Long-Term Feature Banks for Detailed Video Understanding

Chao-Yuan Wu[1,2] Christoph Feichtenhofer[2] Haoqi Fan[2]
Kaiming He[2] Philipp Krähenbühl[1] Ross Girshick[2]

[1]The University of Texas at Austin [2]Facebook AI Research (FAIR)

Abstract

To understand the world, we humans constantly need to relate the present to the past, and put events in context. In this paper, we enable existing video models to do the same. We propose a long-term feature bank*—supportive information extracted over the entire span of a video—to augment state-of-the-art video models that otherwise would only view short clips of 2-5 seconds. Our experiments demonstrate that augmenting 3D convolutional networks with a long-term feature bank yields state-of-the-art results on three challenging video datasets: AVA, EPIC-Kitchens, and Charades. Code is available online*[1].

1. Introduction

What is required to understand a movie? Many aspects of human intelligence, for sure, but *memory* is particularly important. As a film unfolds there's a constant need to relate whatever is happening in the present to what happened in the past. Without the ability to use the past to understand the present, we, as human observers, would not understand what we are watching.

In this paper, we propose the idea of a *long-term feature bank* that stores a rich, time-indexed representation of the entire movie. Intuitively, the long-term feature bank stores features that encode information about past and (if available) future scenes, objects, and actions. This information provides a supportive context that allows a video model, such as a 3D convolutional network, to better infer what is happening in the present (see Fig. 1, 2, and 5).

We expect the long-term feature bank to improve state-of-the-art video models because most make predictions based only on information from a *short* video clip, typically 2-5 seconds [5, 33, 46–48, 51, 52, 56]. The reason for this short-term view is simple: benchmark advances have resulted from training *end-to-end* networks that use some form of 3D convolution, and these 3D convolutions require dense sampling in time to work effectively. Therefore, to fit in GPU memory the video inputs must be short.

[1]https://github.com/facebookresearch/video-long-term-feature-banks

Figure 1. What are these people doing? Current 3D CNN video models operate on short clips spanning only ∼4 seconds. Without observing longer-term context, recognition is difficult. (Video from the AVA dataset [14]; see next page for the answer.)

The long-term feature bank is inspired by works that leverage long-range temporal information by using pre-computed visual features [25, 31, 45, 57]. However, these approaches extract features from isolated frames using an ImageNet pre-trained network, and then use the features as input into a trained pooling or recurrent network. Thus the *same* features represent both the present and the long-term context. In contrast, we propose to *decouple* the two: the long-term feature bank is an auxiliary component that *augments* a standard video model, such as a state-of-the-art 3D CNN. This design enables the long-term feature bank to store flexible supporting information, such as object detection features, that differ from what the 3D CNN computes.

Integrating the long-term feature bank with 3D CNNs is straightforward. We show that a variety of mechanisms are possible, including an attentional mechanism that relates information about the present (from the 3D CNN) to long-range information stored in the long-term feature bank. We illustrate its application to different tasks with different output requirements: we show results on datasets that require object-level as well as frame- or video-level predictions.

Finally, we conduct extensive experiments demonstrating that augmenting 3D CNNs with a long-term feature bank yields state-of-the-art results on three challenging video datasets: AVA spatio-temporal action localization [14], EPIC-Kitchens verb, noun, and action classification [6], and Charades video classification [38]. Our ablation study establishes that the improvements on these tasks arise from the integration of long-term information.

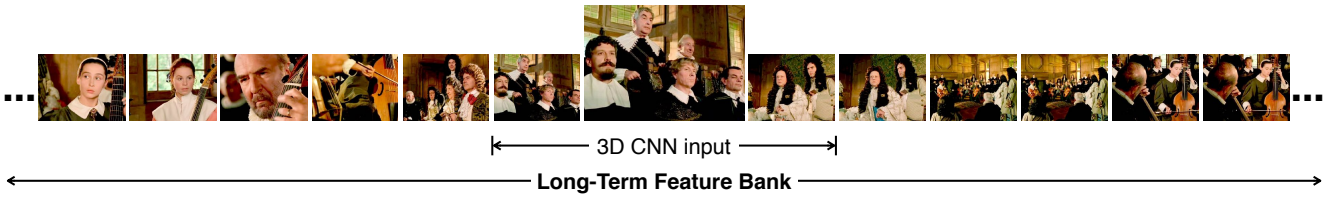


Figure 2. The action becomes clear when relating the target frame to the long-range context. Our **long-term feature bank** provides long-term supportive information that enables video models to better understand the present. (AVA ground-truth label: 'listening to music'.)

2. Related Work

Deep networks are the dominant approach for video understanding [5, 21, 33, 39, 46–48, 50, 51, 56]. This includes the highly successful two-stream networks [21, 39, 50] and 3D convolutional networks [5, 33, 46–48, 51, 56]. In this paper, we use 3D CNNs, but the long-term feature bank can be integrated with other families of video models as well.

Temporal and relationship models include RNNs that model the evolution of video frames [7, 24, 27, 44, 57] and multilayer perceptrons that model ordered frame features [58]. To model finer-grained interactions, a growing line of work leverages pre-computed object proposals [52] or detections [4, 30], and models their co-occurrence [30, 43, 52], temporal order [4], or spatial arrangement [52] within a short clip.

Long-term video understanding with modern CNNs is less explored, in part due to GPU memory constraints. One strategy to overcome these constraints is to use pre-computed features without end-to-end training [25, 31, 45, 57]. These methods do not optimize features for a target task, and thus are likely suboptimal. Another strategy is to use aggressive subsampling [50, 58] or large striding [8]. TSN [50] samples 3-7 frames per video. ST-ResNet [8] uses a temporal stride of 15. To our knowledge, our approach is the first that enjoys the best of three worlds: end-to-end learning for strong short-term features with dense sampling *and* decoupled, flexible long-term modeling.

Spatio-temporal action localization is an active research area [12, 14, 17, 32, 40, 53]. Most recent approaches extend object detection frameworks [10, 34] to first propose tubelets/boxes in a short clip/frame, and then classify the tubelets/boxes into action classes [14, 17, 20, 32, 36, 36]. The detected tubelets/boxes can then be optionally linked to form full action tubes [12, 17, 20, 32, 36, 40]. In contrast to our method, these methods find actions within each frame or clip independently without exploiting long-term context.

Information 'bank' representations, such as *object bank* [26], *detection bank* [1], and *memory networks* [42] have been used as image-level representations, for video indexing and retrieval, and for modeling information in text corpora. We draw inspiration from these approaches and develop methodologies for detailed video understanding tasks.

3. Long-Term Feature Bank Models

For computer vision models to make accurate predictions on long, complex videos they will certainly require the ability to relate what is happening in the present to events that are distant in time. With this motivation in mind, we propose a model with a *long-term feature bank* to explicitly enable these interactions.

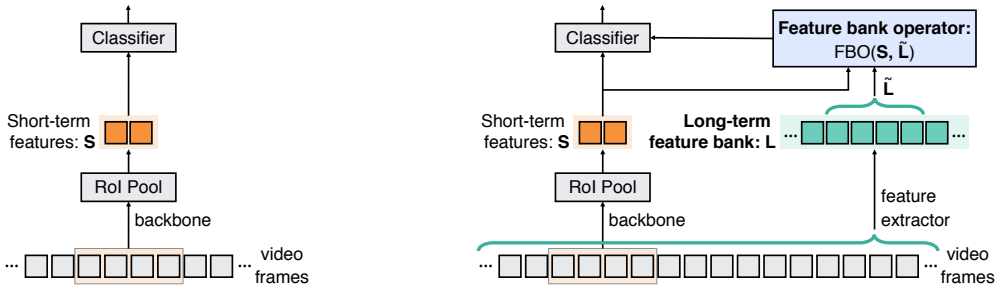
3.1. Method Overview

We describe how our method can be used for the task of *spatio-temporal action localization*, where the goal is to detect all actors in a video and classify their actions. Most state-of-the-art methods [9, 14, 43] combine a 'backbone' 3D CNN (*e.g.*, C3D [46], I3D [5]) with a region-based person detector (*e.g.*, Fast/Faster R-CNN [10, 34]). To process a video, it is split into *short* clips of 2-5 seconds, which are *independently* forwarded through the 3D CNN to compute a feature map, which is then used with region proposals and region of interest (RoI) pooling to compute RoI features for each candidate actor [9, 14]. This approach, which captures only short-term information, is depicted in Fig. 3a.

The central idea in our method is to extend this approach with two new concepts: (1) a **long-term feature bank** that intuitively acts as a 'memory' of what happened during the entire video—we compute this as RoI features from detections at regularly sampled time steps; and (2) a **feature bank operator** (FBO) that computes interactions between the short-term RoI features (describing what actors are doing now) and the long-term features. The interactions may be computed through an attentional mechanism, such as a non-local block [51], or by feature pooling and concatenation. Our model is summarized in Fig. 3b. We introduce these concepts in detail next.

3.2. Long-Term Feature Bank

The goal of the long-term feature bank, L, is to provide relevant contextual information to aid recognition at the current time step. For the task of spatio-temporal action localization, we run a person detector over the entire video to generate a set of detections for each frame. In parallel, we run a standard clip-based 3D CNN, such as C3D [46] or I3D [5], over the video at regularly spaced intervals, such as every one second. We then use RoI pooling to extract fea-



(a) **3D CNN** (b) **3D CNN with a Long-Term Feature Bank (Ours)**

Figure 3. We contrast our model with standard methods. (a) **3D CNN:** vanilla 3D CNNs process a *short* clip from a video (*e.g.*, 2-5 seconds) and use pooling to obtain a representation of the clip (*e.g.*, [5, 46, 51]). (b) **Long-Term Feature Bank (Ours):** We extend the vanilla 3D CNN with a long-term feature bank L and a feature bank operator $\mathrm{FBO}(S, \tilde{L})$ that computes interactions between the short-term and long-term features. Our model is able to integrate information over a long temporal support, lasting minutes or even the whole video.

tures for all person detections at each time-step processed by the 3D CNN. Formally, $L = [L_0, L_1, \ldots, L_{T-1}]$ is a time-indexed list of features for video time steps $0, \ldots, T-1$, where $L_t \in \mathbb{R}^{N_t \times d}$ is the matrix of N_t d-dimensional RoI features at time t. Intuitively, L provides information about when and what all actors are doing *in the whole video* and it can be efficiently computed in a single pass over the video by the detector and 3D CNN.

3.3. Feature Bank Operator

Our model references information from the long-term features L via a *feature bank operator*, $\mathrm{FBO}(S_t, \tilde{L}_t)$. The feature bank operator accepts inputs S_t and $\tilde{L}_t$, where S_t is the short-term RoI pooled feature and $\tilde{L}_t$ is $[L_{t-w}, \ldots, L_{t+w}]$, a subset of L centered at the current clip at t within 'window' size $2w+1$, stacked into a matrix $\tilde{L}_t \in \mathbb{R}^{N \times d}$, where $N = \sum_{t'=t-w}^{t+w} N_{t'}$. We treat the window size $2w+1$ as a hyperparameter that we cross-validate in our experiments. The output is then channel-wise concatenated with S_t and used as input into a linear classifier.

Intuitively, the feature bank operator computes an updated version of the pooled short-term features S_t by relating them to the long-term features. The implementation of FBO is flexible. Variants of attentional mechanisms are an obvious choice and we will consider multiple instantiations in our experiments.

Batch *vs*. Casual. Thus far we have assumed a *batch* setting in which the entire video is available for processing. Our model is also applicable to online, *casual* settings. In this case, $\tilde{L}_t$ contains only past information of window size $2w+1$; we consider both batch and causal modes of operation in our experiments.

3.4. Implementation Details

Backbone. We use a standard 3D CNN architecture from recent video classification work. The model is a ResNet-50 [16] that is pre-trained on ImageNet [35] and 'inflated' into a network with 3D convolutions (over space and time) using the I3D technique [5]. The network structure is modified to include non-local operations [51]. After inflating the network from 2D to 3D, we pre-train it for video classification on the Kinetics-400 dataset [5]. The model achieves 74.9% (91.6%) top-1 (top-5) accuracy on the Kinetics-400 [5] validation set. Finally, we remove the temporal striding for conv_1 and pool_1 following [52], and remove the Kinetics-specific classification layers to yield the backbone model. The exact model specification is given in Supplementary Material. The resulting network accepts an input of shape $32 \times H \times W \times 3$, representing 32 RGB frames with spatial size $H \times W$, and outputs features with shape $16 \times H/16 \times W/16 \times 2048$. The same architecture is used to compute short-term features S and long-term features L. Parameters are not shared between these two models unless otherwise noted.

RoI Pooling. We first average pool the video backbone features over the time axis. We then use RoIAlign [15] with a spatial output of 7×7, followed by spatial max pooling, to yield a single 2048 dimensional feature vector for the RoI. This corresponds to using a temporally straight tube [14].

Feature Bank Operator Instantiations. The feature bank operator can be implemented in a variety of ways. We experiment with the following choices; others are possible.

– **LFB NL:** Our default feature bank operator $\mathrm{FBO}_{\mathrm{NL}}(S_t, \tilde{L}_t)$ is an attention operator. Intuitively, we use S_t to attend to features in $\tilde{L}_t$, and add the attended information back to S_t via a shortcut connection. We use a simple implementation in which $\mathrm{FBO}_{\mathrm{NL}}(S_t, \tilde{L}_t)$ is a stack of up to three non-local (NL) blocks [51]. We replace the self-attention of the standard non-local block [51] with attention between the local features S_t and the long-term feature window $\tilde{L}_t$, illustrated in Fig. 4. In addition, our design uses layer normalization (LN) [3] and dropout [41] to improve regularization. We found these modifications to be important since our target tasks contain relatively few training videos and exhibit overfitting. The stack of modified

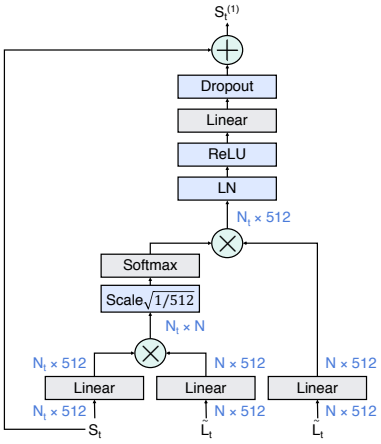


Figure 4. **Our modified non-local block design.** Here we plot the first layer $S_t^{(1)} = \mathrm{NL}'_{\theta_1}(S_t, \tilde{L}_t)$ as an example. '$\otimes$' denotes matrix multiplication, and '$\oplus$' denotes element-wise sum.

non-local blocks, denoted as NL′, is iterated as:

$$S_t^{(1)} = \mathrm{NL}'_{\theta_1}(S_t, \tilde{L}_t),$$
$$S_t^{(2)} = \mathrm{NL}'_{\theta_2}(S_t^{(1)}, \tilde{L}_t),$$
$$\vdots$$

where $\theta_{\{1,2,\dots\}}$ are learnable parameters. Similar to Wang *et al*. [52], we use a linear layer to reduce the $\mathrm{FBO}_{\mathrm{NL}}$ input dimensionality to 512 and apply dropout [41] with rate 0.2. Thus the input of the final linear classifier is 2048 (S_t) + 512 ($\mathrm{FBO}_{\mathrm{NL}}$ output) = 2560 dimensional.

– LFB Avg/Max: This version uses a simpler variant in which $\mathrm{FBO}_{\mathrm{pool}}(S_t, \tilde{L}_t) = \mathrm{pool}(\tilde{L}_t)$, where pool can be either average or max pooling. This implementation results in a classifier input that is 2048 (S_t) + 2048 ($\mathrm{FBO}_{\mathrm{pool}}$ output) = 4096 dimensional.

Training. Joint, end-to-end training of the entire model (Fig. 3b) is not feasible due to the computational and memory complexity of back-propagating through the long-term feature bank. Instead, we treat the 3D CNN and detector that are used to compute L as *fixed* components that are trained offline, but still on the target dataset, and not updated subsequently. We have experimented with alternating optimization methods for updating these models, similar to target propagation [23], but found that they did not improve results. Dataset-specific training details are given later.

A Baseline Short-Term Operator. To validate the benefit of incorporating long-term information, we also study a 'degraded' version of our model that does not use a long-term feature bank. Instead, it uses a *short-term operator* that is identical to $\mathrm{FBO}_{\mathrm{NL}}$, but only references the information from within a clip: $\mathrm{STO}(S_t) := \mathrm{FBO}_{\mathrm{NL}}(S_t, S_t)$. STO is conceptually similar to [52] and allows for back-propagation. We observed substantial overfitting with STO and thus applied additional regularization techniques. See Supplementary Material for details.

4. Experiments on AVA

We use the AVA dataset [14] for extensive ablation studies. AVA consists of 235 training videos and 64 validation videos; each video is a 15 minute segment taken from a movie. Frames are labeled sparsely at 1 FPS. The labels are: one bounding box around each person in the frame together with a multi-label annotation specifying which actions the person in the box is engaged in within ± 0.5 seconds of the labeled frame. The action label space is defined as 80 'atomic' actions defined by the dataset authors.

The task in AVA is spatio-temporal action localization: each person appearing in a test video must be detected in each frame and the multi-label actions of the detected person must be predicted correctly. The quality of an algorithm is judged by a mean average precision (mAP) metric that requires at least 50% intersection over union (IoU) overlap for a detection to be matched to the ground-truth while simultaneously predicting the correct actions.

4.1. Implementation Details

Next, we describe the object detector, input sampling, and training and inference details used for AVA.

Person Detector. We use Faster R-CNN [34] with a ResNeXt-101-FPN [28, 55] backbone for person detection. The model is pre-trained on ImageNet [35] and COCO keypoints [29], and then fine-tuned on AVA bounding boxes; see Supplementary Material for training details. The final model obtains 93.9 AP@50 on the AVA validation set.

Temporal Sampling. Both short- and long-term features are extracted by 3D CNNs that use 32 input frames sampled with a temporal stride of 2 spanning 63 frames ($\sim$2 seconds in 30 FPS video). Long-term features are computed at one clip per second over the whole video, with a 3D CNN model (Fig. 3a) fine-tuned on AVA.

Training. We train our models using synchronous SGD with a minibatch size of 16 clips on 8 GPUs (*i.e*., 2 clips per GPU), with batch normalization [18] layers frozen. We train all models for 140k iterations, with a learning rate of 0.04, which is decreased by a factor of 10 at iteration 100k and 120k. We use a weight decay of 10^{-6} and momentum of 0.9. For data augmentation, we perform random flipping, random scaling such that the short side $\in [256, 320]$ pixels, and random cropping of size 224×224. We use both ground-truth boxes and predicted boxes with scores at least 0.9 for training. This accounts for the discrepancy between ground-truth-box distribution and predicted-box distribution, and we found it beneficial. We assign labels of a ground-truth box to a predicted box if they overlap with IoU at least 0.9. A predicted box might have no labels assigned. Since the number of long-term features N differs from clip to clip, we pad zero-vectors for clips with fewer long-term features to simplify minibatch training.

Support (sec.)	2	3	5	10	15	30	60
3D CNN	22.1	22.2	22.3	20.0	19.7	17.5	15.7
STO	23.2	23.6	23.3	21.5	20.9	18.5	16.9
LFB (causal)	-	24.0	24.3	24.6	24.8	24.6	24.2
LFB (batch)	-	24.2	24.7	25.2	25.3	25.3	**25.5**

(a) **Temporal support** (mAP in %)

	mAP
K400 feat.	19.7
AVA feat.	24.3
LFB	**25.5**

(b) **Feature decoupling**

	mAP
Avg pool	23.1
Max pool	23.2
NL	**25.5**

(c) **LFB operator**

	mAP
Global pool	24.9
2×2 Grid	25.1
4×4 Grid	25.1
Detection	**25.5**

(d) **LFB spatial design**

	mAP
1L	25.1
2L (default)	25.5
2L w/o scale	25.2
2L w/o LN	23.9
2L w/o dropout	25.4
2L (dot product)	25.5
2L (concat)	25.3
3L	**25.8**

(e) **LFB NL design**

	params	FLOPs	mAP
3D CNN	1×	1×	22.1
3D CNN ×2	2×	2×	22.9
STO	1.00×	1.12×	23.2
STO ×2	2.00×	2.24×	24.1
LFB (2L)	2.00×	2.12×	25.5
LFB (3L)	2.00×	2.15×	**25.8**

(f) **Model complexity**

	mAP
R50-I3D-NL	
center-crop (default)	25.8
R101-I3D-NL	
center-crop (default)	26.8
3-crop	27.1
3-crop+flips	27.4
3-crop+flips+3-scale	**27.7**

(g) **Backbone & testing**

model	flow	val	test
AVA [14]	✓	15.6	-
ACRN [43]	✓	17.4	-
RTPR [24]	✓	22.3	-
9-model ens. [19]	✓	25.6	21.1
R50-I3D-NL [19]		19.3	-
RTPR [24]		20.5	-
Girdhar *et al*. [9]		21.9	21.0
LFB (R50)		25.8	24.8
LFB (best R101)		**27.7**	**27.2**

(h) **Comparison to prior work**

Table 1. **AVA ablations and test results**. **STO:** 3D CNN with a non-local (NL) short-term operator; **LFB:** 3D CNN with a **long-term feature bank**; the LFB operator is a two-layer (2L) NL block by default. We perform ablations on the AVA spatio-temporal action localization. The results validate that longer-term information is beneficial, that the improvement is larger than what would be observed by ensembling, and demonstrate various design choices. Finally, we show state-of-the-art results on the AVA test set.

Inference. At test time we use detections with scores $\geq$ 0.85. All models rescale the short side to 256 pixels and use a single center crop of 256×256. For both training and inference, if a box crosses the cropping boundary, we pool the region within the cropped clip. In the rare case where a box falls out of the cropped region, RoIAlign [15] pools the feature at the boundary.

4.2. Ablation Experiments

Temporal Support. We first analyze the impact of increasing temporal support on models both with and without LFB. For models without LFB, we evaluate a vanilla 3D CNN (Fig. 3a) and a 3D CNN extended with STO (denoted 'STO' in tables). To increase their temporal support, we increase the temporal stride but fix the number of input frames so that the model covers a longer temporal range while still being feasible to train. To increase the temporal support of LFB models, we increase the 'window size' $2w + 1$ of $\tilde{L}_t$.

Table 1a compares model performance. Increasing temporal support through striding in fact hurts the performance of both '3D CNN' and 'STO'. Temporal convolution might not be suitable for long-term patterns, since long-term patterns are more diverse, and include challenging scene cuts. On the other hand, increasing temporal support by adding an LFB steadily improves performance, leading to a large gain over the original '3D CNN' (22.1 $\rightarrow$ **25.5**). The online (causal) setting shows a similar trend.

Overall we observe strong improvements given by long-range context even though AVA actions are designed to be 'atomic' and localized within $\pm$0.5 seconds. For the rest of the ablation study, we focus on the batch setting and use a window size of 60 seconds due to the strong performance.

Feature Decoupling. In Table 1b, we compare our decoupled feature approach with prior long-term modeling strategies where a single, pre-computed feature type is used (*e.g.* [25, 31, 45, 57]). To do this, we use the same 3D CNN for both short-term and long-term feature bank computation and keep it fixed during training; only the parameters of the FBO and classifier are updated. We consider two choices: a Kinetics-400 [5] pre-trained 3D CNN ('K400 feat.') and a 3D CNN that is fine-tuned on AVA ('AVA feat.'). Our decoupled approach, which updates the short-term 3D CNN based on the long-term context, works significantly better.

FBO Function Design. We next compare different FBO function designs in Table 1c. We see that a non-local function significantly outperforms pooling on AVA. This is not surprising, since videos in AVA (and videos in general) are multi-actor, multi-action, and can exhibit complex interactions possibly across a long range of time. We expect a more complex function class is required for reasoning through the complex scenes. Nonetheless, pooling still offers a clear improvement. This again confirms the importance of long-term context in video understanding.

FBO Input Design. What spatial granularity is required for complex video understanding? In Table 1d, we compare constructing long-term features using detected objects ('Detection'), regular grids ('Grid'), and non-spatial features ('Global pool'). In the 'Grid' experiments, we divide the feature map into a $k \times k$ grid, for $k = 2, 4$, and average the features within each bin to obtain a localized representation without an object detector (similar to ACRN [43]).

Table 1d shows that the actor-level features ('Detection') works better than coarser regular-grid or non-spatial fea-

Figure 5. **Example Predictions.** We compare predictions made by models using LFB of different window sizes. Through LFB, the model is able to exploit information distant in time, *e.g.*, the zoomed-in frames in example A and C, to improve predictions. We encourage readers to zoom in for details. (Blue: correct labels. Red: incorrect labels. Best viewed on screen.)

tures. We believe this suggests a promising future research direction that moves from global pooling to a more detailed modeling of objects/actors in video.

Non-Local Block Design. Next, we ablate our NL block design. In Table 1e, we see that adding a second layer of NL blocks leads to improved performance. In addition, scaling [49], layer normalization [3], and dropout [41] all contribute to good performance. Layer normalization [3], which is part of our modified NL design, is particularly important. As found in [51], the default embedded Gaussian variant performs similarly to dot product and concatenation. (Later we found that adding a third layer of NL blocks further improves accuracy, but otherwise use two by default.)

Model Complexity. Our approach uses two instances of the backbone model: one to compute the long-term features and another to compute the short-term features. It thus uses around $2\times$ more parameters and computation than our baselines. What if we simply use $2\times$ more computation on baselines through an ensemble? We find that both '3D CNN' or 'STO' are not able to obtain a similar gain when ensembled; the LFB model works significantly better (Table 1f).

Example Predictions. We qualitatively present a few examples illustrating the impact of LFB in Fig. 5. Specifically, we compare predictions made by models with LFB of different window sizes, from 4 seconds to 10 seconds. We see that when observing only short-term information, the model is confused and not able to make accurate predictions in these cases. When observing more context, *e.g.*, zoomed-in frames (example A, C) or frames that give clearer cue (example B, D), an LFB model is able to leverage the information and improve predictions.

Backbone and Testing. So far for simplicity, we have used a relatively small backbone of R50-I3D-NL and performed center-crop testing. In Table 1g, we show that with an R101-I3D-NL backbone, LFB (3L) achieves 26.8 mAP, and with standard testing techniques, 27.7 mAP. We use short side $\in \{224, 256, 320\}$ pixels for 3-scale testing.

Comparison to Prior Work. Finally we compare with other state-of-the-art methods (Table 1h). For fair comparison, we follow Girdhar *et al*. [9] and train on both the training and validation set for test set evaluation.[2] For this model we use a $1.5\times$ longer schedule due to the larger data size.

Our model, using only RGB frames, significantly outperforms all prior work including strong competition winners that use optical flow and large ensembles. Our single model outperforms the best previous single-model entry by Girdhar *et al*. [9] with a margin of 5.8 and 6.2 points mAP on validation and test set respectively.

[2]Test set performance evaluated by ActivityNet server.

5. Experiments on EPIC-Kitchens

The long-term feature bank is a generalizable and flexible concept. We illustrate this point on two tasks in the EPIC-Kitchens dataset [6] for which we store different types of information in the feature bank.

The EPIC-Kitchens dataset consists of videos of daily activities (mostly cooking) recorded in participants' native kitchen environments. Segments of each video are annotated with one verb (*e.g.*, 'squeeze') and one noun (*e.g.*, 'lemon'). The task is to predict the *verb*, *noun*, and the combination (termed *action* [6]) in each segment. Performance is measured by top-1 and top-5 accuracy.

The dataset consists of 39,594 segments in 432 videos. Test set annotations are not released; for validation we split the original training set into a new training/validation split following Baradel *et al*. [4]. We train independent models to recognize *verbs* and *nouns*, and combine their predictions for *actions*. For actions, we additionally use a prior based on the training class frequencies of verbs and nouns; see Supplementary Material for details.

5.1. Implementation Details

Long-Term Feature Banks. Recognizing which object a person is interacting with (the *noun* task) from a short segment is challenging, because the object is often occluded, blurred, or can even be out of the scene. Our LFB is well-suited for addressing these issues, as the long-term supporting information can help resolve ambiguities. For example, if we know that the person took a lemon from the refrigerator 30 seconds ago, then cutting a lemon should be more likely. Based on this motivation, we construct an LFB that contains *object-centric features*. Specifically, we use Faster R-CNN to detect objects and extract object features using RoIAlign from the detector's feature maps (see Supplementary Material for details of our detector).

On the other hand, for recognizing *verbs* we use a *video model* to construct an LFB that captures motion patterns. Specifically, we use our baseline 3D CNN fine-tuned on EPIC-Kitchens verbs to extract clip-level features every 1 second of video. Our default setting uses a window size of 40 seconds for the *verb* model, and 12 seconds for the *noun* model, chosen on the validation set.

Adaptation to Segment-Level Tasks. To adapt our model for segment-level predictions, we replace the RoI pooling by global average pooling, resulting in $S \in \mathbb{R}^{1\times 2048}$. For the STO baseline, we use a slightly modified formulation: $\mathrm{STO}(S'_t) := \mathrm{FBO}_{\mathrm{NL}}(S_t, S'_t)$, where S'_t contains 16 spatially pooled features, each at a temporal location, and S_t is S'_t pooled over the time axis. STO learns to interact with information at different time steps within the short clip. Training and inference procedures are analogous to our experiments on AVA; see Supplementary Material for details.

	Verbs		Nouns		Actions	
	top-1	top-5	top-1	top-5	top-1	top-5
Validation						
Baradel [4]	40.9	-	-	-	-	-
3D CNN	49.8	80.6	26.1	51.3	19.0	37.8
3D CNN ens.	50.7	81.2	27.8	52.8	20.0	39.0
STO	51.0	80.8	26.6	51.5	19.5	38.3
STO ens.	51.9	81.2	27.8	52.5	20.5	39.4
LFB NL	51.7	81.2	29.2	55.3	21.4	40.2
LFB Avg	**53.0**	**82.3**	29.1	55.4	21.2	40.8
LFB Max	52.6	81.2	**31.8**	**56.8**	**22.8**	**41.1**
Δ	*+3.2*	*+1.7*	*+5.7*	*+5.5*	*+3.8*	*+3.3*
Test s1 (seen)						
TSN RGB [6]	45.7	85.6	36.8	64.2	19.9	41.9
TSN Flow [6]	42.8	79.5	17.4	39.4	9.0	21.9
TSN Fusion [6]	48.2	84.1	36.7	62.3	20.5	39.8
LFB Max	**60.0**	**88.4**	**45.0**	**71.8**	**32.7**	**55.3**
Test s2 (unseen)						
TSN RGB [6]	34.9	74.6	21.8	45.3	10.1	25.3
TSN Flow [6]	40.1	73.4	14.5	33.8	6.7	18.6
TSN Fusion [6]	39.4	74.3	22.7	45.7	10.9	25.3
LFB Max	**50.9**	**77.6**	**31.5**	**57.8**	**21.2**	**39.4**

Table 2. **EPIC-Kitchens validation and test server results.** Augmenting 3D CNN with LFB leads to significant improvement.

5.2. Quantitative Evaluation

We now quantitatively evaluate our LFB models. Table 2 shows that augmenting a 3D CNN with LFB significantly boosts the performance for all three tasks. Using object features for the *noun* model is particularly effective, leading to 5.7% (26.1 → **31.8**) absolute improvement over our strong baseline model. On *verb* recognition, LFB with 3D CNN features results in 3.2% (49.8 → **53.0**) improvement and outperforms previous state-of-the-art by Baradel *et al*. [4] by 12.1% (40.9 → **53.0**).

We also observe that $\mathrm{FBO}_{\mathrm{Max}}$ and $\mathrm{FBO}_{\mathrm{Avg}}$ outperform $\mathrm{FBO}_{\mathrm{NL}}$ on EPIC-Kitchens. We conjecture that this is due to the simpler setting: each video has only one person, doing one thing at a time, without the complicated person-person interactions of AVA. Thus a simpler function suffices.

On the test set, our method outperforms prior work by a large margin on both the 'seen kitchens (s1)' and 'unseen kitchens (s2)' settings. Our LFB model outperforms the Two-Stream [39] TSN [50] baseline by 50% relatively for s1, and almost doubles the performance for s2, in terms of top-1 *action* accuracy.

6. Experiments on Charades

Finally we evaluate our approach on the Charades dataset [38]. The Charades dataset contains 9,848 videos with an average length of 30 seconds. In each video, a person can perform one or more actions. The task is to recognize all the actions in the video without localization.

iterations / lr / wd	50k / 0.0025 / 1e-4 [52]	24k / 0.02 / 1.25e-5
3D CNN	33.8	**38.3**
STO	37.8	**39.6**

Table 3. **Training schedule on Charades.** Our 2× shorter schedule works significantly better than the schedule used in STRG [52].

	backbone	modality	train / val	trainval / test
2-Strm. [39] (from [37])	VGG16	RGB+Flow	18.6	-
Asyn-TF [37]	VGG16	RGB+Flow	22.4	-
CoViAR [54]	R50	Compressed	21.9	-
MultiScale TRN [58]	Inception	RGB	25.2	-
I3D [5]	Inception-I3D	RGB	32.9	34.4
I3D [5] (from [51])	R101-I3D	RGB	35.5	37.2
I3D-NL [51] (from [52])	R50-I3D-NL	RGB	33.5	-
I3D-NL [51]	R101-I3D-NL	RGB	37.5	39.5
STRG [52]	R50-I3D-NL	RGB	37.5	-
STRG [52]	R101-I3D-NL	RGB	39.7	-
3D CNN	R50-I3D-NL	RGB	38.3	-
3D CNN ens.	R50-I3D-NL	RGB	39.5	-
STO	R50-I3D-NL	RGB	39.6	-
STO ens.	R50-I3D-NL	RGB	40.0	-
LFB Avg	R50-I3D-NL	RGB	38.4	-
LFB Max	R50-I3D-NL	RGB	38.6	-
LFB NL	R50-I3D-NL	RGB	**40.3**	-
3D CNN	R101-I3D-NL	RGB	40.3	40.8
3D CNN ens.	R101-I3D-NL	RGB	41.7	-
STO	R101-I3D-NL	RGB	41.0	-
STO ens.	R101-I3D-NL	RGB	42.3	-
LFB Avg	R101-I3D-NL	RGB	40.8	-
LFB Max	R101-I3D-NL	RGB	40.9	-
LFB NL	R101-I3D-NL	RGB	**42.5**	**43.4**

Table 4. **Action recognition accuracy on Charades.** (mAP in %)

6.1. Implementation Details

We use the RGB frames at 24 FPS provided by the dataset authors. We sample training and testing clips (32 frames) with a temporal stride of 4 following STRG [52], resulting in input clips spanning 125 frames (∼5.2 seconds). The LFB is sampled at 2 clips per second. We found a post-activation version of NL′ to work better on Charades, so we adopt it in the following experiments. Details and full results of both variants are in Supplementary Material. Other details are identical to the verb model for EPIC-Kitchens.

Training and Inference. We train 3D CNN models for 24k iterations with a learning rate of 0.02 and weight decay of 1.25e-5. Note that these hyperparameters are different from STRG [52], which uses longer schedule (50k iterations), smaller learning rate (0.0025), and larger weight decay (1e-4).[3] Table 3 compares the two settings, and we see that surprisingly our 2× *shorter* schedule works significantly better. With the new schedule, a simple NL model without proposals (STO) works as well as the full STRG method (37.5% mAP) [52]. We observe that the benefit of using the short-term operator becomes smaller when using a stronger baseline. In all following experiments we use our 24k schedule as default, and use a 2-stage training approach similar to STRG [52] for training LFB models; see Supplementary Material for details. At test time, we sample 10 clips per video, and combine the predictions using max pooling following prior work [51, 52]. We use (left, center, right) 3-crop testing following Wang *et al*. [51].

[3]The original STRG [52] uses a batch size of 8. For clear comparison, we use the same batch size as ours (16), but adjust the learning rate and schedule according to 'Linear Scaling Rule' [13]. We verified that the accuracy matches that of the original 4-GPU training.

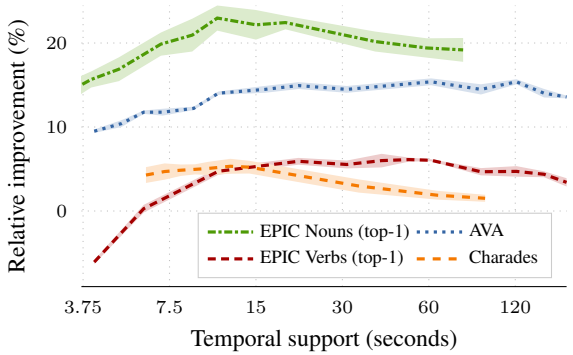


Figure 6. **Relative improvement** of LFB models with different window sizes over vanilla 3D CNN.

6.2. Quantitative Evaluation

For Charades, we experiment with both ResNet-50-I3D-NL and ResNet-101-I3D-NL [5, 16, 51] backbones for a consistent comparison to prior work. Table 4 shows that LFB models again consistently outperform all models without LFB, including prior state-of-the-art on both validation and test sets. The improvement on Charades is not as large as other datasets, in part due to the coarser prediction task (video-level).

7. Discussion

Fig. 6 shows the relative gain of using LFB of different window sizes.[4] We see that different datasets exhibit different characteristics. The *movie* dataset, AVA, benefits from very long context lasting 2+ minutes. To recognize *cooking* activities (EPIC-Kitchens), context spanning from 15 to 60 seconds is useful. Charades videos are much shorter (∼30 seconds), but still, extending the temporal support to 10+ seconds is beneficial. We conjecture that more challenging datasets in the future may benefit even more.

In conclusion, we propose a **Long-Term Feature Bank** that provides long-term supportive information to video models. We show that enabling video models with access to long-term information, through an LFB, leads to a large performance gain and yields state-of-the-art results on challenging datasets like AVA, EPIC-Kitchens, and Charades.

[4]For each dataset, we use its best-performing FBO. Standard error is calculated based on 5 runs. The temporal support here considers the support of each clip used for computing L, so Charades's support starts at a higher value due to its larger 3D CNN clip size (∼5.2 seconds).

References

[1] T. Althoff, H. O. Song, and T. Darrell. Detection bank: an object detection based video representation for multimedia event recognition. In *International Conference on Multimedia*, 2012. 2

[2] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*, 2018.

[3] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 4, 6

[4] F. Baradel, N. Neverova, C. Wolf, J. Mille, and G. Mori. Object level visual reasoning in videos. In *ECCV*, 2018. 2, 7

[5] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 1, 2, 3, 5, 8

[6] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, et al. Scaling egocentric vision: The EPIC-kitchens dataset. In *ECCV*, 2018. 1, 7

[7] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 2

[8] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In *NIPS*, 2016. 2

[9] R. Girdhar, J. Carreira, C. Doersch, and A. Zisserman. A better baseline for AVA. *arXiv preprint arXiv:1807.10066*, 2018. 2, 5, 6, 7

[10] R. Girshick. Fast R-CNN. In *ICCV*, 2015. 2

[11] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He. Detectron, 2018.

[12] G. Gkioxari and J. Malik. Finding action tubes. In *CVPR*, 2015. 2

[13] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 8

[14] C. Gu, C. Sun, D. A. Ross, C. Vondrick, C. Pantofaru, Y. Li, S. Vijayanarasimhan, G. Toderici, S. Ricco, R. Sukthankar, et al. AVA: A video dataset of spatio-temporally localized atomic visual actions. In *CVPR*, 2018. 1, 2, 3, 4, 5

[15] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *ICCV*, 2017. 3, 5

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3, 8

[17] R. Hou, C. Chen, and M. Shah. Tube convolutional neural network (T-CNN) for action detection in videos. In *ICCV*, 2017. 2

[18] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 4

[19] J. Jiang, Y. Cao, L. Song, S. Z. Y. Li, Z. Xu, Q. Wu, C. Gan, C. Zhang, and G. Yu. Human centric spatio-temporal action localization. In *ActivityNet workshop, CVPR*, 2018. 5

[20] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid. Action tubelet detector for spatio-temporal action localization. In *ICCV*, 2017. 2

[21] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 2

[22] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*, 2017.

[23] Y. LeCun. Learning process in an asymmetric threshold network. In *Disordered systems and biological organization*. 1986. 4

[24] D. Li, Z. Qiu, Q. Dai, T. Yao, and T. Mei. Recurrent tubelet proposal and recognition networks for action detection. In *ECCV*, 2018. 2, 5

[25] F. Li, C. Gan, X. Liu, Y. Bian, X. Long, Y. Li, Z. Li, J. Zhou, and S. Wen. Temporal modeling approaches for large-scale youtube-8m video understanding. *arXiv preprint arXiv:1707.04555*, 2017. 1, 2, 5

[26] L.-J. Li, H. Su, L. Fei-Fei, and E. P. Xing. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *NIPS*, 2010. 2

[27] Z. Li, K. Gavrilyuk, E. Gavves, M. Jain, and C. G. Snoek. Videolstm convolves, attends and flows for action recognition. *Computer Vision and Image Understanding*, 2018. 2

[28] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 4

[29] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 4

[30] C.-Y. Ma, A. Kadav, I. Melvin, Z. Kira, G. AlRegib, and H. P. Graf. Attend and interact: Higher-order object interactions for video understanding. In *CVPR*, 2018. 2

[31] A. Miech, I. Laptev, and J. Sivic. Learnable pooling with context gating for video classification. *arXiv preprint arXiv:1706.06905*, 2017. 1, 2, 5

[32] X. Peng and C. Schmid. Multi-region two-stream r-cnn for action detection. In *ECCV*, 2016. 2

[33] Z. Qiu, T. Yao, and T. Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *ICCV*, 2017. 1, 2

[34] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 2, 4

[35] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015. 3, 4

[36] S. Saha, G. Singh, and F. Cuzzolin. AMTnet: Action-micro-tube regression by end-to-end trainable deep architecture. In *ICCV*, 2017. 2

[37] G. A. Sigurdsson, S. K. Divvala, A. Farhadi, and A. Gupta. Asynchronous temporal fields for action recognition. In *CVPR*, 2017. 8

[38] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV*, 2016. 1, 8

[39] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 2, 7, 8

[40] G. Singh, S. Saha, M. Sapienza, P. H. Torr, and F. Cuzzolin. Online real-time multiple spatiotemporal action localisation and prediction. In *ICCV*, 2017. 2

[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014. 4, 6

[42] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus. End-to-end memory networks. In *NIPS*, 2015. 2

[43] C. Sun, A. Shrivastava, C. Vondrick, K. Murphy, R. Sukthankar, and C. Schmid. Actor-centric relation network. In *ECCV*, 2018. 2, 5, 6

[44] L. Sun, K. Jia, K. Chen, D.-Y. Yeung, B. E. Shi, and S. Savarese. Lattice long short-term memory for human action recognition. In *ICCV*, 2017. 2

[45] Y. Tang, X. Zhang, J. Wang, S. Chen, L. Ma, and Y.-G. Jiang. Non-local netvlad encoding for video classification. *arXiv preprint arXiv:1810.00207*, 2018. 1, 2, 5

[46] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 1, 2, 3

[47] D. Tran, J. Ray, Z. Shou, S.-F. Chang, and M. Paluri. Convnet architecture search for spatiotemporal feature learning. *arXiv preprint arXiv:1708.05038*, 2017. 1, 2

[48] G. Varol, I. Laptev, and C. Schmid. Long-term temporal convolutions for action recognition. *PAMI*, 2018. 1, 2

[49] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, 2017. 6

[50] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016. 2, 7

[51] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *CVPR*, 2018. 1, 2, 3, 6, 8

[52] X. Wang and A. Gupta. Videos as space-time region graphs. In *ECCV*, 2018. 1, 2, 3, 4, 8

[53] P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Learning to track for spatio-temporal action localization. In *ICCV*, 2015. 2

[54] C.-Y. Wu, M. Zaheer, H. Hu, R. Manmatha, A. J. Smola, and P. Krähenbühl. Compressed video action recognition. In *CVPR*, 2018. 8

[55] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 4

[56] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking spatiotemporal feature learning for video understanding. In *ECCV*, 2018. 1, 2

[57] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015. 1, 2, 5

[58] B. Zhou, A. Andonian, A. Oliva, and A. Torralba. Temporal relational reasoning in videos. In *ECCV*, 2018. 2, 8