# Survey: Exploiting Data Redundancy for Optimization of Deep Learning

JOU-AN CHEN, Department of Computer Science, North Carolina State University, USA
WEI NIU and BIN REN, Department of Computer Science, William & Mary, USA
YANZHI WANG, Department of Electrical and Computer Engineering, Northeastern University, USA
XIPENG SHEN, Department of Computer Science, North Carolina State University, USA

Data redundancy is ubiquitous in the inputs and intermediate results of **Deep Neural Networks (DNN)**. It offers many significant opportunities for improving DNN performance and efficiency and has been explored in a large body of work. These studies have scattered in many venues across several years. The targets they focus on range from images to videos and texts, and the techniques they use to detect and exploit data redundancy also vary in many aspects. There is not yet a systematic examination and summary of the many efforts, making it difficult for researchers to get a comprehensive view of the prior work, the state of the art, differences and shared principles, and the areas and directions yet to explore. This article tries to fill the void. It surveys hundreds of recent papers on the topic, introduces a novel taxonomy to put the various techniques into a single categorization framework, offers a comprehensive description of the main methods used for exploiting data redundancy in improving multiple kinds of DNNs on data, and points out a set of research opportunities for future exploration.

CCS Concepts: • **Computing methodologies** → **Machine Learning**; **Knowledge representation and reasoning;**

Additional Key Words and Phrases: Data redundancy, representation redundancy, deep neural network, convolutional neural network, transformer

**ACM Reference format:**
Jou-An Chen, Wei Niu, Bin Ren, Yanzhi Wang, and Xipeng Shen. 2023. Survey: Exploiting Data Redundancy for Optimization of Deep Learning. *ACM Comput. Surv.* 55, 10, Article 212 (February 2023), 38 pages.
https://doi.org/10.1145/3564663

## 1 INTRODUCTION

Deep learning, primarily powered by **Deep Neural Networks (DNN)**, has repeatedly demonstrated its success and tremendous potential in revolutionizing the development of Artificial Intelligence and its applications in various domains. At a high level, DNN has two pillars: algorithm

Authors' addresses: J.-A. Chen and X. Shen, Department of Computer Science, North Carolina State University, USA; emails: {jchen73, xshen5}@ncsu.edu; W. Niu and B. Ren, Department of Computer Science, William & Mary, USA; emails: wniu@email.wm.edu, bren@cs.wm.edu; Y. Wang, Department of Electrical and Computer Engineering, Northeastern University, USA; email: yanz.wang@northeastern.edu.

0360-0300/2023/02-ART212 $15.00
https://doi.org/10.1145/3564663

and data. DNN models embody the algorithm, and data is represented by DNN inputs and intermediate results (or called *activation maps*). Both algorithm and data are essential for the quality of a DNN, determining its accuracy, speed, size, energy efficiency, robustness, and so on.

For DNNs, redundancy exists in both models and data. As many studies have shown, reducing some DNN layers or parameters or precision often leaves the model accuracy intact. The observation has prompted a large body of research in DNN compression, which tries to find effective ways to reduce the size of a DNN model and achieve more compact models or/and faster speeds. Many of the efforts resort to model pruning and quantization, while some compress models in the frequency domains [153] or optimize attentions in Transformer models [32, 113, 147]. We call those efforts model redundancy exploitations. Several recent survey papers [9, 28, 31, 51, 61, 76] have provided comprehensive overviews on the topic.

Data redundancy exploitation is no less critical for DNN and has received much and increasing attention in recent several years as well. Many studies have noticed similar patches within an image, activation maps, or between adjacent video frames. Other studies have confirmed that some words in a sentence carry no significant meaning for the sentence. These observations have prompted many recent efforts in creating methods for detecting and leveraging the redundancy of various dimensions in all kinds of data. Hundreds of papers have been published, scattering in many venues over several years. But unlike model redundancy, there is not yet a systematic examination and summary of the many efforts, making it difficult for researchers to get a comprehensive view of the prior work, to learn about state of the art, to understand the differences and common principles among the many published studies, or to find out the areas and directions yet to explore.

To the best of our knowledge, this paper offers the first one-stop resource for people interested in learning about studies on data redundancy for DNN. It provides a holistic view of the various techniques and their connections, offers insights on the limitations of state of the art, and potential directions and opportunities for the future to explore. The paper introduces the first known taxonomy on DNN data redundancy exploitation, putting the various topics' various techniques into a single categorization framework. It offers a comprehensive description of the main techniques used for detecting and exploiting data redundancy in improving multiple kinds of DNNs, from CNNs to RNNs, Transformers, and so on. It presents numerous data redundancy opportunities in images, videos, and texts and how existing studies tap into them with various techniques at a spectrum of granularities and scopes. It discusses the commonalities among the techniques, their differences, limitations, and promising research directions worth future explorations.

We organize the rest of the paper as follows. Section 2 provides a formal definition of data redundancy and defines the scope of this survey. Section 3 presents a taxonomy of studies on DNN data redundancy. Sections 4–6 describe the practical techniques that prior work has developed in detecting and exploiting data redundancy for DNNs respectively on images, videos, and text data. Section 7 discusses the limitations of existing explorations and points out several future directions. Section 8 concludes the survey with a summary.

## 2 TERMINOLOGY AND SCOPE OF DISCUSSION

We start this section with clarifications of several terms essential for the rest of the paper, then define the scope of this survey, and explain the relations with several relevant concepts to our focus.

### 2.1 Terminology

**Activation Map.** In the traditional terminology, an *activation map* (or called feature map) is the activations of filters applied to the input of a DNN, also to the output of the previous layer. For

the visual representation of an image or a single image frame in a video with RGB channels, it is a 3D tensor with $Height \times Width \times Channel$; for a video clip, it is a 4D tensor with $Height \times Width \times Channel \times Depth$, the *Depth* is the number of frames. For ease of explanation, in this article, we extend the term *activation map* to include the inputs to a DNN (the values of the input layer neurons).

**Hidden State.** For text representation, before input to the first layer of the model, the text is transformed into word vectors representation via a *word embedding* (the collective name for a set of language modeling and feature learning techniques in **natural language processing (NLP)** where words or phrases from the vocabulary are mapped to vectors of real numbers [43]). Multiplied with weights, the output activations of a layer are called a *hidden state* (which is also called *encoder state*). It is represented as a 2D tensor with *sequence length* $\times$ *dimensionality of word embedding*.

**Redundancy.** According to the dictionary [117], *redundancy* is *the state of being not or no longer needed or valuable*. It is hence a concept relative to a particular purpose. In the context of DNN, being useful usually refers to contributing to the quality of the DNN outputs, which is often measured by a certain kind of accuracy metric. *Data redundancy* in DNN is hence defined as the data that is not or no longer useful for the quality of the outputs of DNN. *Data redundancy exploitation* for DNN refers to techniques that try to avoid data redundancy while keeping DNN outputs meeting the needs.

### 2.2 Scope

Data redundancy in DNN manifests in different kinds of forms. Regardless of the concrete structure, data redundancy in DNN must fall into one of the following three categories: (1) **Repeated information**: Some part of the data conveys the identical (or similar) information as some other parts do. (2) **Irrelevant information**: The information conveyed by some part of the data is irrelevant to the targeted outputs of the DNN. (3) **Over-detailed information**: The information is conveyed in an unnecessarily detailed manner (e.g., image resolution). So any technique that tries to make a DNN avoid spending time and computations on any of the three types of information can be regarded as a technique for data redundancy exploitation. However, this general definition would blur the boundary between DNN algorithm, model, and data-centered optimizations. Some model optimizations (e.g., channel pruning [66, 70, 71, 101, 110, 148, 164, 170]) are fundamentally driven by data redundancy (e.g., redundancy across channels). There are already surveys, particularly on model optimizations, but this survey focuses on data redundancy exploitation beyond model optimizations. These techniques help DNN avoid data redundancy in its computations to efficiently execute DNN in inference or training or eliminate noise to boost DNN accuracy.

There is a body of work on considering data redundancy in the hardware design of DNN accelerators, such as cache buffer designs for data reuse in 2D CNNs [68, 82, 89, 111, 114, 128, 150] and 3D CNNs [19, 44, 67, 132, 149, 151, 152], or integrating activation pruning into the hardware architecture designs [123, 129]. The techniques reduce the number of computations and bring speed and energy benefits. In this survey, we mainly focus our discussion on *software-based techniques* for exploiting data redundancy.

### 2.3 Relations with Relevant Concepts

Several concepts are closely related to data redundancy exploitation. We next describe the relations with these concepts to further clarify our scope.

**Model Redundancy.** Another aspect of redundancy in DNNs is model redundancy, referring to the redundancy within the parameters and architecture of a DNN. Model compression [1, 9, 25, 28, 31, 51, 61, 76, 90], including knowledge distillation [69, 146], parameter pruning, and weight

quantization, are popular approaches to exploit model redundancy. This article focuses on *data redundancy* of DNN, which refers to the redundancy within the input data of each DNN layer (usually in the form of multi-dimensional tensors).

**Data Preprocessing and Augmentation.** Before data are fed into a DNN model, they often go through a preprocessing process. For training, that process is sometimes part of dataset augmentation [33, 136], increasing data variance to increase the models' generality. In computer vision, example operations include rotating, inverting, equalizing, color changes, brightness adjustment, and so on. The samples created by the transformations naturally carry some redundancy with the original data. For instance, with images inverted, two images contain the same set of pixel values, despite that the positions of individual image patches differ in the images. Such redundancy is implicitly considered when a technique exploits redundancy in input data. This survey exploits data redundancy and leaves data augmentation out of the scope.

**Feature Extraction and Data Embedding.** *Feature extraction* is a term in machine learning, referring to techniques that select or derive some features from a set of data that are regarded as most relevant to a specific machine learning task. Therefore, theoretically speaking, feature extraction can be viewed as a kind of exploitation of data redundancy as it reduces the raw data to an often smaller set of features. In a similar vein, *data embeddings* that map raw data to vectors in a space smaller than the original data space could also be regarded as a kind of reduction of data redundancy. Studies on these topics are usually considered as a separate research area named *feature extraction and representation*. They are typically not designed explicitly to exploit data redundancy, even though they may sometimes show such effects. We leave discussions on these studies out of the main focus of this paper.

**Dataset Selection.** In the learning phase of DNN, *training data selection* [45, 47, 163] is a process that tries to select the training data appropriate for the learner to achieve the learning task. To a certain degree, it may also remove some redundancy in the dataset (e.g., some data items in an over-sampled population). In this survey, we do not focus our discussion on this direction but on how data redundancy is addressed during the execution of the model (after the training dataset is selected or during inference).

## 3 TAXONOMY

Many studies detect or leverage data redundancy in DNN from several angles. A taxonomy that puts all the investigations into one categorization framework is essential for a holistic view of their differences, tradeoffs, and shared principles. The taxonomy also offers a comprehensive view at the various dimensions of DNN data redundancy elimination, which can potentially guide the design of new redundancy elimination techniques.

Figure 1 presents a taxonomy we built after surveying hundreds of papers on DNN data redundancy exploitation. As far as we know, this is the first taxonomy on this topic. It shows the six most essential dimensions in DNN data redundancy exploitation. Data redundancy exploitation is usually the combination of one or more items in each dimension. We explain each of the dimensions next.

### 3.1 Granularity

This dimension refers to the unit of the data examined for data redundancy, which we will call *data unit* in the following discussions. There are five granularities listed below in increasing order in size.

- **Bit:** At this granularity, every bit in a value is the unit for examination. Data quantization and precision relaxation to the shorter representation of a value are commonly seen

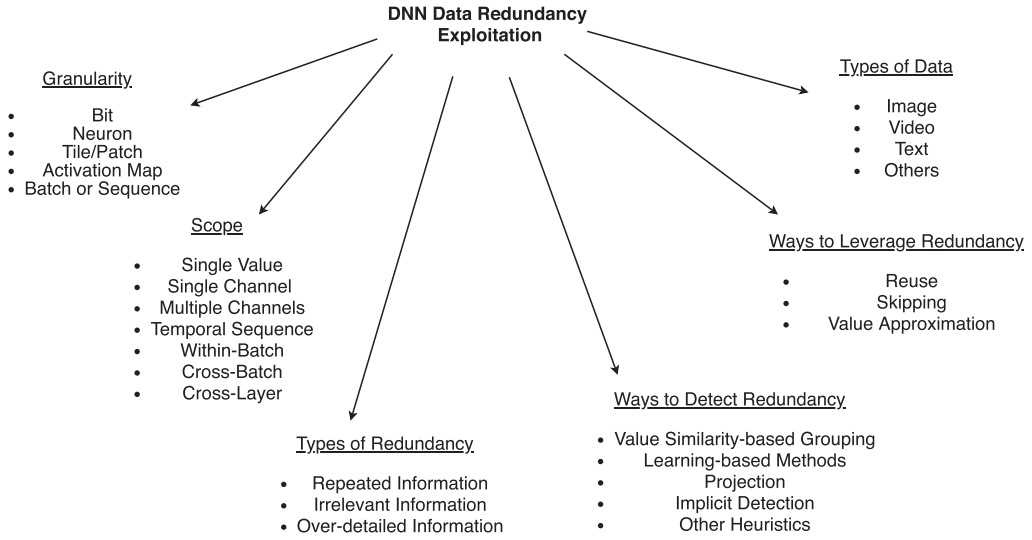


Fig. 1. Taxonomy of data redundancy exploitation in DNN.

strategies exploiting bit-level redundancy. Besides, bit representation can also be used directly for pattern-based bucketing. In RNSNet [128], for instance, an input value is transformed into its n-bits binary format and gets represented with a **residue number system (RNS)**. The multiplications in neural networks are simplified to only addition and memory lookup, allowing memory-friendly operations.

- **Neuron:** The unit at this granularity is the value of a single neuron. An image is a pixel value at one channel or the value carried by a neuron in a DNN.
- **Tile/Patch:** The values carried by a set of neurons, which correspond to a part of an activation map. It could be, for instance, a bounding box or **region of interest (RoI):** or channel in an image; in the case of text, it could be one or multiple sub-word embeddings.
- **Activation Map:** The values in an entire activation map.
- **Batch or Sequence:** A set of activation maps that the DNN carries at a layer, either at once or in order. The activation maps within the set can be either related or unrelated to each other. For example, sequential sentence input in a paragraph to a DNN can be correlated, while in a randomly shuffled set of training images, image activation maps are generally unrelated.

In general, the larger the granularity is, the less redundancy there is, but at the same time, the ratio between the benefits from removing a redundancy and the overhead in finding the redundancy is larger[1]: Avoiding processing a whole batch of images saves more time than avoiding processing a neuron, but the chances of finding two identical batches are generally smaller than finding two identical neurons. On the other hand, the removals of different granularities of redundancy are not exclusive; one optimization could exploit redundancies at multiple granularities.

### 3.2 Scope of Consideration

This dimension is about the scope in which different data units are compared to identify similar units and hence redundancy. This dimension is related to the previous extent, granularity, but

[1]Bit level is special, limited by the number of bits in one value, which is usually up to 32.

differ: For a given data unit, such as a patch of activation map, the comparisons for similar units can still vary (e.g., within one activation map, across activation maps in a batch, or batches). Specifically, the scopes considered in prior studies fall into one or more of the following.

- **Single Value:** Redundancy within a single representation of a value—the corresponding granularity is a bit.
- **Single Channel Activation Map:** In the case of an image, it refers to redundancy within a single channel activation map; in the case of text, it relates to redundancy within a single hidden state. The corresponding granularity can be either neuron or tile/patch.
- **Multiple Channels of Activation Map:** This is for visual data (image and video) only. Redundancy across multiple channels of an activation map between neurons, patches/tiles, or single-channel activation maps.
- **Temporal Sequence of Activation Maps:** In the case of video, it refers to redundancy across a single video clip representation (4D tensor) between neurons, patches/tiles, single-channel activation maps, or activation maps. In the case of sequential text sequence input, it refers to redundancy across hidden states of the same sentence label (token type id), between neurons, sub-word embeddings, or single-sequence hidden states.
- **Within a Batch:** Inputs to a DNN are often provided in a batch each time. The scope for data redundancy consideration can be across inputs within a batch.
- **Cross Batches:** The scope can also cross the boundaries of batches of inputs.
- **Cross Layers:** All the earlier scopes typically assume that the comparisons are about activation maps at the same layer of a Neural Network. Some studies even expand the scope to activation maps on different layers of a Neural Network [38, 119].

In general, a large scope subsumes a smaller scope: Redundancy discoverable in a smaller scope must be discoverable in a larger scope. But on the other hand, checking values in a larger scope also entails more overhead and complexity.

### 3.3 Types of Data Redundancy

Data redundancy exploitation techniques can also be categorized based on what types of data redundancy they exploit. As earlier sections have mentioned, there are mainly three types of data redundancy. We list them here for the completeness of the discussion on the taxonomy.

- **Repeated Information:** Similarity or duplication between data units. When the same operations operate on either identical or similar data units (e.g., multiplications on the same values, convolutions on similar image tiles), the results could be indistinguishable; the computations are redundant.
- **Irrelevant Information:** Unnecessary data units; if they are removed, will not harm the DNN computation results.
- **Over-detailed Information:** High-precision representation is not always necessary. For example, high-resolution images can be replaced with their low-resolution approximation in some application scenarios without causing the DNN to lose accuracy. Activation maps with lower precision sometimes give the same results for a DNN. A video with more frames than needed is another example. A sub-word vector with a higher-dimension representation (e.g., $1 \times 512$ versus $1 \times 256$ for a single word representation) or a higher precision value representation is also an example.

The types of data redundancy define how redundancy appears. These different types of redundancy are complementary to each other; they require different ways to detect but at the same time can be capitalized together. Previous work has each focused on one of the three types of

redundancy; it remains yet to investigate how to effectively combine the removals of multiple types of redundancy in one framework. Among the potential challenges, how to minimize the overhead is one of them.

### 3.4 Ways to Detect Data Redundancy

The techniques can also be classified based on the ways they detect redundancy:

- **Value Similarity-based Grouping:** This includes all kinds of similarity-based data grouping methods (e.g., various data clustering methods).
- **Learning-based Methods:** These methods learn data redundancy on some data samples (often sampled from the training or validation datasets) and apply the knowledge at runtime execution of the model. An example is the learning-based quantization of activation maps.
- **Projection:** These techniques project the data representation to another domain space (e.g., frequency domain) and infer redundancy in that space.
- **Implicit Detection:** These techniques assume redundancy exists in a specific scope based on some prior knowledge about the domain. One of the examples is the techniques that leverage similarities between neighboring video frames.
- **Other Heuristics:** Methods that leverage some other domain-specific heuristics, such as the relative attentions or the average number of zeros appearing at a specific pixel location.

The different ways apply to different domains and scenarios. The first two in the list are general, the third applies to the domains where projection across certain spaces makes sense, and the fourth and fifth apply to domains where there is already certain prior knowledge about the redundancy in the domains.

### 3.5 Ways to Leverage Redundancy

The different techniques take different ways to leverage data redundancy. They, however, in principle fall into the following three main categories.

- **Reuse:** This approach reuses computation results on some data items for other data items. It applies to a similar kind of data redundancy. For example, Deep Reuse [116] applies ***Locality Sensitive Hashing (LSH)*** [56], a fast unsupervised clustering scheme, to cluster data into similarity groups, and then uses the results computed on the cluster centroids as the results for all data items in the same cluster.
- **Skipping:** This approach skips computations on some data items. It applies to data redundancy caused by irrelevant data items. For instance, in Perforated CNN [49], the authors deal with spatial redundancy by masking some pixels in CNN feature maps. It also includes techniques that select a subset of data representations. Some work does ad-hoc sampling, while some others make the selection in a more sophisticated way. In PoWER-BERT [58], for instance, a retention configuration is defined to observe that cosine-similarity between vectors progressively increases as the layer goes deeper. Based on that setting, they employ strategies for word-vector selection. The retention configuration is further designed into learnable hyper-parameters.
- **Value Approximation:** This approach uses an approximation to generate data representation. It corresponds to data redundancy caused by over-detailed information. *Quantization* and *binarization* on data items fall into this category. Previous reviews [76, 124, 137] have provided comprehensive coverage on both research topics, so we do not emphasize them in the following discussion.

These three ways to leverage redundancy are complementary. Even though only the third in the list carries "approximation" in the name, all three could cause accuracy loss: *Reuse* may reuse

results across similar but not identical data, and similarly, *skipping* may skip the computations of some similar but not identical data. These ways of leveraging redundancy can be used together. Some prior studies have already done that. One of them [38], for instance, exploits a *layer selector* ("skipping") and *correlation clustering* ("reuse") practice at the same time.

### 3.6 Types of Data

We can also classify the studies on data redundancy based on the types of data they deal with. Although there are many kinds of data, most existing studies on data redundancy are on the following three types of data:

- **Images.** This type of data includes various kinds of images, including those generated from higher dimension sensors (e.g., those collected through Lidar).
- **Videos.** This type of data features an extra-temporal dimension than images, which provides special opportunities for data redundancy exploitation.
- **Texts.** This type of data consists of texts from various kinds of sources, usually written in Natural Languages.

There are some other types of data that DNN has been applied to, such as graphs, genes, computer programs, and so on. We will briefly discuss them in Section 7.

### 3.7 Use of the Taxonomy

As far as we know, this is the first taxonomy on data redundancy exploitation. It was created based on our survey of hundreds of papers on data redundancy exploitation. The taxonomy can be used to position work in the big picture, to recognize the possible missing opportunities yet to explore for a certain kind of data, to guide the designs of future DNNs for efficiency, and to assist the possible creation of future automatic frameworks that may apply redundancy exploitation for new Deep Learning tasks.

We categorize a set of representative papers listed in Table 1 using the taxonomy. We will discuss them in more detail in the next several sections. Specifically, we organize the following discussions based on several layers of the taxonomy. At the highest level, we divide the studies based on the types of data they handle into three sections, with Section 4 on images, Section 5 on videos, and Section 6 on texts. We organize the various studies based on a dimension that captures the most prominent differences among the studies in each area.

In Section 4, we divide studies on image data redundancy based on the types of redundancy they are dealing with, as the number of works in the three sub-categories (*Repeated Information*, *Irrelevant Information*, and *Over-detailed Information*) are relatively balanced and offers comprehensive coverage of the differences among the techniques. We point out the discrepancies between the work in each group in other dimensions (e.g., data representations, redundancy exploitation techniques) when necessary during the discussion. While many image redundancy exploitation techniques could potentially be applied to each frame in a video, Section 5 focuses on the methods specially designed for videos. These efforts extend the work in Section 4 by exploring how redundancy is leveraged on the patch unit and the temporal dimension. Section 6 discusses how data redundancy in the text is handled. Most of the work is based on Transformer models. In both Sections 5 and 6, the primary dimension we use to group the various explorations is how data redundancy is exploited.

## 4 LEVERAGE REDUNDANCY IN IMAGE DATA

Image data are generally presented as three-dimensional pixel values (height, width, and channel). The inherent locality within data values provides opportunities for advanced optimizations. This

Table 1. Some Representative Papers on Data Redundancy Exploitation

| Data Type | Paper | Redundancy Type | Granularity | Scope | Detection | Exploitation |
|---|---|---|---|---|---|---|
| Image | [116] | Repeated | Patch/Tile | Multiple Channels, Within-Batch, Cross-Batch | Similarity | Reuse |
| | [55] | Irrelevant, Repeated | Bit, Neuron | Single Value, Cross-Layer | Learning, Similarity | Value Approximation, Reuse |
| | [119] | Repeated | Activation Map | Cross-Layer | Similarity | Reuse |
| | [39] | Repeated | Activation Map | Multiple Channels | Similarity | Reuse |
| | [18] | Repeated | Activation Map | Single Channel | Learning | Value Approximation, Reuse |
| | [49] | Irrelevant | Neuron | Multiple Channels | Implicit Detection, Learning | Skipping |
| | [2, 73, 135] | Irrelevant | Neuron | Multiple Channels | Heuristic | Skipping |
| | [22, 143] | Irrelevant | Neuron | Multiple Channels | Learning | Skipping |
| | [54] | Over-detailed | Activation Map | Multiple Channel | Learning | Skipping |
| | [78] | Irrelevant | Neuron | Multiple Channels | Heuristic | Skipping (Subset Selection) |
| | [98] | Irrelevant | Neuron | Cross-Layer | Heuristic | Skipping (Subset Selection) |
| | [75] | Irrelevant | Neuron | Multiple Channels, Cross-Layer | Learning | Skipping (Subset Selection) |
| | [20, 24, 30] | Over-detailed | Activation Map | Multiple Channels | Projection | Value Approximation |
| | [27] | Over-detailed | Activation Map | Multiple Channels | Implicit Detection | Skipping (Ad-hoc Sampling) |
| | [52, 100] | Irrelevant | Patch/Tile | Multiple Channels | Learning | Skipping (Subset Selection) |
| Video | [84] | Irrelevant | Activation Map | Temporal Sequence | Similarity | Skipping |
| | [15, 16] | Irrelevant | Neuron | Temporal Sequence | Similarity | Skipping |
| | [29] | Over-detailed | Patch/Tile | Multiple Channels | Learning | Value Approximation |
| | [162] | Irrelevant | Patch/Tile | Multiple Channels | Implicit Detection | Skipping (Subset Selection) |
| | [112] | Repeated | Patch/Tile | Temporal Sequence | Implicit Detection | Skipping (Subset Selection) |
| | [4, 94, 154, 156, 160] | Over-detailed | Activation Map | Temporal Sequence | Learning | Skipping (Subset Selection) |
| | [142] | Over-detailed | Batch or Sequence | Temporal Sequence | Learning | Skipping (Subset Selection) |
| | [167–169] | Repeated | Activation Map | Temporal Sequence | Implicit Detection | Reuse (Temporal Propagation) |
| | [86, 87] | Repeated | Activation Map | Temporal Sequence | Implicit Detection | Reuse (Temporal Propagation) |
| | [21] | Repeated | Activation Map | Temporal Sequence | Implicit Detection | Reuse (Temporal Propagation) |
| | [165] | Repeated | Activation Map | Temporal Sequence | Implicit Detection | Reuse (Temporal Propagation) |
| Text | [35, 108] | Over-detailed | Patch/Tile | Activation Map | Implicit Detection | Skipping (Ad-hoc Sampling) |
| | [58] | Irrelevant | Patch/Tile | Activation Map | Learning | Skipping (Subset Selection) |
| | [92] | Repeated | Patch/Tile | Activation Map | Similarity | Reuse |
| | [38] | Repeated, Irrelevant | Batch or Sequence | Temporal Sequence, Cross-Layer | Similarity | Reuse, Skipping (Subset Selection) |

section groups the relevant studies at a high level based on the types of redundancy they mainly target: *repeated information*, *irrelevant information*, and *over-detailed information*. The discussion of each kind then divides the techniques to detect and handle data redundancy. Table 2 presents these works by their applications (tasks) and reported performance for fast reference. The second column ("Paper") shows the references to the relevant papers; the third column ("Code") shows the link to the source code of the implementation; and the fourth column ("Key Idea") briefly summarizes the key idea of each work. The fifth column ("Performance") summarizes the performance gains by the data redundant optimization techniques.

### 4.1 Repeated Information

Due to the common coherence of data values in an image, image data often show a certain degree of data locality along the spatial dimensions. The most common optimizations take advantage of the value similarities among pixels. The initial motivation is to minimize the storage space of CNN's intermediate activation maps on a resource-constrained platform and reduce the data movements between CPU and GPU and between processing units and memory. The optimizations also reduce the number of computations and improve computation speeds.

Table (or cache) lookup [39, 68, 111, 114, 119, 125] and clustering [89, 116] are the most typical approaches taken to discover and leverage repeated information in images.

To detect similarity among data points, the various techniques differ in the level of pixel units used for the similarity detection, some at the level of bits [82, 111, 128], some at the level of pixels [125], patches (or tiles, sub-vectors) [68, 89, 116], or even entire feature maps as a whole [39, 82, 114, 119]. Some use clustering with predefined distance metrics and thresholds to identify the similarity between units, while others use hash functions like locality sensitive hashing [116] or bloom filters [82]. RNSnet [128] utilizes a unique bit-handling technique that maps the binary representation into the **Residue Number System (RNS)**. Each binary representation is divided by a modulus set; with the values represented by the corresponding modules and reminders, the technique efficiently identifies similar values. The method could be regarded as a kind of hashing function.

After gathering similar data points, each bucket (or cluster) of the data is represented by usually one or a few data points in the bucket. The determination of such representatives depends on the

Table 2. Summary of Works on Image Data Redundancy

| Application | Paper | Code | Key Idea | Performance |
|---|---|---|---|---|
| image classification | [115, 116] | N/A | *Deep Reuse* clusters the sub-vectors in activation maps by similarity and reuses the cluster centroids' computation results | 2.41×–3.64× speedup for AlexNet and 2.26×–3.35× for VGG-16 on GPU |
| | [55] | N/A | Sparsify the activation maps | 2.32× speedup on LeNet-5, 1.61× on MobileNet-v1, 2.12× on Inception-v3, 1.77× on ResNet-18, and 1.94× on ResNet-34 |
| | [119] | N/A | Use a feature map cache to reuse similar feature maps' computation results | Evaluated on AlexNet, GoogLeNet, VGG-16, VGG-19, ResNet-50, and ResNet-101, giving 1.06×–1.34× average speedup |
| | [49] | [48] | *Perforated CNNs* applies perforation masks to skip pixels or patches on activation maps | AlexNet with 2×–3.6× speedup on CPU and 2×–4× on GPU; VGG-16 with 2×–4× speedup on CPU and 2×–4× on GPU; NIN [103] with 2.2×–4.2× speedup on CPU and 2.1×–3.5× on GPU |
| | [2] | N/A | *SnaPEA* predicts the signs of activation outputs earlier to skip potential zero values that appear after the ReLU activations | 28% speedup and 16% energy reduction with no accuracy loss on CNNs (AlexNet, GoogleLeNet, SqueezeNet, VGGNet), without accuracy loss. 2.02×–3.59× speedup and 1.89×–3.14× energy reduction with 3% accuracy loss |
| | [73] | [72] | Skip some neurons' computation based on Average Percentage of Zeros (APoZ) | 2×–3× fewer parameters with no accuracy loss on LeNet and VGG-16 |
| | [135] | [134] | Zero activation predictor (ZAP) to dynamically predict the zero-valued outputs in the activation maps | 38%–51% MAC reduction on AlexNet, 26.2%–44.5% on VGG-16, and 23.8%–34% on ResNet-18 on GPU (Nvidia Titan V) |
| | [22] | N/A | Layer-by-layer neuron pruning | 5.13× speedup on VGG-16 and 3× speedup on ResNet-50 on GPU (Nvidia Titan X (Pascal)) |
| | [143] | N/A | A spectral pruning method that compresses the CNNs by using their *degrees of freedom* | 32%–55% FLOPs reduction on ResNet-50 |
| | [54] | [53] | Use feature boosting and suppression (FBS) to emphasize salient convolutional channels and skip unimportant ones | 5× speedup on VGG-16 and 2× speedup on ResNet-18 |
| | [78] | N/A | Select neurons by the magnitude of their average activations | 2×–3× training speedup by employing 6×–10× fewer parameters on LeNet, AlexNet, and VGG-16 on GPU (Nvidia Titan X (Pascal)) |
| | [98] | [97] | Apply ranking on the neurons of DNNs based on their contribution to the inference accuracy | 1×–6.7× speedup on LeNet-5, AlexNet, and KWS [127] on GPU (Nividia RTX 2080Ti (Turing) and Jetson Nano)/CPU (x86 and ARM) |
| | [75] | [74] | Propose to add a sparsity regularization on neurons that guides the selection of more informative neurons | ResNet-38 achieves 14% less FLOPs and 0.2% lower top-5 error than DenseNet-121 on GPU |
| | [30] | N/A | Propose splitting and overlap-and-add operations for FFT-based CNNs to eliminate computation redundancy on activation maps | VGG-16 with single channel input achieve over 3× speedup on FPGA (Xilinx XC6VLX240T) |
| medical image segmentation (3D-UNet) | [20] | N/A | *Frequency Domain Compact 3D CNNs (FDC3D)* eliminating redundancy of 3D convolution filters and feature maps by converting them into the frequency domain | 2× speedup ratio on 3D-ResNet-18 |

(Continued)

Table 2. Continued

| Application | Paper | Code | Key Idea | Performance |
|---|---|---|---|---|
| object detection/recognition | [52] | N/A | Use two networks to focus on the region of interest on the input data | reducing the number of processed pixels by 70% and the detection time by over 50% on pedestrian detection on GPU |
| | [100] | [99] | The author implements a zoom network with *map attention decision (MAD) unit* that uses a selective vector to decide the attention on neurons | 78.1% mAP and 79.8% mAP compared to SSD baseline on PASCAL with 76.8% mAP |
| | [39] | N/A | A software-based memoization technique that groups CNN's layer output into range-based clusters and performs table lookup for similar computations | 3.5× speedup with YOLOv3-tiny and 22% energy consumption reduction on CPU (Intel Core i5-7500, 4 cores, 3.40GHz) |
| image classification & object detection/recognition | [27] | [26] | *ViP* leverages virtual pooling that applies larger strides in the convolution layer to reduce computation | VGG-16 with 2.1× (5.23× combined model compression); Faster-RCNN with 1.8× on GPU (Nvidia Titan X and Jetson TX1) |
| | [18] | N/A | *Multi-LayerFeature Federation Network (MuffNet)* splits the activation maps of a layer into several groups and shares the group information with other layers only once | WORK ON small models (under 45 MFLOPs); 0.3% better accuracy than MobileNet with half of its cost. |
| image classification & video action recognition | [24] | [23] | *OctaveConv* factorizes the activation maps into high and low-frequency groups and performs low-cost information exchange and update | 82.9% top-1 classification accuracy by ResNet-152 with 22.2 GFLOPs on GPU (Nvidia V100) |

tasks (e.g., image classification, object detection) that the DNN performs. For a relatively simple image classification task, if the number of buckets is enough to showcase the overall discrepancy between predicted classes without hurting the training or inference accuracy, the representative can be simply an average of data values in that bucket. To the extreme, when the dataset contains a significant proportion of identical data samples, a representative is sometimes set as an arbitrary data point taken from a bucket. The representative is used in place of the other issues in the bucket during the CNN inference or training, such that similar computations can be avoided.

Representative work in this category is the Deep Reuse work by Ning and others [115, 116]. As shown in Figure 2, the approach divides input images or activation maps into sub-vectors during runtime. Based on their similarities, it clusters them into several buckets through efficient online LSH-based clustering [56], with each cluster represented by its centroid. Besides handling repeated information in a 2D scope, Deep Reuse also addresses the redundancy in data batches. Based on the clustering results, Deep Reuse reduces a convolution to multiplications between a small matrix formed by the clusters centroids and the filter matrix. The results are used to reconstruct the full activation map through data duplications. Even with the online clustering overhead, the work shows that the reuse of computation results yields an overall $1.77 - 2\times$ speedup (up to $4.3\times$ layer-wise) with negligible accuracy loss and without model retraining. The method yields more speedups when applied to CNN training in an adaptive manner [115].

A fundamental tradeoff facing the methods in this category is the accuracy and the amount of computation reuse. It is determined by the aggressiveness in similarity-based clustering. For instance, in the Deep Reuse work, the aggressiveness is determined by two hyper-parameters, the length of a sub-vector in an activation map (granularity), and then the number of LSH hashing vectors. As the appropriate values of the hyperparameters are data and model dependent, the previous work [116] employs some offline tuning process to select the values such that the accuracy is not compromised. One of the directions worth some future explorations is to adapt the clustering method to data and models. For instance, in Deep Reuse, the hashing vectors are randomly

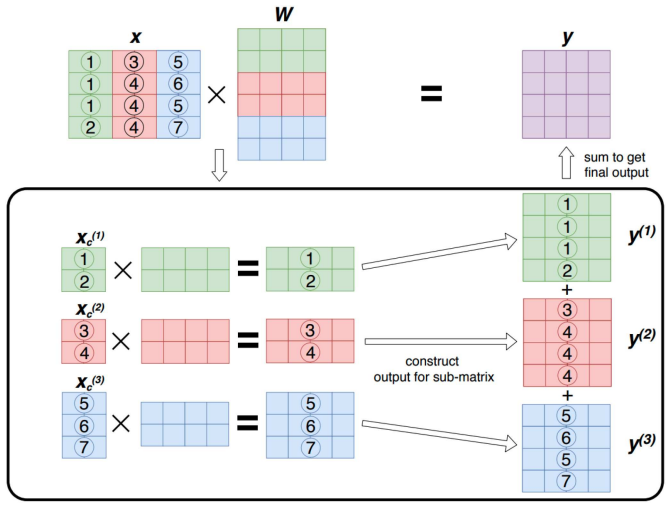


Fig. 2. Illustration of sub-vector clustering in *Deep Reuse*. In a convolution layer, the activation maps of 2D images are unfolded as $x$ and are multiplied by weights $W$ of corresponding filters. With sub-vector clustering, $x$ is first divided into sub-vectors and labeled with corresponding cluster ids. In the matrix multiplication, only the representative of each cluster is used. Afterward, the output activation maps are re-constructed with the representative results of each cluster. Figure adapted from Reference [116].

generated. If they can be learned from data for a given CNN, more aggressive clustering might be possible without causing accuracy losses.

## 4.2 Irrelevant Information

*4.2.1 Feature Pruning.* Image data or intermediate activation maps may have unnecessary information as well. For example, most well-trained CNNs can still maintain their prediction accuracy when some pixels in the activation maps are removed. Subsequently, the skipped portion of the data reduces the amount of computation and thus offers acceleration during inference or training. We introduce some details about how the optimizations in this line identify these irrelevant values at different scales and get rid of them while keeping the overall accuracy.

Hu et al. [73] identify many zeros in the activation of CNNs. They quantify the neurons to be pruned based on the ***Average Percentage of Zeros (APoZ)***. *APoZ* is calculated on a per-layer basis to measure the probability each fixed position appears to be zero-valued in validation data samples. With the same motivation, Akhlaghi et al. [2] and Piyasena et al. [123] propose detecting zero pixels that may occur after the ReLU activations. It is by identifying the negative activations earlier using a low-cost approximation scheme since the negative values turn into zeros after going through the ReLU layers. In particular, Akhlaghi et al. [2] construct *SnaPEA* with a runtime technique to speculate on the convolution outputs' sign before going through negative weights. The aggressiveness of the speculation is controlled by thresholding on parameters and the predefined associated number of MAC operations that can maintain accuracy. The parameters are tuned offline within the search space. Piyasena et al. [123] augment the CNN implementation with a lightweight approximation scheme that consists of *ApproxConv* and *ReLupred* stages. The two operations work sequentially to handle the sign prediction on an approximate convolution computation.

A more direct way to remove irrelevant information is to assume the existence of redundancy based on domain knowledge and prune the data or activation maps at different scales. Figurnov

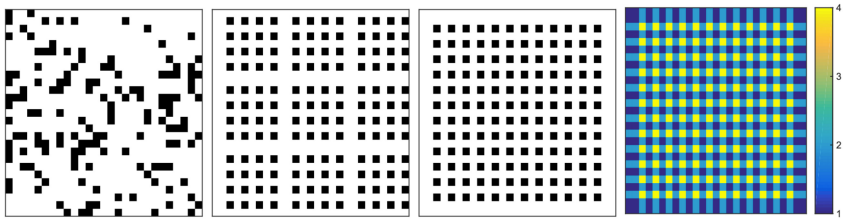


Fig. 3. Input-independent perforation masks in *PerforatedCNNs*. According to the paper, the mass layout is for the AlexNet Conv2 layer. (a) **Uniform perforation mask** is $N$ points chosen randomly without replacement from the set $\Omega$. (b) **Grid perforation mask** is a set of points $I = a(1), \ldots, a(Kx) \times b(1), \ldots, b(Ky)$ the values of a(i), b(i) using the pseudorandom integer sequence generation scheme [59]. (c) **Pooling structure mask** exploits the structure of the overlaps of pooling operators. Denote by $A(x, y)$ the number of times the convolutional layer output is used in the pooling operators. The pooling structure mask is obtained by picking top-N positions with the highest values of $A(x, y)$. (d) **Impact mask** estimates the impact of perforation of each position on the CNN loss function and then removes the least important positions. When activation maps are computed, the coordinates of black pixels are skipped. Figures and description adapted from Reference [49].

et al. [49] prune the data at pixel and patch level. The proposed mechanism inherits the idea of the loop perforation technique in code optimization to skip certain spatial positions in images for CNN classification inference. The selection of pixels to prune is random within a predefined limited number of points. They exploit four input-independent perforation masks: *Uniform mask*, *Grid mask*, *Pooling Structure mask*, and *Impact mask*, as shown in Figure 3. These masks are used to confine the random point selected into corresponding patterns. Afterward, perforated CNN uses interpolation to reconstruct the output feature maps. The CNN wights are retrained to adapt to this optimization as well.

Aside from the pruning technique within the scope of a single activation map, irrelevant information can be addressed across the channel dimensions by, for instance, removing a set of outside 2D feature maps. Specifically, Gao et al. [54] apply feature boosting and suppression by formulating a channel saliency predictor. The predictor serves as an indicator to skip execution at some feature maps at runtime. Chakraborty et al. [17] show that in a simple handwritten digit recognition task, directly discarding redundant feature maps randomly before the full connection does not affect the prediction accuracy much. On the same axis, Singh et al. [140] conduct a comprehensive analysis on pruning redundant activation maps in 1D CNN, SoundNet [5], at a per map level. The pruned feature map selection is based on the following measurements: ANOVA-based method [120], Entropy-based (DE) method [121], Cosine-similarity (CS) based method, and a greedy algorithm for selection of feature maps based on KL divergence. Gaikwad et al. [50] apply the L2 norm to decide feature map importance in pruning feature maps of SqueezeNet [77]. Although the technique is not for image data, the selection criteria are worthy of being noted under the context of irrelevancy elimination. Furthermore, multiple granularities of redundancy can be handled at the same time. For example, Yu et al. [161] combine spatial and channel-wise neuron pruning on activation maps. The criteria of what to be pruned are based on the attention mechanism. The attention mechanism is trained with targeted dropout, enhancing inference time robustness without accuracy loss after pruning.

Moreover, the optimizations can be classified by whether the irrelevancy is subject to the input datasets or not. Namely, they are input-dependent or input-independent. Input-dependent irrelevant information removal can be achieved by detecting data irrelevancy based on

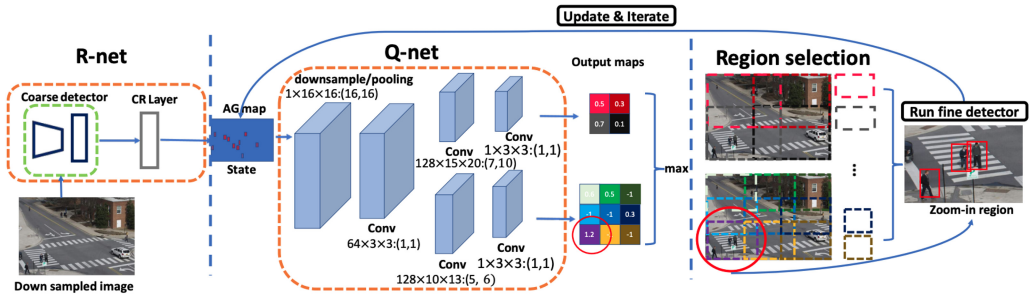


Fig. 4. The workflow of *R-net* and *Q-net*. Given a down-sampled image as input, the R-net generates an initial **accuracy gain (AG)** map indicating the potential zoom-in accuracy gain of different regions (initial state). The Q-net is applied iteratively on the AG map to select regions. Once a region is selected, the AG map will be updated to reflect the history of actions. Two parallel pipelines are used for the Q-net, each of which outputs an action-reward map that corresponds to selecting zoom-in regions with a specific size. The map's value indicates the likelihood that the action will increase accuracy at a low cost. Action rewards from all maps are considered to select the optimal zoom-in region at each iteration. The notation $128 \times 15 \times 20 : (7, 10)$ means 128 convolution kernels with size $15 \times 20$, and stride of 7/10 in height/width. Each grid cell in the output maps is given a unique color, and a bounding box of the same color is drawn on the image to denote the corresponding zoom region size and location. Figure and description adapted from Reference [52].

statistical [54, 73, 140] or learning-based heuristic [2, 161], or estimation of values [123]. On the contrary, input-independent approaches [17, 49] get rid of those irrelevant values by implicitly assuming them to appear randomly with a limit threshold in structural or non-structural manners.

*4.2.2 Neuron or Feature Subset Selection.* Another granularity used in feature map pruning is the selection of neurons (or activation). It could be regarded as a feature selection process at the value granularity. The neurons not chosen can be interpreted as ones being pruned. A subset of activations is determined based on specific criteria in neuron selection. Lee et al. [98] apply ranking on the neurons of DNNs based on their contribution to the inference accuracy. Huang et al. [75] add a sparsity regularization on neurons to force some of them to become zeros and guide the selection of more informative neurons without further tuning. Besides, Ibrokhimov et al. [78] propose the technique that selects neurons based on the magnitude of their average activations and keeps only the exact amount of the most minor activated neurons that work well. They report a reduction in the total number of operations and corresponding speedups.

*4.2.3 Dynamic Region of Interest.* In object detection, dynamic zooming in [52, 100] of a particular region of interest can help reduce computations on irrelevant background information. Gao et al. [52] present such a technique. They employ *zoom-in accuracy gain regression network (R-net)* and *zoom-in Q function network (Q-net)*. The *R-net* learns the correlation between coarse and fine detection and predicts the accuracy gain for zooming in to a specific region. The *Q-net* learns via reinforcement learning. It sequentially selects the optimal zoom locations and scales them correspondingly by analyzing the new output of *R-net* and its history. In the inference phase, a down-sampled image is input to the *R-net* and predicts the accuracy gain. The gain serves as an indicator for the *Q-net* to select the target on regions to concentrate sequentially. The workflow is illustrated in Figure 4. The mechanism reduces the total pixels in computation by over 50% and reduces the average detection time by 25% on the Caltech Pedestrian Detection dataset. It also reduces the pixels by 70% and obtains over 50% detection time reduction on the YFCC100M dataset.

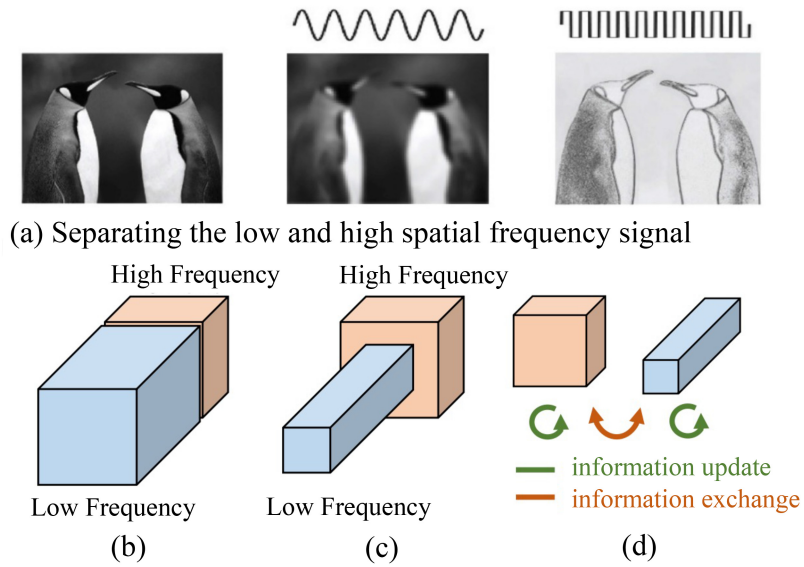


Fig. 5. Illustration of *Octave Conv* in the factorization of high and low spatial frequency. (a) Motivation. The spatial frequency model for vision [11, 40] shows that raw images can be decomposed into a low and a high spatial frequency part. (b) The output maps of a convolution layer can also be factorized and grouped by their spatial frequency. (c) The proposed multifrequency feature representation stores the smoothly changing, low-frequency maps in a low-resolution tensor to reduce spatial redundancy. (d) The proposed Octave Convolution operates directly on this representation. It updates the information for each group and further enables information exchange between groups. Figures and descriptions adapted from Reference [24].

### 4.3 Over-detailed Information

Yet another direction in image data redundancy exploitation is eliminating over-detailed information, represented by adaption to variant image resolutions, especially resolution lowering. Chen et al. [24] propose **Octave convolution (*OctConv*)** (Figure 5) to leverage the spatial redundancy and factorize the activation maps into high and low frequency (resolution) groups to save both memory and computation cost. The operations at each layer enable the information to be exchanged within both groups during computation. It is implemented as a portable plug-and-play convolution unit and can be applied together with a compressed model or optimization that reduces channel-wise redundancy. An OctConv-equipped ResNet-152 can achieve 82.9% top-1 classification accuracy on ImageNet with 22.2 GFLOPs.

***ViP*** proposed by Chen et al. [27] also targets the spatial redundancy in feature maps. It takes advantage of **virtual pooling**, as shown in Figure 6, such that the larger stride is used to obtain a smaller feature map and at the same time save convolution computation. To recover the output feature map dimension, linear interpolation is applied. ViP delivers 2.1× speedup with less than 1.5% accuracy degradation in ImageNet classification on VGG16, and 1.8× speedup with 0.025 mAP degradation in PASCAL VOC object detection with Faster-RCNN. ViP also reduces mobile GPU and CPU energy consumption by up to 55% and 70%, respectively.

## 5 LEVERAGE DATA REDUNDANCY IN DNN FOR VIDEOS

Existing DNN frameworks for video analysis are built on well-developed CNNs such as VGG [139] or ResNet [65] and image-based object detection algorithms such as Faster-RCNN [126]. They treat the video as a continuous input of images. Intuitively, the data redundancy in this line of

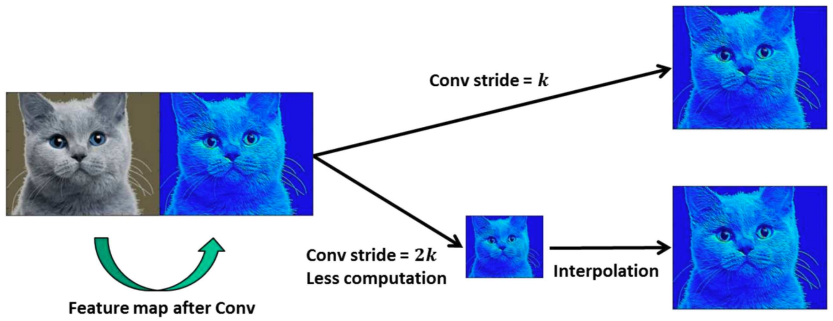


Fig. 6. Illustration of virtual pooling [104]. By using a larger stride, it saves computation in conv layers and, to recover the output size, it uses linear interpolation which is fast to compute. Figure and description adapted from Reference [27].

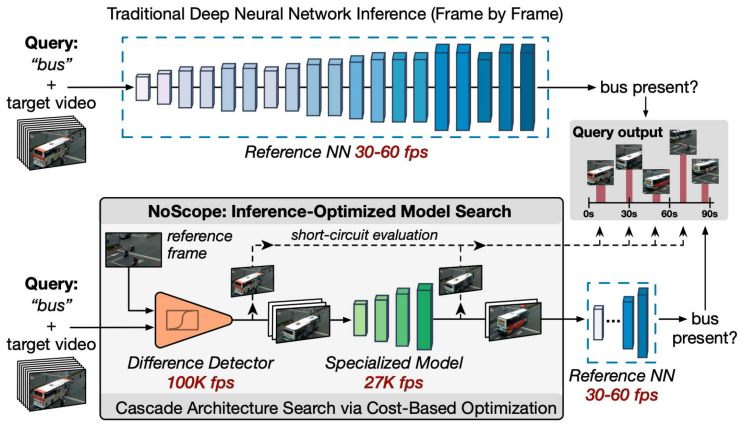


Fig. 7. Illustration of the *NoScope* framework. *NoScope* is a system for accelerating neural network analysis over videos via an inference-optimized model search. Given an input video, target object, and reference neural network, *NoScope* automatically searches for and trains a cascade of models–including difference detectors and specialized networks–that can reproduce the binarized outputs of the reference network with high accuracy–but up to three orders of magnitude faster. Figure and description adapted from Reference [84].

work, especially the duplication along the temporal axis, provides much room for performance enhancement. DNN frameworks that leverage data redundancy in video data can be categorized based on how they exploit redundancy. These techniques fall into the three categories listed in our taxonomy (skipping, reuse, and approximation); this section groups them more detailed to capture the more complicated differences among the techniques. Table 3 shows the works with video data redundancy exploitation by their applications.

### 5.1 Pruning

A way to avoid recomputing similar values across video frames is through skipping some of them along the temporal dimension (or the batch dimension in 2D CNN). Kang et al. propose *NoScope* [84] to optimize the processing of video querying. As shown in Figure 7, they insert a difference detector and a specialized model with short-circuit evaluation that allows computation reduction on almost identical neighboring frames in a video sequence.

Table 3. Summary of Works on Video Data Redundancy

| Application | Paper | Code | Key Idea | Performance |
|---|---|---|---|---|
| surveillance video processing | [16] | [13] | *CBInfer* utilizes change-based evaluation for CNNs to detect redundant pixels on video tasks | Evaluated on a self-constructed 5-layer CNN with input image size 871 × 541, provides speedup 8.6× over the CUDA baseline. Energy efficiency 10× higher (328 GOp/s/W on Nvidia Jetson TX1) |
| | [15] | [14] | | |
| video object detection | [29] | N/A | *AdaScale* trains a object detector and a regressor to dynamically adjust the optimal detector scale for video frames | Improve the speed by 25%; RFCN + AdaS 20 fps; SeqNMS [64] + AdaS 16 fps; DFF [168] + AdaS 60 fps on GPU (Nvidia GTX 1080Ti) |
| | [21] | N/A | *Scale-Time Lattice* performs expensive detection with fewer times and propagates the results across scale and time to balance the performance and accuracy tradeoff | On ImageNet VID, the approach achieves 79.6% at 20 fps or 79.0% at 62 fps on GPU |
| | [165] | [106] | A convolutional LSTM to assist Single Shot Detector (SSD) on refining feature maps and reuse history information | 15 fps with MobileNet-SSD on CPU |
| video object tracking | [112] | N/A | *CaTDeT* applies a tracker to assist refining intermediate feature maps for better reuse | Constructed Res10a and Res10b as proposal net, ResNet50 as refinement net, giving 81.4% and 81.5% mAP with MAC size 49.3 Gops and 29.3 Gops respectively on GPU (Nvidia Titan X (Maxwell)) |
| | [162] | N/A | *Kill Two Bird with One Stone* aggregates moving objects as patch-of-interest to construct a compact patch that skips duplicate image patchs' calculation | SSD with 300 × 300 subframe size at 13.7-24.2 fps and 500 × 500 subframe size at 23.8–25.7 fps on GPU (Nvidia Titan X) |
| temporal action localization | [160] | [159] | An end-to-end model for temporal action localization that generates compact video representations | Requiring 2% or less frames. |
| | [4] | [3] | *Action Search* uses LSTMs to predict next search location from current frame's location and the history | Observing an average 17.3% of the video 30.8% mAP on human activities recognition |
| frame selection | [156] | N/A | *AdaFrame* uses memory-augmented LSTM to adaptively select relevant frames for fast prediction | On FCVID [81] and ActivityNet dataset, achieving the same performance with only 8.21 and 8.65 frames, respectively. |
| | [94] | N/A | *SCSampler* is a lightweight model for clip sampling that invokes recognition on only the most salient clips | Improve the state-of-art 3D CNN video classification models (ResNet3D (R3D), Mixed Convolutional Network (MC3), and R(2+1)D [42]) on Sports1M by up to 7% and reduces the computation cost by more than 15× on GPU (32 Nvidia P100) |
| | [84] | [83] | *NoScope* adopts a difference detector and short-circuit evaluation on the model that allows skipping identical neighboring video frames | YOLOv2 with more than 30 fps and less than 5% accuracy loss on GPU (Nvidia P100) |
| action recognition | [154] | N/A | The authors use gated recurrent units (GRUs) and policy networks in reinforcement learning to adjust video sampling location per time step | 79.1% mAP for ActivityNet-v1.3 on YouTube Birds and 79.77% on YouTube Cars on GPU (Nvidia Titan X) |
| streaming activity recognition & untrimmed video activity detection | [142] | N/A | An active mechanism that prioritizes frames and makes timely action prediction | Achieving comparable accuracy on 32 fps and 64 fps CNN video classification on CNNs |

(Continued)

Table 3. Continued

| Application | Paper | Code | Key Idea | Performance |
|---|---|---|---|---|
| video object detection & video semantic segmentation | [169] | [166] | *deep feature flow network (DFF)* propagates neighboring frames information with a flow estimation algorithm | 20.25 fps and 73.1% mAP on ResNet-101-R-FCN on GPU (Nvidia K40) |
| | [168] | | Assemble the appearance information in consecutive frames | 74.0% mAP and 76.3% mAP on ResNet-50-R-FCN and ResNet-101-R-FCN respectively on GPU (Nvidia K40) |
| | [167] | | A unified framework that includes the above two techniques | 77.8% mAP on ResNet-101-DCN on GPU (Nvidia K40) |
| video object detection & video object localization | [86, 87] | [85] | Use LSTMs for temporal information propagation in order to improve video detection accuracy | Evaluated on DeepID-Net [118] with CRAFT [157] (pretrained Faster-RCNN), offering 67.82% mAP and 68.4% mAP on GPU (4 Nvidia Titan X) |

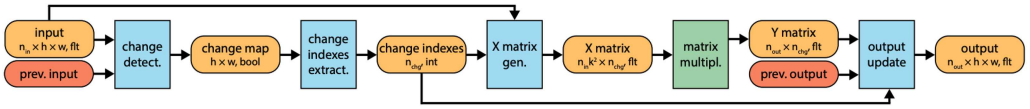


Fig. 8. Processing flow of the change-based convolution algorithm. Custom processing kernels are shown in blue, processing steps using available libraries are shown in green, variables sharable among layers are shown in yellow, and variables to be stored per layer are colored orange. The size and data type of the tensor storing the intermediate results is indicated below each variable name. Figure and description adapted from Reference [16].

The redundancy across continuous frames can also be addressed at the pixel level. Cavigelli et al. propose *CBinfer* [15, 16], a change-based evaluation of CNN for video data recorded with a static camera, to exploit the spatial-temporal sparsity of pixel changes. As shown in Figure 8, they modify each convolution layer to include an additional change detection mechanism and change indexes extraction before the matrix multiplication. The output is updated with both non-changed output pixels from the previous frame and changed output pixels calculated at this time. *CBinfer* achieves an average speed-up of 8.6× over a cuDNN baseline on a realistic benchmark with a negligible accuracy loss of less than 0.1% and no network retraining. The resulting energy efficiency is 10× higher than per-frame evaluation and reaches an equivalent of 328 $GOp/s/W$ on the Nvidia Tegra X1 platform.

### 5.2 Approximate Representation

Regional re-scaling is a type of approximate representation. Chin et al. propose *AdaScale* [29] that exploits the potential benefit bought by adaptive image scaling. They resort to image down-sampling to realize performance improvement on both accuracy and speedup for video object detection. To accomplish this, *AdaScale* adopts an object detector to generate the optimal scale labels for images carried out by a learning-based method. The generated optimal scale is later used to train the *scale regressor* that dynamically re-scales the image. The visualization of the effect is shown in Figure 9. When the images are down-sampled, they are supposed to reduce the number of false positives introduced by overly detail-oriented messages, which in turn dismiss the redundancy. The image scaling increases the number of true positives by turning the objects too large to smaller ones for the detector to be more confident. For ImageNet VID and mini YouTube-BB datasets, AdaScale demonstrates 1.3% and 2.7% mAP improvement with 1.6× and 1.8× speedup respectively. The framework can also work complementary with video

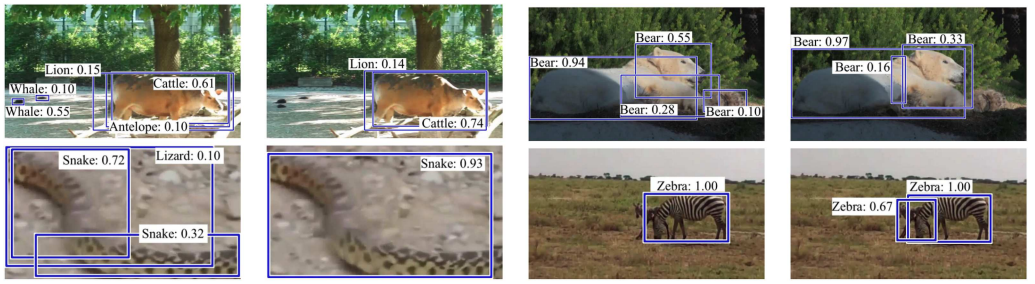


Fig. 9. *AdaScale* detection result visualization. Examples where down-sampled images have better detection results. Blue boxes are the detection results, and the numbers are the confidence. The detector is trained on a single scale (pixels of the shortest side) of 600. Column (a) and (c) are tested at scale of 600. Column (b) is tested at scale 240, and column (d) is tested at scale 480. Figures and descriptions adapted from Reference [29].

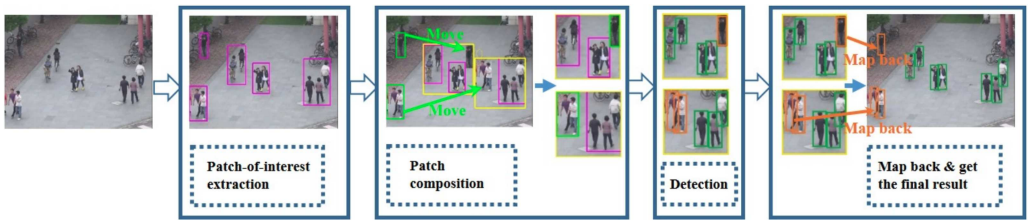


Fig. 10. Patch-of-Interest composition. (a) The input image. (b) Detected patches. (c) The patch composition (left) and sub-frames(right). (d) Detection results on sub-frames. (e) Map back and get the final result on the original image. Figures and descriptions adapted from Reference [162].

acceleration work [169], of which they experience a speedup gain by 25% and a slight mAP boost on the ImageNet VID dataset.

### 5.3 Subset Selection

Subset selection is for obtaining a subgroup of potentially representative video frames or corresponding feature maps. These techniques adopt a variety of dimensions in preference, from the spatial to the temporal, batch, and streaming.

#### *5.3.1 Selection in the Spatial Dimension.*

*Patch-of-Interest.* Zhang et al. [162] propose *Kill Two Birds with One Stone* to aggregate patch-of-interest (usually the moving object within images) for construction of a compact patch composition for video object detection as shown in Figure 10. In the detection steps, the parts of the spatial dimension data patches are eliminated to save unnecessary computation. The set of patch-of-interest is mapped back by their numbers and location offsets to reconstruct the final result.

*Patch Refinement.* Mao et al. [112] present ***CaTDeT (Cascaded Tracking Detector)***, which applies a tracker as assistance for refining intermediate feature maps in video object detection. The illustration of the algorithm is shown in Figure 11. The tracker provides information on the region of interest based on historical detection. It reduces the operation count by 5.1× to 8.7× with a slight delay.

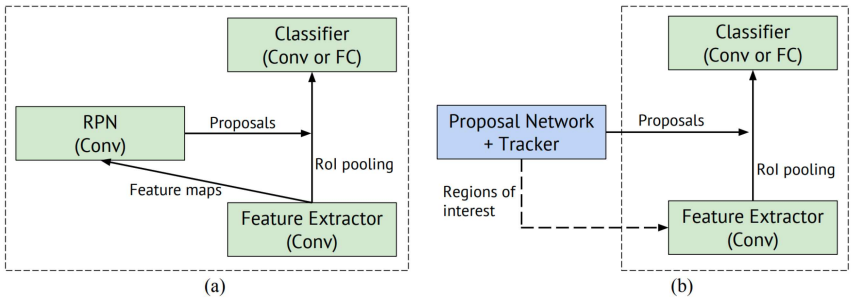


Fig. 11. Illustration of algorithm flow in CaTDeT. Compare the inference-time workflows of the standard Faster R-CNN model and the refinement network. (a) Standard Faster R-CNN detector: the RPN takes the feature maps from the feature extractor. (b) Faster R-CNN detector for the refinement network: the proposals from the proposal network and the tracker instruct the feature extractor only to compute features on regions of interest. Regions of interest are a mask of all proposals over the frame. Figures and descriptions adapted from Reference [112].

*5.3.2 Selection in the Temporal Dimension.* Korbar et al. propose *SCSampler* [94], a lightweight model for clip sampling that can invoke recognition on only the most salient clips. The paper targets scenarios where the videos are untrimmed and long. They aim to reduce the high inference cost when every clip is executed on the clip classifier. They apply both visual and audio samplers and combine their saliency for prediction. The visual sampler proposes a learning-based clip-level saliency model that provides each clip a *saliency score* between [0,1]. In particular, the model takes the input clip features that are fast to compute from the raw clip and have low dimensionality (lower than that of the classifier) to analyze each clip very efficiently. The clips with top-K *saliency scores* are extracted with features by the classifier and aggregated together to form the representation for prediction. The sampler learning phase labels a *saliency score* of video with 1 if the action is contained in that clip; otherwise, it is labeled as 0. The approach elevates the accuracy of an already state-of-the-art action classifier by 7% and reduces its computational cost by more than 15 times.

On the other hand, subset selection in the temporal dimension of the video can sometimes help improve task accuracy. This work typically involves the addition of some sophisticated mechanisms into the DNN. As a result, they often slow down the DNN rather than speed it up. But their goal is on the accuracy of the outputs rather than speed. A class of the work uses recurrent Neural Networks, such as **Long Short Term Memory networks (LSTM)**, for capturing long-term dependencies in sequence data. They use them to generate compact video representations along with CNN activation maps [4, 156, 160]. They mimic the behavior of humans viewing videos that often consist of a glimpse followed by several refinements. In addition, some work [154] combines **gated recurrent units (GRUs)** and policy networks in reinforcement learning to adjust video sampling location accordingly at each time step.

*5.3.3 Selection in the Streaming Scenario.* The selection in a streaming scenario is more challenging. For each selection, the mechanisms need to make decisions on future frames based on historical records. Su et al. [142] introduce an active mechanism that prioritizes *"which feature to compute when"* to make timely action predictions. The core idea is to learn a policy that dynamically schedules the sequence of features to compute on selected frames of a given test video. The selection procedure can be invoked for either batch or streaming dimension. They formulate the problem with a **Markov decision process (MDP)** to learn the feature prioritization policy. The MDP sequentially selects frames with the most promising bag-of-objects or CNN features

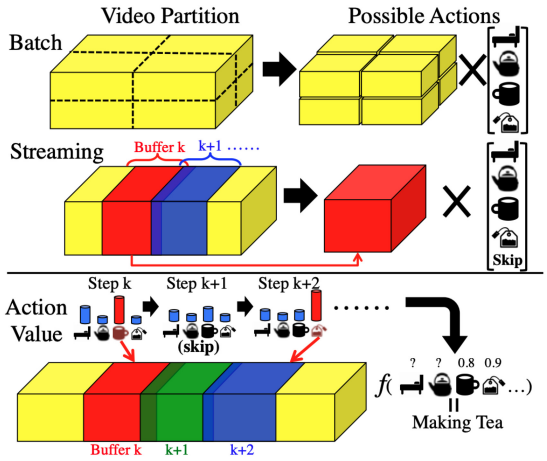


Fig. 12. Action spaces. Top: In batch, the whole video is divided into sub-volumes, and actions are defined by the volume and object category to detect. Middle: In streaming, the video is divided into segments by the buffer at each step, and actions are the object category to detect in the buffer plus a "skip" action. Bottom: Our method learns a video-specific policy to select a sequence of valuable features to extract dynamically. Figure and description adapted from Reference [142].

that can improve accuracy when combined with accumulated history results. Figure 12 illustrates the action spaces explored in the mechanism. On two challenging datasets (Activities of Daily Living [122] and UCF-101 [141]), their method provides significantly better accuracy than previous techniques under given computational budgets on two challenging datasets (Activities of Daily Living and UCF-101).

### 5.4 Temporal Propagation

The neighboring frames in a video usually show some inherent continuity. In this type of optimization, the intermediate computation results of adjacent video frames such as feature maps, bounding boxes, or classification probability and confidence are propagated along the temporal dimension to reduce the need for re-computation on similar contents. Usually, some key frames are chosen, and the results are propagated from them to other frames. The optimizations are different in the representations used in propagation and update. The representations of frame features include *Optical Flow*, ***Motion History Image (MHI)***, and *Convolutional LSTMs*. To clarify, in *Convolutional LSTMs*, frames are represented in normal CNN feature maps, but the corresponding information is updated via LSTM networks.

*5.4.1 Using Optical flow.* Zhu et al. [169] propose a ***deep feature flow network (DFF)***. In *DFF*, neighboring frames are propagated with feature maps via a flow field for partial updates. The updates are obtained by a flow estimation algorithm like *SIFT Flow* [105] or *FlowNet* [41]. Figure 13 shows how information is propagated within neighboring frames. Compared to re-computing each frame through convolution layers, *DFF* is 10× faster with only a few percent accuracy loss. Subsequently, the team comes up with the ***flow-guided feature aggregation (FGFA)*** [168] that associates and assembles the rich appearance information in consecutive frames to improve feature representation and accuracy. Afterward, the authors unify the solutions into a common framework [167], and achieve 77.8% mAP score at a speed of 15.22 frame per second for video object detection.

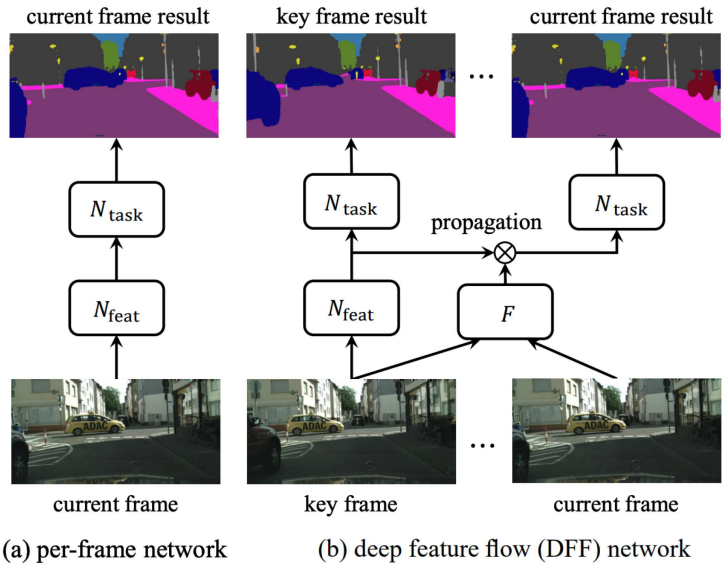


Fig. 13. Illustration of temporal propagation in *deep feature flow*. (a) Video recognition using per-frame network evaluation. (b) The proposed *deep feature flow*. Figures and descriptions adapted from Reference [169].

Kang et al. [87] leverage motion-guided propagation for minor temporal information refinement across adjacent frames. The assistance of temporal information reduces the potential false negatives prediction when frames are processed per frame.

*5.4.2 Using Motion History Image.* Unlike previous approaches that propagate key frames information at fixed intervals, Chen et al. [21] propose *Scale-Time Lattice* that selects the key frames adaptively. The *Scale-Time Lattice* is a directed acyclic graph as shown in Figure 14. Inside the graph, the node stands for the bounding box detection result at a particular spatial resolution and time point; the edge represents an operation that performs temporal propagation or spatial refinement. During the execution, given a video input, the framework first applies expensive object detectors to the key frames selected sparsely and adaptively based on the object motion and scale. Next, within the lattice, those bounding boxes are propagated to intermediate frames and refined across scales (from coarse to fine) via cheaper networks. A ***Propagation Refinement Unit (PRU)*** takes the detection results of two consecutive frames as input, propagates them to other frames, and re-scales them. The authors adopt ***Motion History Image (MHI)*** [10] since the motion representation computation is relatively cheaper than optical flows. Overall, the proposed method yields 79.6 mAP at 20 FPS and 79.0 mAP at 62 FPS on ImageNet VID dataset.

*5.4.3 Using Convolutional LSTMs.* Liu et al. [165] argue that temporal cues can be efficiently propagated through an LSTM network. The LSTM serves as an augmented structure to assist in refining temporal context for the generated features of each frame in the video. In the presented architecture in Figure 15, a hypothesis feature map for each frame is generated by the CNN feature extractor and then fed into the LSTM to be fused with temporal context from previous frames. The output for that frame is a temporally-aware refined feature map. The experiment integrates the convolutional LSTMs with ***Single Shot Detector (SSD)*** [109] and provides a quick inference speed with only 15 FPS on a mobile CPU.

Kang et al. [86] leverage LSTMs instead of bounding box proposal refinement in video object recognition. The temporal information is propagated across each proposal tublet to improve

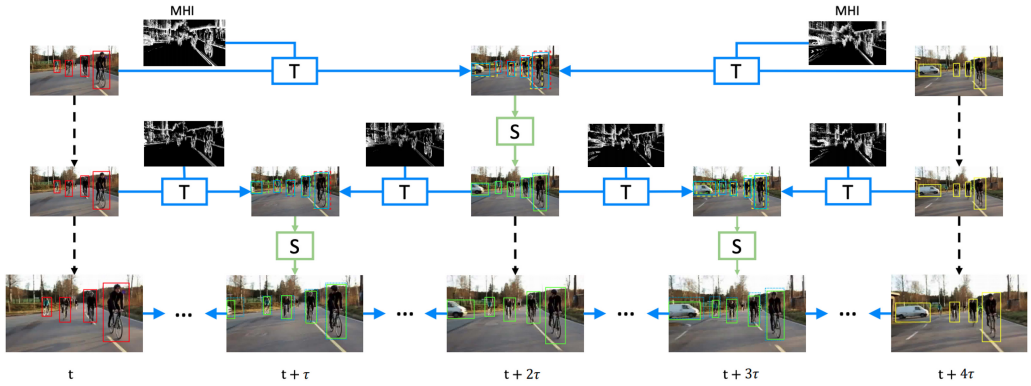


Fig. 14. The *Scale-Time Lattice*, where each node represents the detection results at a specific scale and time point, and each edge represents an operation from one node to another. In particular, the horizontal edges (in blue color) represent the temporal propagation from one time step to the next, while the vertical edges (in green color) represent the spatial refinement from low to high resolutions. Given a video, the image-based detection is only done at sparsely chosen key frames, and the results are propagated along a pre-defined path to the bottom row. The final results at the bottom cover all the time points. Figure and description adapted from Reference [21].

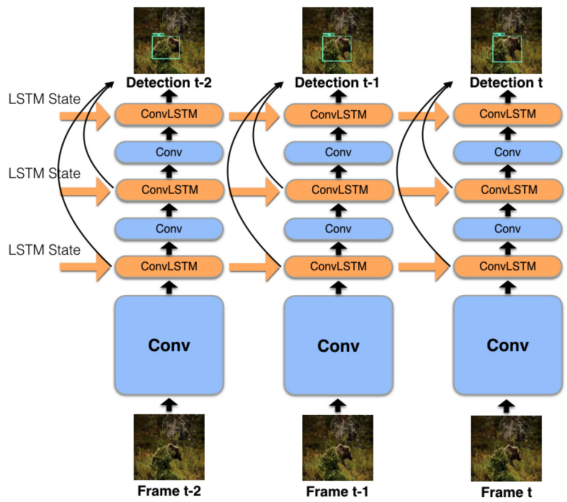


Fig. 15. An example illustration of our joint LSTM-SSD model. Multiple Convolutional LSTM layers are inserted into the network. Each propagates and refines feature maps at a particular scale. Figure and description adapted from Reference [165].

accuracy. As shown in Figure 16, an encoder-decoder LSTM structure is placed after the *Tublet Proposal Network*. It generates the output probability of each class label as a prediction result for each proposal.

### 5.5 Data Redundancy based Optimizations on 3D CNNs

So far, we have discussed how the performance of 2D CNN is advanced with works that exploit redundancy, especially at the temporal dimension where contiguous frames involve much identical information. Most of the optimizations are for the performance of video object detection, which

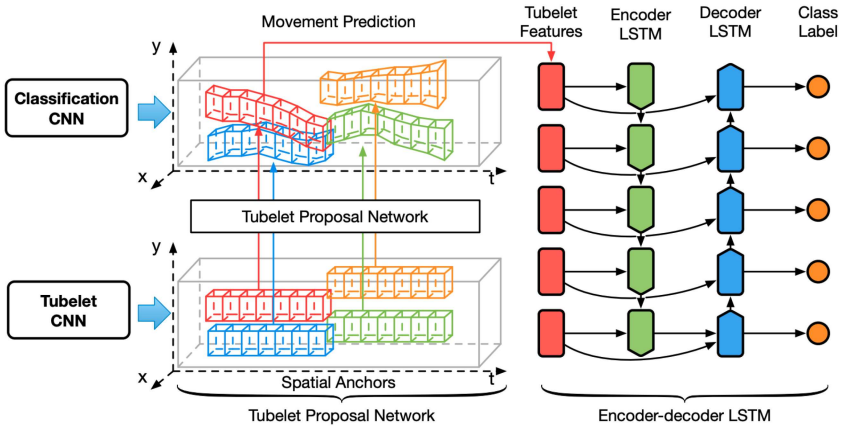


Fig. 16. The proposed object detection system consists of two main parts. The first is a tubelet proposal network to generate tubelet proposals efficiently. The tubelet proposal network extracts multi-frame features within the spatial anchors, predicts the object motion patterns relative to the spatial anchors, and generates tubelet proposals. The gray box indicates the video clip, and different colors indicate the proposal process of other spatial anchors. The second part is an encoder-decoder CNN-LSTM network to extract tubelet features and classify each proposal box into different classes. The tubelet features are first fed into the encoder LSTM by a forward pass to capture the appearance features of the entire sequence. Then the states are copied to the decoder LSTM for a backward pass with the tubelet features. The encoder-decoder LSTM processes the whole clip before outputting class probabilities for each frame. Figure and description reference adapted from Reference [86].

is rather heavy on computation as the DNN has an additional region proposal phase. Still, some DNNs enable the analytical capability for videos with either 3D CNN [80, 145] or two-stream models [88, 138]. Unlike how 2D CNN is operated on video data, 3D CNN can end-to-end training and inference on videos. It is augmented with capturing semantics along the temporal dimension using stridden convolution filters.

Redundancy removal on video data for 3D CNN has been less explored at software-based optimizations. Some works [12, 93] propose to analyze the potential redundancy of the underlying network structures by progressively expanding the operator dimension from a 2D CNN basis. For example, Feichtenhofer et al. [46] explore the 2D to 3D extension by considering the necessity of parameters or data in multiple dimensions. These include frame rate under fixed duration, temporal duration, spatial resolution, number of layers per residual stage, number of channels, and inner channel width in a residual block. These initiatives introduce another angle of viewing data redundancy on video DNNs. Future works offer prospective clues for advancing redundancy removal-based algorithms on deep models.

Still, existing works [19, 44, 67, 132, 149, 151, 152] have developed performance optimizations for 3D CNN that extend the fruitful efforts on the 2D CNN counterparts in hardware-based optimizations. The promising outcomes lie in that computation along the temporal dimensions is only required on the deltas (value differences) between adjacent video frames. They effectively alleviate the memory consumption and inference time bottleneck on resource-constraint devices.

## 6 LEVERAGE DATA REDUNDANCY IN TRANSFORMER OPTIMIZATION FOR TEXT

Text-based redundancy optimization can be applied to **Recurrent Neural Networks (RNNs)**, the widely used deep learning model for texts before the invention of the Transformer-based model. Guan et al. [60] propose to leverage **context-free grammar (CFG)** and a hierarchical

Table 4. Summary of Works on Text Data Redundancy

| Application | Paper | Code | Key Idea | Performance |
|---|---|---|---|---|
| text summarization | [108] | [107] | *T-DMCA* is a modified multi-head self-attention structure to reduce the memory footprint in self-attention layers | possible to learn on sequence length of 12,000 on GPU (Nvidia P100) |
| text classification | [35] | [34] | *Funnel-Transformer* augments the transformer model with an encoder to gradually pooled the representation to reduce the sequence-length of the hidden states as the layer goes deeper | 1.3×–1.5× speedup on GPU/TPU |
| | [58] | [57] | *PoWER-BERT* applies extract layers and learning-based retention configuration to retain only the key embedding vectors in the transformer model | 4.5× speedup on BERT-base models (6.8× with ALBERT) on GPU (Nvidia K80) |
| | [38] | [36] | Exploit *LayerSelector (LS)* and *Correlation Clustering Feature Selection (CCFS)* to select the essence layers and features for transformer models | Evaluated on BERT and XLNET. Reduce forward pass parameters up to 47%. Reduce feature set to 4% (sequence labeling), <1% (sequence classification) on GPU (Nvidia Titan X) |
| question answering | [92] | [91] | Propose *Locality-sensitive Hashing Attention* and *Reversible Transformer* to address data redundancy | Attention speed can maintain as low as 0.1–0.5 seconds per step when sequence length per batch scale from 32 to 32768s (base-line is 0.2–5 seconds per step) on GPU/TPU |

compression algorithm to squeeze the repeated contexts in sequence prediction tasks. The optimization accelerates RNN inference up to a thousand times and expends the prediction scope without losing task accuracy. Similar optimization insights can be found in many recent Transformer models as well. Aside from reducing model execution time or increasing task-specific accuracy, many data redundancy-based optimizations in Transformer models are initially motivated by the need to scale up the capacity of the models. Each time, the model can only process a sequence limited in length (the number of token representations) for token-level text inputs, subject to the hardware capability. The problem is especially crucial when it is necessary to process the representations of a paragraph or an extended sequence at once. As a result, researchers have systematically identified redundancy within the raw data or intermediate data representations to scale the model correctly.

Unlike CNN optimizations which leverage data redundancy at input data or intermediate feature maps between convolution layers, optimizations on Transformers can occur at different stages of interim data representations. A sequence of text input to the transformer model goes through several encoding phases, each capturing an extra level of meaning. The stages focus on the initial sentences or sub-words input, the sub-words embeddings, the hidden states produced by positional encoding, multi-head attention, or the encoder layer as a whole (multi-head attention and feed-forward layer together). Data redundancy can exist in each of the phases. Existing studies leverage the redundancy of various insights.

For the first stage, sub-word segmentation algorithms have been carried out with the motivation to remove redundancy in text data. ***Byte-Pair Encoding (BPE)*** [131] is one such case. For the second stage, redundancy may exist in input embeddings. Compression at the vector representation of word embeddings was introduced long before the wide use of Transformer models [1, 25, 90]. For the remaining stages, there are optimizations that leverage redundancy at intermediate representation, *"hidden state"* (a single sample output of a transformer layer; the unit is the same as activation map in CNN), of the model in general. Optimizations that leverage data redundancy in Transformer are classified based on how they exploit redundancy: clustering for reuse, skipping, and other ad-hoc sampling techniques. Table 4 lists the surveyed work by their applications for better reference.

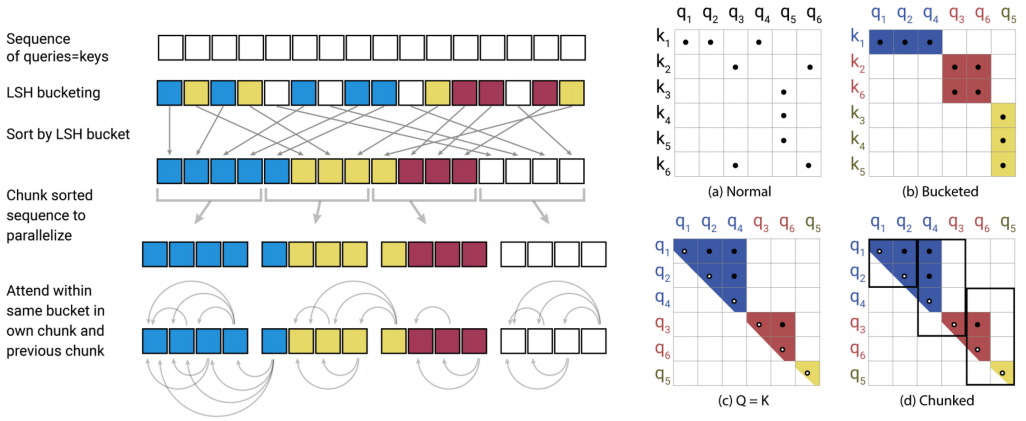


Fig. 17. Locality-sensitive hashing attention in *Reformer*. LSH Attention shows the hash-bucketing, sorting, and chunking steps and the resulting causal attentions. (a–d) Attention matrices for these varieties of attention. Figures adapted from Reference [92].

### 6.1 Reuse via Clustering

Kitaev and Kaiser et al. propose *Reformer* [92] with two main contributions in improving the transformer model: *Locality-sensitive Hashing Attention* and *Reversible Transformer*. In particular, *Locality-sensitive Hashing Attention* tackles data redundancy with a specific function design. The function uses an approximation in the *scaled dot-product attention* in a transformer model. The *scaled dot-product attention* is defined as $\frac{QK^T}{\sqrt{d_k}}V$, which $Q$, $K$, and $V$ are the query, key, and value matrices, respectively. The authors identify that the computation in the softmax function is dominated by key-query pairs that are in proximity. It indicates the calculation can pay attention to only a small subset of the closet key-query pairs. As exemplified in Figure 17, ***locality-sensitive hashing (LSH)*** [56] is employed to first cluster the keys and queries into hash buckets. Queries and keys in each bucket are sorted to form chunks to avoid sparsity in the attention matrix. Afterward, the *scaled dot-product attention* is substituted by allowing multiplication only between pairs in the same hash buckets (or in chunks in the attention matrix). Furthermore, the *Locality-sensitive Hashing Attention* introduces a *multi-round LSH attention* that enables multiple rounds of hashing to alleviate the problems that similar items may fall into different buckets after hashing. In the decoder of the transformer model, a masking mechanism is implemented to re-order the position indices by the same permutations that were previously applied to sort the key and query vectors.

### 6.2 Skipping

Output representations from attention layers in Transformer models have been recognized as containing much redundant information [113, 147]. They have been addressed through ways of attention head pruning [113, 147]. Recent studies have investigated the criteria for pruning neurons (activations) in Transformer models. Specifically, Gupta et al. [63] suggest neurons are to be pruned when either the importance of that neuron is low or when the same activation multiplies with the same weights. The neuron importance function [130] is defined on entropy, output-weights norm, or the input-weights norm.

The hidden state of a sequence of word vectors may contain repetitive information themselves. Goyal et al. [58] identify that the pair-wise cosine similarity between word vectors in hidden states increases as layers go progressively deeper (encode by more phases). They propose *PoWER-BERT* to exploit the redundancy to reduce the inference time of the BERT model in text classification

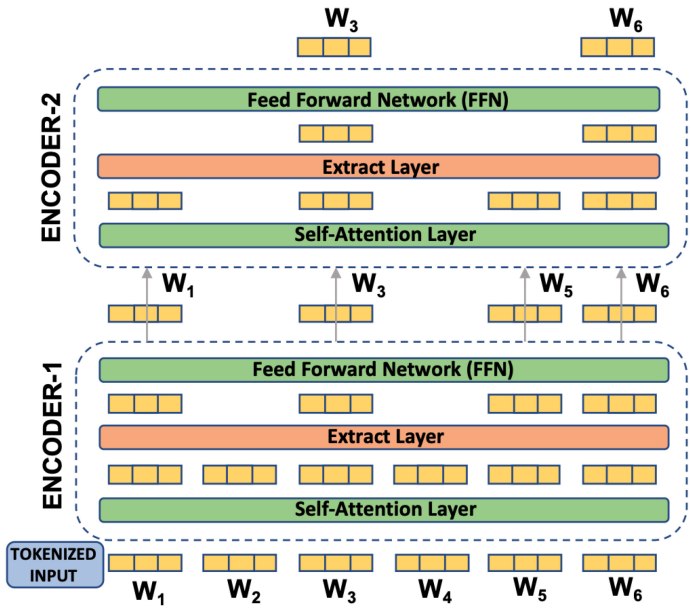


Fig. 18. Illustration of *PoWER-BERT*. Word-vector selection over the first two encoders. Here, $N = 6$, $l_1 = 4$ and $l_2 = 2$. The first encoder eliminates two word-vectors $w2$ and $w4$ with least significance scores; the second encoder further eliminates word-vectors $w1$ and $w5$. Figure and description adapted from Reference [58].

and regression tasks. As illustrated in Figure 18, *PoWER-BERT* uses an *Extract Layer* to manage the retention configuration after the *Self-Attention Layer* and before the ***Feed Forward Network (FFN)*** in each encoder layer of the Transformer model. The retention configuration can be set manually or set with parameters from training. The model is trained with a loss function proposed to obtain the learning-based design. This method avoids the exponential search space in retention configuration. Furthermore, they employ two kinds of strategies to retain the word representations: the *static* and the *dynamic strategy*. The *static strategy* keeps the word vectors at the same positions across all input sequences in the dataset. The *dynamic strategy* retains the word vectors based on the attention scoring and a soft-extract layer. In the experiment, they obtain a 4.5× inference time acceleration over the baseline BERT model with less than 1% accuracy loss. The proposed technique is also compatible with the compression scheme such as *ALBERT* [96]. The combination of both compression methods achieves a 6.8× inference time reduction with less than 1% accuracy loss.

Dalvi et al. [38] propose an efficient transfer learning method that exploits ***LayerSelector (LS)*** and ***Correlation Clustering Feature Selection (CCFS)*** to select the essence layers and features required during transformer model execution. There are three main steps in the selection: (1) *LayerSelector (LS)* uses the layer-classifier to select the lowest layer that maintains oracle performance. (2) *Correlation Clustering* filters out redundant neurons given the hidden state's output of the previous layer. (3) ***Feature Selection (FS)*** selects a minimal set of neurons required to reach optimum performance on the given task.

For the first step, the authors analyze task-specific layer-level redundancy by training linear probing classifiers [7, 133] on each layer $l_i$ as *layer-classifier*. They use the *layer-classifier* to select the lowest layer that maintains oracle performance (maintaining 99% performance). In *LS*, a concentration of all layers until the chosen layer is used.

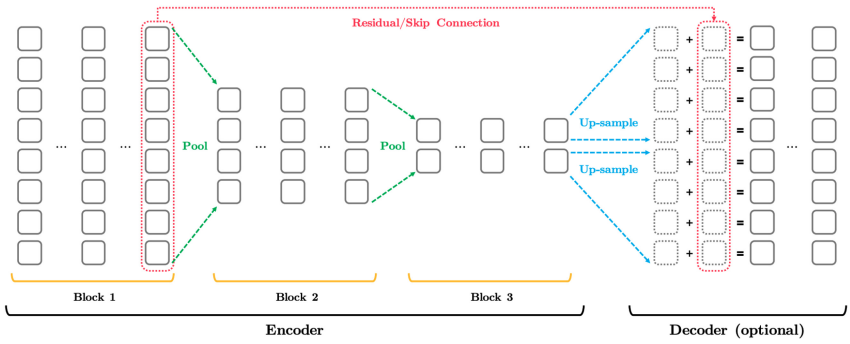


Fig. 19. Illustration of Pooling in Funnel-Transformer. The encoder on the left consists of several blocks of consecutive Transformer layers. Within each block, the sequence length of the hidden states always remains the same. But when going from a lower-level block to a higher-level block, the size of the hidden sequence is reduced by performing a particular type of pooling along the sequence dimension. The right part showcases the up-sampling to re-construct the arrangement of the original length. Figure and description adapted from Reference [35].

For the following two steps, the *CCFS* exploits data redundancy by a combination of clustering and subset selection:

- ***Correlation Clustering (CC)* [6]:** For every length-N sequence vector representation, the product-moment correlation between each of the two features is calculated. This gives a $N$-by-$N$ matrix $corr(x, y)$ representing the correlation between $fx$ and $fy$. The correlation value ranges from $-1$ to 1, which provides a relative scale to compare any two neurons. The distance metric $cdist(x, y)$ between these two features is defined by $1 - |corr(x, y)|$. A hyper-parameter $ct$ defines the maximum distance between any two features to be considered as a cluster.
- ***Feature Selection (FS)***: It identifies a minimal set of neurons that match the oracle performance. The *Linguistic Correlation Analysis* [37] ranks the neurons concerning a downstream task.

Overall, the work finds out that up to 85% and 95% neurons are redundant in BERT and XL-Net [158], respectively. In the experiment, the proposed method accelerates sequence labeling tasks by 2.8× and 6.2× on BERT and XLNet, respectively. While for sequence classification tasks, the average speedups are 1.1× and 2.8× correspondingly.

### 6.3 Other Sampling

Humans implicitly use prior linguistic knowledge to merge nearby tokens or words in paragraphs. This, in turn, forms more extensive semantic phrases or units to understand sentences or paragraphs in a general form. Inspired by that, Dai et al. [35] have proposed *Funnel-Transformer*. As shown in Figure 19, the *Funnel-Transformer* is equipped with an additional encoder to gradually pool the representation to reduce the sequence length of the hidden states as the layer goes deeper. The sub-modules have an extra residual connection and layer normalization operation in the *Self-Attention Layer* and the ***Feed Forward Network (FFN)*** of the Transformer model. The decoder is applied to reconstruct the full-sequence of representation when the token-level outputs are required, such as pretraining.

To achieve the generation of Wikipedia-styled summarization over multi-documents, Liu et al. [108] introduce a compressive scheme for long text sequences. The ***Transformer Decoder***

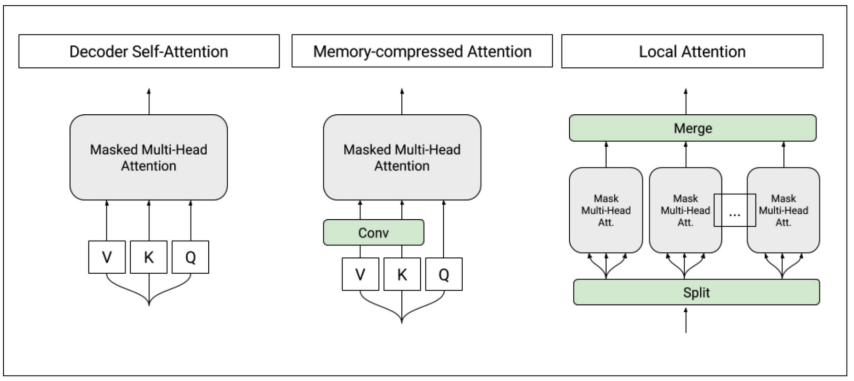


Fig. 20. Transformer Decoder with Memory-Compressed Attention(T-DMCA). Every attention layer takes a sequence of tokens as input and produces a sequence of similar length as the output. Left: Original self-attention as used in the transformer decoder. Middle: Memory-compressed attention, which reduces the number of keys/values. Right: Local attention, which splits the sequence into individual smaller sub-sequences. The sub-sequences are then merged to get the final output sequence. Figure and reference adapted from Reference [108].

***with Memory-Compressed Attention(T-DMCA)*** is a modified multi-head self-attention structure to reduce the memory footprint via limiting the dot product between the query and key of a *Self-Attention Layer*. The mechanism consists of the *Memory-Compressed Attention* and the *Local Attention* as shown in Figure 20. The *Memory-Compressed Attention* reduces the number of keys and values by using a stridden convolution (for sampling). The *Local Attention* divides a sequence of tokens into blocks of similar length of tokens (they use 256 tokens) to allow constant attention memory cost per block regardless of sequence length. After feeding the sub-sequence to their corresponding *Multi-head Attention* to capture local information, the method merges the results to get the final output sequence. For both *Local* and *Memory-Compressed Attention*, masking is cast to prevent the queries from attending to future keys and values. Finally, the design enables the model to process 3×-longer sequences than the default model.

## 7 FUTURE RESEARCH DIRECTIONS

The many previous studies have clearly shown the significant benefits of exploitation of data redundancy in Deep Learning. In creating the summary of the many explorations from various aspects, we have recognized the broad coverage of the existing innovations, but at the same, observed some open problems and several research directions worth pursuing in the future.

**(i) Principled understanding of the relations between data redundancy exploitation and accuracy.** Data redundancy exploitation could cause changes in the output of a DNN because the removed data are often not guaranteed to be redundant. In most cases, it faces the risks of degrading the accuracy, but in some cases, as mentioned earlier [29, 94, 142], it may also improve the accuracy for its noise removal effects. There is a lack of principled understanding of how data redundancy elimination affects model accuracy. The accuracy of a DNN model can be affected by existing variance related to data. Dataset itself may already contain bias over DNN models [144]. Data preprocessing, such as data augmentation [33, 136], makes the discussion even more complicated. Previous work has either resorted to empirical tuning to reach a satisfying point or used some heuristic methods to rate the salience of some data. An open question is how to achieve a principled understanding of the impact and transform the knowledge into principled data redundancy exploitation techniques. Some rigorous studies may help.

**(ii) Enhancing interpretability.** Related to the first direction, the second direction that may be worth exploring is interpretability, which refers to both the interpretability of neuron representation and the interpretability of DNN in general. Utilizing evidence in data redundancy has led to some initial success in improving the interpretability of CNN's feature representation, contributing to the CNN model pruning processes. For example, Li et al. [102] propose a ***kernel sparsity and entropy indicator (KSE)*** to quantify the importance of each pixel in the activation map to provide feature-agonistic guidance in weight compression. Identifications of data redundancy also benefit the interpretation of the importance of each word in text processing. For example, Dalvi et al. [37] present *Linguistic Correlation Analysis* and *Cross-model Correlation Analysis* to recognize the relative importance of a neuron in a Transformer model. Their method offers an ablation view on the saliency of words in the paragraphs to guide future representation development. These several studies show some promise in exploring data redundancy for the interpretability of deep learning. As interpretability is important for connecting Deep Learning with domain understanding, more efforts are worthwhile to develop further.

**(iii) Other data types and models.** A large portion of the prior work on data redundancy is about image and video data. There is still some room left for more explorations on these data types. An example is learning from point cloud [62] where data redundancy has not yet been explored. But in comparison to images and videos, redundancy in text data is much less explored, as Transformer is more recently developed than the more mature models used in images and videos. More explorations are especially in demand to better understand data redundancy in texts. Moreover, besides the three types of data, there are other types of data on which DNN is also playing an important role. These data include graphs, scientific simulation results, genes, computer programs, and so on. Correspondingly, the special properties of these data have prompted the development of some new DNN models, such as **Graph Convolutional Networks (GCN)** [155]. The differences in data properties, DNN models, and learning tasks suggest that some research could be fruitful in understanding and exploiting data redundancy in these areas.

**(iv) Relations among optimizations and systematic solutions.** One of the questions that has not received much attention is the relations among the many kinds of optimizations on data redundancy elimination, as well as their relations with model optimizations. On images, for instance, as Section 4 and Table 1 show, there are many ways to exploit data redundancy from various angles. Can these optimizations be used together? Is it worth doing that? Would there be some conflicts besides that they may all affect the model accuracy? How to use them together in the best way? More ambitiously, is it possible to build up frameworks that automatically apply one or more kinds of data redundancy exploitation techniques on a given dataset and learning task? How about the interplay with model optimizations? Answers to these questions may significantly advance the current understanding of data redundancy exploitation and tap into the potential better.

**(v) Synergy with DNN accelerator designs.** Yet another direction worth looking into is the synergy between the many data redundancy exploitation techniques and DNN hardware designs. In modern **application-specific integrated circuit (ASIC)** design, dataflow analysis [8, 79, 95] has been recognized as a crucial consideration. Some work on edge devices has tried to address memory capacity issues by exploiting data redundancy (e.g., through data compression) [44, 67, 68, 114, 128, 132, 149, 151, 152]. But in general, there is no systematic understanding of the relations of the many data redundancy exploitation techniques and the designs of DNN hardware accelerators. On the one hand, do those techniques stay profitable when DNN runs on hardware accelerators? On the other hand, can hardware accelerator designs gain some efficiency benefits by adopting some of the ideas in those techniques? Answers to these questions could lead to the novel synergy between the two strands of efforts.

## 8 CONCLUSION

Machine learning is based on two pillars, models and data. As an essential part of deep learning, data plays an important role. Making the best use of data determines the quality, speed, efficiency, and interpretability of machine learning. This survey summarizes the efforts in the research community in detecting and leveraging data redundancy in a variety of dimensions, introduces the first known taxonomy to categorize the existing techniques, and points out a set of directions yet to explore. As the landscape of machine learning evolves continuously at an unprecedented speed, we hope that this survey can provide a one-stop resource for researchers to attain a quick understanding of state-of-the-art and open issues, and hence benefit the industry practitioners and research community in selecting existing solutions for solving a problem at hand or creating new solutions to advance this critical field further.

## REFERENCES


[1] Anish Acharya, Rahul Goel, Angeliki Metallinou, and Inderjit Dhillon. 2019. Online embedding compression for text classification using low rank matrix factorization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6196–6203.

[2] Vahideh Akhlaghi, Amir Yazdanbakhsh, Kambiz Samadi, Rajesh K. Gupta, and Hadi Esmaeilzadeh. 2018. SnaPEA: Predictive early activation for reducing computation in deep convolutional neural networks. In *Proceedings of the 45th ACM/IEEE Annual International Symposium on Computer Architecture (ISCA'18)*. 662–673.

[3] Humam Alwassel. 2017. GitHub. Action Search: Spotting Targets in Videos and Its Application to Temporal Action Localization. https://github.com/HumamAlwassel/action-search.

[4] Humam Alwassel, Fabian Caba Heilbron, and Bernard Ghanem. 2018. Action search: Spotting actions in videos and its application to temporal action localization. In *Proceedings of the European Conference on Computer Vision (ECCV'18)*. 251–266.

[5] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. 2016. SoundNet: Learning sound representations from unlabeled video. In *Advances in Neural Information Processing Systems*. 892–900.

[6] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. 2004. Correlation clustering. *Machine Learning* 56, 1–3 (2004), 89–113.

[7] Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, 861–872.

[8] Tal Ben-Nun, Johannes de Fine Licht, Alexandros N. Ziogas, Timo Schneider, and Torsten Hoefler. 2019. Stateful dataflow multigraphs: A data-centric model for performance portability on heterogeneous architectures. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–14.

[9] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. 2020. What is the state of neural network pruning? *Proceedings of Machine Learning and Systems* 2 (2020), 129–146.

[10] Aaron F. Bobick and James W. Davis. 2001. The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23, 3 (2001), 257–267.

[11] Fergus W. Campbell and John G. Robson. 1968. Application of Fourier analysis to the visibility of gratings. *The Journal of Physiology* 197, 3 (1968), 551.

[12] Joao Carreira and Andrew Zisserman. 2017. Quo vadis, action recognition? A new model and the kinetics dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'17)*. 6299–6308.

[13] Lukas Cavigelli. 2017. Papers with Code. CBinfer: Change-Based Inference for Convolutional Neural Networks on Video Data. https://paperswithcode.com/paper/cbinfer-change-based-inference-for.

[14] Lukas Cavigelli. 2018. Papers with Code. CBinfer: Exploiting Frame-to-Frame Locality for Faster Convolutional Network Inference on Video Streams. https://paperswithcode.com/paper/cbinfer-exploiting-frame-to-frame-locality.

[15] Lukas Cavigelli and Luca Benini. 2019. CBinfer: Exploiting frame-to-frame locality for faster convolutional network inference on video streams. *IEEE Transactions on Circuits and Systems for Video Technology* 30, 5 (2019), 1451–1465.

[16] Lukas Cavigelli, Philippe Degen, and Luca Benini. 2017. CBinfer: Change-based inference for convolutional neural networks on video data. In *Proceedings of the 11th International Conference on Distributed Smart Cameras* (Stanford, CA, USA) *(ICDSC'17)*. Association for Computing Machinery, New York, NY, USA, 1–8.

[17] Sinjan Chakraborty, Sayantan Paul, Ram Sarkar, and Mita Nasipuri. 2019. Feature map reduction in CNN for handwritten digit recognition. In *Recent Developments in Machine Learning and Data Analytics*. Springer, 143–148.

[18] Hesen Chen, Ming Lin, Xiuyu Sun, Qian Qi, Hao Li, and Rong Jin. 2019. MuffNet: Multi-layer feature federation for mobile deep learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 0–0.

[19] Huixiang Chen, Mingcong Song, Jiechen Zhao, Yuting Dai, and Tao Li. 2019. 3D-based video recognition acceleration by leveraging temporal locality. In *Proceedings of the 46th ACM/IEEE Annual International Symposium on Computer Architecture (ISCA'19)*. 79–90.

[20] Hanting Chen, Yunhe Wang, Han Shu, Yehui Tang, Chunjing Xu, Boxin Shi, Chao Xu, Qi Tian, and Chang Xu. 2020. Frequency domain compact 3D convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'20)*. 1641–1650.

[21] Kai Chen, Jiaqi Wang, Shuo Yang, Xingcheng Zhang, Yuanjun Xiong, Chen Change Loy, and Dahua Lin. 2018. Optimizing video object detection via a scale-time lattice. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'18)*. 7814–7823.

[22] Weijie Chen, Yuan Zhang, Di Xie, and Shiliang Pu. 2019. A layer decomposition-recomposition framework for neuron pruning towards accurate lightweight networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3355–3362.

[23] Yunpeng Chen. 2019. Papers with Code. Drop an Octave: Reducing Spatial Redundancy in Convolutional Neural Networks with Octave Convolution. https://paperswithcode.com/paper/drop-an-octave-reducing-spatial-redundancy-in.

[24] Yunpeng Chen, Haoqi Fan, Bing Xu, Zhicheng Yan, Yannis Kalantidis, Marcus Rohrbach, Shuicheng Yan, and Jiashi Feng. 2019. Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV'19)*. 3435–3444.

[25] Yunchuan Chen, Lili Mou, Yan Xu, Ge Li, and Zhi Jin. 2016. Compressing neural language models by sparse word representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 226–235.

[26] Zhuo Chen. 2020. Papers with Code. ViP: Virtual Pooling for Accelerating CNN-based Image Classification and Object Detection. https://paperswithcode.com/paper/vip-virtual-pooling-for-accelerating-cnn.

[27] Zhuo Chen, Jiyuan Zhang, Ruizhou Ding, and Diana Marculescu. 2020. ViP: Virtual pooling for accelerating CNN-based image classification and object detection. In *The IEEE Winter Conference on Applications of Computer Vision*. 1180–1189.

[28] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. 2018. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine* 35, 1 (2018), 126–136.

[29] Ting-Wu Chin, Ruizhou Ding, and Diana Marculescu. 2019. AdaScale: Towards real-time video object detection using adaptive scaling. In *Proceedings of Machine Learning and Systems 2019*. 431–441.

[30] Kamran Chitsaz, Mohsen Hajabdollahi, Nader Karimi, Shadrokh Samavi, and Shahram Shirani. 2020. Acceleration of convolutional neural network using FFT-based split convolutions. *arXiv preprint arXiv:2003.12621* (2020).

[31] Tejalal Choudhary, Vipul Mishra, Anurag Goswami, and Jagannathan Sarangapani. 2020. A comprehensive survey on model compression and acceleration. *Artificial Intelligence Review* (2020), 1–43.

[32] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? An analysis of BERT's attention. In *BlackBoxNLP@ACL*.

[33] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. 2019. AutoAugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'19)*. 113–123.

[34] Zihang Dai. 2020. GitHub. Funnel-Transformer. https://github.com/laiguokun/Funnel-Transformer.

[35] Zihang Dai, Guokun Lai, Yiming Yang, and Quoc Le. 2020. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. In *Advances in Neural Information Processing Systems*.

[36] Fahim Dalvi. 2020. Papers with Code. Analyzing Redundancy in Pretrained Transformer Models. https://paperswithcode.com/paper/exploiting-redundancy-in-pre-trained-language.

[37] Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, Anthony Bau, and James Glass. 2019. What is one grain of sand in the desert? Analyzing individual neurons in deep NLP models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6309–6317.

[38] Fahim Dalvi, Hassan Sajjad, Nadir Durrani, and Yonatan Belinkov. 2020. Analyzing redundancy in pretrained transformer models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP'20)*. 4908–4926.

[39] Rafael Fão de Moura, Paulo C. Santos, João Paulo C. de Lima, Marco A. Z. Alves, Antonio C. S. Beck, and Luigi Carro. 2019. Skipping CNN convolutions through efficient memoization. In *International Conference on Embedded Computer Systems*. Springer, 65–76.

[40] Russell L. De Valois and Karen K. De Valois. 1980. Spatial vision. *Annual Review of Psychology* 31, 1 (1980), 309–341.

[41] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. 2015. FlowNet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV'15)*. 2758–2766.

[42] Heng Wang Du Tran, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. 2017. A closer look at spatiotemporal convolutions for action recognition. 2018 IEEE. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'17)*. 6450–6459.
[43] Word Embedding. 2020. Retrieved from https://en.wikipedia.org/wiki/Word_embedding.
[44] Hongxiang Fan, Xinyu Niu, Qiang Liu, and Wayne Luk. 2017. F-C3D: FPGA-based 3-dimensional convolutional neural network. In *2017 27th International Conference on Field Programmable Logic and Applications (FPL'17)*. IEEE, 1–4.
[45] Yang Fan, Fei Tian, Tao Qin, Jiang Bian, and Tie-Yan Liu. 2017. Neural data filter for boot-strapping stochastic gradient descent. In *Proceedings of the 5th International Conference on Learning Representations (ICLR'17)*.
[46] Christoph Feichtenhofer. 2020. X3D: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'20)*.
[47] Shuo Feng, Huiyu Zhou, and Hongbiao Dong. 2019. Using deep neural network with small dataset to predict material defects. *Materials & Design* 162 (2019), 300–310.
[48] Michael Figurnov. 2016. GitHub. perforated-cnn-matconvnet. https://github.com/mfigurnov/perforated-cnn-matconvnet.
[49] Mikhail Figurnov, Aizhan Ibraimova, Dmitry P. Vetrov, and Pushmeet Kohli. 2016. PerforatedCNNs: Acceleration through elimination of redundant convolutions. In *Advances in Neural Information Processing Systems*. 947–955.
[50] Akash Sunil Gaikwad and Mohamed El-Sharkawy. 2019. Pruning the convolution neural network (SqueezeNet) based on L 2 normalization of activation maps. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC'19)*. IEEE, 0392–0396.
[51] Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Hassan Sajjad, Preslav Nakov, Deming Chen, and Marianne Winslett. 2021. Compressing large-scale transformer-based models: A case study on BERT. *Transactions of the Association for Computational Linguistics* 9 (2021), 1061–1080.
[52] Mingfei Gao, Ruichi Yu, Ang Li, Vlad I. Morariu, and Larry S. Davis. 2018. Dynamic zoom-in network for fast object detection in large images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'18)*. 6926–6935.
[53] Xitong Gao. 2019. Papers with Code. Dynamic Channel Pruning: Feature Boosting and Suppression. https://paperswithcode.com/paper/dynamic-channel-pruning-feature-boosting-and.
[54] Xitong Gao, Yiren Zhao, Lukasz Dudziak, Robert D. Mullins, and Cheng-Zhong Xu. 2019. Dynamic channel pruning: Feature boosting and suppression. In *Proceedings of the 7th International Conference on Learning Representations (ICLR'19)*.
[55] Georgios Georgiadis. 2019. Accelerating convolutional neural networks via activation map compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'19)*. 7085–7095.
[56] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 1999. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB'99)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 518–529.
[57] Saurabh Goyal. 2020. Papers with Code. PoWER-BERT: Accelerating BERT Inference via Progressive Word-vector Elimination. https://paperswithcode.com/paper/power-bert-accelerating-bert-inference-for.
[58] Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan Chakaravarthy, Yogish Sabharwal, and Ashish Verma. 2020. PoWER-BERT: Accelerating BERT inference via progressive word-vector elimination. In *Proceedings of the International Conference on Machine Learning*, Vol. 119. 3690–3699.
[59] Benjamin Graham. 2014. Fractional max-pooling. *CoRR* abs/1412.6071 (2014).
[60] Hui Guan, Umang Chaudhary, Yuanchao Xu, Lin Ning, Lijun Zhang, and Xipeng Shen. 2021. Recurrent neural networks meet context-free grammar: Two birds with one stone. In *2021 IEEE International Conference on Data Mining (ICDM'21)*. 1078–1083.
[61] Yunhui Guo. 2018. A survey on methods and theories of quantized neural networks. *CoRR* abs/1808.04752 (2018).
[62] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. 2020. Deep learning for 3D point clouds: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).
[63] Manish Gupta, Vasudeva Varma, Sonam Damani, and Kedhar Nath Narahari. 2020. Compression of deep learning models for NLP. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 3507–3508.
[64] Wei Han, Pooya Khorrami, Tom Le Paine, Prajit Ramachandran, Mohammad Babaeizadeh, Honghui Shi, Jianan Li, Shuicheng Yan, and Thomas S. Huang. 2016. Seq-NMS for video object detection. *CoRR* abs/1602.08465 (2016).
[65] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'16)*. 770–778.
[66] Yihui He, Xiangyu Zhang, and Jian Sun. 2017. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV'17)*. 1389–1397.

[67] Kartik Hegde, Rohit Agrawal, Yulun Yao, and Christopher W. Fletcher. 2018. Morph: Flexible acceleration for 3D CNN-based video understanding. In *Proceedings of the 51tst Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'18)*. 933–946.
[68] Kartik Hegde, Jiyong Yu, Rohit Agrawal, Mengjia Yan, Michael Pellauer, and Christopher Fletcher. 2018. UCNN: Exploiting computational reuse in deep neural networks via weight repetition. In *Proceedings of the 45th ACM/IEEE Annual International Symposium on Computer Architecture (ISCA'18)*. 674–687.
[69] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2014. Distilling the knowledge in a neural network. In *Advances in Neural Information Processing Systems Deep Learning Workshop*.
[70] Zejiang Hou and Sun-Yuan Kung. 2020. A feature-map discriminant perspective for pruning deep neural networks. *CoRR* abs/2005.13796 (2020).
[71] Zejiang Hou and Sun-Yuan Kung. 2020. Efficient image super resolution via channel discriminative deep neural network pruning. In *ICASSP 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'20)*. IEEE, 3647–3651.
[72] Hengyuan Hu. 2016. Papers with Code. Network Trimming: A Data-Driven Neuron Pruning Approach Towards Efficient Deep Architectures. https://paperswithcode.com/paper/network-trimming-a-data-driven-neuron-pruning.
[73] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. 2016. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *CoRR* abs/1607.03250 (2016).
[74] Zehao Huang. 2018. Papers with Code. Data-Driven Sparse Structure Selection for Deep Neural Networks. https://paperswithcode.com/paper/data-driven-sparse-structure-selection-for.
[75] Zehao Huang and Naiyan Wang. 2018. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV'18)*. 304–320.
[76] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2017. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research* 18, 1 (2017), 6869–6898.
[77] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size. *CoRR* abs/1602.07360 (2016).
[78] Bunyodbek Ibrokhimov, Cheonghwan Hur, and Sanggil Kang. 2020. Effective node selection technique towards sparse learning. *Applied Intelligence* (2020).
[79] Andrei Ivanov, Nikoli Dryden, Tal Ben-Nun, Shigang Li, and Torsten Hoefler. 2021. Data movement is all you need: A case study on optimizing transformers. In *Proceedings of Machine Learning and Systems*, Vol. 3. 711–732.
[80] S. Ji, W. Xu, M. Yang, and K. Yu. 2013. 3D convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 1 (2013), 221–231.
[81] Yu-Gang Jiang, Zuxuan Wu, Jun Wang, Xiangyang Xue, and Shih-Fu Chang. 2017. Exploiting feature and class relationships in video categorization with regularized deep neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 2 (2017), 352–364.
[82] Xun Jiao, Vahideh Akhlaghi, Yu Jiang, and Rajesh K. Gupta. 2018. Energy-efficient neural networks using approximate computation reuse. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1223–1228.
[83] Daniel Kang. 2017. GitHub. NoScope. https://github.com/stanford-futuredata/noscope.
[84] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. NoScope: Optimizing neural network queries over video at scale. *Proc. VLDB Endow.* 10, 11 (Aug. 2017), 1586–1597.
[85] Kai Kang. 2017. GitHub. TPN: Tubelet Proposal Network. https://github.com/myfavouritekk/TPN.
[86] Kai Kang, Hongsheng Li, Tong Xiao, Wanli Ouyang, Junjie Yan, Xihui Liu, and Xiaogang Wang. 2017. Object detection in videos with tubelet proposal networks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'17)* (Jul. 2017).
[87] Kai Kang, Hongsheng Li, Junjie Yan, Xingyu Zeng, Bin Yang, Tong Xiao, Cong Zhang, Zhe Wang, Ruohui Wang, Xiaogang Wang, et al. 2018. T-CNN: Tubelets with convolutional neural networks for object detection from videos. *IEEE Transactions on Circuits and Systems for Video Technology* 28, 10 (Oct. 2018), 2896–2907.
[88] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'14)*.
[89] Sangyeob Kim, Juhyoung Lee, Sanghoon Kang, Jinsu Lee, and Hoi-Jun Yoo. 2020. A power-efficient CNN accelerator with similar feature skipping for face recognition in mobile devices. *IEEE Transactions on Circuits and Systems I: Regular Papers* 67, 4 (2020), 1181–1193.
[90] Yeachan Kim, Kang-Min Kim, and SangKeun Lee. 2020. Adaptive compression of word embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 3950–3959.

[91] Nikita Kitaev. 2020. Papers with Code. Reformer: The Efficient Transformer. https://paperswithcode.com/paper/reformer-the-efficient-transformer-1.
[92] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *Proceedings of the 8th International Conference on Learning Representations (ICLR'20)*.
[93] Okan Kopuklu, Neslihan Kose, Ahmet Gunduz, and Gerhard Rigoll. 2019. Resource efficient 3D convolutional neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*.
[94] Bruno Korbar, Du Tran, and Lorenzo Torresani. 2019. SCSampler: Sampling salient clips from video for efficient action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV'19)*. 6232–6242.
[95] Hyoukjun Kwon, Prasanth Chatarasi, Michael Pellauer, Angshuman Parashar, Vivek Sarkar, and Tushar Krishna. 2019. Understanding reuse, performance, and hardware cost of DNN dataflow: A data-centric approach. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* (Columbus, OH, USA) *(MICRO'52)*. Association for Computing Machinery, New York, NY, USA, 754–768.
[96] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *Proceedings of the 8th International Conference on Learning Representations (ICLR'20)*.
[97] Seulki Lee. 2020. GitHub. [RTAS 2020] SubFlow: A Dynamic Induced-Subgraph Strategy Toward Real-Time DNN Inference and Training. https://github.com/learning1234embed/SubFlow.
[98] Seulki Lee and Shahriar Nirjon. 2020. SubFlow: A dynamic induced-subgraph strategy toward real-time DNN inference and training. In *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'20)*. IEEE, 15–29.
[99] Hongyang Li. 2019. GitHub. Zoom-out-and-in Network for Region Proposal and Object Detection. https://github.com/hli2020/zoom_network.
[100] Hongyang Li, Yu Liu, Wanli Ouyang, and Xiaogang Wang. 2019. Zoom out-and-in network with map attention decision for region proposal and object detection. *International Journal of Computer Vision* 127, 3 (2019), 225–238.
[101] Hang Li, Chen Ma, Wei Xu, and Xue Liu. 2020. Feature statistics guided efficient filter pruning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. International Joint Conferences on Artificial Intelligence Organization, 2619–2625. https://doi.org/10.24963/ijcai.2020/363 Main track.
[102] Yuchao Li, Shaohui Lin, Baochang Zhang, Jianzhuang Liu, David Doermann, Yongjian Wu, Feiyue Huang, and Rongrong Ji. 2019. Exploiting kernel sparsity and entropy for interpretable CNN compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'19)*. 2800–2809.
[103] Min Lin, Qiang Chen, and Shuicheng Yan. 2014. Network in network. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR'14)*.
[104] LinkViz. 2020. Retrieved from https://hackernoon.com/visualizing-parts-of-convolutional-neural-networks-using-keras-and-cats5cc01b214e59.
[105] Ce Liu, Jenny Yuen, Antonio Torralba, Josef Sivic, and William T. Freeman. 2008. Sift flow: Dense correspondence across different scenes. In *Proceedings of the European Conference on Computer Vision (ECCV'08)*. 28–42.
[106] Mason Liu. 2017. Papers with Code. Mobile Video Object Detection with Temporally-Aware Feature Maps. https://paperswithcode.com/paper/mobile-video-object-detection-with-temporally.
[107] Peter J. Liu. 2018. Papers with Code. Generating Wikipedia by Summarizing Long Sequences. https://paperswithcode.com/paper/generating-wikipedia-by-summarizing-long.
[108] Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating Wikipedia by summarizing long sequences. In *Proceedings of the 6th International Conference on Learning Representations (ICLR'18)*.
[109] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. 2016. SSD: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision (ECCV'16)*. Springer, 21–37.
[110] Zhenhua Liu, Jizheng Xu, Xiulian Peng, and Ruiqin Xiong. 2018. Frequency-domain dynamic pruning for convolutional neural networks. In *Advances in Neural Information Processing Systems*. 1043–1053.
[111] Dongning Ma, Xunzhao Yin, Michael Niemier, X. Sharon Hu, and Xun Jiao. 2020. AxR-NN: Approximate computation reuse for energy-efficient convolutional neural networks. In *Proceedings of the 2020 on Great Lakes Symposium on VLSI*. 363–368.
[112] Huizi Mao, Taeyoung Kong, and Bill Dally. 2019. CaTDet: Cascaded tracked detector for efficient object detection from video. In *Proceedings of Machine Learning and Systems 2019, MLSys 2019, Stanford, CA, USA, March 31 - April 2, 2019*. mlsys.org.
[113] Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one?. In *Advances in Neural Information Processing Systems*. 14014–14024.

[114] Luca Mocerino, Valerio Tenace, and Andrea Calimera. 2019. Energy-efficient convolutional neural networks via recurrent data reuse. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 848–853.
[115] L. Ning, H. Guan, and X. Shen. 2019. Adaptive deep reuse: Accelerating CNN training on the fly. In *2019 IEEE 35th International Conference on Data Engineering (ICDE'19)*. 1538–1549.
[116] Lin Ning and Xipeng Shen. 2019. Deep reuse: Streamline CNN inference on the fly via coarse-grained computation reuse. In *Proceedings of the ACM International Conference on Supercomputing*. 438–448.
[117] Definition of Redundancy by Oxford Dictionary. 2020. Retrieved from https://www.lexico.com/en/definition/redundancy.
[118] Wanli Ouyang, Xiaogang Wang, Xingyu Zeng, Shi Qiu, Ping Luo, Yonglong Tian, Hongsheng Li, Shuo Yang, Zhe Wang, Chen-Change Loy, and Xiaoou Tang. 2015. DeepID-Net: Deformable deep convolutional neural networks for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'15)*. 2403–2412.
[119] Keunyoung Park and Doo-Hyun Kim. 2019. Accelerating image classification using feature map similarity in convolutional neural networks. *Applied Sciences* 9, 1 (2019), 108.
[120] W. Penny and R. Henson. 2006. Analysis of variance. *Statistical Parametric Mapping: The Analysis of Functional Brain Images* (2006), 166–177.
[121] Fernando Pérez-Cruz. 2009. Estimation of information theoretic measures for continuous random variables. In *Advances in Neural Information Processing Systems*. 1257–1264.
[122] Hamed Pirsiavash and Deva Ramanan. 2012. Detecting activities of daily living in first-person camera views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'12)*. IEEE, 2847–2854.
[123] Duvindu Piyasena, Rukshan Wickramasinghe, Debdeep Paul, Siew-Kei Lam, and Meiqing Wu. 2019. Reducing dynamic power in streaming CNN hardware accelerators by exploiting computational redundancies. In *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 354–359.
[124] Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, and Nicu Sebe. 2020. Binary neural networks: A survey. *Pattern Recognition* (2020), 107281.
[125] Mohammad Samragh Razlighi, Mohsen Imani, Farinaz Koushanfar, and Tajana Rosing. 2017. LookNN: Neural network with no multiplication. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 1775–1780.
[126] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, Vol. 28. Curran Associates, Inc.
[127] Tara N. Sainath and Carolina Parada. 2015. Convolutional neural networks for small-footprint keyword spotting. In *Sixteenth Annual Conference of the International Speech Communication Association*.
[128] Sahand Salamat, Mohsen Imani, Sarangh Gupta, and Tajana Rosing. 2018. RNSnet: In-memory neural network acceleration using residue number system. In *2018 IEEE International Conference on Rebooting Computing (ICRC'18)*. IEEE, 1–12.
[129] Kruttidipta Samal, Marilyn Wolf, and Saibal Mukhopadhyay. 2020. Attention-based activation pruning to reduce data movement in real-time AI: A case-study on local motion planning in autonomous vehicles. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* (2020).
[130] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. In *5th Workshop on Energy Efficient Machine Learning and Cognitive Computing NeurIPS 2019*.
[131] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 1715–1725.
[132] Junzhong Shen, You Huang, Zelong Wang, Yuran Qiao, Mei Wen, and Chunyuan Zhang. 2018. Towards a uniform template-based architecture for accelerating 2D and 3D CNNs on FPGA. In *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. 97–106.
[133] Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax?. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 1526–1534.
[134] Gil Shomron. 2019. GitHub. Thanks for Nothing: Predicting Zero-Valued Activations with Lightweight Convolutional Neural Networks. https://github.com/gilshm/zap.
[135] Gil Shomron, Ron Banner, Moran Shkolnik, and Uri Weiser. 2020. Thanks for nothing: Predicting zero-valued activations with lightweight convolutional neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV'20)*. Springer, 234–250.
[136] Connor Shorten and Taghi M. Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. *Journal of Big Data* 6, 1 (2019), 1–48.
[137] Taylor Simons and Dah-Jye Lee. 2019. A review of binarized neural networks. *Electronics* 8, 6 (2019), 661.

[138] Karen Simonyan and Andrew Zisserman. 2014. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 568–576.
[139] Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR'15)*.
[140] Arshdeep Singh, Padmanabhan Rajan, and Arnav Bhavsar. 2019. Deep hidden analysis: A statistical framework to prune feature maps. In *ICASSP 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'19)*. IEEE, 820–824.
[141] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. 2012. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR* abs/1212.0402 (2012).
[142] Yu-Chuan Su and Kristen Grauman. 2016. Leaving some stones unturned: Dynamic feature prioritization for activity detection in streaming video. In *European Conference on Computer Vision*. Springer, 783–800.
[143] Taiji Suzuki, Hiroshi Abe, Tomoya Murata, Shingo Horiuchi, Kotaro Ito, Tokuma Wachi, So Hirai, Masatoshi Yukishima, and Tomoaki Nishimura. 2020. Spectral pruning: Compressing deep neural networks via spectral analysis and its generalization error. In *IJCAI*.
[144] Antonio Torralba and Alexei A. Efros. 2011. Unbiased look at dataset bias. In *CVPR 2011*. IEEE, 1521–1528.
[145] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning spatiotemporal features with 3D convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV'15)*. 4489–4497.
[146] Gregor Urban, Krzysztof J. Geras, Samira Ebrahimi Kahou, Ozlem Aslan, Shengjie Wang, Rich Caruana, Abdelrahman Mohamed, Matthai Philipose, and Matt Richardson. 2017. Do deep convolutional nets really need to be deep and convolutional? In *Proceedings of the 5th International Conference on Learning Representations (ICLR'17)*.
[147] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 5797–5808.
[148] Dong Wang, Lei Zhou, Xueni Zhang, Xiao Bai, and Jun Zhou. 2018. Exploring linear relationship in feature map subspace for ConvNets compression. *CoRR* abs/1803.05729 (2018).
[149] Hai Wang, Mengjun Shao, Yan Liu, and Wei Zhao. 2017. Enhanced efficiency 3D convolution based on optimal FPGA accelerator. *IEEE Access* 5 (2017), 6909–6916.
[150] Ying Wang, Shengwen Liang, Huawei Li, and Xiaowei Li. 2019. A none-sparse inference accelerator that distills and reuses the computation redundancy in CNNs. In *Proceedings of the 56th Annual Design Automation Conference 2019*. 1–6.
[151] Yongchen Wang, Ying Wang, Huawei Li, Cong Shi, and Xiaowei Li. 2019. Systolic cube: A spatial 3D CNN accelerator architecture for low power video analysis. In *Proceedings of the 56th Annual Design Automation Conference 2019*. 1–6.
[152] Ying Wang, Yongchen Wang, Cong Shi, Long Cheng, Huawei Li, and Xiaowei Li. 2020. An edge 3D CNN accelerator for low power activity recognition. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2020).
[153] Yunhe Wang, Chang Xu, Shan You, Dacheng Tao, and Chao Xu. 2016. CNNpack: Packing convolutional neural networks in the frequency domain. In *Advances in Neural Information Processing Systems*. 253–261.
[154] Wenhao Wu, Dongliang He, Xiao Tan, Shifeng Chen, and Shilei Wen. 2019. Multi-agent reinforcement learning based frame sampling for effective untrimmed video recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV'19)*. 6222–6231.
[155] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S. Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* (2020).
[156] Zuxuan Wu, Caiming Xiong, Chih-Yao Ma, Richard Socher, and Larry S. Davis. 2019. AdaFrame: Adaptive frame selection for fast video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'19)*. 1278–1287.
[157] Bin Yang, Junjie Yan, Zhen Lei, and Stan Z. Li. 2016. Craft objects from images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'16)*. 6043–6051.
[158] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R. Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*. 5753–5763.
[159] Serena Yeung. 2015. GitHub. End-to-end Learning of Action Detection from Frame Glimpses in Videos. https://github.com/syyeung/frameglimpses.
[160] Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. 2016. End-to-end learning of action detection from frame glimpses in videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'16)*. 2678–2687.

[161] Fuxun Yu, Chenchen Liu, Di Wang, Yanzhi Wang, and Xiang Chen. 2020. AntiDote: Attention-based dynamic optimization for neural network runtime efficiency. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE'20)*. IEEE, 951–956.

[162] Shihao Zhang, Weiyao Lin, Ping Lu, Weihua Li, and Shuo Deng. 2017. Kill two birds with one stone: Boosting both object detection accuracy and speed with adaptive patch-of-interest composition. In *2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW'17)*. IEEE, 447–452.

[163] Jian Zheng, Wei Yang, and Xiaohua Li. 2017. Training data reduction in deep neural networks with partial mutual information based feature selection and correlation matching based active learning. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'17)*. IEEE, 2362–2366.

[164] Yuefu Zhou, Ya Zhang, Yanfeng Wang, and Qi Tian. 2019. Accelerate CNN via recursive Bayesian pruning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV'19)*. 3306–3315.

[165] Menglong Zhu and Mason Liu. 2018. Mobile video object detection with temporally-aware feature maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'18)*. 5686–5695.

[166] Xizhou Zhu. 2017. Papers with Code. Towards High Performance Video Object Detection for Mobiles. https://paperswithcode.com/paper/towards-high-performance-video-object.

[167] Xizhou Zhu, Jifeng Dai, Lu Yuan, and Yichen Wei. 2018. Towards high performance video object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'18)*. 7210–7218.

[168] Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. 2017. Flow-guided feature aggregation for video object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV'17)*. 408–417.

[169] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. 2017. Deep feature flow for video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'17)*. 2349–2358.

[170] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. 2018. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems*. 875–886.

Received 3 November 2020; revised 21 February 2022; accepted 7 September 2022