0.508
HLR	0.740	0.503	0.737	0.502	0.736	0.511
	Average		**Average + MLP**		**Transformer**	
Best	0.742	0.479	0.749	0.493	0.759	0.508
HLR	0.725	0.486	0.728	0.494	0.736	0.511
	1		**2**		**3**	
Best	0.756	0.493	0.797	0.498	0.759	0.508
HLR	0.737	0.498	0.732	0.497	0.736	0.511

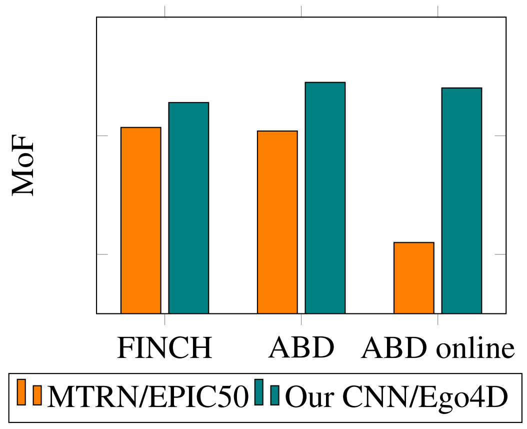


Figure 4.6: Performance of SoTA segmentation approaches with STREAMER weights. Results show performance increase of SoTA clustering-based methods when using STREAMER's pretrained 4-layer CNN features.

Chapter 5: Predictive Attractor Models

"In general we are least aware of what our minds do best."

— Marvin Minsky

Sequential memory, the ability to form and accurately recall a sequence of events or stimuli in the correct order, is a fundamental prerequisite for biological and artificial intelligence as it underpins numerous cognitive functions (e.g., language comprehension, planning, episodic memory formation, etc.) However, existing methods of sequential memory suffer from catastrophic forgetting, limited capacity, slow iterative learning procedures, low-order Markov memory, and, most importantly, the inability to represent and generate multiple valid future possibilities stemming from the same context. Inspired by biologically plausible neuroscience theories of cognition, we propose *Predictive Attractor Models (PAM)*[4], a novel sequence memory architecture with desirable generative properties. PAM is a streaming model that learns a sequence in an online, continuous manner by observing each input *only once*. Additionally, we find that PAM avoids catastrophic forgetting by uniquely representing past context through lateral inhibition in cortical minicolumns, which prevents new memories from overwriting previously learned knowledge. PAM generates future predictions by sampling from a union set of predicted possibilities; this generative ability is realized through an attractor model trained alongside the predictor. We show that PAM is trained with local computations through Hebbian plasticity rules in a biologically plausible framework. Other desirable traits (e.g., noise tolerance, CPU-based learning, capacity scaling) are discussed throughout the chapter. Our findings suggest that PAM represents a significant step forward in the pursuit of biologically plausible and computationally efficient sequential memory models, with broad implications for cognitive science and artificial intelligence research.

[4]This chapter was published in the Neural Information Processing Systems conference. The author owns the copyright of the published material.

5.1 Introduction

Modeling the temporal associations between consecutive inputs in a sequence (i.e., sequential memory) enables biological agents to perform various cognitive functions, such as episodic memory formation [35, 202, 249], complex action planning [78] and translating between languages [4]. For example, playing a musical instrument requires remembering the sequence of notes in a piece of music; similarly, playing a game of chess requires simulating, planning, and executing a sequence of moves in a specific order. While the ability to form such memories of static, unrelated events has been extensively studied [211, 214, 242, 265, 275], the ability of biologically-plausible artificial networks to learn and recall temporally-dependent patterns has not been sufficiently explored in literature [241]. The task of sequential memory is considered challenging for models operating under biological constraints (i.e., local synaptic computations) for many reasons, including catastrophic forgetting, ambiguous context representations, multiple future possibilities, etc.

In addition to the biological constraint, we impose the following set of desirable characteristics as learning constraints on sequence memory models.

- The learning of one sequence does not overwrite the previously learned sequences. This property is defined under the continual learning framework as avoiding catastrophic forgetting and evaluated with the Backward Transfer (BWT) metric [149].
- The model should uniquely encode inputs based on their context in a sequence. Consider the sequence of letters “EVER”; the representation of “E” at position 1 should be different from “E” at position 3, thus resulting in different predictions: “V” and “R”. Moreover, the representation of “E” at position 3 in “EVER” should be different from “E” at position 3 in “CLEVER”. Therefore, positional encoding is not a valid solution.
- When presented with multiple valid, future possibilities, the model should learn to represent each possibility separately yet stochastically sample a single valid possibility. Consider the two sequences “THAT” and “THEY”; after seeing “TH”, the model should learn to generate either “A” or “E”, but not an average [132] or a union of both [83].

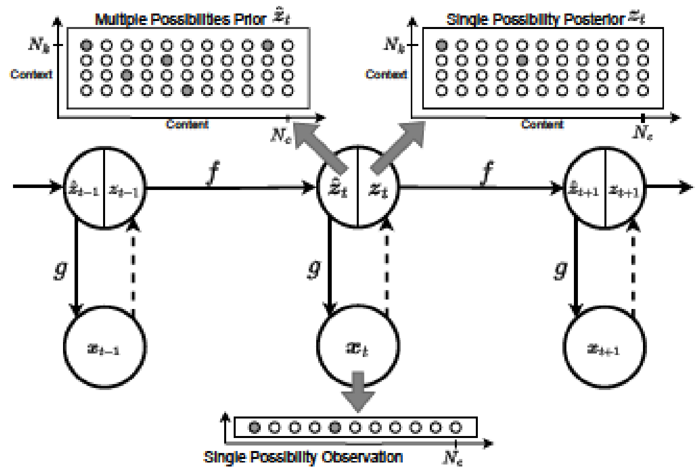
- The model should be capable of incrementally learning each transition without seeing the whole sequence or revisiting older sequence transitions that are previously learned. This property falls under online learning constraints, also called stream learning [83, 167].
- The learning algorithm should be tolerant and robust to significant input noise. A model should continuously clean the noisy inputs using learned priors and beliefs, thus performing future predictions based on the noise-free observations [83].

We propose *Predictive Attractor Models (PAM)*, which consists of a state prediction model and a generative attractor model. The predictor in PAM is inspired by the Hierarchical Temporal Memory (HTM) [83] learning algorithm, where a group of neurons in the same cortical minicolumn share the same receptive feedforward connection from the sensory input on their proximal dendrites. The depolarization of the voltage of any neuron in a single minicolumn (i.e., on distal dendrites) primes this depolarized neuron to fire first while inhibiting all the other neurons in the same minicolumn from firing (i.e., competitive learning). The choice of which neurons fire within the same minicolumn is based on the previously active neurons and their trainable synaptic weights to the depolarized neurons, which gives rise to a unique context representation for every input. The sparsity of representations (discussed later in Section 5.2.2) allows for multiple possible predictions to be represented as a union of individual cell assemblies. The Attractor Model learns to disentangle possibilities by strengthening the synaptic weights between active neurons of input representations and inhibiting the other predicted possibilities from firing, effectively forming fixed point attractors during online learning. During recall, the model uses these learned conditional attractors to sample one of the valid predicted possibilities or uses the attractors as prior beliefs for removing noise from sensory observations.

PAM satisfies the above-listed constraints for a sequential memory model, whereas the current state-of-the-art models fail in all or many of the constraints, as shown in the experiments. Our contributions can be summarized as follows: (1) Present the novel *PAM* learning algorithm that can explicitly represent context in memory without backpropagation, avoid catastrophic forgetting, and perform stochastic generation of multiple future possibilities. (2) Perform extensive evaluation of PAM on multiple tasks (e.g., sequence capacity, sequence generation, catastrophic forgetting, noise robustness, etc.) and different data types (e.g., protein sequences, text, vision). (3) Formulate PAM and its learning rules as a State Space Model grounded in variational inference and the Free Energy Principle [63].

5.2 Predictive Attractor Models

5.2.1 State Space Model (SSM) Formulation



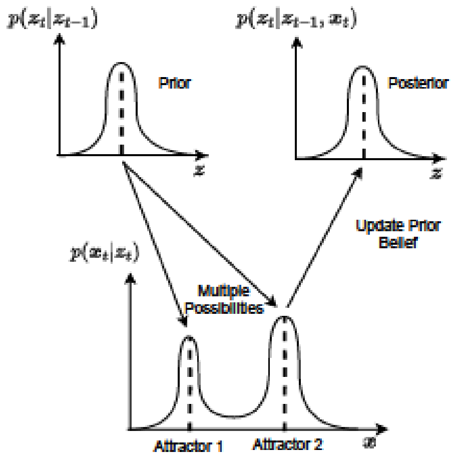


Figure 5.1: State Space Model (SSM) formulation of PAM. (Left): Dynamical system represented by first-order Markov chain of latent states z with transition function f and an emission function g which projects to the observation states x. (Right): Gaussian form assumptions for the prior $\hat{z}$ and posterior z latent states, and the Mixture of Gaussian model representing the conditional probability of multiple possibilities $p(x|z)$

PAM can be represented as a dynamical system with its structure depicted by a Bayesian probabilistic graphical [121, 131] model, more specifically, a State Space Model, where we can perform Bayesian inference on the latent variables and derive learning rules using Variational Inference. Formally, we define a state space model as a tuple $(\mathcal{Z}, \mathcal{X}, f, g)$, where $\mathcal{Z}$ is the latent state space, $\mathcal{X}$ is the observation space, and f and g are the transition and emission functions respectively (similar to HMM [187]). We consider a Gaussian form with white Gaussian noise covariance for the latent states. However, we assume a latent state can generate multiple valid possibilities. Therefore, we model the conditional probability as a Multivariate Gaussian Mixture Model (GMM), where each mode is considered a possibility or a fixed-point attractor in an associative memory model. The GMM has components with means , covariances and component weights . The SSM dynamics can be formally represented with the following equations:

$$\mathcal{N} \quad \text{and} \quad \mathcal{N} \tag{5.1}$$

where $\mathcal{Z}$ and $\mathcal{X}$. From the Bayesian inference viewpoint, we are interested in the posterior . Since the functions and are non-linear, the computation of this posterior is intractable (unlike a LG-SSM, such as Kalman Filter [261]). Therefore, we utilize variational inference to approximate the posterior by assuming the surrogate posterior has a Gaussian form, and minimize the Variational Free Energy (VFE) [65]. The minimization of VFE (in equation 5.2) minimizes the KL-divergence between the approximate posterior and the true posterior . Derivation 2 of Variational Free Energy is provided in Appendix C.3.1

$$\underbrace{\qquad\qquad}_{\text{Variational Free Energy}} \quad \underbrace{\qquad\qquad}_{\text{Latent State Error}} \quad \underbrace{\qquad\qquad}_{\text{Observation Error}} \quad \mathcal{H} \tag{5.2}$$

where and $\mathcal{H}$ is the Entropy of the approximate posterior . The assumption of Gaussian forms for the latent and observable states can further simplify the negative log-likelihood terms (i.e., Latent State Accuracy and Observation Accuracy) to prediction errors. This learning objective encourages the approximate posterior to assign a high probability to states that explain the observations well and follow the latent dynamics of the system. We minimize the prediction errors (i.e., learn the transition and emission functions) through Hebbian rules as shown in equations 5.6 and 5.7.

Theorem 1. *Assume the likelihood in eqn 5.2 represents multiple possibilities using a Gaussian Mixture Model (GMM) conditioned on the latent state , as shown in eqn 5.1. The maximization of such log-likelihood function (i.e., —) w.r.t a query observation state is equivalent to the Hopfield recall function (i.e., eqn 2.1) with the means of the GMM representing the attractors of a Hopfield model. Formally, the weighted average of the GMM means (i.e.,), with the weights being a similarity measure, maximizes the log-likelihood of under the mixture model.*

$$\underbrace{\frac{\mathcal{N}}{\mathcal{N}}}_{\textit{similarity score}} \quad \textit{projection} \tag{5.3}$$

Proof: See Derivation 3 in Appendix C.3.2 for the full proof.

5.2.2 Preliminaries and Notations

5.2.2.1 Sparse Distributed Representation (SDR)

Inspired by the sparse coding principles observed in the brain, SDRs encode information using a small set of active neurons in high dimensional binary representation. We adopt SDRs as a more biologically plausible cell assembly representation [113]. SDRs have desirable characteristics, such as a low chance of false positives and collisions between multiple SDRs and high representational capacity [5] (More on SDRs in Appendix C.6). An SDR is parameterized by the total number of neurons and the number of active neurons . The ratio denotes the SDR sparsity. A 1-dimensional SDR can be indexed as , whereas a 2-dimensional SDR can be indexed as . To identify the active neurons, we define the function to represent the indices of the active neurons in an SDR as .

5.2.2.2 Context as Orthogonal Dimension

We transform the high-order Markov dependencies between observation states into a first-order Markov chain of latent states by storing context information in those latent states. The latent states SDRs, , are represented with two orthogonal dimensions, where content information about the input is stored in one dimension with size , while context related information is stored in an orthogonal dimension with size . Therefore, the projection of the latent state on the first dimension (i.e.,) removes all context information from the state. In contrast, adding context information to an observation state expands the dimensionality of the state (i.e.,) such that context can be encoded without affecting its content. Competitive learning is enforced on the context dimension through lateral inhibition, effectively minimizing the collisions between contexts of multiple SDRs. We define a projection operator . Additionally, we define a projection operator for 1-dimensional SDRs (i.e.,) as shown in equation 5.4.

if s.t. if (5.4)
otherwise, otherwise,

5.2.3 Sequence Learning

Given a sequence of SDR patterns , where , the sequence can be learned by modeling the context-dependent transitions between consecutive inputs within the sequence. We define learnable weight parameters for transition and emission functions, respectively. A single latent state transition is defined as , where is a threshold function transforming the logits to the predicted SDR state . The full sequence learning algorithm is provided in algorithm 3.

5.2.3.1 Context Encoding through Competitive Learning

Every observation contains only content information about the input; we embed the observation with context by expanding the state with an orthogonal dimension (i.e.,) which activates all neurons in the minicolumns at the indices . Then, for each active minicolumn, the neuron in a predictive state (i.e., higher than the prediction threshold) fires and inhibits all the other neurons in the same minicolumn from firing (i.e., lateral inhibition), as shown in Equation 5.5. If none - or more than one - of the neurons are in a predictive state, random Gaussian noise ( ) acts as a tiebreaker to select a context neuron. We *do not* allow multiple neurons to fire in the same minicolumn, which is different from HTM [83], where multiple cells can fire in any minicolumn (e.g., bursting).

$$\begin{cases} & \text{if} \\ & \text{otherwise,} \end{cases} \quad \setminus \qquad (5.5)$$

Algorithm 3 : **Sequence Learning**. Given a sequence of T+1 patterns, this algorithm learns the Transition and Emission synaptic weights (and). Fixed start context is initialized for all learned sequences.

```
 1: procedure TRAIN( )
 2:             \                                        Eqn. 5.5
 3:     for     to     do
 4:
 5:                 \
 6:                                            Update    via Eqn. 5.6
 7:
 8:                                            Update    via Eqn. 5.7
 9:     end for
10: end procedure
```

Algorithm 4 : **Sequence Generation**. Given a noisy sequence (i.e., online), or the first input in a sequence (i.e., offline). The model uses the learned functions and to generate the full sequence. denotes sampled from (eqn 5.8).

```
1:  procedure GENERATE( or )
2:          \                                          Eqn. 5.5
3:      for to do
4:
5:
                     online
6:
                     offline                           Eqn. 5.8
7:          for i=1 to iers do                         Attractors iterations
8:                      \
9:          end for
10:                 \
11:     end for
12: end procedure
```

5.2.3.2 State Transition Learning

The transition between latent states is learned through local computations with Hebbian-based rules. We modify the synaptic weights to model the transition between pre-synaptic neurons and post-synaptic neurons . Only the synapses with active pre-synaptic neurons are modified. The weights operate on context-dependant latent states (i.e.,); thus, the learning of one transition does not overwrite previously learned transition of different contexts, regardless of the input contents (i.e.,). We use the learning constant coefficients and to independently control learning and forgetting behavior, as shown in equation 5.6. A lower encourages learning multiple possibilities by slowing down the forgetting behavior.

$$\underbrace{\frac{}{}\ \frac{}{}}_{\text{increase}}\qquad\underbrace{\frac{}{}\ \frac{}{}}_{\text{decrease}} \tag{5.6}$$

5.2.3.3 Contrastive Attractors Formation

The attractors are formed in an online manner by contrasting the input observation $\boldsymbol{x}_t$ to the predicted union set of possibilities $\downarrow \hat{\boldsymbol{z}}_t$. The goal is to learn excitatory synapses between active neurons of $\boldsymbol{x}_t$, and bidirectional inhibitory synapses between $\boldsymbol{x}_t$ and the union set of predicted possibilities *excluding* the $\boldsymbol{x}_t$ possibility (i.e., $(\downarrow \hat{\boldsymbol{z}}_t) \setminus \boldsymbol{x}_t$), as shown in equation 5.7.

$$\Delta \boldsymbol{B} = \underbrace{\eta_B^+ \cdot \boldsymbol{x}_t \cdot \boldsymbol{x}_t{}^T}_{\Delta \boldsymbol{B}_{\text{increase}}} + \underbrace{\eta_B^- \cdot [\boldsymbol{x}_t \cdot ((\downarrow \hat{\boldsymbol{z}}_t) \setminus \boldsymbol{x}_t)^T + ((\downarrow \hat{\boldsymbol{z}}_t) \setminus \boldsymbol{x}_t) \cdot \boldsymbol{x}_t^T]}_{\Delta \boldsymbol{B}_{\text{decrease}}} \tag{5.7}$$

5.2.4 Sequence Generation

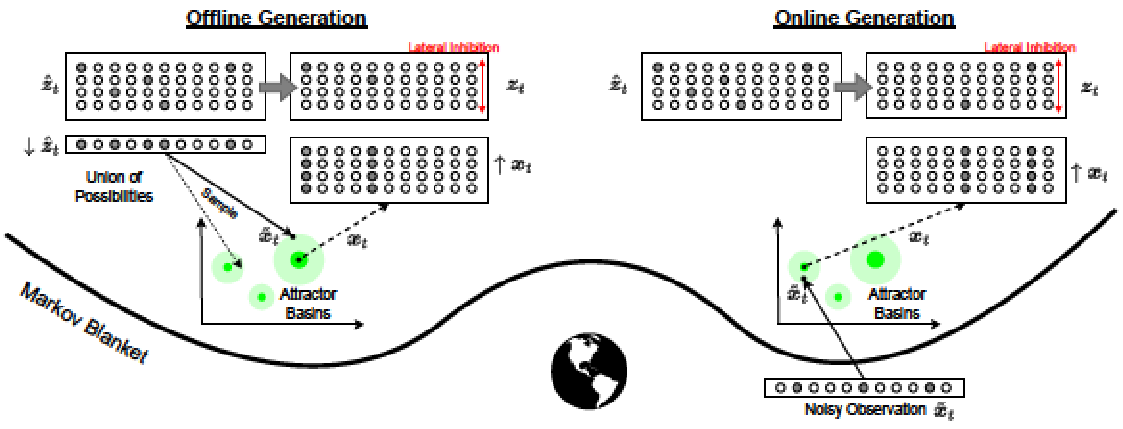


Figure 5.2: Sequence generation in PAM. (Left): Offline generation by sampling a single possibility (i.e., attractor point) from a union of predicted possibilities. (Right): Online generation by removing noise from an observation using the prior beliefs about the observed state. Markov Blanket separates the agent's latent variables from the world observable states.

After learning one or multiple sequences using algorithm 3, we use algorithm 4 to generate sequences. First, we define two generative tasks: online and offline. In online sequence generation, a noisy version of the sequence is provided as input, and the model is expected to generate the original learned sequence. In offline sequence generation, the model is only provided with the first input, and it is expected to generate the entire sequence auto-regressively. For cases with equally valid future predictions (e.g., “a” and “e” after “TH” in “THAT” and “THEY”), the model is expected to stochastically generate either one of the possibilities (i.e., “THAT” or “THEY”). The online generation task is a more challenging extension of the online recall task in [241], where the noise-free inputs are provided, and the model only makes a 1-step prediction into the future. During offline sequence generation, the model randomly samples from the union set of predictions a single SDR with active neurons (equation 5.8) to initialize the iterative attractor procedure. denotes a random permutation function. This random permutation function allows the model to randomly generate a different sequence with every generation.

$$\begin{array}{l} \text{if} \\ \text{otherwise} \end{array} \tag{5.8}$$

5.3 Experimental Evaluation

5.3.1 Evaluation and Metrics

5.3.1.1 Metrics

To evaluate the similarity of two SDRs, we use the Jaccard Index (i.e., IoU), which focuses on the active bits in sparse binary representations. Since the sparsity of the representations can change across experiments and methods, we normalize the IoU by the expected IoU (Derived in Theorem 2 in Appendix C.3.3) of two random SDRs at their specified sparsities. The normalized IoU is computed as $\frac{\text{IoU} \quad \text{IoU}}{\text{IoU}}$. We use the Backward Transfer [149] metric in evaluating catastrophic forgetting. Mean Squared Error (MSE) is used to compare images for vision datasets.

5.3.1.2 Datasets

We perform evaluations on synthetic and real datasets. The synthetic datasets allow us to manually control variables (e.g., sequence length, correlation, noise, input size) to better understand the models' behavior across various settings. Additionally, we evaluate on real datasets of various types (e.g., protein sequences, text, vision) to benchmark PAM's performance relative to other models on more challenging and real sequences. For all vision experiments, we use an SDR autoencoder to learn a mapping between images and SDRs (Details on the autoencoder are provided in Appendix C.4). We run all experiments for 10 different seeds and report the mean and standard deviation in all the figures. More experimental details and results are provided in Appendices C.4 and C.5.

5.3.2 Results

We align our evaluation tasks with the desired characteristics of a biologically plausible sequence memory model, as listed in the introduction. We show that PAM outperforms current predictive coding and associative memory SoTA approaches on all tasks. Most importantly, PAM is capable of long context encoding, multiple possibilities generation, and learning continually and efficiently while avoiding catastrophic forgetting. These tasks pose numerous significant challenges to other methods.

5.3.2.1 Offline Sequence Capacity

We evaluate the models' capacity to learn long sequences by varying the input size , model parameters (e.g.,), and sequence correlation. The correlation is increased by reducing the number of unique patterns (i.e., vocab) used to create a sequence of length . Correlation is computed as $\frac{\text{vocab}}{}$. In Figure 5.3 *A*, we vary the input size and ablate the models to find the maximum sequence length to be encoded and retrieved, in an *offline* manner, with a Normalized IoU higher than 0.9. We set the number of active bits to be 5 unless otherwise specified. Results

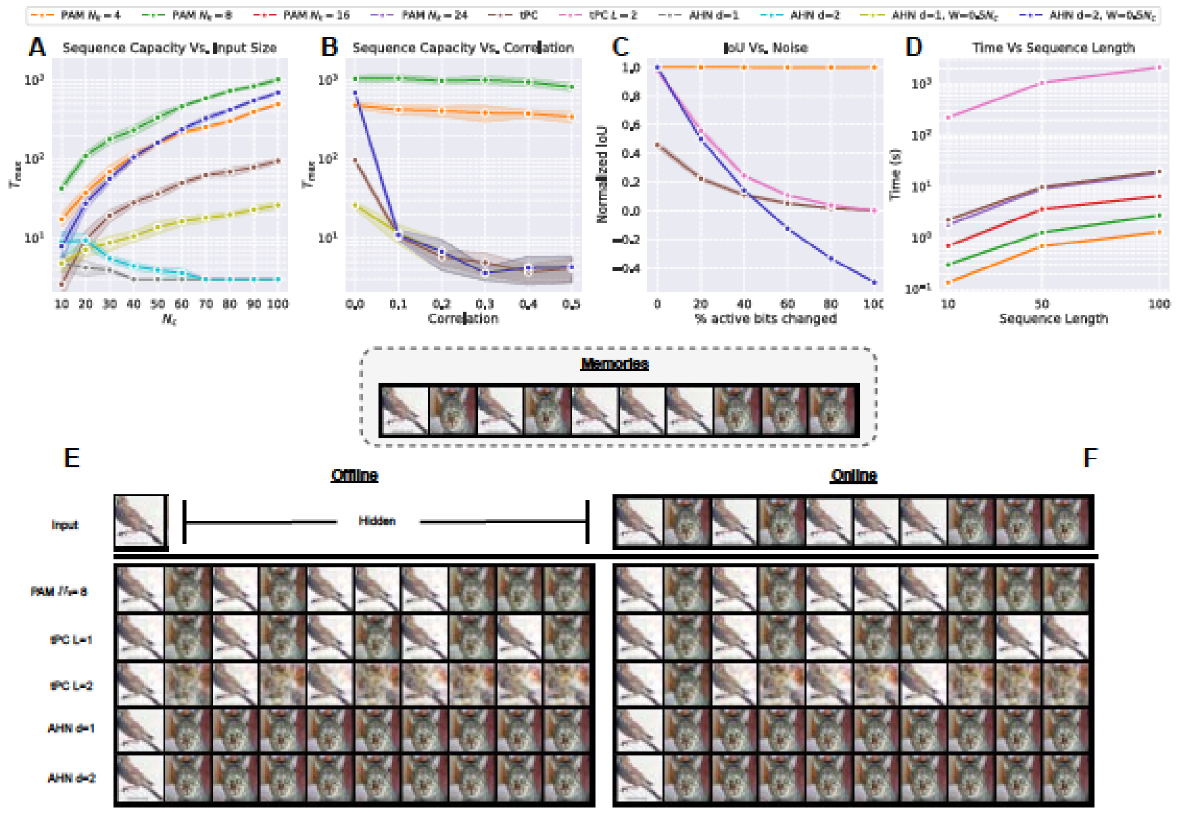


Figure 5.3: Quantitative and qualitative results and comparisons of PAM and other approaches. Quantitative results reported on (*A*-*B*) Offline sequence capacity, (*C*) Noise robustness, and (*D*) Time of sequence learning and recall. Qualitative results reported on highly correlated CIFAR sequence in (*E*) offline and (*F*) online settings. The mean and standard deviation of 10 trials are reported for all plots.

show that Hopfield Networks (HN) fail to learn with sparse representations; therefore, we use W of $0.5N_c$ only for HN and normalize the IoU metric accordingly. PAM's capacity significantly increases with context neurons N_k, as expected. HN's capacity also increases with the polynomial degree d of its separation function; however, as shown in Figure 5.3 *B*, the capacity sharply drops as correlation increases. PAM retains its capacity with increasing correlation, reflecting its ability to encode context in long sequences (i.e., high-order Markov memory). This context encoding property is also demonstrated in the qualitative CIFAR [124] results in Figure 5.3 *E* and *F*, where a short sequence of images with high correlation is used. The model must uniquely encode the context to correctly predict at every step in the sequence. While PAM correctly predicts the full context, single layer tPC learns to indefinitely alternate between the patterns, while two-layered tPC attempts to average its predictions. AHN shows similar low performance and failure mode as in [241].

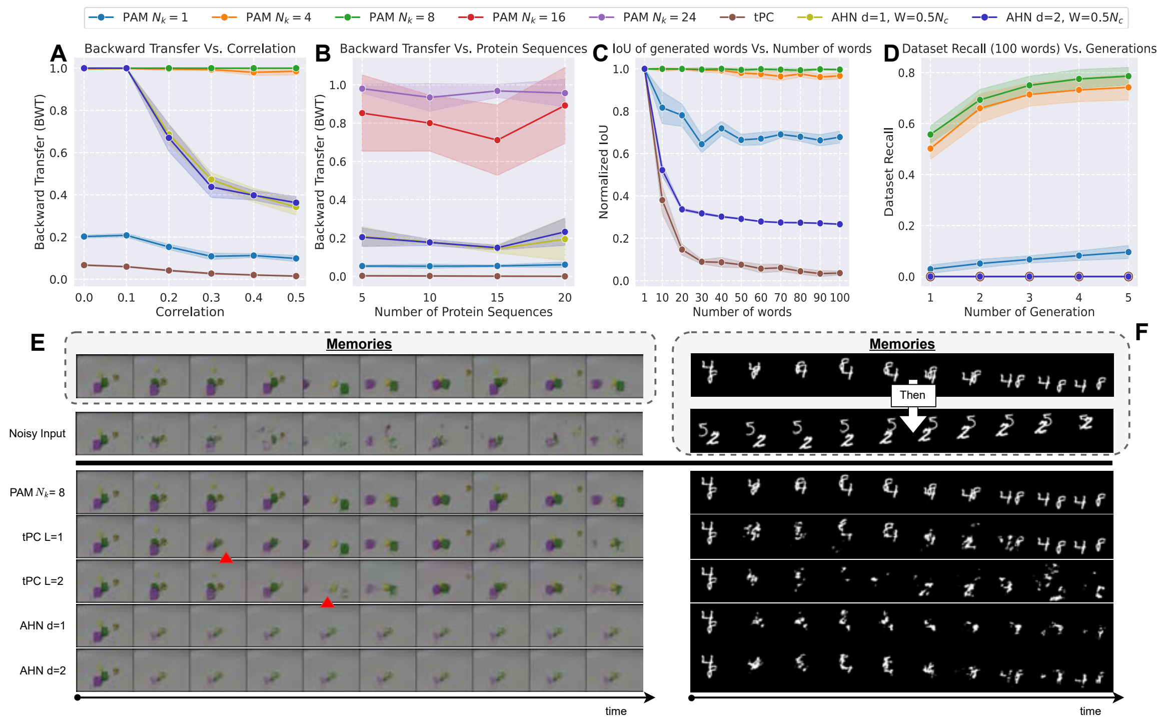


Figure 5.4: Additional comparisons on continual learning and multiple possibilities generation. Qualitative results on (*A*) synthetic and (*B*) protein sequences backward transfer, and (*C*-*D*) multiple possibilities generation on text datasets. Qualitative results on (*E*) noise robustness on CLEVRER sequence, and (*F*) catastrophic forgetting on Moving MNIST dataset. ▲ highlights the first frame with significant error. The mean and standard deviation of 10 trials are reported for all plots.

5.3.2.2 Catastrophic Forgetting

To asses the model's performance in a continual learning setup, we sequentially train each model on multiple sequences and compute the Backward Transfer (BWT) [149] metric by reporting the average normalized IoU on previously learned sequences after learning a new one. In Figure 5.4 *A*, we report BWT for 50 synthetically generated sequences with varying correlation. AHN can avoid catastrophic forgetting when the patterns are not correlated, whereas tPC fails to retain previous knowledge regardless of the correlation value. PAM, with high enough context , does not overwrite or forget previously learned sequences after learning new ones but performs poorly when , as expected. We repeat the experiment on more challenging sequences from ProteinNet [10], which contains highly correlated ( ), and long, sequences (details in appendix). The results in Figure 5.4 *B* show a similar trend with PAM requiring more context neurons to handle the more challenging data. Qualitative results on moving-MNIST [236] in Figure 5.4 *F* further demonstrate the catastrophic forgetting challenge where the learning of the second sequence overwrites the learned sequence. PAM successfully retrieves the previously learned sequence while the other models fail.

5.3.2.3 Multiple Possibilities Generation

In addition to accurately encoding input contexts, PAM is designed to represent multiple valid possibilities and sample a single possibility. We perform evaluation on a dataset of four-letter English words (details in appendix), which includes many possible future completions (e.g., “th[is]”, “th[at]”, “th[em]”, etc.) We train PAM on the list of letters sequentially (i.e., one word at a time); the other methods are trained in a simpler batched setup as in [241] because they suffer from catastrophic forgetting. This puts PAM at a disadvantage, but as shown in Figure 5.4 *C*, PAM still significantly outperforms the other methods in accurately generating valid words (high average IoU) in an offline manner. Both tPC and AHN fail to generate meaningful words when trained on sequences with multiple future possibilities. Figure 5.4 *D* further demonstrates the stochastic generative property of PAM. We show PAM’s ability to recall more of the dataset as it repeats the generation process, whereas PC and AHN fail in the dataset recall task.

5.3.2.4 Online Noise Robustness

The online generation ability of PAM shown in Figure 5.2 allows the model to use the learned attractors to clean the noisy observations before using them as inputs to the predictor. This step allows the model to use its prior belief about future observations to modify the noisy inputs. In Figure 5.3 *C*, we evaluate the models' performances by changing a percentage of the active bits during online generation. PAM is able to completely replace the noisy input with its prior belief if it does not exist in the predicted set of possibilities . In contrast, the other methods use the noisy inputs, thus hindering their performances. We provide qualitative results on CLEVRER [273] in Figure 5.4 *E*; PAM retrieves the original sequence despite getting noisy inputs (40% noise), and outperforms the other models. Interestingly, tPC performs reasonably well on this task despite the added noise.

5.3.2.5 Efficiency

We report, in Figure 5.3 *D*, the time each model requires to learn and recall a sequence. For this study, we use input size and vary the sequence length. PAM operates entirely on *CPU*. The results show that a single-layer tPC model requires more time than all PAM variants ( ). Additionally, a two-layered tPC requires two to three orders of magnitude more time than PAM or single-layered tPC, significantly limiting its scalability and practicality when applied to real data with long sequences.

Chapter 6: Conclusion

"We are all agreed that your theory is crazy. The question which divides us is whether it is crazy enough to have a chance of being correct."

— Niels Bohr

In this dissertation, we have explored several predictive learning frameworks to advance hierarchical event segmentation, representation learning, and sequential memory models. At the core of these works lies a common principle of leveraging predictive mechanisms to process streaming data efficiently. We have addressed challenges across various domains, from video event understanding to biologically inspired memory models.

We first introduced a stream learning framework that segments long sequences of activities into discrete events and spatially segments objects in video frames. This predictive approach not only models the temporal dependencies between events but also segments data at different granularities through a gating mechanism on the loss signal. By anticipating patterns in the sequence and adjusting thresholds dynamically, our model adapts to varying temporal resolutions. We introduced a new, long-duration wildlife monitoring dataset showcasing how predictive learning can handle the complexities of real-world, unstructured data. Our approach, particularly in the context of ethogramming, has the potential to significantly reduce the manual labor required for annotating datasets, emphasizing the practical utility of predictive event segmentation framework.

Next, we presented STREAMER, a self-supervised hierarchical model that learns event representations from streaming data by predicting future inputs. STREAMER builds on the idea of temporal hierarchies, where nested events are represented at multiple levels of granularity and abstraction. Through a gradient scaling mechanism and biologically-inspired learning constraints, the model effectively learns from and predicts long-term temporal dependencies. STREAMER processes each input in a streaming manner, predicting and segmenting events as they unfold. This streaming capability makes the model well-suited for applications in dynamic environments, such as egocentric video analysis, where the ability to predict and react to unfolding events is critical.

Finally, we proposed PAM, a generative model rooted in predictive learning principles. PAM draws on theories of cognition and neuroscience, predicting multiple future outcomes by representing possibilities as a union of sparse distributed representations (SDRs). This predictive capacity allows PAM to model several potential continuations of a sequence, overcoming challenges such as sequence correlation and noise. Importantly, PAM does not suffer from catastrophic forgetting, a common issue in sequential learning, enabling the model to maintain predictions over multiple sequences. PAM's predictive framework offers a scalable, biologically plausible solution to generative modeling, with potential future applications in hierarchical sensory processing and higher-order predictive models.

6.1 Limitations

While the predictive methods presented in this dissertation demonstrate significant advancements in event segmentation, representation learning, and sequential memory, they also come with several limitations that are important to address. These limitations highlight both the current boundaries of the models and potential areas for future development.

One key limitation of the single-layer predictive approach is its inability to label the events and objects segmented from videos. While these models are capable of detecting temporal events and spatially localizing objects within frames, they do not provide any labels for these events or objects. As a result, when multiple objects are present in an event, the model outputs a single, combined mask (i.e., the union of individual masks) to represent all objects involved. This limitation prevents the models from performing instance segmentation, where differentiating between multiple objects in the same scene is necessary. Another limitation is the reliance on motion cues for learning object representations. While the models are flexible enough to detect objects even when they are not moving, the initial learning process still depends heavily on motion as a signal to segment objects from their background. This reliance could hinder performance in cases where objects or events are entirely static. As a result, the current approach might struggle with scenarios where key objects are present but do not exhibit any movement over time.

STREAMER, while performing well at representation learning and hierarchical event segmentation in streaming environments, requires large amounts of streaming data to effectively model complex, high-level causal structures. The training time increases as additional layers are introduced to the model, which can be computationally expensive. Moreover, although STREAMER is designed for online learning from large, unlabeled datasets, it remains highly dependent on the availability of such data to function optimally. The need for substantial amounts of data could limit its scalability in more resource-constrained settings.

PAM, the biologically plausible sequential memory model, also has its limitations. PAM's reliance on sparse distributed representations (SDRs) means it cannot directly process dense input data, such as raw images. Instead, PAM requires inputs to be encoded into sparse binary representations before learning. This limitation prevents the model from being directly applied to input spaces like images without preprocessing. While SDRs are argued to be a more biologically realistic approach for cognitive models, they restrict PAM's application to domains where such representations can be easily constructed. Future work will need to explore methods for encoding high-dimensional inputs, such as images, into SDRs that retain essential compositional structures.

6.2 Broader Impact

The work presented in this dissertation has the potential to significantly impact a wide range of fields, from machine learning and computer vision to theoretical and computational neuroscience. By advancing predictive methods for hierarchical segmentation, representation learning, and generative modeling, the research not only addresses technical challenges in these domains but also opens new avenues for applying these techniques to real-world problems.

One of the most immediate and tangible impacts of this work lies in its potential to transform how large-scale video data processing is handled. The ability to segment and categorize events in streaming data has broad applications, ranging from automated wildlife monitoring to egocentric video analysis. For instance, the stream learning framework introduced here can dramatically reduce the manual labor required to annotate vast amounts of behavioral data, particularly in fields like ethology, where understanding animal behavior is crucial for ecological studies. This automation of event detection and segmentation in videos could facilitate more efficient monitoring of species in their natural habitats, contributing to conservation efforts and ecological research.

The broader significance of the STREAMER model lies in its ability to discover hierarchical, nested structures in data, beyond just its streaming capabilities. While this dissertation focuses on its application for segmenting temporal events, the same hierarchical predictive approach can be extended to other types of data. For example, STREAMER could be used to detect part-whole relationships in visual scenes, identifying how objects are composed of smaller components. In audio, it could uncover nested phonetic structures, and in natural language processing, it could identify the compositional hierarchy of phrases and sentences. What makes this approach powerful is its flexibility: by applying a simple predictive objective, STREAMER can learn across different data types and build connections between them at multiple levels of abstraction. This ability to bridge different modalities and uncover compositional structures is a major step toward addressing one of artificial intelligence's long-standing challenges.

Moreover, this hierarchical approach inherently improves the explainability and interpretability of AI models. By breaking down complex data into more understandable sub-components and identifying patterns at different levels, STREAMER provides clearer insights into how the model makes decisions, making it easier for users to trace the reasoning behind its predictions. This enhanced interpretability is crucial for building trust in AI systems, especially in critical applications.

On a more theoretical level, the biologically inspired aspects of the models developed in this work, such as PAM, offer valuable insights into cognitive processes. By modeling how the brain might predict and encode sequential events without catastrophic forgetting, these frameworks have the potential to contribute to research in computational neuroscience and artificial intelligence. PAM's use of Sparse Distributed Representations (SDRs) mimics neural activity and suggests that efficient, scalable memory systems can be developed based on biologically plausible learning principles. This insight could inform future research on how to develop AI systems that are more robust, efficient, and capable of learning across longer timescales without degradation in performance.

Moreover, the integration of predictive learning across these different models highlights the versatility of prediction as a core mechanism in intelligent systems. Prediction underlies many aspects of human cognition — such as perception, decision-making, and memory — and the models developed here offer a solution to bringing similar predictive capabilities into artificial systems.

6.3 Ethical Considerations

As with any advanced technological development, the predictive models and methods presented in this dissertation raise several important ethical considerations that must be carefully addressed, particularly given their potential applications in sensitive areas such as surveillance and behavior monitoring. The ability of these models to segment events, predict future outcomes, and generate representations from streaming data introduces both significant opportunities and risks, particularly in how these technologies might impact privacy and fairness.

6.3.1 Privacy Concerns

One of the most pressing ethical concerns arises from the use of these predictive models in domains where personal or sensitive data is involved, such as surveillance systems or egocentric video analysis. The models described in this dissertation have the ability to process and analyze vast amounts of video data in real time, detecting and segmenting events as they unfold. While this capability is valuable for applications like security or wildlife monitoring, it also raises concerns about the invasion of privacy, particularly when applied to human subjects. The automatic detection of events in public or private spaces without explicit consent could lead to misuse or overreach, especially in cases where individuals are not aware they are being monitored. Additionally, as these models evolve to handle multi-modal data (e.g., integrating audio or biometric data), the potential for privacy violations increases.

However, the privacy concerns traditionally associated with such models are significantly mitigated by the nature of the models proposed in this dissertation. All of the models are trained in a streaming fashion, which means they process data continuously and without storing the input data after it has been processed. This approach inherently lowers privacy risks by reducing the possibility of sensitive information being stored or re-used inappropriately. Unlike batch learning models that often retain entire datasets for retraining or validation, the models developed here do not maintain any explicit record of the data they process.

To further mitigate these risks, future implementations of these models must be accompanied by robust privacy protections. This could involve designing systems that anonymize data, use ethical guidelines for video collection, and provide transparency to individuals about how their data is being used. Furthermore, regulatory frameworks should be established to ensure that predictive models are deployed responsibly, particularly in areas involving human surveillance.

6.3.2 Bias and Fairness

Another significant ethical consideration is the potential for bias in the predictive algorithms. While the models in this dissertation are designed to be adaptive and generalizable, they rely heavily on the data they are trained on. In domains like wildlife monitoring, this might not pose a significant issue. However, in applications involving human subjects, such as surveillance or behavioral analysis, the risk of bias becomes more pronounced. Predictive models can inadvertently inherit biases from the datasets they are trained on, leading to skewed outcomes in event detection or segmentation, particularly if the training data does not adequately represent diverse populations or behaviors.

For example, in an egocentric video analysis context, if the model is predominantly trained on data from a specific demographic or environment, it may perform less accurately when applied to different demographics or settings, reinforcing societal biases. If those datasets are not sufficiently diverse, the model's predictions could disproportionately benefit or disadvantage certain groups. Addressing this concern requires careful curation of training data, ongoing evaluation of model performance across diverse groups, and potentially incorporating fairness constraints into the design of future predictive algorithms.

6.3.3 Long-Term Societal Impact

Finally, as predictive technologies become increasingly embedded in the infrastructure of daily life, it is important to consider their long-term societal impacts. The advancements in predictive learning explored in this dissertation could contribute to large-scale societal shifts, such as increasing reliance on automated systems, changes in labor markets, and shifts in how surveillance is conducted. While automation of processes like video analysis or behavioral monitoring can bring efficiencies, it could also lead to job displacement in fields where manual monitoring is currently the norm.

References

[1] Sathyanarayanan N Aakur and Sudeep Sarkar. “A Perceptual Prediction Framework for Self Supervised Event Segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1197–1206.

[2] Sathyanarayanan Aakur and Sudeep Sarkar. “Action localization through continual predictive learning”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 300–317.

[3] *ActEV: Activities in Extended Video*. URL: https://actev.nist.gov/.

[4] Valentin Afraimovich, Xue Gong, and Mikhail Rabinovich. “Sequential memory: Binding dynamics”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 25.10 (2015).

[5] Subutai Ahmad and Jeff Hawkins. “How do neurons operate on sparse distributed representations? A mathematical theory of sparsity, neurons and active dendrites”. In: *arXiv preprint arXiv:1601.00720* 10 (2016).

[6] Subutai Ahmad and Jeff Hawkins. “Properties of sparse distributed representations and their application to hierarchical temporal memory”. In: *arXiv preprint arXiv:1503.07469* (2015).

[7] Subutai Ahmad and Luiz Scheinkman. “How can we be so dense? the benefits of using highly sparse representations”. In: *arXiv preprint arXiv:1903.11257* (2019).

[8] Anurag Ajay et al. “Compositional foundation models for hierarchical planning”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[9] Hüseyin Gökhan Akçay et al. “Automated bird counting with deep learning for regional bird distribution mapping”. In: *Animals* 10.7 (2020), p. 1207.

[10] Mohammed AlQuraishi. “ProteinNet: a standardized data set for machine learning of protein structure”. In: *BMC bioinformatics* 20 (2019), pp. 1–10.

[11] Jean-Baptiste Alayrac et al. “Unsupervised learning from narrated instruction videos”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4575–4583.

[12] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. “Task-free continual learning”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 11254–11263.

[13] Evlampios Apostolidis et al. “Combining Global and Local Attention with Positional Encoding for Video Summarization”. In: *2021 IEEE International Symposium on Multimedia (ISM)*. Ieee. 2021, pp. 226–234.

[14] Anurag Arnab et al. “ViViT: A Video Vision Transformer”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 6836–6846.

[15] Mahmoud Assran et al. “Self-supervised learning from images with a joint-embedding predictive architecture”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 15619–15629.

[16] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint arXiv:1409.0473* (2014).

[17] Adrien Bardes, Jean Ponce, and Yann LeCun. “Mc-jepa: A joint-embedding predictive architecture for self-supervised learning of motion and content features”. In: *arXiv preprint arXiv:2307.12698* (2023).

[18] Adrien Bardes, Jean Ponce, and Yann LeCun. “VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning”. In: *International Conference on Learning Representations*. 2022.

[19] Adrien Bardes et al. “V-JEPA: Latent Video Prediction for Visual Representation Learning”. In: (2023).

[20] András A Benczúr, Levente Kocsis, and Róbert Pálovics. “Online machine learning in big data streams”. In: *arXiv preprint arXiv:1802.05872* (2018).

[21] David Beniaguev, Idan Segev, and Michael London. “Single cortical neurons as deep artificial neural networks”. In: *Neuron* 109.17 (2021), pp. 2727–2739.

[22] Bharat Lal Bhatnagar et al. “Unsupervised Learning of Deep Feature Representation for Clustering Egocentric Actions.” In: *Ijcai*. 2017, pp. 1447–1453.

[23] Piotr Bojanowski et al. “Weakly supervised action labeling in videos under ordering constraints”. In: *European Conference on Computer Vision*. Springer. 2014, pp. 628–643.

[24] Elizabeth Bondi et al. “BIRDSAI: A Dataset for Detection and Tracking in Aerial Thermal Infrared Videos”. In: *Wacv*. 2020.

[25] Léon Bottou. “Large-scale machine learning with stochastic gradient descent”. In: *Proceedings of COMPSTAT’2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*. Springer. 2010, pp. 177–186.

[26] Tom B. Brown et al. “Language Models are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems 33 (NeurIPS)*. Vol. 33. 2020, pp. 1877–1901.

[27] Neil Burgess, JL Shapiro, and MA Moore. “Neural network models of list learning”. In: *Network: Computation in Neural Systems* 2.4 (1991), pp. 399–422.

[28] Xavier P. Burgos-Artizzu et al. “Social behavior recognition in continuous video”. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 1322–1329. DOI: 10.1109/cvpr.2012.6247817.

[29] Mathilde Caron et al. “Emerging Properties in Self-Supervised Vision Transformers”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 9650–9660.

[30] Hamza Chaudhry et al. “Long sequence Hopfield memory”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[31] Xinlei Chen and Kaiming He. “Exploring simple siamese representation learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 15750–15758.

[32] Kyunghyun Cho et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).

[33] Melanie Clapham et al. “Automated facial recognition for wildlife that lack unique markings: A deep learning approach for brown bears”. In: *Ecology and evolution* 10.23 (2020), pp. 12883–12892.

[34] Andy Clark. “Whatever next? Predictive brains, situated agents, and the future of cognitive science”. In: *Behavioral and brain sciences* 36.3 (2013), pp. 181–204.

[35] Martin A Conway. “Episodic memories”. In: *Neuropsychologia* 47.11 (2009), pp. 2305–2313.

[36] Kellie Corona et al. “Meva: A large-scale multiview, multimodal video dataset for activity detection”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021, pp. 1060–1068.

[37] Corinna Cortes. “Support-Vector Networks”. In: *Machine Learning* (1995).

[38] Kenneth James Williams Craik. *The nature of explanation*. Vol. 445. CUP Archive, 1967.

[39] Francis Crick. “The recent excitement about neural networks.” In: *Nature* 337.6203 (1989), pp. 129–132.

[40] Zihang Dai et al. “Transformer-xl: Attentive language models beyond a fixed-length context”. In: *arXiv preprint arXiv:1901.02860* (2019).

[41] Dima Damen et al. “Scaling egocentric vision: The epic-kitchens dataset”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 720–736.

[42] Anna Dawid and Yann LeCun. “Introduction to latent variable energy-based models: A path towards autonomous machine intelligence”. In: *arXiv preprint arXiv:2306.02572* (2023).

[43] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[44] Li Deng. “The mnist database of handwritten digit images for machine learning research”. In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[45] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *North American Chapter of the Association for Computational Linguistics*. 2019. URL: https://api.semanticscholar.org/CorpusID:52967399.

[46] James J DiCarlo and David D Cox. “Untangling invariant object recognition”. In: *Trends in cognitive sciences* 11.8 (2007), pp. 333–341.

[47] Li Ding and Chenliang Xu. “Weakly-supervised action segmentation with iterative soft boundary assignment”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6508–6516.

[48] Shibhansh Dohare et al. “Loss of plasticity in deep continual learning”. In: *Nature* 632.8026 (2024), pp. 768–774.

[49] Alexey Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *International Conference on Learning Representations*. 2020.

[50] Zexing Du et al. “Fast and unsupervised action boundary detection for action segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 3323–3332.

[51] Sai Kumar Dwivedi et al. “Learning to regress bodies from images using differentiable semantic rendering”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 11250–11259.

[52] Peter Elias. “Predictive coding–I”. In: *IRE transactions on information theory* 1.1 (1955), pp. 16–24.

[53] Patrick Esser, Robin Rombach, and Bjorn Ommer. “Taming transformers for high-resolution image synthesis”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 12873–12883.

[54] Mark Everingham et al. “The pascal visual object classes (voc) challenge”. In: *International journal of computer vision* 88.2 (2010), pp. 303–338.

[55] Jiri Fajtl et al. “Summarizing videos with attention”. In: *Asian Conference on Computer Vision*. Springer. 2018, pp. 39–54.

[56] Abassin Sourou Fangbemi et al. “Zoobuilder: 2d and 3d pose estimation for quadrupeds using synthetic data”. In: *arXiv preprint arXiv:2009.05389* (2020).

[57] Oleg V Favorov and Mathew E Diamond. “Demonstration of discrete place-defined columns segregates in the cat SI”. In: *Journal of Comparative Neurology* 298.1 (1990), pp. 97–112.

[58] André C Ferreira et al. “Deep learning-based methods for individual recognition in small birds”. In: *Methods in Ecology and Evolution* 11.9 (2020), pp. 1072–1085.

[59] Peter Foldiak. “Sparse coding in the primate cortex”. In: *The handbook of brain theory and neural networks* (2003).

[60] Robert M French. “Catastrophic forgetting in connectionist networks”. In: *Trends in cognitive sciences* 3.4 (1999), pp. 128–135.

[61] Karl J Friston, N Trujillo-Barreto, and Jean Daunizeau. “DEM: a variational treatment of dynamic systems”. In: *Neuroimage* 41.3 (2008), pp. 849–885.

[62] Karl Friston. “Does predictive coding have a future?” In: *Nature neuroscience* 21.8 (2018), pp. 1019–1021.

[63] Karl Friston. “The free-energy principle: a unified brain theory?” In: *Nature reviews neuroscience* 11.2 (2010), pp. 127–138.

[64] Karl Friston and Stefan Kiebel. “Predictive coding under the free-energy principle”. In: *Philosophical transactions of the Royal Society B: Biological sciences* 364.1521 (2009), pp. 1211–1221.

[65] Karl Friston et al. “Variational free energy and the Laplace approximation”. In: *Neuroimage* 34.1 (2007), pp. 220–234.

[66] Kunihiko Fukushima. “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position”. In: *Biological cybernetics* 36.4 (1980), pp. 193–202.

[67] João Gama et al. “A survey on concept drift adaptation”. In: *ACM computing surveys (CSUR)* 46.4 (2014), pp. 1–37.

[68] Saeed Ghadimi and Guanghui Lan. “Stochastic first-and zeroth-order methods for nonconvex stochastic programming”. In: *SIAM journal on optimization* 23.4 (2013), pp. 2341–2368.

[69] Gabriel Goh et al. “Multimodal neurons in artificial neural networks”. In: *Distill* 6.3 (2021), e30.

[70] Ian Goodfellow, Patrick McDaniel, and Nicolas Papernot. “Making machine learning robust against adversarial inputs”. In: *Communications of the ACM* 61.7 (2018), pp. 56–66.

[71] Ian Goodfellow et al. “Generative adversarial networks”. In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[72] Kristen Grauman et al. “Ego4d: Around the world in 3,000 hours of egocentric video”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 18995–19012.

[73] Roddy M Grieves and Kate J Jeffery. “The representation of space in the brain”. In: *Behavioural processes* 135 (2017), pp. 113–131.

[74] Jean-Bastien Grill et al. “Bootstrap your own latent-a new approach to self-supervised learning”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 21271–21284.

[75] Roman Gula et al. “An audio/video surveillance system for wildlife”. In: *European Journal of Wildlife Research* 56 (2010), pp. 803–807.

[76] Semih Günel et al. “DeepFly3D, a deep learning-based approach for 3D limb and appendage tracking in tethered, adult *Drosophila*”. In: *eLife* (2019).

[77] Torkel Hafting et al. “Microstructure of a spatial map in the entorhinal cortex”. In: *Nature* 436.7052 (2005), pp. 801–806.

[78] Patrick Haggard. “Planning of action sequences”. In: *Acta Psychologica* 99.2 (1998), pp. 201–215.

[79] Kuan Han et al. “Deep predictive coding network with local recurrent processing for object recognition”. In: *Advances in neural information processing systems* 31 (2018).

[80] Tengda Han, Weidi Xie, and Andrew Zisserman. “Video representation learning by dense predictive coding”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019, pp. 0–0.

[81] James Harrison et al. “Continuous meta-learning without tasks”. In: *Advances in neural information processing systems* 33 (2020), pp. 17571–17581.

[82] Jeff Hawkins. *A thousand brains: A new theory of intelligence*. Basic Books, 2021.

[83] Jeff Hawkins and Subutai Ahmad. “Why neurons have thousands of synapses, a theory of sequence memory in neocortex”. In: *Frontiers in neural circuits* 10 (2016), p. 23.

[84] Jeff Hawkins, Subutai Ahmad, and Yuwei Cui. “A theory of how columns in the neocortex enable learning the structure of the world”. In: *Frontiers in neural circuits* 11 (2017), p. 81.

[85] Jeff Hawkins and Sandra Blakeslee. *On intelligence*. Macmillan, 2004.

[86] Jeff Hawkins et al. “A framework for intelligence and cortical function based on grid cells in the neocortex”. In: *Frontiers in neural circuits* 12 (2019), p. 431889.

[87] Wayne D Hawkins and Sarah E DuRant. “Applications of machine learning in behavioral ecology: Quantifying avian incubation behavior and nest conditions in relation to environmental temperature”. In: *Plos one* 15.8 (2020), e0236925.

[88] Benjamin Y Hayden, Hyun Soo Park, and Jan Zimmermann. “Automated pose estimation in primates”. In: *American journal of primatology* (2021), e23348.

[89] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[90] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology press, 2005.

[91] David J Heeger. “Theory of cortical function”. In: *Proceedings of the National Academy of Sciences* 114.8 (2017), pp. 1773–1782.

[92] Nikolas Hesse et al. “Learning and tracking the 3D body shape of freely moving infants from RGB-D sequences”. In: *IEEE transactions on pattern analysis and machine intelligence* 42.10 (2019), pp. 2540–2551.

[93] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. “Transforming auto-encoders”. In: *Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21*. Springer. 2011, pp. 44–51.

[94] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. “Matrix capsules with EM routing”. In: *International conference on learning representations*. 2018.

[95] Geoffrey Hinton. “Distilling the Knowledge in a Neural Network”. In: *arXiv preprint arXiv:1503.02531* (2015).

[96] Geoffrey Hinton. “How to represent part-whole hierarchies in a neural network”. In: *Neural Computation* (2022), pp. 1–40.

[97] Geoffrey Hinton. “Some demonstrations of the effects of structural descriptions in mental imagery”. In: *Cognitive Science* 3.3 (1979), pp. 231–250.

[98] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[99] Tin Kam Ho. “Random decision forests”. In: *Proceedings of 3rd international conference on document analysis and recognition*. Vol. 1. IEEE. 1995, pp. 278–282.

[100] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[101] Kjell Jørgen Hole and Subutai Ahmad. “A thousand brains: toward biologically constrained ai”. In: *SN Applied Sciences* 3.8 (2021), p. 743.

[102] Jason Holmberg, Bradley Norman, and Zaven Arzoumanian. “Estimating population size, structure, and residency time for whale sharks Rhincodon typus through collaborative photo-identification”. In: *Endangered Species Research* 7.1 (2009), pp. 39–53.

[103] John J Hopfield. “Neural networks and physical systems with emergent collective computational abilities.” In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.

[104] Toshihiko Hosoya, Stephen A Baccus, and Markus Meister. “Dynamic predictive coding by the retina”. In: *Nature* 436.7047 (2005), pp. 71–77.

[105] De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. “Connectionist temporal modeling for weakly supervised action labeling”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 137–153.

[106] David H Hubel and Torsten N Wiesel. “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex”. In: *The Journal of physiology* 160.1 (1962), p. 106.

[107] Abhiram Iyer et al. “Avoiding catastrophe: Active dendrites enable multi-task learning in dynamic environments”. In: *Frontiers in neurorobotics* 16 (2022), p. 846219.

[108] Zhong Ji et al. “Video summarization with attention-based encoder–decoder networks”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 30.6 (2019), pp. 1709–1717.

[109] John Jumper et al. “Highly accurate protein structure prediction with AlphaFold”. In: *nature* 596.7873 (2021), pp. 583–589.

[110] Jon H Kaas. “The functional organization of somatosensory cortex in primates”. In: *Annals of Anatomy-Anatomischer Anzeiger* 175.6 (1993), pp. 509–518.

[111] Hiroshi Kage. “Implementing associative memories by Echo State Network for the applications of natural language processing”. In: *Machine Learning with Applications* 11 (2023), p. 100449.

[112] Angjoo Kanazawa et al. “Learning 3D Deformation of Animals from 2D Images”. In: *Comput. Graph. Forum* 35.2 (May 2016), pp. 365–374. ISSN: 0167-7055. DOI: 10.1111/cgf.12838. URL: https://doi.org/10.1111/cgf.12838.

[113] Pentti Kanerva. *Sparse distributed memory*. MIT press, 1988.

[114] Hyolim Kang et al. “Winning the CVPR’2021 Kinetics-GEBD Challenge: Contrastive Learning Approach”. In: *arXiv preprint arXiv:2106.11549* (2021).

[115] Ekkehard M Kasper et al. “Pyramidal neurons in layer 5 of the rat visual cortex. II. Development of electrophysiological properties”. In: *Journal of Comparative Neurology* 339.4 (1994), pp. 475–494.

[116] Benjamin Kellenberger et al. “21 000 birds in 4.5 h: efficient large-scale seabird detection with machine learning”. In: *Remote Sensing in Ecology and Conservation* (2021).

[117] Diederik P Kingma. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[118] Thomas N Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: *International Conference on Learning Representations*. 2017.

[119] James Kirkpatrick et al. “Overcoming catastrophic forgetting in neural networks”. In: *Proceedings of the national academy of sciences* 114.13 (2017), pp. 3521–3526.

[120] Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. “Vibe: Video inference for human body pose and shape estimation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 5253–5263.

[121] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[122] Bart Kosko. “Bidirectional associative memories”. In: *IEEE Transactions on Systems, man, and Cybernetics* 18.1 (1988), pp. 49–60.

[123] Moritz Köster et al. “Making sense of the world: infant learning from a predictive processing perspective”. In: *Perspectives on psychological science* 15.3 (2020), pp. 562–571.

[124] Alex Krizhevsky, Geoffrey Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009).

[125] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012).

[126] Dmitry Krotov and John J Hopfield. “Dense associative memory for pattern recognition”. In: *Advances in neural information processing systems* 29 (2016).

[127] Hilde Kuehne, Ali Arslan, and Thomas Serre. “The language of actions: Recovering the syntax and semantics of goal-directed human activities”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 780–787.

[128] Christopher A Kurby and Jeffrey M Zacks. “Segmentation in the perception and memory of events”. In: *Trends in cognitive sciences* 12.2 (2008), pp. 72–79.

[129] Wang Lam et al. “Muppet: MapReduce-style processing of fast data”. In: *arXiv preprint arXiv:1208.4175* (2012).

[130] Alex M Lamb et al. “Professor forcing: A new algorithm for training recurrent networks”. In: *Advances In Neural Information Processing Systems*. 2016, pp. 4601–4609.

[131] Steffen L Lauritzen. *Graphical models*. Vol. 17. Clarendon Press, 1996.

[132] Yann LeCun. “A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27”. In: *Open Review* 62 (2022).

[133] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[134] Colin Lea et al. “Segmental spatiotemporal cnns for fine-grained action segmentation”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 36–52.

[135] Colin Lea et al. “Temporal convolutional networks for action segmentation and detection”. In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 156–165.

[136] Niels Leadholm, Marcus Lewis, and Subutai Ahmad. “Grid cell path integration for movement-based visual object recognition”. In: *arXiv preprint arXiv:2102.09076* (2021).

[137] Stephan Lewandowsky and Shu-Chen Li. “Catastrophic interference in neural networks: Causes, solutions, and data”. In: *Interference and inhibition in cognition*. Elsevier, 1995, pp. 329–361.

[138] Marcus Lewis et al. “Locations in the neocortex: a theory of sensorimotor object recognition using cortical grid cells”. In: *Frontiers in neural circuits* 13 (2019), p. 22.

[139] Shuyuan Li et al. “ATRW: a benchmark for Amur tiger re-identification in the wild”. In: *arXiv preprint arXiv:1906.05586* (2019).

[140] Siyuan Li et al. “Deformation-aware unpaired image translation for pose estimation on laboratory animals”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 13158–13168.

[141] Yanghao Li et al. “Improved multiscale vision transformers for classification and detection”. In: *arXiv preprint arXiv:2112.01526* (2021).

[142] Zhizhong Li and Derek Hoiem. “Learning without forgetting”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.12 (2017), pp. 2935–2947.

[143] Timothy P Lillicrap et al. “Backpropagation and the brain”. In: *Nature Reviews Neuroscience* 21.6 (2020), pp. 335–346.

[144] Tianwei Lin et al. “BMN: Boundary-matching network for temporal action proposal generation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019.

[145] Tianwei Lin et al. “BSN: Boundary Sensitive Network for Temporal Action Proposal Generation”. In: *European conference on computer vision*. 2018.

[146] Yuanze Lin, Xun Guo, and Yan Lu. “Self-supervised video representation learning with meta-contrastive network”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 8239–8249.

[147] Ze Liu et al. “Swin Transformer V2: Scaling Up Capacity and Resolution”. In: *arXiv preprint arXiv:2111.09883* (2021).

[148] Michael London and Michael Häusser. “Dendritic computation”. In: *Annu. Rev. Neurosci.* 28.1 (2005), pp. 503–532.

[149] David Lopez-Paz and Marc’Aurelio Ranzato. “Gradient episodic memory for continual learning”. In: *Advances in neural information processing systems* 30 (2017).

[150] Malte Lorbach et al. "Learning to recognize rat social behavior: Novel dataset and cross-dataset application". In: *Journal of Neuroscience Methods* 300 (2018). Measuring Behaviour 2016, pp. 166–172. ISSN: 0165-0270. DOI: https://doi.org/10.1016/j.jneumeth.2017.05.006. URL: https://www.sciencedirect.com/science/article/pii/S0165027017301255.

[151] Lester C Loschky et al. "The scene perception & event comprehension theory (SPECT) applied to visual narratives". In: *Topics in cognitive science* 12.1 (2020), pp. 311–351.

[152] William Lotter, Gabriel Kreiman, and David Cox. "Deep Predictive Coding Networks for Video Prediction and Unsupervised Learning". In: *International Conference on Learning Representations*. OpenReview.net, 2017.

[153] Guy Major, Matthew E Larkum, and Jackie Schiller. "Active properties of neocortical pyramidal neuron dendrites". In: *Annual review of neuroscience* 36.1 (2013), pp. 1–24.

[154] Jonathan Malmaud et al. "What's Cookin'? Interpreting Cooking Videos using Text, Speech and Vision". In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2015.

[155] Joseph R Manns and Howard Eichenbaum. "Evolution of declarative memory". In: *Hippocampus* 16.9 (2006), pp. 795–808.

[156] Mackenzie Weygandt Mathis and Alexander Mathis. "Deep learning tools for the measurement of animal behavior in neuroscience". In: *Current opinion in neurobiology* 60 (2020), pp. 1–11.

[157] Stewart M McCauley and Morten H Christiansen. "Computational investigations of multi-word chunks in language learning". In: *Topics in Cognitive Science* 9.3 (2017), pp. 637–652.

[158] Michael McCloskey and Neal J Cohen. "Catastrophic interference in connectionist networks: The sequential learning problem". In: *Psychology of learning and motivation*. Vol. 24. Elsevier, 1989, pp. 109–165.

[159] Warren S McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5 (1943), pp. 115–133.

[160] Declan McIntosh et al. “Movement Tracks for the Automatic Detection of Fish Behavior in Videos”. In: *arXiv preprint arXiv:2011.14070* (2020).

[161] Katherine Metcalf and David Leake. “Modeling Unsupervised Event Segmentation: Learning Event Boundaries from Prediction Errors.” In: *CogSci*. 2017.

[162] Beren Millidge et al. “Universal hopfield networks: A general framework for single-shot associative memory models”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 15561–15583.

[163] Ana Garcia del Molino, Joo-Hwee Lim, and Ah-Hwee Tan. “Predicting visual context for unsupervised event segmentation in continuous photo-streams”. In: *Proceedings of the 26th ACM international conference on Multimedia*. 2018, pp. 10–17.

[164] Hans Moravec. “Mind Children: The Future of Robot and Human Intelligence”. In: *Harvard UP* (1988).

[165] Ramy Mounir, Sathyanarayanan Aakur, and Sudeep Sarkar. “Self-supervised temporal event segmentation inspired by cognitive theories”. In: *Advanced Methods and Deep Learning in Computer Vision*. Elsevier, 2022, pp. 405–448.

[166] Ramy Mounir and Sudeep Sarkar. “Predictive Attractor Models”. In: *Advances in Neural Information Processing Systems* 37 (2025).

[167] Ramy Mounir, Sujal Vijayaraghavan, and Sudeep Sarkar. “STREAMER: Streaming representation learning and event segmentation in a hierarchical manner”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[168] Ramy Mounir et al. “Spatio-temporal event segmentation for wildlife extended videos”. In: *International Conference on Computer Vision and Image Processing*. Springer. 2021, pp. 48–59.

[169] Ramy Mounir et al. “Towards Automated Ethogramming: Cognitively-Inspired Event Segmentation for Streaming Wildlife Video Monitoring”. In: *International Journal of Computer Vision* (2023), pp. 1–31.

[170] Vernon B Mountcastle. “The columnar organization of the neocortex.” In: *Brain: a journal of neurology* 120.4 (1997), pp. 701–722.

[171] *NOAA Arctic Seals*. URL: https://lila.science/datasets/arcticseals.

[172] JP Nadal et al. “Networks of formal neurons and memory palimpsests”. In: *Europhysics letters* 1.10 (1986), p. 535.

[173] *Noninvasive bee tracking in videos: deep learning algorithms and cloud platform design specifications*. 2021.

[174] J O’Neal. “Entropy coding in speech and television differential PCM systems (Corresp.)” In: *IEEE Transactions on Information Theory* 17.6 (1971), pp. 758–761.

[175] Bruno A Olshausen and David J Field. “Sparse coding of sensory inputs”. In: *Current opinion in neurobiology* 14.4 (2004), pp. 481–487.

[176] Tian Pan et al. “Videomoco: Contrastive video representation learning with temporally adversarial examples”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 11205–11214.

[177] Frank Papenmeier, Alisa Brockhoff, and Markus Huff. “Filling the gap despite full attention: The role of fast backward inferences for event completion”. In: *Cognitive Research: Principles and Implications* 4 (2019), pp. 1–17.

[178] German I Parisi et al. “Continual lifelong learning with neural networks: A review”. In: *Neural networks* 113 (2019), pp. 54–71.

[179] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. “On the difficulty of training recurrent neural networks”. In: *International conference on machine learning*. Pmlr. 2013, pp. 1310–1318.

[180] Malte Pedersen et al. "3d-zef: A 3d zebrafish tracking benchmark dataset". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2426–2436.

[181] Pierre Perruchet et al. "New evidence for chunk-based models in word segmentation". In: *Acta psychologica* 149 (2014), pp. 1–8.

[182] Mathis Petrovich, Michael J Black, and Gül Varol. "Action-conditioned 3d human motion synthesis with transformer vae". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10985–10995.

[183] Panayiota Poirazi, Terrence Brannon, and Bartlett W Mel. "Pyramidal neuron as two-layer neural network". In: *Neuron* 37.6 (2003), pp. 989–999.

[184] Alon Polsky, Bartlett W Mel, and Jackie Schiller. "Computational subunits in thin dendrites of pyramidal cells". In: *Nature neuroscience* 7.6 (2004), pp. 621–627.

[185] Will Price, Carl Vondrick, and Dima Damen. "UnweaveNet: Unweaving Activity Stories". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 13770–13779.

[186] Jielin Qiu, Ge Huang, and Tai Sing Lee. "A Neurally-Inspired Hierarchical Prediction Network for Spatiotemporal Sequence Learning and Prediction". In: *arXiv:1901.09002* (2019).

[187] Lawrence Rabiner and Biinghwang Juang. "An introduction to hidden Markov models". In: *ieee assp magazine* 3.1 (1986), pp. 4–16.

[188] Alec Radford et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[189] Gabriel A Radvansky, Sabine A Krawietz, and Andrea K Tamplin. "Walking through doorways causes forgetting: Further explorations". In: *Quarterly journal of experimental psychology* 64.8 (2011), pp. 1632–1645.

[190] Gabriel A. Radvansky and Jeffrey M. Zacks. “Event Cognition”. In: Oxford University Press, 2014.

[191] Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. “Making gradient descent optimal for strongly convex stochastic optimization”. In: *Proceedings of the 29th International Coference on International Conference on Machine Learning*. 2012, pp. 1571–1578.

[192] Hubert Ramsauer et al. “Hopfield Networks is All You Need”. In: *International Conference on Learning Representations*. 2020.

[193] Rajesh PN Rao. “A sensory–motor theory of the neocortex”. In: *Nature Neuroscience* 27.7 (2024), pp. 1221–1235.

[194] Rajesh PN Rao and Dana H Ballard. “Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects”. In: *Nature neuroscience* 2.1 (1999), pp. 79–87.

[195] Rajesh PN Rao, Dimitrios C Gklezakos, and Vishwas Sathish. “Active predictive coding: A unifying neural model for active perception, compositional learning, and hierarchical planning”. In: *Neural Computation* 36.1 (2023), pp. 1–32.

[196] Roger Ratcliff. “Connectionist models of recognition memory: constraints imposed by learning and forgetting functions.” In: *Psychological review* 97.2 (1990), p. 285.

[197] Malika Nisal Ratnayake, Adrian G Dyer, and Alan Dorin. “Tracking individual honeybees among wildflower clusters with computer vision-facilitated pollinator monitoring”. In: *Plos one* 16.2 (2021), e0239504.

[198] Sylvestre-Alvise Rebuffi et al. “icarl: Incremental classifier and representation learning”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2017, pp. 2001–2010.

[199] Alexander Richard, Hilde Kuehne, and Juergen Gall. “Weakly supervised action learning with rnn based fine-to-coarse modeling”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 754–763.

[200] Herbert Robbins and Sutton Monro. “A stochastic approximation method”. In: *The annals of mathematical statistics* (1951), pp. 400–407.

[201] Domingo S. Rodriguez-Baena et al. “Identifying livestock behavior patterns based on accelerometer dataset”. In: *Journal of Computational Science* 41 (2020), p. 101076. ISSN: 1877-7503. DOI: https://doi.org/10.1016/j.jocs.2020.101076. URL: https://www.sciencedirect.com/science/article/pii/S1877750318311359.

[202] Edmund T Rolls. “A computational theory of episodic memory formation in the hippocampus”. In: *Behavioural brain research* 215.2 (2010), pp. 180–196.

[203] Paul E Rose and Lisa M Riley. “Conducting behavioural research in the zoo: A guide to ten important methods, concepts and theories”. In: *Journal of Zoological and Botanical Gardens* 2.3 (2021), pp. 421–444.

[204] David A Rosenbaum, Sandra B Kenny, and Marcia A Derr. “Hierarchical control of rapid movement sequences.” In: *Journal of Experimental Psychology: Human Perception and Performance* 9.1 (1983), p. 86.

[205] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386.

[206] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), pp. 533–536.

[207] Olga Russakovsky et al. “Imagenet large scale visual recognition challenge”. In: *International journal of computer vision* 115 (2015), pp. 211–252.

[208] Andrei A Rusu et al. “Progressive neural networks”. In: *arXiv preprint arXiv:1606.04671* (2016).

[209] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. “Dynamic routing between capsules”. In: *Advances in neural information processing systems* 30 (2017).

[210] Doyen Sahoo et al. “Online Deep Learning: Learning Deep Neural Networks on the Fly”. In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18* (July 2018).

[211] Tommaso Salvatori et al. “Associative memories via predictive coding”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 3874–3886.

[212] Tommaso Salvatori et al. “Brain-inspired computational intelligence via predictive coding”. In: *arXiv preprint arXiv:2308.07870* (2023).

[213] Tommaso Salvatori et al. “Incremental predictive coding: A parallel and fully automatic learning algorithm”. In: *arXiv preprint arXiv:2212.00720* (2022).

[214] Tommaso Salvatori et al. “Learning on arbitrary graph topologies via predictive coding”. In: *Advances in neural information processing systems* 35 (2022), pp. 38232–38244.

[215] Artsiom Sanakoyeu et al. “Transferring dense pose to proximal animal classes”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 5233–5242.

[216] Raphaël Sarfati et al. “Spatio-temporal reconstruction of emergent flash synchronization in firefly swarms via stereoscopic 360-degree cameras”. In: *Journal of The Royal Society Interface* 17.170 (2020), p. 20200179.

[217] Saquib Sarfraz, Vivek Sharma, and Rainer Stiefelhagen. “Efficient parameter-free clustering using first neighbor relations”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 8934–8943.

[218] Saquib Sarfraz et al. “Temporally-weighted hierarchical clustering for unsupervised action segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 11225–11234.

[219] *Scenedetect: Video Scene Cut Detection and Analysis Tool*. URL: https://github.com/Breakthrough/PySceneDetect.

[220] Eric Schulz, Francisco Quiroga, and Samuel J Gershman. “Communicating compositional patterns”. In: *Open Mind* 4 (2020), pp. 25–39.

[221] Nicolas Schweighofer, K Doya, and F Lay. “Unsupervised learning of granule cell sparse codes enhances cerebellar adaptive control”. In: *Neuroscience* 103.1 (2001), pp. 35–50.

[222] Fadime Sener and Angela Yao. “Unsupervised learning and segmentation of complex activities from video”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8368–8376.

[223] Dian Shao et al. “Intra- and Inter-Action Understanding via Temporal Action Parsing”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[224] Hanul Shin et al. “Continual learning with deep generative replay”. In: *Advances in neural information processing systems* 30 (2017).

[225] Mike Zheng Shou et al. “Generic Event Boundary Detection: A Benchmark for Event Segmentation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), pp. 8075–8084.

[226] David Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *nature* 529.7587 (2016), pp. 484–489.

[227] Tanja Berger-Wolf Michael J. Black Silvia Zuffi Angjoo Kanazawa. “Three-D Safari: Learning to Estimate Zebra Pose, Shape, and Texture from Images ”In the Wild””. In: *The IEEE International Conference on Computer Vision (ICCV)*. 2019.

[228] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[229] Yosef Singer et al. “Sensory cortex is optimized for prediction of future input”. In: *elife* 7 (2018), e31557.

[230] Fabian H Sinz et al. “Engineering a less artificial intelligence”. In: *Neuron* 103.6 (2019), pp. 967–979.

[231] Haim Sompolinsky and Ido Kanter. “Temporal association in asymmetric neural networks”. In: *Physical review letters* 57.22 (1986), p. 2861.

[232] Sen Song, Kenneth D Miller, and Larry F Abbott. “Competitive Hebbian learning through spike-timing-dependent synaptic plasticity”. In: *Nature neuroscience* 3.9 (2000), pp. 919–926.

[233] Yuhang Song et al. “Can the brain do backpropagation?—exact implementation of back-propagation in predictive coding networks”. In: *Advances in neural information processing systems* 33 (2020), pp. 22566–22579.

[234] Nicole K Speer, Khena M Swallow, and Jeffery M Zacks. “Activation of human motion processing areas during event perception”. In: *Cognitive, Affective, & Behavioral Neuroscience* 3.4 (2003), pp. 335–345.

[235] Mandyam Veerambudi Srinivasan, Simon Barry Laughlin, and Andreas Dubs. “Predictive coding: a fresh view of inhibition in the retina”. In: *Proceedings of the Royal Society of London. Series B. Biological Sciences* 216.1205 (1982), pp. 427–459.

[236] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. “Unsupervised learning of video representations using lstms”. In: *International conference on machine learning*. PMLR. 2015, pp. 843–852.

[237] Jennifer J Sun et al. “The Multi-Agent Behavior Dataset: Mouse Dyadic Social Interactions”. In: *arXiv preprint arXiv:2104.02710* (2021).

[238] Alexandra Swanson et al. “Snapshot Serengeti, high-frequency annotated camera trap images of 40 mammalian species in an African savanna”. In: *Scientific data* 2.1 (2015), pp. 1–14.

[239] C Szegedy. “Intriguing properties of neural networks”. In: *arXiv preprint arXiv:1312.6199* (2013).

[240] Christian Szegedy et al. “Rethinking the inception architecture for computer vision”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[241] Mufeng Tang, Helen Barron, and Rafal Bogacz. “Sequential memory with temporal predictive coding”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[242] Mufeng Tang et al. “Recurrent predictive coding models for associative memory employing covariance learning”. In: *PLoS computational biology* 19.4 (2023), e1010719.

[243] Camille Testard, Sébastien Tremblay, and Michael Platt. “From the field to the lab and back: neuroethology of primate social behavior”. In: *Current opinion in neurobiology* 68 (2021), pp. 76–83.

[244] Sebastian Thrun and Tom M Mitchell. “Lifelong robot learning”. In: *Robotics and autonomous systems* 15.1-2 (1995), pp. 25–46.

[245] Trond A Tjøstheim and Andreas Stephens. “Intelligence as Accurate Prediction”. In: *Review of Philosophy and Psychology* (2021), pp. 1–25.

[246] Edward C Tolman. “Cognitive maps in rats and men.” In: *Psychological review* 55.4 (1948), p. 189.

[247] Mikhail V Tsodyks and Mikhail V Feigel’man. “The enhanced storage capacity in neural networks with low activity level”. In: *Europhysics Letters* 6.2 (1988), p. 101.

[248] Devis Tuia et al. “Perspectives in machine learning for wildlife conservation”. In: *Nature communications* 13.1 (2022), pp. 1–15.

[249] E Tulving. “Episodic and semantic memory”. In: *Organization of memory/Academic Press* (1972).

[250] Alan M Turing. *Computing machinery and intelligence*. Springer, 2009.

[251] John Joseph Valletta et al. “Applications of machine learning in animal behaviour studies”. In: *Animal Behaviour* 124 (2017), pp. 203–220.

[252] Aaron Van Den Oord, Oriol Vinyals, et al. “Neural discrete representation learning”. In: *Advances in neural information processing systems* 30 (2017).

[253] Cristina Vasconcelos, Vighnesh Birodkar, and Vincent Dumoulin. “Proper Reuse of Image Classification Features Improves Object Detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 13628–13637.

[254] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[255] Gido M Van de Ven, Tinne Tuytelaars, and Andreas S Tolias. “Three types of incremental learning”. In: *Nature Machine Intelligence* 4.12 (2022), pp. 1185–1197.

[256] Rosaura G. VidalMata et al. “Joint Visual-Temporal Embedding for Unsupervised Learning of Actions in Untrimmed Sequences”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2021, pp. 1238–1247.

[257] Guiming Wang. “Machine learning for inferring animal behavior from location and movement data”. In: *Ecological informatics* 49 (2019), pp. 69–76.

[258] Xiao Wang et al. “CoSeg: Cognitively Inspired Unsupervised Generic Event Segmentation”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[259] Yunbo Wang et al. “Predrnn++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning”. In: *arXiv preprint arXiv:1804.06300* (2018).

[260] Geoffrey I Webb et al. “Characterizing concept drift”. In: *Data Mining and Knowledge Discovery* 30.4 (2016), pp. 964–994.

[261] Greg Welch, Gary Bishop, et al. “An introduction to the Kalman filter”. In: (1995).

[262] P. Welinder et al. *Caltech-UCSD Birds 200*. Tech. rep. CNS-TR-2010-001. California Institute of Technology, 2010.

[263] P. Welinder et al. *Caltech-UCSD Birds 200*. Tech. rep. Cns-tr-2010-001. California Institute of Technology, 2010.

[264] Tobias Weyand et al. “Google landmarks dataset v2-a large-scale benchmark for instance-level recognition and retrieval”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 2575–2584.

[265] James CR Whittington and Rafal Bogacz. “An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity”. In: *Neural computation* 29.5 (2017), pp. 1229–1262.

[266] James CR Whittington and Rafal Bogacz. “Theories of error back-propagation in the brain”. In: *Trends in cognitive sciences* 23.3 (2019), pp. 235–250.

[267] James CR Whittington et al. “The Tolman-Eichenbaum machine: unifying space and relational memory through generalization in the hippocampal formation”. In: *Cell* 183.5 (2020), pp. 1249–1263.

[268] Shuchen Wu et al. “Learning Structure from the Ground up—Hierarchical Representation Learning by Chunking”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 36706–36721.

[269] Yongqin Xian et al. “Zero-Shot Learning–A Comprehensive Evaluation of the Good, the Bad and the Ugly”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.9 (2019), pp. 2251–2265. DOI: 10.1109/tpami.2018.2857768.

[270] Shen Yan et al. “Multiview transformers for video recognition”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 3333–3343.

[271] Zhilin Yang et al. “Xlnet: Generalized autoregressive pretraining for language understanding”. In: *Advances in neural information processing systems*. Vol. 32. 2019, pp. 5754–5764.

[272] Yuan Yao et al. “OpenMonkeyChallenge: Dataset and Benchmark Challenges for Pose Tracking of Non-human Primates”. In: *bioRxiv* (2021).

[273] Kexin Yi et al. “CLEVRER: Collision Events for Video Representation and Reasoning”. In: *International Conference on Learning Representations*. 2019.

[274] Daniel Yon, Cecilia Heyes, and Clare Press. “Beliefs and desires in the predictive brain”. In: *Nature Communications* 11.1 (2020), p. 4404.

[275] Jinsoo Yoo and Frank Wood. “BayesPCN: A continually learnable predictive coding associative memory”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 29903–29914.

[276] Jaehong Yoon et al. “Lifelong Learning with Dynamically Expandable Networks”. In: *International Conference on Learning Representations*. 2018.

[277] Jeffrey M Zacks. “Using movement and intentions to understand simple events”. In: *Cognitive Science* 28.6 (2004), pp. 979–1008.

[278] Jeffrey M Zacks and Khena M Swallow. “Event segmentation”. In: *Current directions in psychological science* 16.2 (2007), pp. 80–84.

[279] Jeffrey M Zacks and Barbara Tversky. “Event structure in perception and conception.” In: *Psychological bulletin* 127.1 (2001), p. 3.

[280] Jeffrey M Zacks, Barbara Tversky, and Gowri Iyer. “Perceiving, remembering, and communicating structure in events.” In: *Journal of experimental psychology: General* 130.1 (2001), p. 29.

[281] Jeffrey M Zacks et al. “Event perception: a mind-brain perspective.” In: *Psychological bulletin* 133.2 (2007), p. 273.

[282] Jure Zbontar et al. “Barlow twins: Self-supervised learning via redundancy reduction”. In: *International Conference on Machine Learning*. Pmlr. 2021, pp. 12310–12320.

[283] Libby Zhang et al. “Animal pose estimation from video data with a hierarchical von Mises-Fisher-Gaussian model”. In: *International Conference on Artificial Intelligence and Statistics*. Pmlr. 2021, pp. 2800–2808.

[284] Bolei Zhou et al. “Temporal relational reasoning in videos”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 803–818.

[285] Jinghao Zhou et al. “ibot: Image bert pre-training with online tokenizer”. In: *International Conference on Learning Representations*. 2021.

[286] Wencheng Zhu et al. “Dsnet: A flexible detect-to-summarize network for video summarization”. In: *IEEE Transactions on Image Processing* 30 (2020), pp. 948–962.

[287] Zoran Zivkovic and Ferdinand Van Der Heijden. “Efficient adaptive density estimation per image pixel for the task of background subtraction”. In: *Pattern recognition letters* 27.7 (2006), pp. 773–780.

[288] Silvia Zuffi, Angjoo Kanazawa, and Michael J Black. “Lions and tigers and bears: Capturing non-rigid, 3d, articulated shape from images”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2018, pp. 3955–3963.

[289] Silvia Zuffi et al. “3D Menagerie: Modeling the 3D Shape and Pose of Animals”. In: *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

Appendix A: Single-Layered Model

A.1 Implementation Details

In our experiments, we use an Inception V3 [240] encoding model (pretrained on the ImageNet dataset) to transform input images from raw pixel representation to a higher-level feature representation. We freeze the encoder's parameters (weights and biases) and remove the last layer. We resize the input image to . The output of the Inception V3 model is a feature tensor, which we reshape to feature vectors. Each 2048 feature vector requires one LSTM cell for future feature prediction. In other words, the encoded input frame ( ) is provided with 64 LSTM cells (sharing the same weights), each processing a 2048 features vector (hidden state size) simultaneously. During a single optimization step, the model receives eight current frames and predicts eight future frames. The next training step slides the model 8 frames into the future. Hidden states of the LSTM are copied (i.e., stop gradient operation) to initialize the LSTMs of future training steps (i.e., stateful LSTM). We use a 0.4 drop rate (recurrent dropout) on the hidden states to prevent overfitting, which may easily occur due to the stateful LSTM nature of the model and the dataset size. LSTMs' hidden states are initialized to zero. Teacher forcing [130] approach is utilized by concatenating the weighted encoded input image ( ) with the encoded input image ( ) instead of concatenating it with its prediction from the previous time step ( ). Adam optimizer is used with a learning rate of for the gradient descent algorithm. We do not use data augmentations. The dataset is divided into four equal portions and simultaneously trained on four Nvidia GTX 1080 GPUs.

A.2 Relevant Work

This section presents a literature review of relevant works on event segmentation, and animal-related computer vision works. A comprehensive review of predictive learning works is provided in Chapter 2. We begin by introducing supervised approaches targeting event segmentation and boundary detection in Appendix A.2.1, followed by a review of self-supervised event segmentation approaches in Appendix A.2.2. We conclude by presenting animal-related computer vision works and comparison of the Kagu video monitoring dataset to other animal datasets provided in different formats (e.g., images, videos, features) in Appendix A.2.3 and Appendix A.2.4.

A.2.1 Fully Supervised Approaches

A.2.1.1 Supervised Temporal Event Segmentation

Supervised approaches use direct labeling (of frames) to segment videos into smaller constituent events. Fully supervised models are heavily dependent on a vast amount of training data to achieve good segmentation results. Different model variations and approaches have been tested, such as using an encoder-decoder temporal convolutional network (ED-TCN) [135] or a spatiotemporal CNN model [134]. To alleviate the need for expensive direct labeling, weakly supervised approaches [23, 47, 105, 199] have emerged with an attempt to use metadata (such as captions or narrations) to guide the training process without the need for explicit training labels [11, 154]. However, such metadata are not always available as part of the dataset, making weakly supervised approaches inapplicable to most practical applications. UnweaveNet [185] introduces the task unweaving, which defines each activity as a single thread and builds a thread bank to segment and label parts of activities in an untrimmed video. UnweaveNet uses a supervised transformer architecture to learn temporal dependencies between clips.

A.2.1.2 Supervised Boundary Detection

Supervised boundary detection approaches use boundary labels as a supervised learning signal for the model. For example, [134, 135, 145, 223, 225] use a binary classifier head on the extracted features to classify the state of every frame (boundary/non-boundary). The undesirable need for direct labeling gave rise to a family of approaches categorized as "weakly-supervised" methods. Similar to event segmentation, weakly supervised approaches [47, 105] make use of certain metadata, such as video narrations or temporal ordering of frames, to provide a learning signal.

A.2.2 Self-Supervised Approaches

A.2.2.1 Self-Supervised Temporal Event Segmentation

Self-supervised event segmentation methods attempt to completely eliminate the need for annotations [163, 222]. Many approaches rely heavily on higher-level features clustering of frames to sub-activities [22, 256]. The performance of the clustering algorithms in unsupervised event segmentation is proportional to the performance of the embedding/encoding model that transforms frames into higher-level feature representations. Clustering algorithms can be computationally expensive depending on the number of frames to be clustered. Recent work [1] uses a self-supervised perceptual predictive model to detect event boundaries; we improve upon this model to include an attention unit (other differences are discussed in Section 3.4.1.3), which helps the model focus on the main event-causing object and allows for locating it in each frame. Other work [161] uses a self-supervised perceptual prediction model that is refined over a significant amount of reinforcement learning iterations. Recently, [225] proposed the use of context embedding differences before and after the target frame to classify the state of each frame (boundary/non-boundary). Despite using a backbone pretrained [80] on the same domain (Kinetics400), we still outperform [225] using our online training approach.

Recent work [2] has used the prediction loss, with the assistance of pretrained region proposal networks (RPNs) and multi-layer LSTM units, to localize actions. We eliminate the need for RPNs and multi-layer LSTM units by extracting Bahdanau [16] attention weights prior to the LSTM prediction layer, which allows our model to localize objects of interest, even when stationary. From our experiments, we found that the prediction loss attention tends to fade away as moving objects become stationary, which makes its attention map more similar to results extracted from background subtraction or optical flow. In contrast, our model successfully attends to moving and stationary objects despite variations in environmental conditions, such as moving shadows and lighting changes, as presented in the supplementary videos.

A.2.3 Computer Vision for Animals

Machine learning models, specifically deep learning approaches, have been rising in popularity among ethologists. Recent models have targeted pose estimation [56, 283], tracking [160, 180, 197] and other tasks [33, 216]. However, most approaches rely heavily on full supervision and manually annotated datasets. Similarly, a few works [9, 58, 87, 156, 248, 251, 257] have used manual annotations to train deep learning architectures and facilitate behavioral analysis of animals. In contrast to these approaches, we propose a stream learning model that learns and adapts to the data in an online manner without the need for task-specific labels.

A.2.4 Animal Datasets

The field of animal video monitoring and analysis has recently started growing in popularity in the vision community [76, 112, 140, 215, 227, 288, 289]. However, most works focus on image-level species classification and, more recently, pose and keypoint estimation. The types of annotations provided with datasets limit the available tasks to be tackled by researchers. As can be seen from Table A.1, large-scale datasets targeting event localization are almost nonexistent - limiting the research in the field of animal behavior analysis. The majority of datasets collect

Table A.1: Review of wildlife datasets (Part 1). We provide a review of related datasets, types, lengths, and number of classes.

Title	Type	Length	Classes
Mouse Resident-Intruder [28]	RGB Video	88 hrs.	13
Mouse Social Interaction [237]	RGB Video	-	3
RatSI [150]	RGB Video	1.75 hrs.	8
Boxes on Bees and Pollen [173]	RGB Video	-	1
Conservation Drones [24]	UAV TIR Videos	-	2
Snapshot Serengeti [238]	RGB Images	-	48
Caltech Camera Traps [262]	RGB Images	-	21
Open Monkey [272]	RGB Images	-	4
Amur Tiger Re-id [139]	RGB Images	-	1
Whale Shark ID [102]	RGB Images	-	1
Caltech-UCSD Birds 200 [262]	RGB Images	-	200
NOAA Arctic Seals 2019 [171]	Aerial Images	-	-
Seabirds W. Africa [116]	Aerial GeoTIFF Image	-	6
Animals with Attributes2 [269]	Feature Rep.	-	50
Livestock Behavior Patterns [201]	Acceleration	12 days	2
Nest of the Kagu	**Continuous Video**	***252.8 hrs***	**5**

animal images through camera traps, which are motion-activated. The collected images are then manually filtered to remove false positives triggered by wind or lighting conditions (e.g., moving clouds and falling leaves). The animals detected in camera trap datasets are then manually labeled for classification (i.e., species [238, 263] or ID annotations [102, 139]), localization (i.e., bounding box [102, 139, 171, 263], masks [263], point annotations [116]), or pose estimation [139, 272].

Video data of continuous monitoring of animal behavior in their natural habitat are hard to find; very few annotated datasets [28, 150, 237] provide a collection of short videos captured in a controlled environment with event/behavior labels. The scarcity of extensive wildlife video datasets, targeting events and behavioral analysis, led to the collection and annotation of our dataset. Our dataset features ten consecutive days (11 nights) of continuous monitoring of a Kagu nest. We provide two types of annotations: bounding boxes for spatial segmentation and frame-level event labels for temporal event segmentation. Both types of annotations are provided for the whole

Table A.2: Review of wildlife datasets (Part 2). We provide a review of related datasets, amounts and types of annotations, as well as the locations for each dataset

Title	Labelled Frames	Annotations	Location
Mouse Resident-Intruder [28]	8M	Action	Controlled Env.
Mouse Social Interaction [237]	1M	Behavior	-
RatSI [150]	160K	Action	Controlled Env.
Boxes on Bees and Pollen [173]	5K	B.Box	-
Conservation Drones [24]	62K	B.Box	Southern Africa
Snapshot Serengeti [238]	322.7K	Species	Tanzania
Caltech Camera Traps [262]	243K	B.Box	Southwestern USA
Open Monkey [272]	195K	Pose	-
Amur Tiger Re-id [139]	9.5K	B.Box / Pose / 92 IDs	10 China Zoos
Whale Shark ID [102]	7.7K	B.Box / ID	Western Australia
Caltech-UCSD Birds 200 [262]	6K	B.Box / Mask / Species	-
NOAA Arctic Seals 2019 [171]	14K	B.Box	Alaska, USA
Seabirds W. Africa [116]	1	21K Point Annot.	West Africa
Animals with Attributes2 [269]	37.3K	Numeric attributes	-
Livestock Behavior Patterns [201]	-	Activity	Farmhouse
Nest of the Kagu	**23M**	**B.Box / Event**	**New Caledonia**

dataset (23 million frames). In addition to the dataset's scale, length, and quantity of annotations, our dataset is collected in the wild in a non-invasive manner.

Appendix B: Multi-Layered Model

B.1 Implementation Details

We resize video frames to and use a 4-layer CNN autoencoder (only for the first level) to project every frame to a single feature vector of dimension for temporal processing. For predictive-based models (STREAMER and LSTM+AL), we sample frames at 2 fps, whereas for clustering-based models, we use a higher sampling rate (5 fps) to reduce noise during clustering. We use cosine similarity as the distance measure ( ) and use the Adam optimizer with a constant learning rate of for training. We do not use batch normalization, regularization (*i.e.*, dropout, weight decay), learning rate schedule, or data augmentation during training. We use transformer encoder architecture for functions and ; however, ablations show different architectural choices. A window size of 50 inputs (timescale respective) is used to compute the running average in Equation (4.5), and a new layer is added to the stack after layer has processed 50K inputs.

The official implementations of *TW-FINCH* and *LSTM-AL* are available on GitHub (TW-FINCH, LSTM-AL). We use the provided code to run our comparisons. For LSTM-AL, the required number of clusters for each video being clustered was set to [1]. For LSTM-AL, the order of peak detection was set to frames (sampled at 2 fps) to optimize the best results. Both offline and online clustering versions of *ABD* had to be re-implemented based on the implementation details of the paper. For offline clustering, the window size was set to 5, the order of non-max suppression to 10, and the average number of actions to . For the online clustering variant, the window size was set

[1] is the total number of narrations in the ground truth for a given video plus one (for the background).

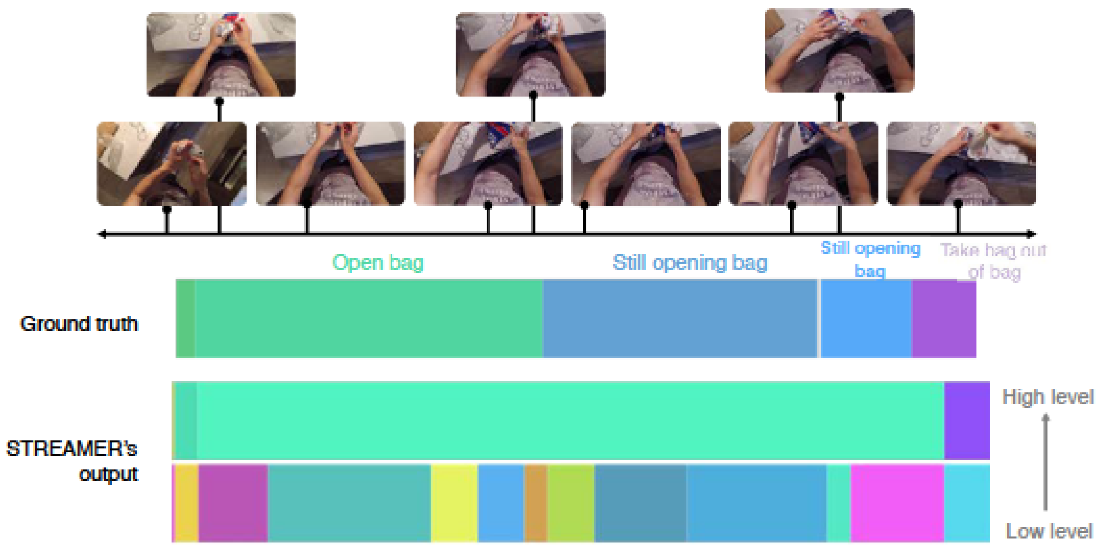


Figure B.1: An illustration of the effect of inconsistent ground truth on the model's evaluation performance. In this segment of a video from EPIC-KITCHENS, the ground truth consists of the same narration annotated thrice in succession (*open bag* 🟩, *still opening bag* 🟦, *still opening bag* 🟦). Although our model could correctly detect this narration to its entirety (the middle row 🟩), its IoU is low, thus affecting its overall evaluation score. Such inconsistencies and redundancies are prevalent throughout the dataset.

Given a video snippet of an event (which we will refer to as a 'query'), can the model retrieve semantically similar video snippets from across the dataset? To determine this, we perform

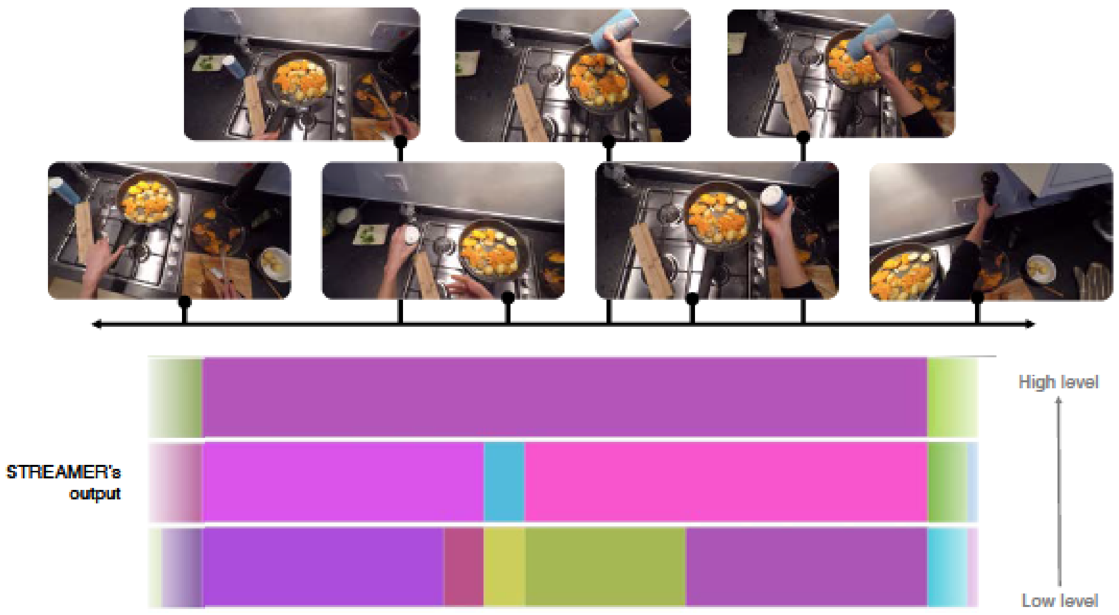


Figure B.2: Qualitative example of STREAMER hierarchical temporal segmentation. Given a sequence of temporal perceptual inputs (*e.g.*, video), our model learns to represent them at varying levels of detail. This figure illustrates the predictions made by our model on a video from EPIC-KITCHENS at three levels: the highest level (the top row in the bar chart) captures a high-level, low-detail concept (seasoning vegetables 🟪); the middle row captures events at relatively finer detail (mixing vegetables 🟪 and adding salt 🟪); and the last row captures the events in much more granular detail. Video available as supplementary.

event retrieval by representation: we first generate a representation for a random query which is then compared with the representations of events from all the videos in the dataset. Based on an appropriate similarity measure as required by the model, we select the top-few nearest matches and qualitatively examine the result.

Figure B.3 shows an example of the top-three similar matches compared with ABD. Figure B.4 shows more examples of STREAMER's event retrieval, displaying the best of the top three matches. Distance in feature space is calculated by the cosine similarity for our method and the Euclidean distance for ABD.

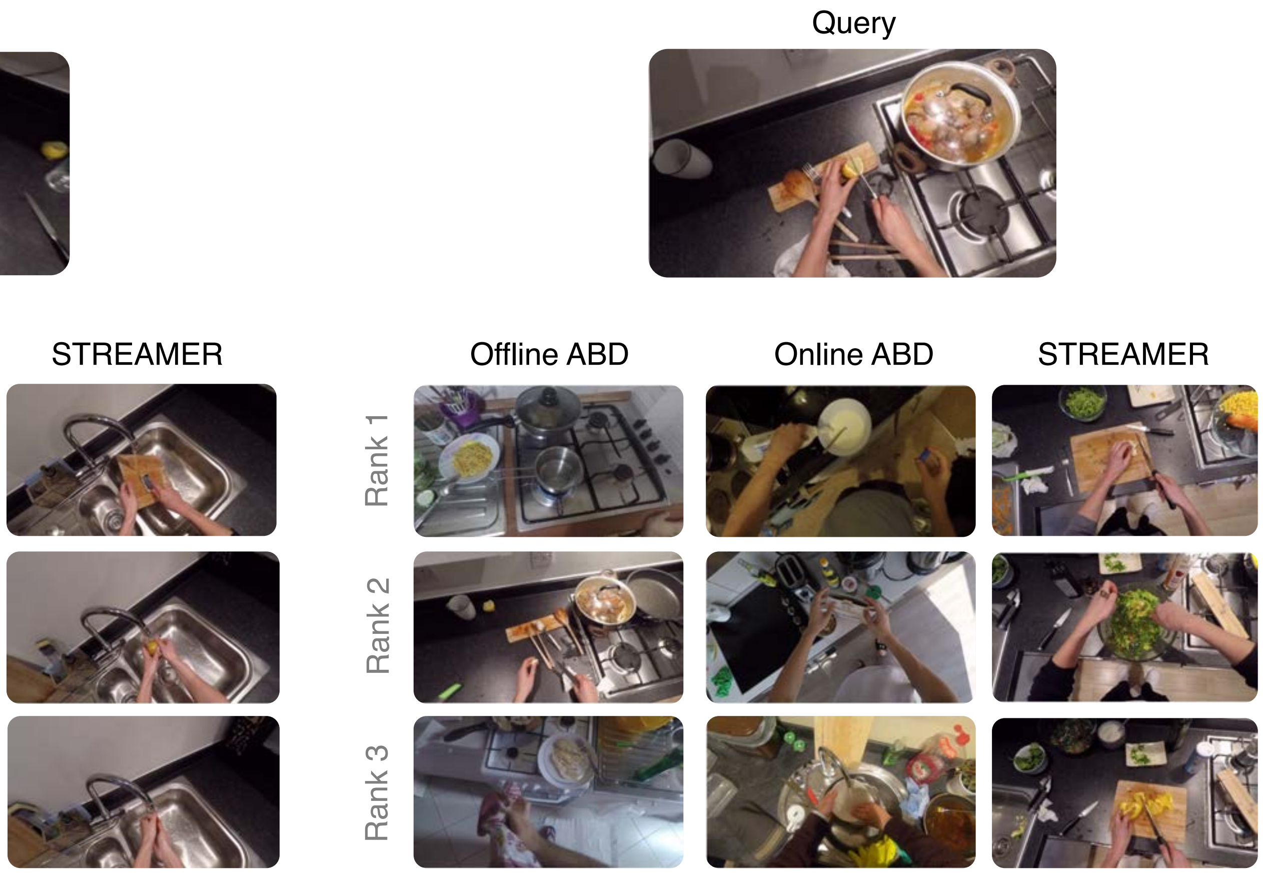


Figure B.3: Qualitative top three results for event retrieval.. A comparison of top-three nearest neighbors retrievals with ABD.

B.3 Retrieval Quantitative Analysis

In addition to the qualitative results, we perform more quantitative experiments on retrieval. As described in the main text of this work, we use the large language model (LLM) GPT 3.5 to create a dataset of event labels from EPIC-KITCHENS ranked by the semantic similarity. The dataset contains 1K comparisons where each comparison comprises a 'query' narration and a set of 10 'key' narrations, and each key is ranked by its similarity to the query according to the LLM. The keys are ranked according to the distance of their representations in the feature space. The two rank lists are compared based on (1) Mean Squared Error (MSE) and (2) the Levenshtein edit distance.

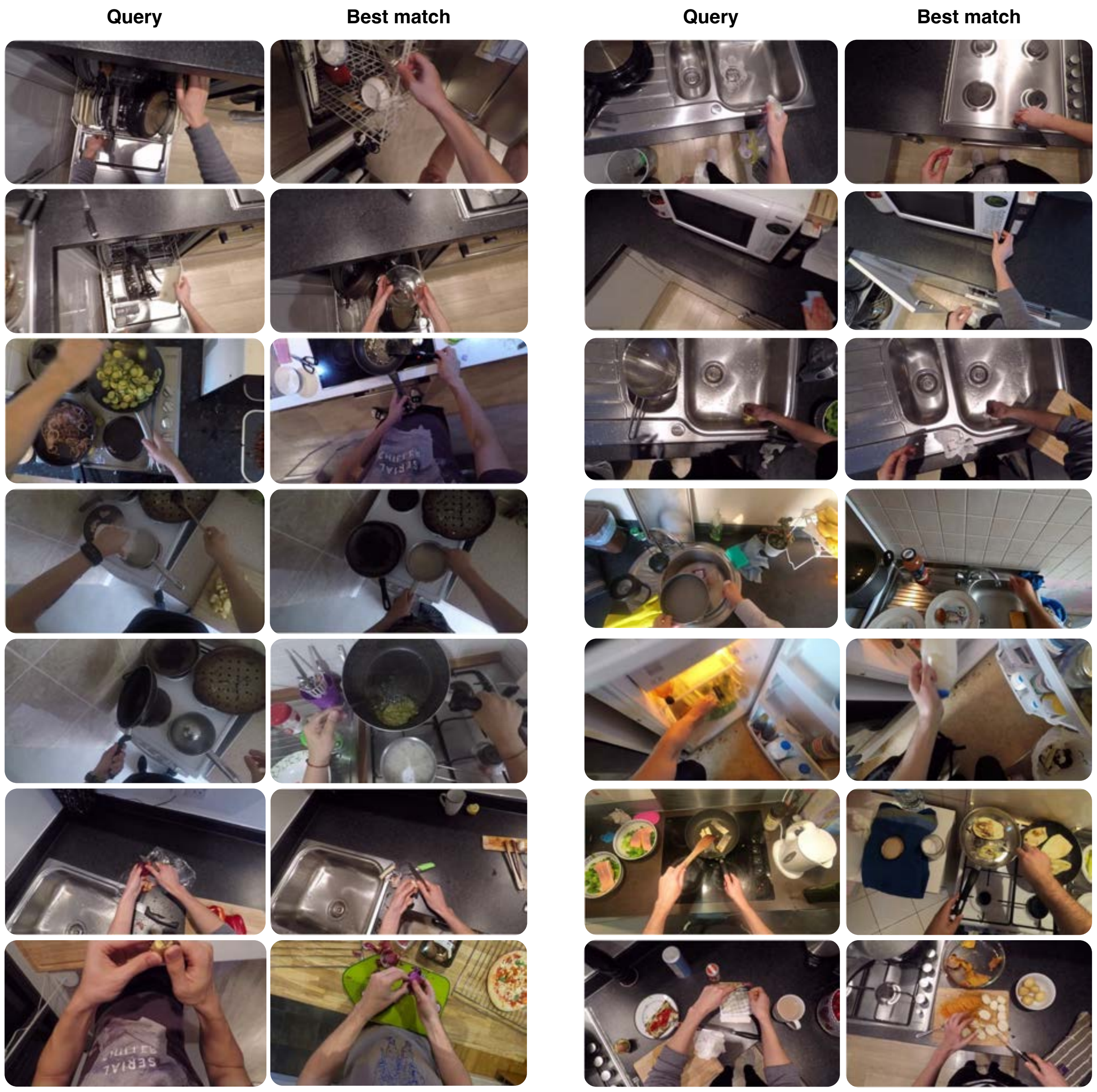


Figure B.4: Additional qualitative results for event retrieval. Random queries and the corresponding best matches, chosen from a set of top-three candidates for each query on EPIC-KITCHENS.

Figure B.5 shows the prompt used to generate the dataset and Table B.2 shows some examples of LLM similarity rankings in the created dataset.

```
prompt = f"""
Given a list of phrase pairs, compute the semantic similarity between the
↪ phrases in each pair and rank in the continuous range of 0 to 10 where
↪ 10 is most similar.

The list is: {queries}

Just return a list of decimal numbers. No explanation.\n"""
```

Figure B.5: The LLM prompt used to generate the dataset for retrieval quality evaluation. We use GPT 3.5 model in this work.

B.4 Relevant Datasets

STREAMER is a self-supervised architecture that relies on predictive learning for hierarchical segmentation. In our model, higher-order predictive layers receive sparser learning signals than lower-order layers, because the first layer directly predicts frames, whereas higher layers only receive events that cannot be predicted at lower levels. Short videos do not allow for higher order predictions and learning long-term temporal dependencies. Therefore, training higher levels in the hierarchy requires a large dataset (*i.e.*, total number of hours) and longer videos (*i.e.*, average video duration) in order to model high-level events. These requirements constraint the choice of datasets on which we can run and evaluate STREAMER.

Based on our review of available datasets for both egocentric and exocentric settings, as shown in the Table B.1, many of the available datasets, typically used in event segmentation, do not provide long enough videos. MovieNet and NewsNet are two large datasets with long videos but have not yet been released to the public. In addition, MovieNet does not contain action segments; it contains coarse scene segments. The only available options to train and evaluate STREAMER is large-scale egocentric datasets, where the available datasets provide large enough scale (*i.e.*, total number of hours) with a long average duration per video for streaming and high-order temporal prediction.

Table B.1: Review of available streaming vision datasets. The table shows various egocentric and exocentric datasets with total hours of recording and average duration statistics.   Not released as of this writing.

	Dataset	Total Hours	Avg. Duration (min)	Large Datasets	Long Videos
Egocentric	Ego4D	3670	23		
	EPIC 55	55	10.5		
	EPIC 100	100	8.5		
	GTEA	0.58	1.23		
Exocentric	Breakfast Action	77	2.3		
	YouTube Instructional	5	2		
	Hollywood Extended	3.7	0.23		
	50 Salads	4.5	4.8		
	FineGym	708	0.91		
	ActivityNet Captions	849	2.5		
	MovieNet	2174	117		
	NewsNet	946	57		

B.5 Glossaries

- *Contextualized inference* refers to the ability of the model to predict a future event by using contextual representations at various levels of the event hierarchy.
- *Contextualized optimization* refers to the ability of a layer to optimize the representations of other layers through its own prediction loss.
- *An Event* is defined as “a segment in time that is perceived by an observer to have a beginning and an end” [281]
- *An Event model*, in cognitive psychology literature, is defined as “a representation of what is happening now, which is robust to transient variability in the sensory input” [281]; in this work, we use ‘event model’ and ‘representation’ interchangeably.

- *Event demarcation and event segmentation*: Event demarcation is the process of detecting event boundaries by using the prediction loss, whereas *event segmentation* is the task of segmenting videos (or sensory inputs) into meaningful events. In other words, event segmentation is the goal, and event demarcation is one way to achieve it: event segmentation could be performed in other ways, such as labeling each frame in a supervised framework.
- *Predictive learning* refers to the brain's ability to generate predictions about future events based on past experiences.
- *A segmentation boundary* is the end of an event and the beginning of the next, marking an event transition.

Table B.2: Examples of similarity values of key and query events. This table shows some examples of the similarity of narrations of 'key events' to a 'query event' as determined by GPT 3.5 (`text-davinci-003`). A 1 scoring means most similar, and a 0 least.

Query	Keys									
open bottle	stop processor	open freezer	enter kitchen	open door	pick up bowl	open fridge	clean kitchen counter	move cheese	put down sausage	put down pan
	0.25	0.8	0.5	0.7	0.4	0.85	0.45	0.3	0.45	0.4
open drawer	pick up stone	check chicken	still cleaning chopping board	shake off courgette	rinse pot	pick up	close dishwasher	take out pasta	still scoop the kiwi	pick up something
	0.75	0.55	0.65	0.65	0.7	0.8	0.7	0.85	0.65	0.8
wash two leaves	wash knife	put down pepper	rinse pan	put away spoon	wash lid	grab bag	get kettle	pour olive oil in pan	stir courgette	pick up forks
	0.75	0.45	0.85	0.65	0.95	0.35	0.45	0.75	0.65	0.55
wash pot	wash pan	open jar	open cupboard	pick up mug	pick up aubergine	pour milk into cereal bowl	rinse cloth	lay aubergine	close fridge	open cupboards
	0.85	0.65	0.75	0.8	0.85	0.9	0.85	0.8	0.75	0.75
pick up bowl	turn off tap	put down salt	cut tomato	stir onion	pick up bowl	stir chicken	place pan	grab salt container	open cupboard	pour milk into glass
	0.75	0.55	0.45	0.65	1.0	0.7	0.6	0.65	0.5	0.65
get chopping board	move chair	open washing machine door	rinse hands	check oil	pour detergent	rinse hands	check temperature	put bread onto tray	wash chopping board with sponge	select schedule
	0.25	0.5	0.8	0.3	0.5	0.8	0.6	0.7	0.9	0.4
put pot	turn on tap	clean cooker	still rinsing spatula	put colander on pot	put fork in bowl	put lid on pot	put lid on sun-dried tomatoes	rinse hands	wipe down counter	move mouse
	0.75	0.45	0.55	0.85	0.65	0.95	0.45	0.55	0.65	0.45
open drawer	take box	pick up jar	close drawer	rinse mug	close hob cover	scrape tomato	lift plate	wash knife	get plastic trash bag	check timer
	0.75	0.85	0.95	0.65	0.75	0.65	0.75	0.85	0.65	0.85
close jug	shake coffee maker	pick up plate	take napkin	get weighing	still taking skin off meat with knife	pick up lid	close tap	get chopping board	put down spoon	put down bowl
	0.25	0.5	0.5	0.25	0.05	0.75	0.9	0.5	0.75	0.75
wash bowl	wash coffee pot	put down knife	put down detergent	throw away onion skin	turn on tap	rinse chopping block	mix nuts with oats	dry hands	place sponge away	open tap
	0.95	0.65	0.85	0.45	0.95	0.85	0.65	0.95	0.85	0.95

Appendix C: Predictive Attractor Models

C.1 Implementation Details

In this section we describe the implementation details and hyperparameters of each method. For each model, we optimize a single set of hyperparameters for all the experiments.

C.1.1 Predictive Attractor Models

The neurons in both, transition and emission, functions are fully connected. We *do not* assume any of the weight matrices are symmetric. All synaptic weights are initialized by sampling from a normal distribution with a mean of 0.0 and a standard deviation of 0.1. All values in Equations 5.6 & 5.7 are set to . is set to , while is set to to avoid forgetting previous possibilities when learning new transitions; PAM learns a union of possibilities. The threshold for the function is set as a function of the SDR sparsity. For the transition function, we use a threshold of , where is the active number of bits in the latent state ( ) SDR. For the emission function, we use a threshold of , where is the active number of bits in the observation state ( ) SDR. During offline generation we sample an initial from with active neurons. During generation, we set the maximum number of attractor iterations to 100, but stop iterating when the energy of the state converges to a local minimum. During sequence learning, we update and iteratively until the transition is learned, before learning the next transition. This iterative weight update makes the model insensitive to the hyperparameter values . Both and are always clamped in the range . The states are flattened into a single dimension before applying the learning rule in Equation 5.6. Binary representations (i.e., 0, 1) are used as inputs.

C.1.2 Temporal Predictive Coding

For the tPC architecture, we use learning rate of 1e-4 for 800 learning iterations. When a 2-layer tPC model is used, the inference learning rate is set to 1e-2 for 400 inference iterations. Also, the hidden size is set to twice the input size. We found that these parameters work best for all of the experiments and allow the model to fully converge. Bipolar representations (i.e., -1, 1) are used as inputs.

C.1.3 Asymmetric Hopfield Network

The Hopfield model does not require hyperparameters other than the ablated separation function. In many experiments, we use a polynomial separation function with degree set to 1 or 2. Bipolar representations (i.e., -1, 1) are used as inputs.

C.2 Notations

The notations used in our paper is summarized in Table C.1.

C.3 Theorems and Derivations

C.3.1 Variational Free Energy

C.3.1.1 Predictive Coding

Consider a hierarchical generative model with hidden states , where denotes the level in the hierarchy. The conditional probability is assumed to be a multivariate Gaussian distribution with its mean calculated as a function of the higher-level hidden representation and covariance as shown in equation C.1.

Table C.1: Table of notations for PAM. The table shows the symbols used in the main paper and a description of each.

Symbol	Description
	Learnable transition weight matrix
	Learnable emission weight matrix
	Observation at time
	Noisy observation at time during recall
	Posterior Latent state after observing (i.e., single possibility)
	Prior Latent state before observing (i.e., multiple possibilities)
	Predicted latent logits at time (i.e.,) before applying threshold
	Threshold function: or
	Hebbian learning strength for adjusting the synaptic weights
	Projection operator adds context to a 1d observation state
	Projection operator removes context from a 2d latent state
	Function for computing the indices of active bits in an SDR
	Random permutation function
	Input size of a pattern
	Number of neurons per minicolumn for context encoding
	Number of active bits in a Sparse Distributed Representation (SDR)
	Sparsity of SDR, calculated as
	Number of patterns in one sequence (i.e., sequence length)
	Degree of polynomial in Hopfield separation function
	Number of Layers used in temporal Predictive Coding (tPC)

$$\mathcal{N} \tag{C.1}$$

The goal is to calculate the posterior of hidden states given an observation , formally . Since the prediction function contains a non-linear activation, we cannot analytically compute the posterior and we have to approximate it with a surrogate posterior (i.e., by maximizing the Evidence Lower Bound (ELBO). We apply the mean field approximation to factorize this joint posterior probability into conditionally independent posteriors , and apply the Laplace approximation to use Gaussian forms for the approximate distribution [61, 65, 212]. Through these approximations, we can maximize the ELBO, or equivalently minimize the Variational Free Energy, in equation C.2.

$$\underbrace{\qquad\quad\qquad\quad}_{\text{Variational Free Energy}} \quad \underbrace{\qquad\quad\qquad\quad}_{} \quad \underbrace{\qquad\quad\qquad\quad}_{\text{Accuracy}} \tag{C.2}$$

The variational free energy can be reduced to minimizing the negative log-likelihood (Accuracy term), which is simply the prediction error when the likelihood is assumed to take a Gaussian Form. Therefore, minimizing the prediction error reduces to the sum of the squared prediction error of every neuron.

Derivation 1. *Variational Free Energy derivation for the predictive coding objective function in equation C.2. We approximate the true posterior with a surrogate posterior . The objective is to minimize the reverse KL divergence .*

(KL Divergence definition)

(Bayes Theorem)

(Linearity of expectations)

Variational Free Energy

(Evidence does not depend on)

To minimize the KL divergence, we can minimize the Variational Free energy instead because the Evidence term ( ) is constant negative term. The Variational Free Energy can be further simplified as follows:

(Linearity of Expectations)

(KL Divergence definition)

Variational Free Energy *Error*

We arrive at equation C.2. Minimizing the error term (i.e., negative log-likelihood) is equivalent to minimizing the Sum of Squared Error (SSE) when a Gaussian form is assumed for the likelihood .

Derivation 2. *Variational Free Energy derivation for a State Space Model (SSM) in equation 5.2. Latent states are denoted with , whereas observations are denoted with . We assume non-linear transition and emission function (i.e., and), therefore a variational approximation is needed to approximate the true posterior _ with a surrogate posterior . As in derivation 1, the goal is to minimize the divergence between the true posterior and the approximate posterior (i.e., _). Note that, for notation brevity, .*

(KL Divergence definition)

(Bayes Theorem)

(Conditional Independence)

$$\underbrace{\qquad\qquad\qquad}_{\textit{Variational Free Energy}}$$

(Linearity of expectations)

We can minimize the Variational Free Energy term which reduces to log-likelihood of two prediction error terms and the negative entropy of the approximate posterior as shown below.

$$\underbrace{\qquad\qquad}_{\textit{Latent State Error}} \quad \underbrace{\qquad\qquad}_{\textit{Observation Error}} \quad \underbrace{\qquad\quad}_{\textit{entropy } \mathcal{H}}$$

Minimizing the Variational Free Energy above forces to better approximate the true posterior.

C.3.2 Gaussian Mixture Model and Hopfield Recall

Derivation 3. *We derive theorem 1 which states that the maximization of the log-likelihood of in the form of a Gaussian Mixture Model is equivalent to the recall function in Hopfield networks (i.e., eqn 2.1), where the means of the GMM (i.e.,) represents the attractors of a Hopfield model. To maximize the log-likelihood, we compute its partial derivative with respect to .*

$$\frac{}{} \qquad \frac{}{} \qquad \mathcal{N}$$

$$\frac{}{\mathcal{N}} \qquad \frac{}{} \qquad \mathcal{N}$$

$$\frac{}{\mathcal{N}} \qquad \frac{}{}\mathcal{N}$$

$$\frac{}{\mathcal{N}} \qquad \mathcal{N}$$

$$\frac{\mathcal{N}}{\mathcal{N}}$$

By setting the partial derivative of the log-likelihood to , we can estimate the value of which maximizes the function .

$$\frac{}{}$$

$$\frac{\mathcal{N}}{\mathcal{N}}$$

$$\frac{\mathcal{N}}{\mathcal{N}} \qquad \frac{\mathcal{N}}{\mathcal{N}}$$

Finally, we can rearrange the equation in terms of and show that it is equivalent to the Hopfield recall function where the recall value equals a weighted average of the attractors (i.e., means of GMM), with the weights being a similarity score function.

$$\boldsymbol{x} = \frac{\sum_{c=1}^{C} w_c \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \cdot \boldsymbol{\Sigma}_c^{-1} \boldsymbol{\mu}_c}{\sum_{c=1}^{C} w_c \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \cdot \boldsymbol{\Sigma}_c^{-1}}$$

$$\boldsymbol{x} = \sum_{c=1}^{C} \underbrace{\frac{w_c \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \cdot \boldsymbol{\Sigma}_c^{-1}}{\sum_{c=1}^{C} w_c \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \cdot \boldsymbol{\Sigma}_c^{-1}}}_{\textit{similarity score}} \cdot \underbrace{\boldsymbol{\mu}_c}_{\textit{projection}}$$

C.3.3 Expected IoU of Random SDRs

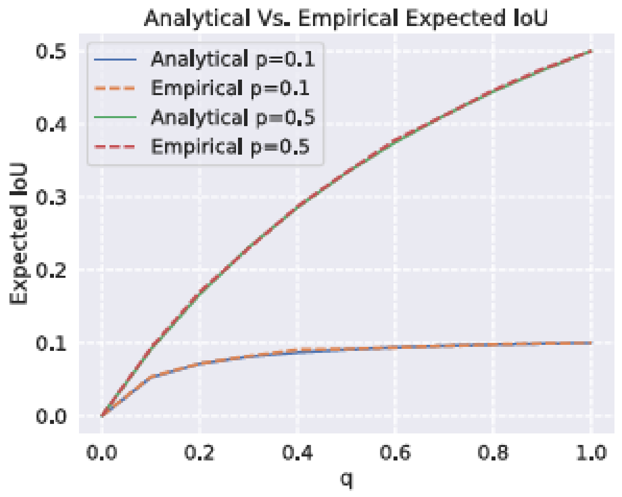


Figure C.1: Empirical validation of the expected IoU theorem. Results show that the analytical formula for calculating the Expected IoU matches with empirical calculations.

Theorem 2. *Consider two SDRs with sparsity defined as random variables* $p \sim \mathcal{U}(0,1)$ *and* $q \sim \mathcal{U}(0,1)$*, the expected Jaccard Index (i.e., IoU) of the two random SDRs follows:*

$$\frac{pq}{p+q-pq}$$

C.3.3.1 Proof

Given the sparsity random variables of both SDRs (i.e., and) and the SDR size , the number of active bits at the same location in both SDRs is equal to the joint probability of both SDRs being active multiplied by the SDR size (i.e.,). The union of both SDRs is the total number of active bits minus the active bits in both SDRs, which is equal to . Therefore, the expected intersection over union is ——— ——

C.3.3.2 Empirical Validation

We perform empirical validation of the above theorem as shown in figure C.1. The sparsity of the first SDR is fixed at 0.1 and 0.5. We vary the sparsity of the second SDR between 0.0 and 1.0 in steps of 0.1 and calculate the average IoU over a population of 1000 pairs of SDR for every setting. Empirical results agree with the derived formulation in theorem 2.

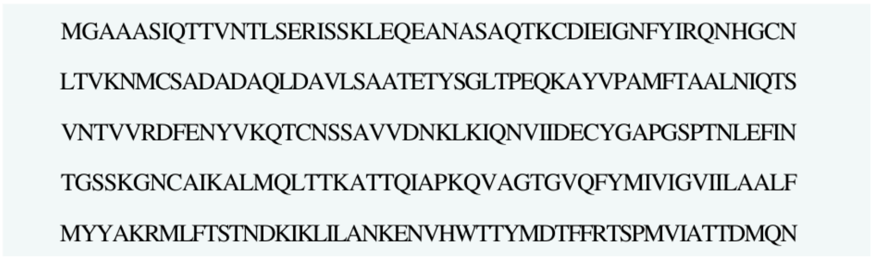
C.4 Datasets Details

C.4.1 Synthetic Datasets

For synthetic experiments, we generate SDRs with the specified size and uniformly initialized active bits to match the required sparsity . In many of the experiments, is set to 100 with 5 active bits, unless otherwise specified. For Hopfield experiments, we set the sparsity to 50% to improve its performance.

C.4.2 Protein Sequences

We use the dataset ProteinNet 7 [10] to extract protein sequences. Each sequence consists of a chain of Amino Acids. In the dataset there are only 20 different types of Amino Acids (i.e., vocabulary) creating long protein sequences with hundreds of Amino Acids. The dataset is reported in the fasta format, where each Amino Acid is represented with a single-letter code. We create a dictionary mapping from the Amino Acid types to random SDRs with and to train the models. When choosing the sequences, we ensure that the starting Amino Acid is unique for all the dataset sequences to avoid ambiguous predictions in the continual learning evaluation. A sample of the protein sequence is provided in Figure C.2.

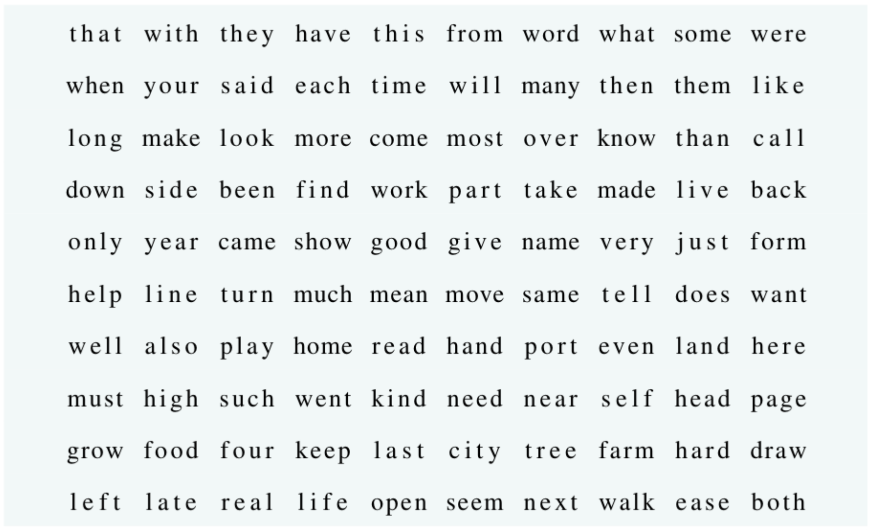


MGAAASIQTTVNTLSERISSKLEQEANASAQTKCDIEIGNFYIRQNHGCN
LTVKNMCSADADAQLDAVLSAATETYSGLTPEQKAYVPAMFTAALNIQTS
VNTVVRDFENYVKQTCNSSAVVDNKLKIQNVIIDECYGAPGSPTNLEFIN
TGSSKGNCAIKALMQLTTKATTQIAPKQVAGTGVQFYMIVIGVIILAALF
MYYAKRMLFTSTNDKIKLILANKENVHWTTYMDTFFRTSPMVIATTDMQN

Figure C.2: Sample protein sequence from ProteinNet 7. Each letter represents an Amino Acid type based on the FASTA format.

C.4.3 Text Dataset

To evaluate the generative ability of PAM, we use a dataset of most frequently used English words. For preprocessing, we extract one hundred 4-letter words from the dataset and create a mapping dictionary from all the unique letters in the dataset to random SDRs with and (except for AHN;). The dataset contains many words with ambiguous future possibilities. The selected words are provided in Figure C.3.



that	with	they	have	this	from	word	what	some	were
when	your	said	each	time	will	many	then	them	like
long	make	look	more	come	most	over	know	than	call
down	side	been	find	work	part	take	made	live	back
only	year	came	show	good	give	name	very	just	form
help	line	turn	much	mean	move	same	tell	does	want
well	also	play	home	read	hand	port	even	land	here
must	high	such	went	kind	need	near	self	head	page
grow	food	four	keep	last	city	tree	farm	hard	draw
left	late	real	life	open	seem	next	walk	ease	both

Figure C.3: The text dataset used in PAM. The dataset contains the 100 most frequently used words.

C.4.4 Vision Datasets

In our experiment, we evaluate on sequences extracted from Moving MNIST [236] and CLEVRER [273] as well as synthetically generated sequences of CIFAR [124] images. In order to convert images to SDRs and SDRs back to images while encoding semantics into the SDRs, we design an SDR AutoEncoder. The goal is to force the bottleneck representation of the autoencoder to become a sparse binary representation, where two visually similar images would result in two SDRs with high overlap of active neurons. We simply design a CNN autoencoder with 3-layer CNN encoder and 3-layer CNN decoder, and apply top K binarization operation on the bottleneck embedding during training. The full architecture of the SDR autoencoder is shown in Figure C.4.

In practice, we use a weighted average of the SDR and Dense representation to allow gradients of the reconstruction loss to freely propagate into the encoder. The weight of the SDR (i.e.,   ) is gradually and linearly increased (from 0.0 to 1.0) with the number of training epochs. This gradual increase in fundamental to the training of the SDR Autoencoder as it smooths the loss landscape and allows the model to distribute the active bits on the full SDR. The total mse loss

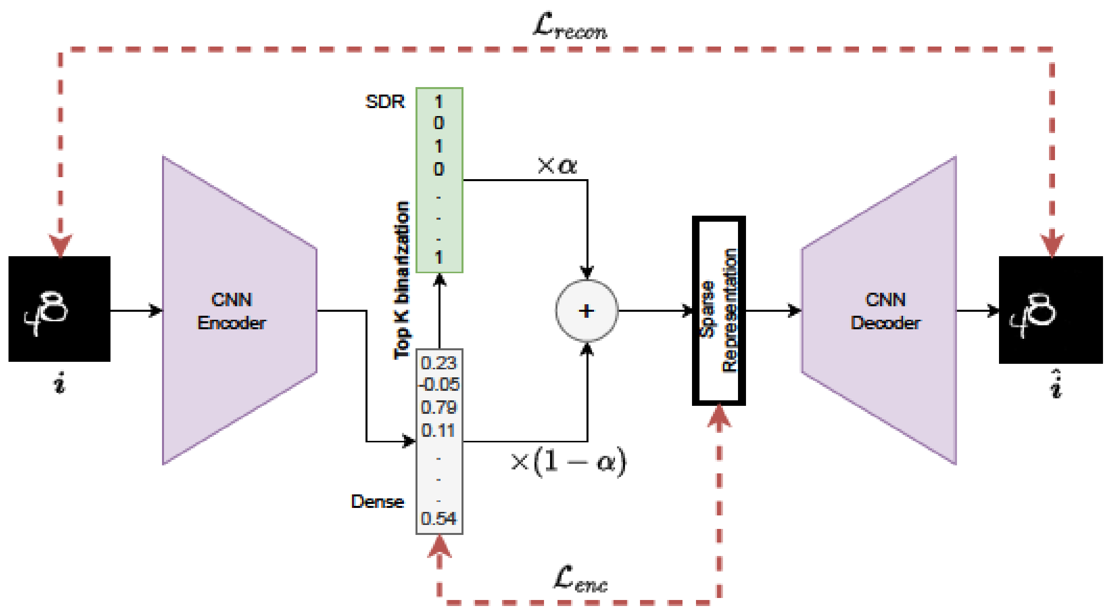


Figure C.4: Overview of the SDR autoencoder architecture. The model consists of a CNN autoencoder and a Top-K binarization operation.

becomes $\mathcal{L}_{enc} + \mathcal{L}_{recon}$. We use Adam optimizer with a learning rate of 1×10^{-4}. For Moving MNIST we use a bottleneck embedding (i.e., N_c) of size 100 with 5 active bits, whereas for more complex datasets (i.e., CLEVRER, CIFAR), we use an SDR of size 200 with 10 active bits. We show examples of the autoencoder reconstruction with full binary SDR (i.e., $\alpha = 1$) for all three datasets in Figure C.5.

C.5 Experiments

In this section we describe the setup of each figure in the main paper and provide additional quantitative and qualitative results for each task. *All experiments are run for 10 different seeds/trials. We report the mean and standard deviation in all the figures and tables.*

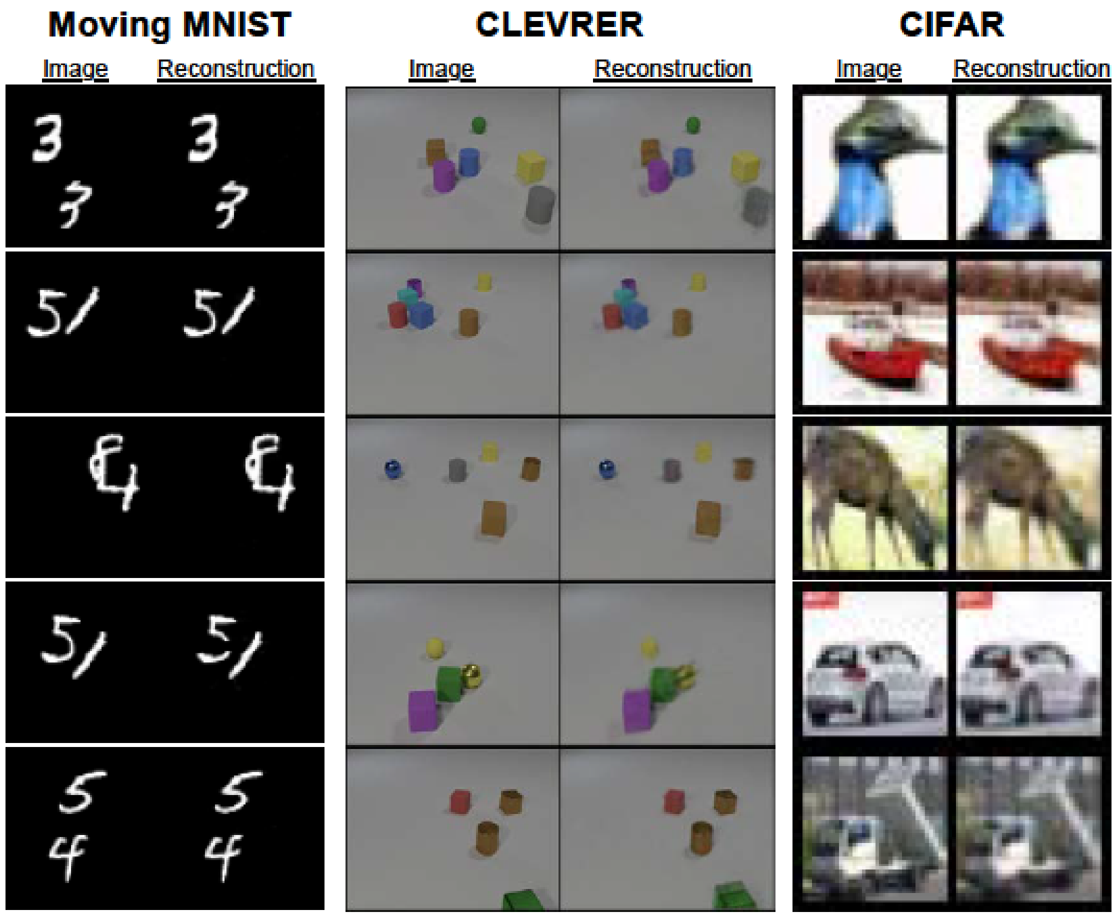


Figure C.5: Examples of autoencoder reconstructions from SDRs for three vision datasets. Results demonstrate the ability of SDRs to store visual information and reconstruct them. Similarity between frames is encoded as bit overlap in the SDRs.

C.5.1 Sequence Capacity

In Figure 5.3 A, we plot the maximum offline sequence length (i.e., sequence capacity, T_{max}) at different input sizes. The input size N_c is varied from 10 to 100 while the number of active bits W is fixed to 5. We compare variants of our model with N_k set to 4 and 8 to temporal predictive coding (tPC) and Asymmetric Hopfield Network (AHN). We observe that AHN completely fails as the sparsity S of the pattern decreases, therefore we also compare to AHN with the sparsity set to

50% (i.e.,). For AHN models, we experiment with a polynomial exponential function with degree and , as recently proposed [30] and used for evaluation in recent papers [241]. All models in this experiment are set to recall/generate in an offline manner, where only the first input is provided. PAM outperforms all other methods and has the potential to improve further by expanding the context neurons . The patterns in this experiment are uncorrelated such that each pattern has active bits uniformly initialized.

In Figure 5.3 B, we plot the effect of sequence correlation on the maximum offline capacity. The higher the correlation value, the more exact repetitions of patterns are available in the sequence. We enforce correlation by limiting the number of unique patterns (i.e., vocab) used to create the sequence. All patterns in this experiment are set to a size of and (except for AHN which is set at). Results show that the capacity of all other methods sharply drops when correlation is introduced. PAM retains most of its original capacity.

In Figure 5.3 E & F, we provide a qualitative example of a short sequence ( ) with high correlation ( ). The sequence is learned by all the methods, then we perform offline (*E*) and online (*F*) recall on the sequence. We use the SDR autoencoder to create SDRs from these CIFAR images for training and recall. The SDRs have a size of 200 and . In the offline recall, only the first input is provided and the model auto-regressively generates the full sequence using its own predictions at every time step. In online recall, the models perform a single step prediction and always uses the groundtruth input at every time step to perform predictions. Results show that only PAM can retain a context of correlated sequence and accurately predicts into the future based on this context.

In Figure C.6 A, we show the effect of scaling the model context memory beyond a simple and . We show that when using and , PAM can model much longer sequences. We vary the input size from 10 to 50 and report the offline sequence capacity of the model as ablations.

In Figure C.6 B, we report the sequence capacity with input size , similar to the experiment plotted in Figure 5.3 *A*. However, this experiment evaluates the online generation capacity, where the

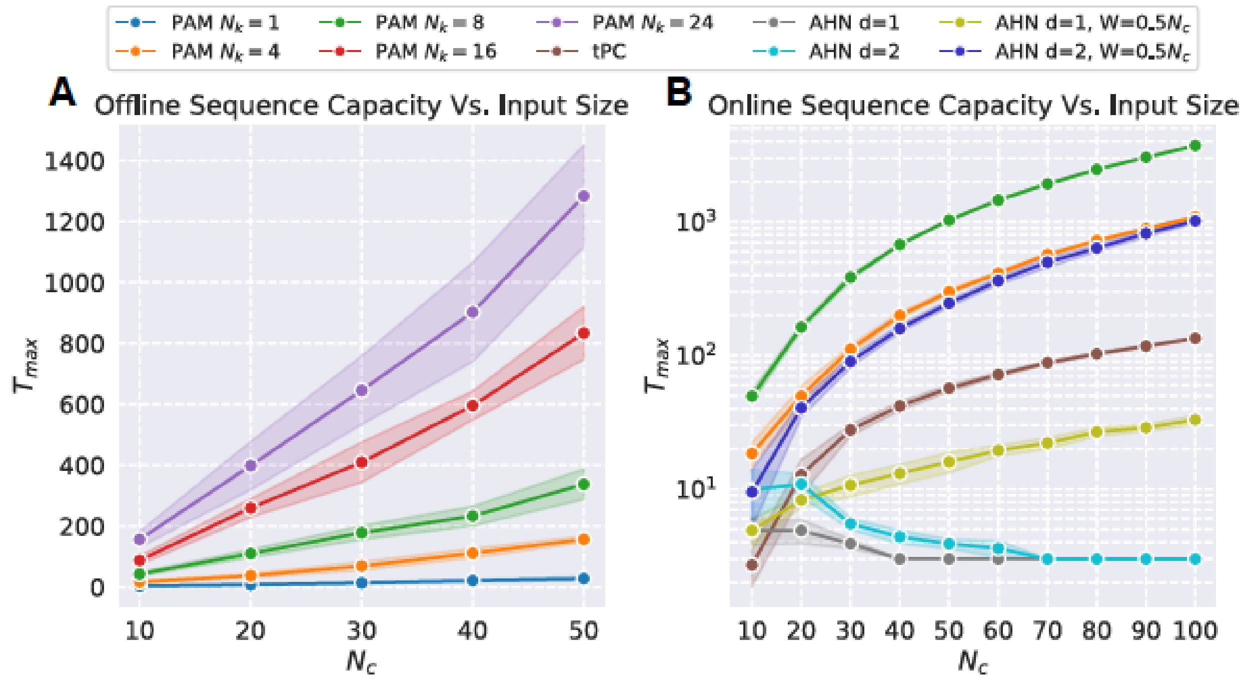


Figure C.6: Additional sequence capacity experiments. *A*: scaling of the offline sequence capacity with context memory size N_k and input size N_c. *B*: Online sequence capacity of various methods.

model uses the correct pattern at every prediction time step instead of using its own prediction from the previous time step. Results show that PAM significantly increased in capacity (three times in some cases), whereas the other methods have not increased as much in modeling longer sequences.

In Figure C.7, we provide additional qualitative example on a different highly correlated sequence. The result shows a different failure mode for AHN, whereas PAM still performs well.

C.5.2 Catastrophic Forgetting

In Figure 5.4 A, we benchmark the performance of difference models in the challenging continual learning setup. The models are expected to avoid catastrophic forgetting by not overwriting previously learned sequences. In this experiment, we use 50 sequences, each with size $N_c = 100$ and length $T = 10$. We vary the correlation of the sequences from 0.0 to 0.5 and compute the backward transfer metric with the normalized IoU as the measure of similarity. Results show that

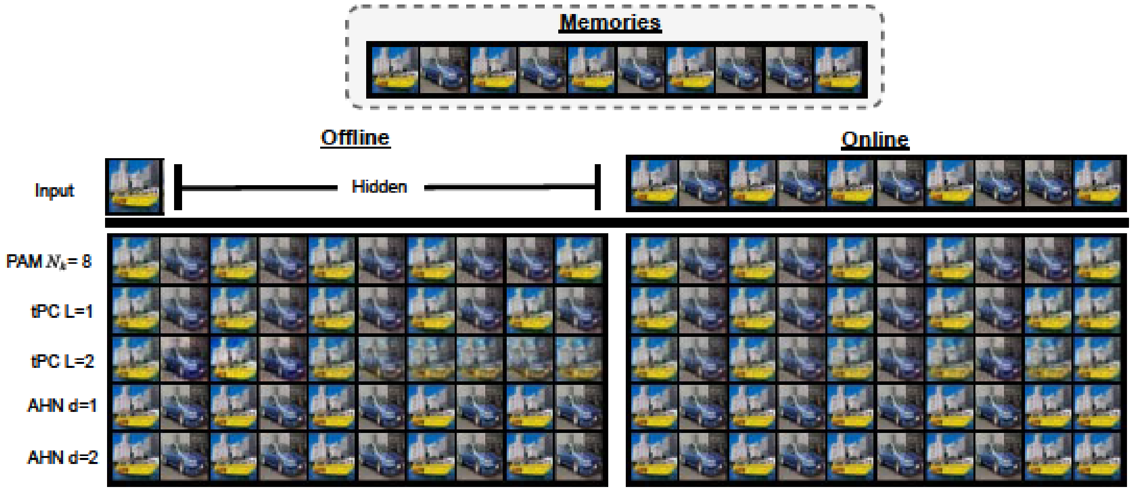


Figure C.7: Additional qualitative example of correlated sequential memory with CIFAR images.

AHN can avoid catastrophic forgetting when the sequence are uncorrelated, but quickly drops in performance with correlation. tPC fails in retaining learned sequences regardless of correlation. PAM performs well with more context neurons N_k. When setting N_k to 1, the model fails to retain its knowledge due to the decreased context modeling capability with a single context neuron.

In Figure 5.4 B, we report the performance of the models on protein sequences. This is a more challenging setup due to the long sequence (few hundreds on average) with high correlation (only 20 unique Amino Acids). We show a similar trend, where the other methods fail due to high correlation or sequence lengths. PAM outperforms the other methods when using context memory N_k of 16 or 24. All Amino Acids types are converted to fixed and randomly initialized SDRs with $N_c = 100$. The sparsity is set similar to sequence capacity experiments (i.e., $W = 5$ and $W = 0.5N_c$).

In Figure 5.4 F, we provide qualitative results on a simple experiment with 2 sequences from moving MNIST. The models learn the first sequence then learn the second sequence. The models are not allowed to train on the first sequence after they have trained on the second sequence. We then perform online generation on the first sequence with all models. We use the SDR autoencoder to generate SDRs for all images in the sequences, the SDRs have $N_c = 100$ with $W = 5$ (for all methods except AHN). The results show that PAM can recall the full sequence even after being trained on another sequence. Other methods fail in this simple task even in online recall setup.

In Figure C.8, we provide continual learning results similar to Figures 5.4 *A* & *B*; however, instead of offline generation, we perform the evaluation in onine manner.

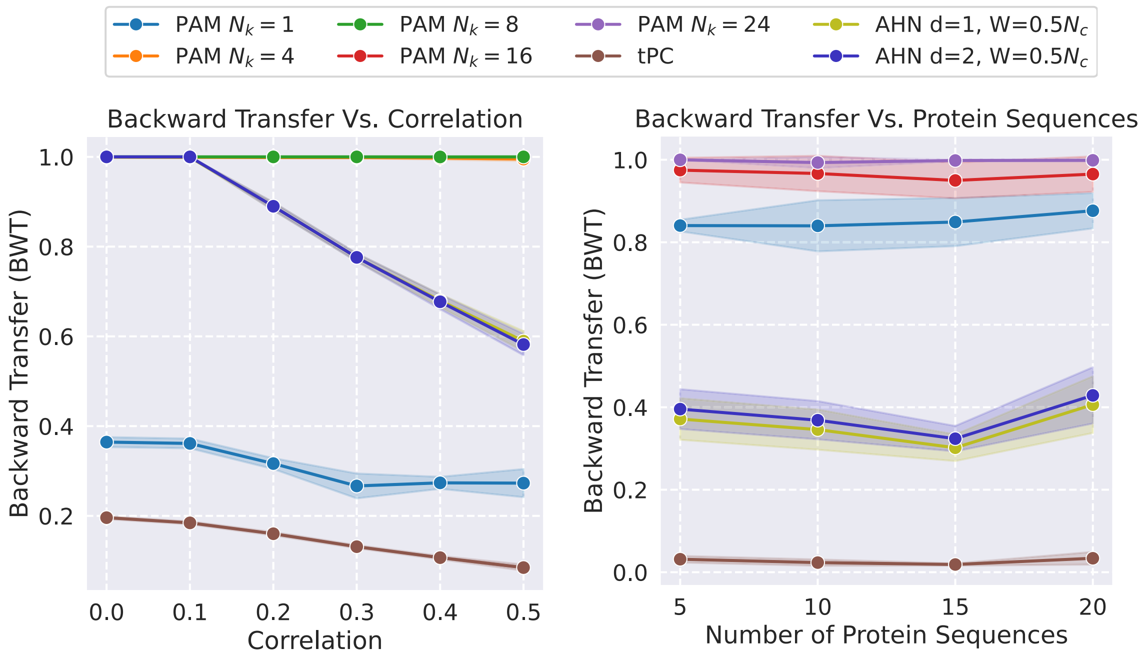


Figure C.8: Catastrophic forgetting experiments for online generation. Results are shown on synthetic dataset of SDR sequences with different correlations and on protein sequences.

In Figure C.9, we provide additional qualitative results on Moving MNIST, as well as quantitative results averaged over 10 trials of MNIST sequence pairs. These quantiative results are reported as the Mean Squared Error of the reconstructed image. Results show that PAM reports the lowest error with a much smaller variance.

In Table C.2, Table C.3, Table C.4, & Table C.5, we report detailed catastrophic forgetting results. The Backward Transfer metric (BWT) to evaluate catastrophic forgetting is computed by taking the average of the performance on previously learned sequences after training on a new sequence. In addition to plotting this average in previous experiments, we provide the full tables

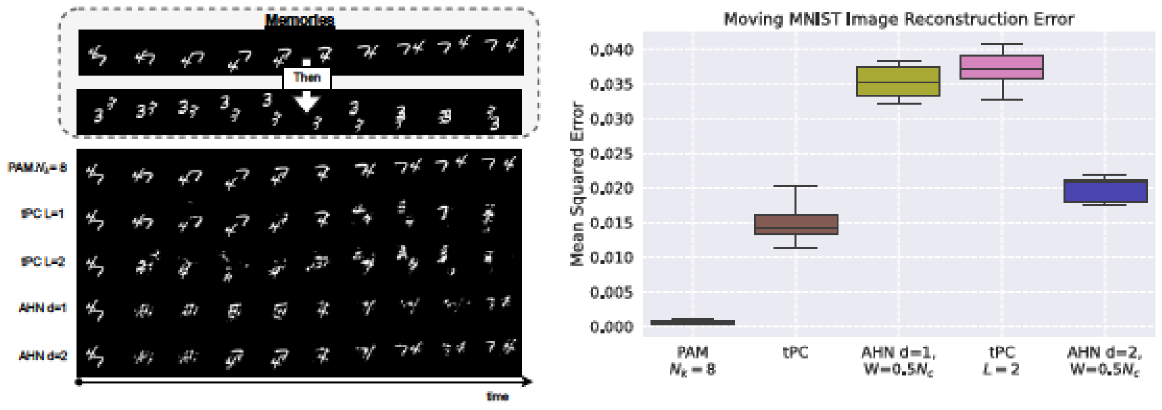


Figure C.9: Additional qualitative and quantitative results on Moving-MNIST. Qualitative catastrophic forgetting visualization on Moving-MNIST and quantitative results on the reconstruction error of 10 Moving-MNIST random examples

for one of experiment as an example. All tables show results on 10 sequences and $N_c = 100$. The BWT metric is calculated as the average of the similarity metric reported in these tables.

Table C.2: Catastophic forgetting experiment results on 10 sequences for PAM with $N_k = 1$. The table shows the mean normalized IoU and standard deviation of previous learned sequences after training on new sequences. The Backward Transfer metric is the average of all the shown numbers. Results are averaged over 10 trials.

	Test / Train	Sequence ID									
		1	2	3	4	5	6	7	8	9	10
Sequence ID	1	-	-	-	-	-	-	-	-	-	-
	2	0.692 ± 0.220	-	-	-	-	-	-	-	-	-
	3	0.654 ± 0.252	0.586 ± 0.314	-	-	-	-	-	-	-	-
	4	0.581 ± 0.200	0.619 ± 0.411	0.759 ± 0.229	-	-	-	-	-	-	-
	5	0.634 ± 0.236	0.553 ± 0.348	0.817 ± 0.200	0.823 ± 0.215	-	-	-	-	-	-
	6	0.617 ± 0.224	0.501 ± 0.322	0.695 ± 0.289	0.546 ± 0.329	0.637 ± 0.299	-	-	-	-	-
	7	0.660 ± 0.246	0.545 ± 0.343	0.706 ± 0.265	0.490 ± 0.239	0.604 ± 0.325	0.716 ± 0.293	-	-	-	-
	8	0.631 ± 0.231	0.657 ± 0.317	0.693 ± 0.292	0.527 ± 0.236	0.633 ± 0.300	0.571 ± 0.297	0.748 ± 0.268	-	-	-
	9	0.700 ± 0.237	0.531 ± 0.357	0.723 ± 0.294	0.645 ± 0.286	0.584 ± 0.277	0.605 ± 0.284	0.613 ± 0.265	0.539 ± 0.267	-	-
	10	0.659 ± 0.235	0.488 ± 0.317	0.602 ± 0.326	0.498 ± 0.260	0.532 ± 0.246	0.630 ± 0.257	0.664 ± 0.327	0.660 ± 0.330	0.575 ± 0.313	-

Table C.3: Catastophic forgetting experiment results on 10 sequences for PAM with . The table shows the mean normalized IoU and standard deviation of previous learned sequences after training on new sequences. The Backward Transfer metric is the average of all the shown numbers. Results are averaged over 10 trials.

	Test / Train	Sequence ID									
		1	2	3	4	5	6	7	8	9	10
Sequence ID	1	-	-	-	-	-	-	-	-	-	-
	2	1.000 ± 0.000	-	-	-	-	-	-	-	-	-
	3	1.000 ± 0.000	1.000 ± 0.000	-	-	-	-	-	-	-	-
	4	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	-	-	-	-	-	-	-
	5	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	-	-	-	-	-	-
	6	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	-	-	-	-	-
	7	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	-	-	-	-
	8	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	-	-	-
	9	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	-	-
	10	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	-

Table C.4: Catastophic forgetting experiment results on 10 sequences for tPC. The table shows the mean normalized IoU and standard deviation of previous learned sequences after training on new sequences. The Backward Transfer metric is the average of all the shown numbers. Results are averaged over 10 trials.

	Test / Train	Sequence ID									
		1	2	3	4	5	6	7	8	9	10
Sequence ID	1	-	-	-	-	-	-	-	-	-	-
	2	0.452 ± 0.230	-	-	-	-	-	-	-	-	-
	3	0.180 ± 0.135	0.360 ± 0.270	-	-	-	-	-	-	-	-
	4	0.148 ± 0.102	0.326 ± 0.257	0.462 ± 0.311	-	-	-	-	-	-	-
	5	0.095 ± 0.047	0.148 ± 0.040	0.253 ± 0.166	0.393 ± 0.339	-	-	-	-	-	-
	6	0.055 ± 0.038	0.102 ± 0.052	0.213 ± 0.245	0.211 ± 0.199	0.344 ± 0.256	-	-	-	-	-
	7	0.068 ± 0.037	0.062 ± 0.050	0.102 ± 0.067	0.103 ± 0.087	0.215 ± 0.197	0.383 ± 0.189	-	-	-	-
	8	0.038 ± 0.035	0.041 ± 0.045	0.052 ± 0.034	0.080 ± 0.074	0.073 ± 0.062	0.232 ± 0.138	0.461 ± 0.326	-	-	-
	9	0.021 ± 0.022	0.032 ± 0.022	0.056 ± 0.042	0.057 ± 0.027	0.079 ± 0.086	0.170 ± 0.129	0.290 ± 0.159	0.284 ± 0.169	-	-
	10	0.016 ± 0.019	0.023 ± 0.028	0.039 ± 0.028	0.032 ± 0.039	0.058 ± 0.058	0.133 ± 0.059	0.187 ± 0.134	0.288 ± 0.302	0.261 ± 0.237	-

Table C.5: Catastophic forgetting experiment results on 10 sequences for AHN with and . The table shows the mean normalized IoU and standard deviation of previous learned sequences after training on new sequences. The Backward Transfer metric is the average of all the shown numbers. Results are averaged over 10 trials.

	Test / Train	Sequence ID									
		1	2	3	4	5	6	7	8	9	10
Sequence ID	1	-	-	-	-	-	-	-	-	-	-
	2	0.689 ± 0.192	-	-	-	-	-	-	-	-	-
	3	0.689 ± 0.192	0.595 ± 0.334	-	-	-	-	-	-	-	-
	4	0.689 ± 0.192	0.595 ± 0.334	0.765 ± 0.238	-	-	-	-	-	-	-
	5	0.689 ± 0.192	0.595 ± 0.334	0.765 ± 0.238	0.780 ± 0.226	-	-	-	-	-	-
	6	0.689 ± 0.192	0.595 ± 0.334	0.765 ± 0.238	0.780 ± 0.226	0.692 ± 0.268	-	-	-	-	-
	7	0.689 ± 0.192	0.595 ± 0.334	0.765 ± 0.238	0.780 ± 0.226	0.692 ± 0.268	0.667 ± 0.226	-	-	-	-
	8	0.689 ± 0.192	0.595 ± 0.334	0.765 ± 0.238	0.780 ± 0.226	0.692 ± 0.268	0.667 ± 0.226	0.734 ± 0.267	-	-	-
	9	0.689 ± 0.192	0.595 ± 0.334	0.765 ± 0.238	0.780 ± 0.226	0.692 ± 0.268	0.667 ± 0.226	0.734 ± 0.267	0.558 ± 0.257	-	-
	10	0.689 ± 0.192	0.595 ± 0.334	0.765 ± 0.238	0.780 ± 0.226	0.692 ± 0.268	0.667 ± 0.226	0.734 ± 0.267	0.558 ± 0.257	0.659 ± 0.293	-

C.5.3 Multiple Possibilities Generation

In this task, we evaluate the models' ability to generate meaningful sequences and recall the full dataset despite presented with multiple valid possibilities. Ideally, the models are expected to sample a single possibility if trained on sequences with ambiguous future continuations (equally valid possibilities). This is a challenging task for most biologically plausible (e.g., tPC, AHN, etc.) and implausible (e.g., transformers, RNNs, etc) models. Most approaches assume the existence of a full set of possibilities and transform the task from regression to classification (e.g., LLM). For vision tasks, some methods (VQ-VAE and its variants) cluster the dense representations to create this set of possibilities and perform classification. We *do not* assume the existence of a full set of possibilities, but instead perform a true generative evaluation as a regression task.

In Figure 5.4 C, we compute the average normalized IoU of the generated words. The models ability to generate a full sequence with high IoU means it can produce sharp single predictions despite being trained on multiple equally valid future predictions. As the number of words increase, the performance of other models decrease as they struggle to model ambiguous future predictions; however, PAM outperforms the other approaches by sampling from these possibilities.

In Figure 5.4 D, we evaluate the ability of the models to recall the dataset words. We compute the recall as the number of valid unique words generated divided by the total number of words in the dataset. Since PAM is a generative stochastic model, the recall increases with every generation. The other methods are deterministic, therefore do not report an increase in dataset recall with more generations. The other methods completely fail in generating any meaningful words. We use an average IoU threshold of 0.9 to classify a generated word as correct, similar to sequence capacity experiments.

In Figure C.10, we provide qualitative results by showing the unique generated words by different models after 5 dataset generations. PAM generates some of the dataset words, but also generates many wrong words. By increasing the context memory neurons to , the model generates many more correct words and reduces the false positives. The other methods cannot generate meaningful words.

Full Dataset	part - call - live - side - both - play - from - move - open - city - left - over - name - form - were - port - more - make - came - hard - most - take - line - some - life - self - like - made - same - near - than - many - late - land - does - this - down - will - farm - hand - turn - work - very - then - been - tell - year - good - must - draw - show - when - said - also - grow - last - know - them - only - read - want - with - what - back - just - such - need - ease - keep - well - tree - look - head - kind - give - page - even - food - home - four - they - long - here - time - that - much - come - have - seem - next - mean - your - walk - went - help - word - find - each - real - high
PAM-4	meem - next - bany - evbe - page - keep - word - came - open - read - kind - went - some - late - seem - tell - city - turn - more - will - bada - move - grow - were - here - keee - near - draw - mean - tree - left - real - such - does - meee - hand - four - very - much - come - back - head - looo - lina - show - with - most - land - from - higa - port - ward - food - whee - good - even - ease - your - only - play - both - name - walk - what - meel - evbr - give - down - year - find - life - want - home - farm - just - form - alea - call - meea - must - hard - alei - each - high - meei - been
PAM-8	name - most - when - turn - more - will - from - land - next - move - side - grow - port - walk - evaf - were - what - here - look - give - near - year - food - find - life - mean - draw - want - page - line - home - farm - just - keep - tree - word - left - came - real - have - call - open - form - good - such - time - help - does - kind - went - even - must - ease - hard - four - hand - also - some - very - high - your - play - come - back - late - both - seem - tell - head - well - city - been - show - with
tPC	hbaa - oaaa - naaa - keaa - laaa - gaaa - thaa - eaaa - mbaa - caaa - raaa - saaa - paaa - veaa - faaa - jaaa - yaaa - waaa - aaaa - daaa - baaa
AHN	oaaa - dcaa - naaa - keaa - laaa - maaa - eaaa - raaa - caaa - saaa - yeaa - paaa - veaa - faaa - haaa - taaa - juca - gcaa - waaa - aaaa - baaa

Figure C.10: Qualitative results showing the generated words from PAM, tPC [241] and AHN [30]. Words highlighted in green are available in the dataset (i.e., True positives). Words highlighted in red are not available in the dataset (i.e., False positives).

C.5.4 Noise Robustness

In Figure 5.3 C, we evaluate noise robustness by plotting each model's performance (Normalized IoU) with varying levels of noise added in an online generation setting. The noisy inputs are created by changing a percentage of the active neurons to different uniformly chosen neurons in the SDR. The noise is computed as a percentage for a fair comparison across different SDR sparsities (e.g., tPC vs. AHN). We show the results for sequences of lengths and no correlation. PAM has the ability to compare the noisy input representation to the learned attractors to recover the correct clean input. Therefore, even when the SDR is completely changed, PAM relies on its predictions and completely ignores the noisy input. During generation, PAM always generates (or corrects a noisy input) from within the predicted set of possibilities. The other approaches use the noisy inputs during recall which affects their performance.

In Figure 5.4 E, we provide qualitative results on the CLEVRER dataset. The memories sequence is learned by all the models, then a noisy sequence is used during generation. We only add noise starting from the second pattern in the input sequence. The results show tPC models performing relatively well, yet still outperformed by PAM . We set in the SDR autoencoder to learn the SDRs used in this experiment. We use 40% noise in this experiment.

In Figure C.11, we perform additional experiments on varying the sequence lengths and the correlation in the sequence, all the other settings remain the same as in the experiment of Figure 5.3 *C*. The results show that with shorter sequences ( ), no noise and no correlation, all the models recall the learned sequence well. When higher correlation is used, 2-layered tPC performs relatively well with short sequences (i.e.,), but fails with longer sequences (i.e.,). The hopfield model fails more with correlation than sequence length. The added noise affects all reported methods except for PAM, due to its ability to rely on its predictions and attractors to clean the noisy signal.

In Figure C.12, we provide an additional qualitative example with similar trend to Figure 5.4 *E*. We also provide quantitative results of CLEVRER averaged over 10 experiments. The mean squared error of the generated sequence for multiple models at different noise levels is reported. It is clear that PAM outperforms all methods, and a 2-layered tPC is the second best.

C.5.5 Efficiency

In Figure 5.3 D, we compare the efficiency of the models and show that PAM is at least two order of magnitude more efficient than tPC with 2 layers. A single layer tPC is almost equivalent to PAM with high context memory of . AHN is highly efficient as the model is not usually trained, but the recall equation is used instead. Therefore, we exclude AHN from the comparison.

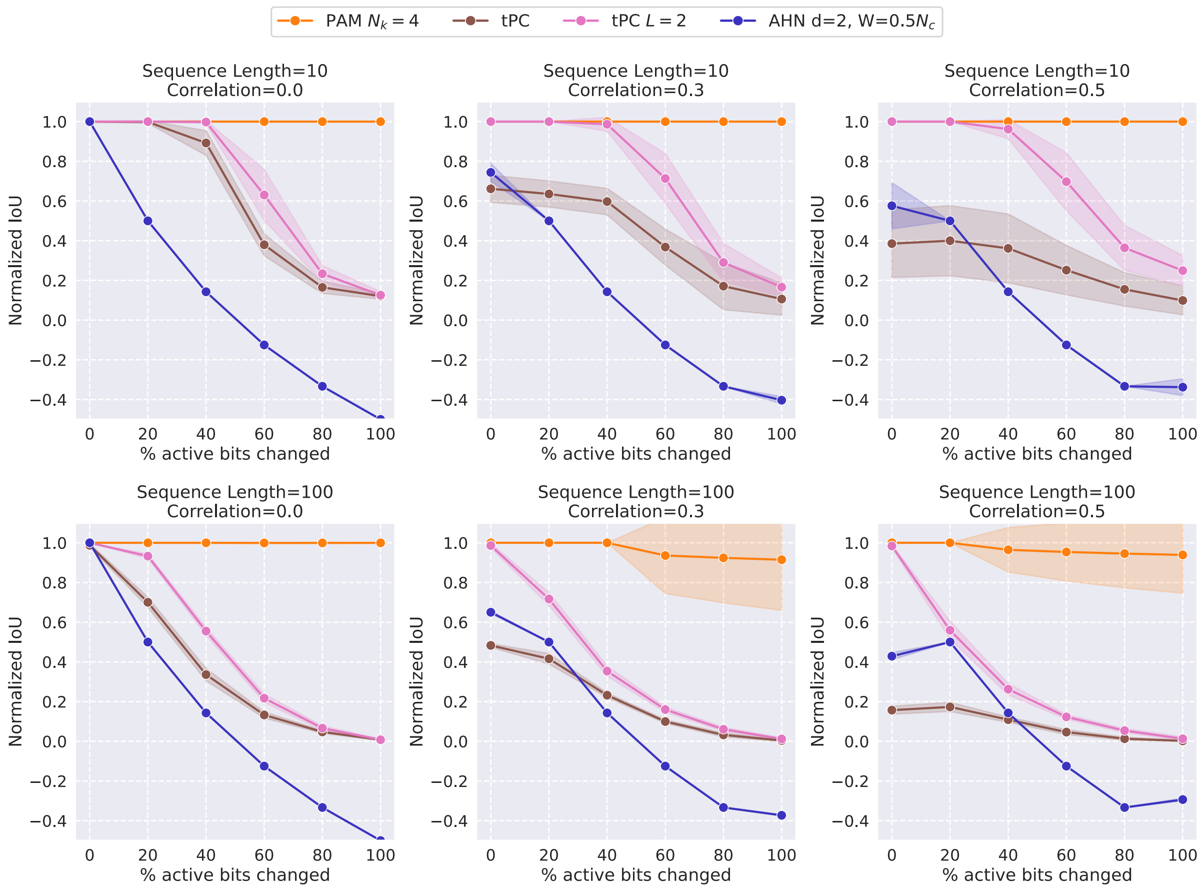


Figure C.11: The effect of noise on online generation with varying sequence lengths and sequence correlations.

C.6 Sparse Distributed Representations

The neocortex stores and represents information using sparse activity patterns, as demonstrated by empirical evidence [6]. Inspired by HTM [83] and neuroscience-based theories of cortical function, we use Sparse Distributed Representations (SDRs) as the main representation format of PAM. An SDR is a sparse binary representation of a cell assembly where only a small fraction of the neurons in the SDR are active at any time. The location of these active neurons encodes the information that is represented by this SDR. In this section we describe some useful properties of SDRs and discuss their robustness to noise as opposed to dense representations.

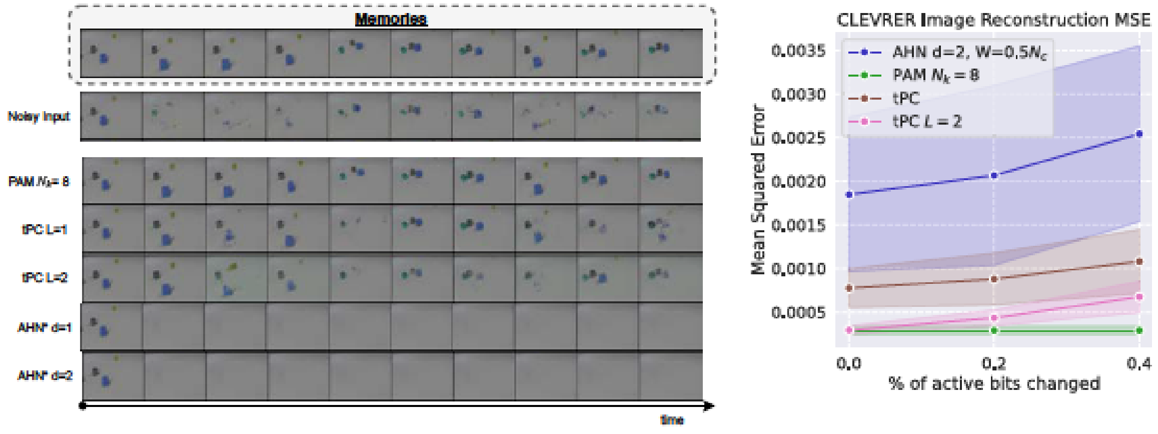


Figure C.12: Additional qualitative and quantitative results on CLEVRER dataset. Qualitative example of online generation with noise on CLEVRER dataset, and quantitative results of reconstruction error over 10 CLEVRER examples at different noise levels.

C.6.1 SDR Properties

SDRs are used to represent rich sensory information in the neocortex as a sparse activity pattern. Therefore, from the mathematical viewpoint, an SDR must have the ability to represent many patterns and easily distinguish between them. The capacity of an SDR can be calculated as the possible combinations of locations where neurons can be active. Consider an SDR with size N and number of active neurons W. The total capacity of this SDR is computed as shown in Equation C.3.

$$\binom{N}{W} = \frac{N!}{W!(N-W)!} \tag{C.3}$$

Based on the above Binomial coefficient equation, it may seem that sparsity is not optimal for capacity as the capacity will be the highest when W is exactly half of N. While capacity is important, we aim to represent multiple possibilities as a union of SDRs and therefore minimize the overlap between them. From an information-theoretic viewpoint, the goal is to minimize *mutual information* between SDRs to ensure that each SDR carries unique information and the union represents a more comprehensive and diverse set of features. We can minimize the expected IoU by using lower sparsities as shown in Theorem 2. In our experiments we use $N = 100$ and $W = 5$, which results in capacity of and an expected IoU of . However, when scaled up to more typical values of SDR sizes and sparsities in the neocortex [6, 83] (i.e., ,), we get capacity of (more than the estimated number of atoms in the observable universe) and expected IoU of . A sparsity of maximizes the mutual information and results in an expected IoU of which cannot be used to represent multiple possibilities as a union of SDRs, in spite of the optimal capacity. In practice, the size of the SDR is increased to increase the capacity, and the sparsity is decreased to minimize the expected overlap.

C.6.2 The Robustness of SDRs

Sparse representations naturally minimize the overlap between random SDRs, therefore they are very tolerant to noise. To visualize this robustness property of SDRs, we design an experiment (Figure C.13) where we train an SDR autoencoder with different sparsities and then decode SDRs at various levels of noise added. When the sparsity is increased to , there is a high chance of overlap between SDRs, therefore a small amount of noise can cause collisions between SDRs. However, a sparsity can tolerate much more noise without overlapping with other SDRs.

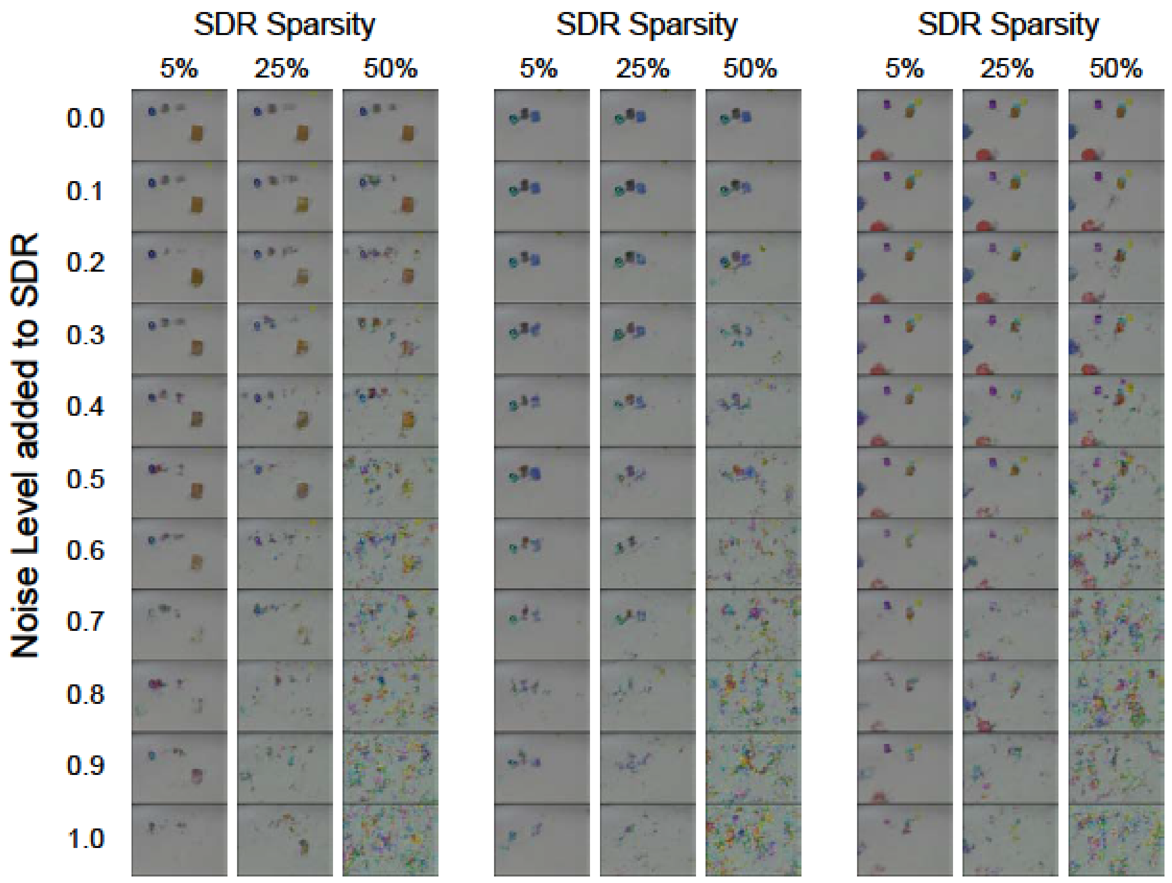


Figure C.13: Three examples of decoding an SDR with different noise levels. The results are shown for SDRs with different sparsities trained in an SDR autoencoder on CLEVRER dataset.

Appendix D: Copyright Clearance Forms

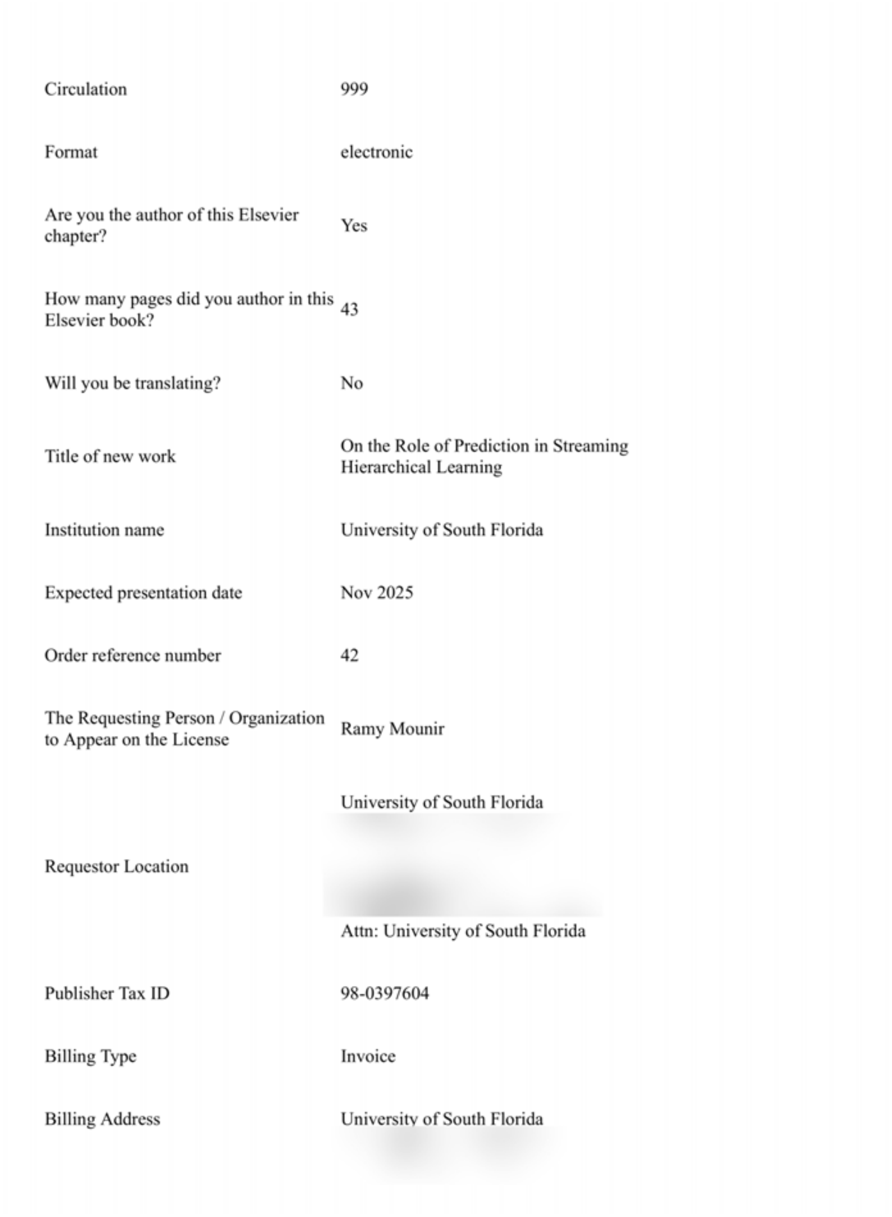
The permissions below is for the use of material in Chapter 1 and 2.

ELSEVIER LICENSE
TERMS AND CONDITIONS

Oct 26, 2024

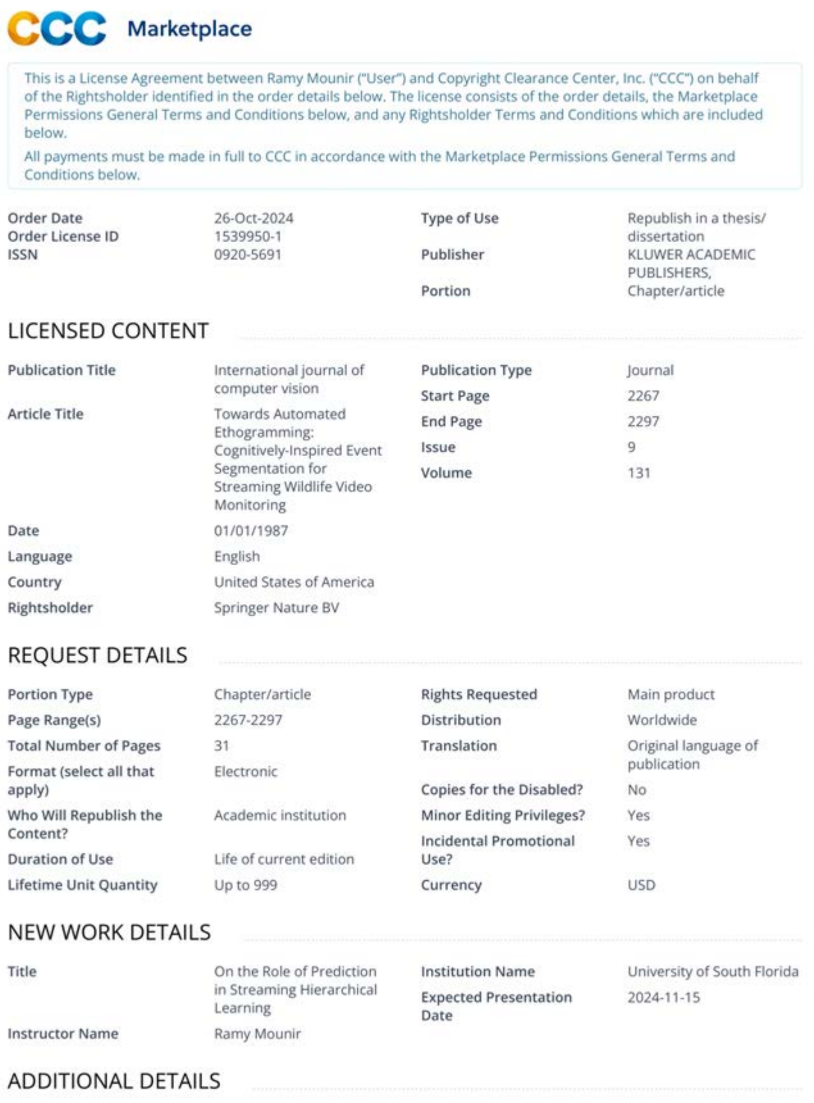
This Agreement between Ramy Mounir ("You") and Elsevier ("Elsevier") consists of your license details and the terms and conditions provided by Elsevier and Copyright Clearance Center.

License Number	5894810896420
License date	Oct 23, 2024
Licensed Content Publisher	Elsevier
Licensed Content Publication	Elsevier Books
Licensed Content Title	Advanced Methods and Deep Learning in Computer Vision
Licensed Content Author	Ramy Mounir,Sathyanarayanan Aakur,Sudeep Sarkar
Licensed Content Date	Jan 1, 2022
Licensed Content Pages	44
Start Page	405
End Page	448
Type of Use	reuse in a thesis/dissertation
Portion	full chapter



Circulation	999
Format	electronic
Are you the author of this Elsevier chapter?	Yes
How many pages did you author in this Elsevier book?	43
Will you be translating?	No
Title of new work	On the Role of Prediction in Streaming Hierarchical Learning
Institution name	University of South Florida
Expected presentation date	Nov 2025
Order reference number	42
The Requesting Person / Organization to Appear on the License	Ramy Mounir
Requestor Location	University of South Florida [redacted] Attn: University of South Florida
Publisher Tax ID	98-0397604
Billing Type	Invoice
Billing Address	University of South Florida [redacted]

The permissions below is for the use of material in Chapter 3.



Marketplace

This is a License Agreement between Ramy Mounir ("User") and Copyright Clearance Center, Inc. ("CCC") on behalf of the Rightsholder identified in the order details below. The license consists of the order details, the Marketplace Permissions General Terms and Conditions below, and any Rightsholder Terms and Conditions which are included below.

All payments must be made in full to CCC in accordance with the Marketplace Permissions General Terms and Conditions below.

Order Date	26-Oct-2024	Type of Use	Republish in a thesis/ dissertation
Order License ID	1539950-1	Publisher	KLUWER ACADEMIC PUBLISHERS,
ISSN	0920-5691	Portion	Chapter/article

LICENSED CONTENT

Publication Title	International journal of computer vision	Publication Type	Journal
Article Title	Towards Automated Ethogramming: Cognitively-Inspired Event Segmentation for Streaming Wildlife Video Monitoring	Start Page	2267
		End Page	2297
		Issue	9
		Volume	131
Date	01/01/1987		
Language	English		
Country	United States of America		
Rightsholder	Springer Nature BV		

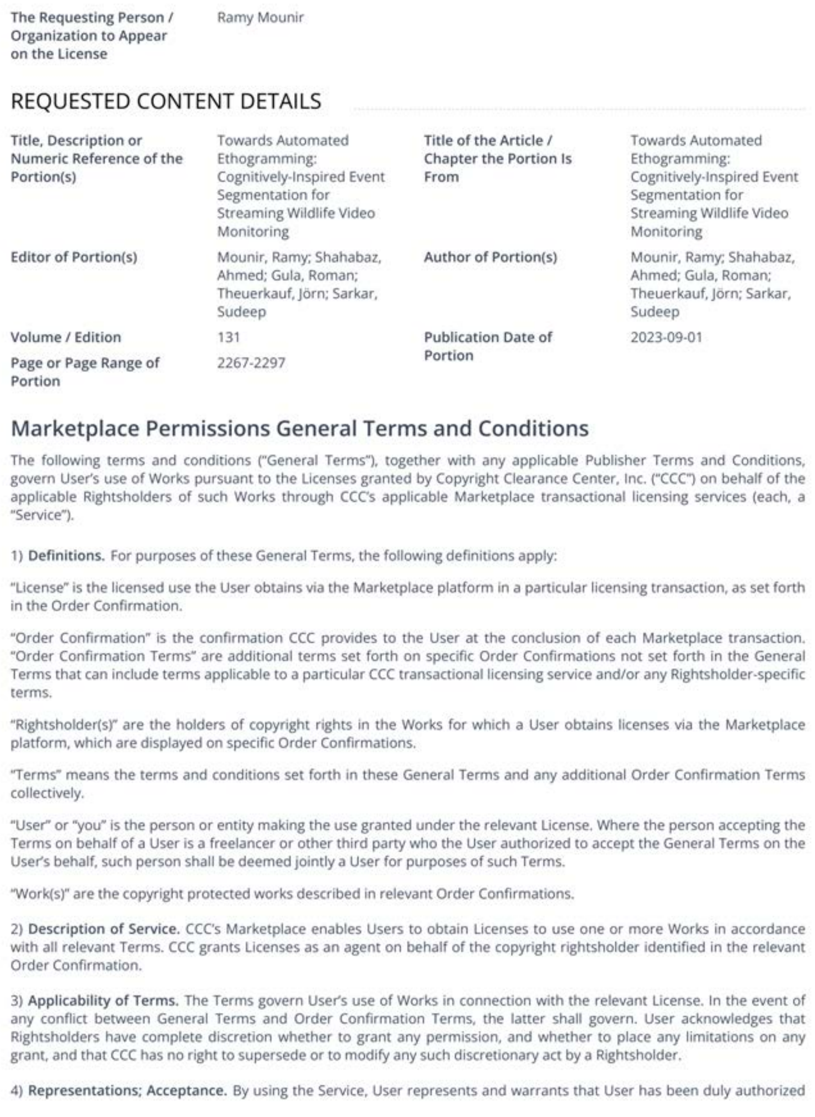
REQUEST DETAILS

Portion Type	Chapter/article	Rights Requested	Main product
Page Range(s)	2267-2297	Distribution	Worldwide
Total Number of Pages	31	Translation	Original language of publication
Format (select all that apply)	Electronic	Copies for the Disabled?	No
Who Will Republish the Content?	Academic institution	Minor Editing Privileges?	Yes
		Incidental Promotional Use?	Yes
Duration of Use	Life of current edition		
Lifetime Unit Quantity	Up to 999	Currency	USD

NEW WORK DETAILS

Title	On the Role of Prediction in Streaming Hierarchical Learning	Institution Name	University of South Florida
		Expected Presentation Date	2024-11-15
Instructor Name	Ramy Mounir		

ADDITIONAL DETAILS



The Requesting Person / Organization to Appear on the License	Ramy Mounir		

REQUESTED CONTENT DETAILS

Title, Description or Numeric Reference of the Portion(s)	Towards Automated Ethogramming: Cognitively-Inspired Event Segmentation for Streaming Wildlife Video Monitoring	**Title of the Article / Chapter the Portion Is From**	Towards Automated Ethogramming: Cognitively-Inspired Event Segmentation for Streaming Wildlife Video Monitoring
Editor of Portion(s)	Mounir, Ramy; Shahabaz, Ahmed; Gula, Roman; Theuerkauf, Jörn; Sarkar, Sudeep	**Author of Portion(s)**	Mounir, Ramy; Shahabaz, Ahmed; Gula, Roman; Theuerkauf, Jörn; Sarkar, Sudeep
Volume / Edition	131	**Publication Date of Portion**	2023-09-01
Page or Page Range of Portion	2267-2297		

Marketplace Permissions General Terms and Conditions

The following terms and conditions ("General Terms"), together with any applicable Publisher Terms and Conditions, govern User's use of Works pursuant to the Licenses granted by Copyright Clearance Center, Inc. ("CCC") on behalf of the applicable Rightsholders of such Works through CCC's applicable Marketplace transactional licensing services (each, a "Service").

1) **Definitions.** For purposes of these General Terms, the following definitions apply:

"License" is the licensed use the User obtains via the Marketplace platform in a particular licensing transaction, as set forth in the Order Confirmation.

"Order Confirmation" is the confirmation CCC provides to the User at the conclusion of each Marketplace transaction. "Order Confirmation Terms" are additional terms set forth on specific Order Confirmations not set forth in the General Terms that can include terms applicable to a particular CCC transactional licensing service and/or any Rightsholder-specific terms.

"Rightsholder(s)" are the holders of copyright rights in the Works for which a User obtains licenses via the Marketplace platform, which are displayed on specific Order Confirmations.

"Terms" means the terms and conditions set forth in these General Terms and any additional Order Confirmation Terms collectively.

"User" or "you" is the person or entity making the use granted under the relevant License. Where the person accepting the Terms on behalf of a User is a freelancer or other third party who the User authorized to accept the General Terms on the User's behalf, such person shall be deemed jointly a User for purposes of such Terms.

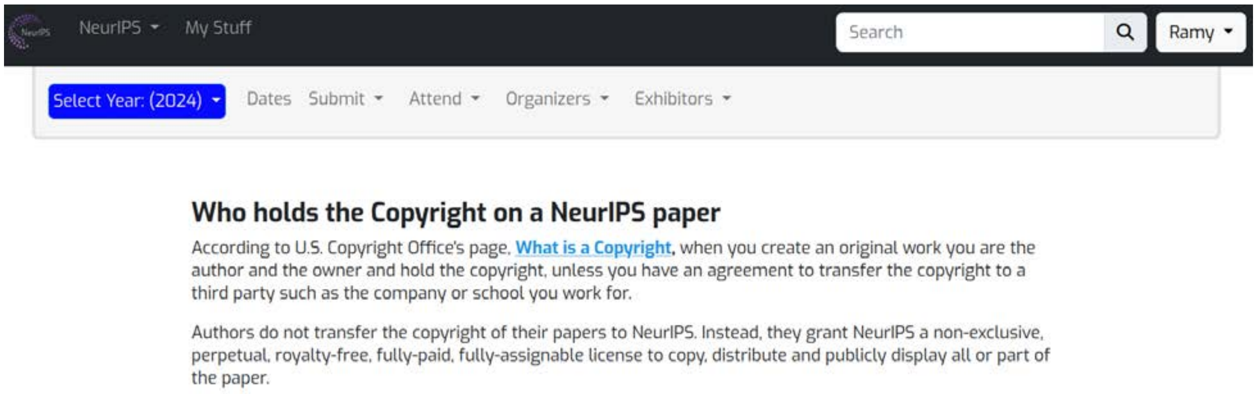
"Work(s)" are the copyright protected works described in relevant Order Confirmations.

2) **Description of Service.** CCC's Marketplace enables Users to obtain Licenses to use one or more Works in accordance with all relevant Terms. CCC grants Licenses as an agent on behalf of the copyright rightsholder identified in the relevant Order Confirmation.

3) **Applicability of Terms.** The Terms govern User's use of Works in connection with the relevant License. In the event of any conflict between General Terms and Order Confirmation Terms, the latter shall govern. User acknowledges that Rightsholders have complete discretion whether to grant any permission, and whether to place any limitations on any grant, and that CCC has no right to supersede or to modify any such discretionary act by a Rightsholder.

4) **Representations; Acceptance.** By using the Service, User represents and warrants that User has been duly authorized

The author owns the copyright of published material in Chapters 4 and 5.



Who holds the Copyright on a NeurIPS paper

According to U.S. Copyright Office's page, What is a Copyright, when you create an original work you are the author and the owner and hold the copyright, unless you have an agreement to transfer the copyright to a third party such as the company or school you work for.

Authors do not transfer the copyright of their papers to NeurIPS. Instead, they grant NeurIPS a non-exclusive, perpetual, royalty-free, fully-paid, fully-assignable license to copy, distribute and publicly display all or part of the paper.