Coarse-Fine Networks for Temporal Activity Detection in Videos

Kumara Kahatapitiya and Michael S. Ryoo
Stony Brook University, Stony Brook, NY 11794, USA

\{kkahatapitiy,mryoo\}@cs.stonybrook.edu

Abstract

In this paper, we introduce Coarse-Fine Networks, a two-stream architecture which benefits from different abstractions of temporal resolution to learn better video representations for long-term motion. Traditional Video models process inputs at one (or few) fixed temporal resolution without any dynamic frame selection. However, we argue that, processing multiple temporal resolutions of the input and doing so dynamically by learning to estimate the importance of each frame can largely improve video representations, specially in the domain of temporal activity localization. To this end, we propose (1) ‘Grid Pool’, a learned temporal downsampling layer to extract coarse features, and, (2) ‘Multi-stage Fusion’, a spatio-temporal attention mechanism to fuse a fine-grained context with the coarse features. We show that our method can outperform the state-of-the-arts for action detection in public datasets including Charades with a significantly reduced compute and memory footprint.

1. Introduction

Learning to represent videos is important. It requires embedding both spatial and temporal information in a sequence of frames, often implemented with 3D convolutions. Learning to build good video representations is crucial for various vision tasks including action classification, video object segmentation, and complex human activity recognition as well as temporal localization of such activities.

One of the main challenges in video representation learning is in capturing long-term motion from a continuous video. In order for a convolutional neural network to abstract long-term motion information across many frames, a large number of (spatio-)temporal conv. layers (or such layers with large kernels) are necessary, requiring many parameters. This, combined with the difficulty in obtaining large-scale annotated videos and increased computation time, makes the learning of the video representation very challenging for non-atomic activities. This is even more challenging for temporal activity detection (i.e., localization), as the activities may very often temporally overlap. A mechanism to reliably and efficiently capture various motion in videos is necessary.

Use of frame striding or temporal pooling (i.e., lowering the frame rate) has been a successful strategy to cover a larger time interval without increasing the number of model parameters. Since such striding loses fine details of frame changes, it was often paired with another CNN tower taking an input with a higher frame rate, forming a two-stream (or multi-stream) CNN architecture as was done in SlowFast [7] and AssembleNet [29]. These models confirmed the benefits of frame striding as well as multi-stream architectures to combine representations with multiple temporal resolutions.

However, although using temporal striding (with a multi-stream multi-resolution architecture) allows the model to more easily process long-term motion, they are limited as it ignores ‘importance’ of each frame. Informativeness of each frame is different. It is often unnecessary and redundant to feed almost identical frames as an input to the model when there is no/little motion in video frames. On the other hand, if a human in the video is displaying a rapid motion, taking all such frames into consideration is desired. Uniform temporal striding or pooling is incapable of such dynamic frame selection.

In this paper, we propose (1) a new approach that allows a learnable dynamic selection of temporal frames within the model, as well as (2) a method to fuse such sampled (i.e., temporally ‘coarse’) representations with conventional, more temporally ‘fine’ representations. We introduce the Coarse-Fine Networks. A new component named temporal...
Coarse-Fine Networks explore how video architectures can benefit from different abstractions of temporal resolution and long-term temporal information. As shown in Fig. 1, we do this by processing the information at two different temporal resolutions: coarse and fine, in a two-stream architecture. The Coarse stream learns to (differentiably) select the most informative frame locations, essentially performing a learned temporal downsampling to abstract a lower temporal resolution. In contrast, the Fine stream processes the input at the original temporal resolution and provide a fine-grained context to the Coarse stream through a fusion mechanism. To abstract this context information, the Fine stream always looks at the full temporal duration of the input clip (which later gets pooled with Gaussians), whereas the Coarse stream can either look at a shorter clip or the entire clip depending on the inference interval.

In Coarse-Fine Networks, we address two key challenges: (i) how to abstract the information at a lower temporal resolution meaningfully, and, (ii) how to utilize the fine-grained context information effectively. First, to abstract coarse information, we propose Grid Pool (Sec. 3.1), a learnable temporal downsampling operation which adaptively samples the most informative frame locations with a differentiable process. Secondly, to effectively use the fine-grained context provided by the Fine stream, we introduce Multi-stage Fusion (Sec. 3.2), a set of lateral connections between the Coarse and Fine streams, which looks at multiple abstraction levels of fine-grained information.

### 3.1. Grid Pool

Our temporal Grid Pool operation learns the most informative frame locations from a given input clip, and samples the representations corresponding to the locations based on interpolation. In fact, it can be viewed as a learnable temporal downsampling layer with a small compute overhead, which can replace the conventional temporal pooling operations. However, in contrast to these pooling operations, our Grid Pool samples by interpolating on a non-uniform grid with learnable (and adaptive) grid locations as shown in Fig. 3. First, a lightweight head \(h\) projects the input feature \(X_C\) of temporal length \(T\) to a set of confidence values \(\{p_t\}_{t=1,\ldots,T}\), where \(\alpha < 1\) and \(\alpha T\) is an integer (e.g., \(\alpha = 1/4\) and \(T = 128\)). These confidence values represent how informative each temporal interval with a size of \(1/\alpha\) (e.g., 4 frames if \(\alpha = 1/4\)) is, and is modeled as a function...
The intuition here is to sample frames at a higher frame rate where the confidence (i.e., informativeness) is high and at a lower frame rate where it is low. In other words, the stride between the interpolated frame locations should be small where the confidence is high, and vice-versa. We normalize these confidence values \( p_i \) since we need the relative (not absolute) confidence to capture the relative importance of frames. To get a set of \( \alpha T \) grid locations based on confidence values, we consider the Cumulative Distribution Function \( \{ \text{cdf}(1 - p_i) \}_{i=1,\ldots,\alpha T} \), which is a non-uniform and monotonically-increasing function. The input of the Grid Pool layer \( X^C \) is sampled/interpolated based on these grid locations to get the output \( \tilde{X}^C \), while making it fully differentiable for backpropagation. This process can be represented as,

\[
\{ p_i \}_{i=1,\ldots,\alpha T} = h(X^C) .
\]

\[ p_i = T \cdot \text{cdf}(1 - p_i) = T \cdot \frac{\sum_{i=1}^{\alpha T}(1 - p_i)}{\sum_{i=1}^{\alpha T}(1 - p_i)} ,
\]

\[
\tilde{X}^C = I(X^C, \{ q_t \}_{t=1,\ldots,\alpha T}) ,
\]

where \( q_t \) represents the grid location \( t \), and \( I(\cdot) \) represent the temporal sampling function. Here, when a grid location is non-integer, the corresponding sampled frame is a temporal interpolation between the adjacent frames. We do not perform any spatial sampling in the Grid Pool layer.

**Grid Unpooling:** A temporal interpolation based on a non-uniform grid as such can affect the temporal structure of the propagated features. Before bifurcating the final output, the frame-wise predictions of the network should be re-aligned properly for activity detection tasks. To do this, we introduce a Grid Unpool operation, which is coupled with the grid locations learned by the Grid Pool layer. This does not have any learnable parameters and simply performs the inverse operation of the former. First, \( \alpha T \) re-sample grid locations are calculated based on the inverse mapping of the \( \text{cdf} \), based on which, the logits are re-sampled to acquire the original temporal structure. The idea is to re-sample with a low frame-rate in the regions where we used a high frame-rate in Grid Pool, and vice-versa. Any non-integer frame location is temporally interpolated similar to Eq. 2. Finally, these logits are uniformly upsampled through interpolation to fit the input temporal resolution. For classification tasks, the Grid Unpool operation may not be necessary as a global pooling of the logits is considered as the prediction.

### 3.2. Multi-stage Fusion

We introduce Multi-stage Fusion, a set of lateral connections between the two streams as shown in Fig. 4, to fuse the context from the Fine stream with the Coarse stream. We consider three main design choices here: (i) it should be capable of filtering out which fine-grained information should be passed down to the Coarse stream, (ii) it should have a calibration step to properly align the fine features with the coarse features based on their relative temporal locations, and (iii) it should be able to learn and benefit from multiple abstraction-levels of fine-grained context at each fuse-location in the Coarse stream. Our design tries to address these aspects.

**Filtering fine-grained information:** First, to decide which fine-grained context should be passed through to the fusion process, the fine feature \( X^F_{l_i} \) at abstraction-level \( l_i \) is multiplied with a self-attention mask. This mask is calculated by processing the fine feature through a lightweight head (\( g \)) consisting of point-wise convolutional layers followed by a sigmoid non-linearity.

\[
\tilde{X}^F_{l_i} = X^F_{l_i} \cdot g(X^F_{l_i})
\]

**Fine-to-Coarse correspondence:** The attention-weighted fine feature \( \tilde{X}^F_{l_i} \) still needs to be calibrated for the temporal location of each coarse feature. Since the Coarse and Fine streams does not necessarily process the same, properly aligned temporal length because of our non-uniform Grid Pooling, we need to explicitly compute the frame correspondence. To make this calibration, we use a set of temporal Gaussian distributions centered at each coarse frame location \( \{ \mu^C_j \}_{j=1,\ldots,\alpha T} \) which abstract a location-dependent weighted average of the fine feature. We use \( \alpha T \) such Coarse-centric Gaussians, each having a temporal length of \( T' \) and a standard deviation \( \sigma \) which is a fraction of this length. We found that considering the center and scale of these Gaussians to be hyperparameters rather than making them learnable, gives a better performance, possibly due to relatively simpler training. This calibration step can be viewed as,
The feature $\hat{X}_i^F$ still corresponds to a single abstraction-level $l_i$ of fine features, where we have Multi-stage Fusion connections in multiple abstraction-levels, i.e., depths of the network. Therefore, we allow each fusion connection to look at the features from all abstraction levels by concatenating them channel-wise (after adjusting the spatial resolution through max pooling), and performing point-wise (i.e. $1 \times 1 \times 1$) convolutions to get the final scale $(A_i)$ and shift $(B_i)$ features at each fusion location. This can be represented as,

$$\hat{X}_i^F = \bigoplus_{i=1}^n \hat{X}_i^f,$$

$$A_{i_l} = f_{i_l}^A(\hat{X}_i^F), B_{i_l} = f_{i_l}^B(\hat{X}_i^F),$$

$$\hat{X}_{i_l}^C = A_{i_l} \hat{X}_{i_l}^C + B_{i_l}.$$

where $\bigoplus$ is the channel-wise concatenation of features from $n$ abstraction-levels and, $f_{i_l}^A$ and $f_{i_l}^B$ represent projection heads to calculate the scale and shift features at each fusion location $l_i$, respectively. This design enables Multi-stage Fusion to process multiple abstraction-levels of fine-grained context through filtering and temporal calibration.

### 3.3. Model Details

The network architecture used as the backbone in Coarse-Fine Networks is adopted from X3D [6], which follow a ResNet [13] structure, but designed for efficiency in video models. Both Coarse and Fine streams are initialized with separate sets of parameters, but have the same number of layers and filters as shown in Table 1, except for the addition of Grid Pool layer and Grid Unpool operation in the Coarse stream. The Fine stream process the entire temporal length of the input $T'$ to provide a fine-grained context, whereas the Coarse stream can look at a segmented clip of length $T$, for which the frame-wise predictions are required. Here, $\alpha < 1$ and $\alpha T$ is an integer. The kernel shapes follow the standard notation $\{T' \times S', C\}$.

| Stage | Filters | Output size $T \times S^2$ |
|-------|---------|---------------------------|
| input | stride 10, 1 | $T \times 22^2$ |
| conv1 | $1 \times 3^2, 3 \times 1, 24$ | $T \times 112^2$ |
| res2 | $1 \times 1^2, 54$ | $T \times 56^2$ |
| grid pool | stride $1/\alpha, 1^2$ | $\alpha T \times 56^2$ |
| res3 | $1 \times 1^2, 108$ | $\alpha T \times 28^2$ |
| res4 | $1 \times 1^2, 216$ | $\alpha T \times 14^2$ |
| res5 | $1 \times 1^2, 432$ | $\alpha T \times 7^2$ |
| conv5 | $1 \times 1^2, 432$ | $\alpha T \times 7^2$ |
| pool5 | None $\times 2^2$ | $\alpha T \times 7^2$ |
| fc1 | $1 \times 1^2, 2048$ | $\alpha T \times 1^2$ |
| fc2 | $1 \times 1^2, \text{classes}$ | $\alpha T \times 1^2$ |

Table 1. Coarse-Fine Network Architecture is adopted from X3D [6], more specifically from the version X3D-M. Both streams have the same design and parameters, except for the addition of Grid Pool layer and Grid Unpool operation in the Coarse stream. The Fine stream process the entire temporal length of the input $T'$ to provide a fine-grained context, whereas the Coarse stream can look at a segmented clip of length $T$, for which the frame-wise predictions are required. Here, $\alpha < 1$ and $\alpha T$ is an integer. The kernel shapes follow the standard notation $\{T' \times S', C\}$.
the compute, but at the same time, having good enough fea-
tures to learn the grid locations. Thus, we place the Grid
Pool layer after the first residual block res2. We find that
downsampling by a factor of 4 works well in practice, to
have a good compute/performance trade-off (Table 3e). To
calculate the confidence values \((p)\) in the Grid Pool layer, we
use a lightweight head \((h)\) of 3 strided convolutions with a
total temporal stride of 4 and a spatial stride of 8, followed
by spatial average pooling and a sigmoid non-linearity. The
Grid Unpool operation has no learnable parameters. It is cou-
pled with the grid locations predicted by the Grid Pool layer
to perform the inverse operation of the former to recover the
original temporal structure at the logits level.

We try to follow a lightweight design in Multi-stage Fu-
non as well. The self-attention mask \(\hat{X}^F_i\) is calculated
through a head \(g\) of 2 point-wise (i.e. \(1 \times 1 \times 1\)) conv-
olutions followed by a sigmoid non-linearity. The Coarse-
centric Gaussians \((G^C)\) have no learnable parameters, and
the peak magnitude of each mask is normalized to 1. The
standard deviation \(\sigma\) is set to be \(T/8\), empirically. The two
heads \(f^A_i\), and \(f^A_{i+1}\) at each fusion location which project
the concatenated multi-stage features \((\hat{X}^F)\) to scale \((A_i)\)
and shift \((B_i)\) features contain a single point-wise convo-
lution each. The scale features go through an additional
sigmoid non-linearity. We further discuss the complexity
(compute and parameters) of these operations in our abla-
tions (subsection 4.2).

We plan to release the code as well as the trained models
together with the final version of the paper.

4. Experiments

We evaluate Coarse-Fine Networks on two large-scale
benchmarks for activity detection: Charades \([33]\) and Multi-
THUMOS \([46]\). Note that we focus on the temporal detection
(i.e., localization) task of generating multi-label activity an-
notations at every time step, which is more challenging than
video classification. Activities may temporally overlap (e.g.,
sitting and eating), and the model must be trained to annotate
all of them at each time step.

4.1. Charades

Dataset: Charades \([33]\) is a large scale dataset consisting of
\(~9.8k\) continuous videos with frame-wise annotations of \(~157\)
common household activities. The dataset is split as \(~7.9k\)
training and \(~1.8k\) validation videos. Each video contains an
average of \(6.8\) activity instances, often with multiple activity
classes per frame, and has longer clips averaging a duration of
\(~30\) seconds. Such a long duration makes it a suitable
dataset to test Coarse-Fine Networks.

Training: We initialize both Coarse and Fine streams of
our network with the X3D backbone pretrained on Kinetics-
400 \([18]\). For the actual training of the Coarse-Fine network
as well as baselines, we follow a two-stage training pro-
cess: first, training the two streams separately, followed by
finetuning the combined streams.

In the first stage, the Coarse stream considers an input of
\(64\) frames sampled at a stride of 10, whereas the Fine stream
considers \(16\) frames sampled at the same stride. This allows
both streams to process same-sized features after Grid Pool
layer. We use \(\alpha = 1/4\) in our experiments. Each stream is trained for \(100\) epochs with a batch size of \(16\) at a learning
rate of \(0.02\) at the start, which is decreased by a factor of \(10\)
at \(60\) and \(80\) epochs.

In the second stage, the two streams are trained together
as Coarse-Fine Networks, with Multi-stage Fusion param-
eters initialized from scratch. We train this for another \(100\)
epochs with the same schedule and batch size, but use \(10\times
increased learning rate for the newly-initialized parameters
of the fusion layers. Here, the Fine stream process the en-
tire duration of the input, which is capped at \(128\) frames
(sampled at a stride of 10) for Charades \([33]\). During both
stages, each input is randomly sampled in \([256,320]\) pixels,
spatially cropped to \(224 \times 224\) and applied a random hori-
tonal flip. We use a dropout rate of 0.5 before the logits
layer. The logits go through a sigmoid to make multi-label
predictions for each frame. We use an average of classifi-
cation and localization loss for training, similar to previous
methods \([25,27]\).

Inference: At inference, we make predictions for every
frame. Even though our input is sampled at a stride of
\(10\), we consider the labels for all frames (at a stride of \(1\))
and interpolate the logits to fit the original temporal length.
In other words, we evaluate our models so that the predic-
tions are more fine-grained at original temporal resolution.
However, all state-of-the-art methods in Table 2 report the
performance for \(25\) equally-sampled frames per each input,
which is the original Charades localization evaluation \([33]\)
setting. Therefore, to make a fair comparison, we evaluate
our models in the same setting, only making predictions for
\(25\) equally-sampled frames per input. The evaluation script
from the Charades challenge scales the Average Precision for
each class with a corresponding class weights. At inference,
the inputs are center-cropped to \(224 \times 224\). We report the
performance as mean Average Precision \(\text{(mAP)}\) and measure
the compute requirement to process an input clip of \(128 \times 10\)
frames, for which our network processes only \(128\) frames
due to input striding. The compute is reported as GFLOPs
\((\text{floating-point operations} \times 10^9)\) and the number of learn-
able parameters in millions \(\text{(M)}\), i.e., \(\times 10^6\). We do not take
advantage of any multi-crop inference.

Results: We compare the performance of the Coarse-Fine
Networks with the state-of-the-art methods on Charades \([33]\)
localization task (i.e., temporal activity detection) in Table 2.
For this evaluation, we use the standard test setting (i.e., the
official ‘Charades_v1 localize’ evaluation) where we make
predictions for equally-sampled \(25\) frames in the validation
set. This is the same procedure followed in previous work.
# Comparison with the state-of-the-art methods for activity detection on Charades [33]

We report the performance (mAP), compute requirement to process a clip of $128 \times 10$ frames (GFLOPs), and the number of parameters (M). These results correspond to the original Charades localization evaluation settings. Coarse-Fine Networks significantly outperform the previous state-of-the-art with +1.4% mAP relative improvement, while reducing the compute requirement by more than one order of magnitude. It is worth noting that we do not use additional input modalities, i.e., optical-flow or object detections as any of the previous methods. The source of [23] is not available to us to calculate its exact complexity values.

| model                  | mAP (%) | GFLOPs | Param (M) |
|------------------------|---------|--------|-----------|
| I3D (Inception) [2]    | 15.63   | 2223.03| 12.45     |
| Two-stream I3D [2]     | 17.22   | 4446.10| 24.90     |
| 3D ResNet-50 [13, 38]  | 18.60   | 3187.63| 46.52     |
| X3D-L [6]              | 18.87   | 37.96  | 3.29      |
| X3D-L [6]              | 20.03   | 147.04 | 5.78      |
| I3D + super-events [25]| 19.41   | 4446.15| 26.18     |
| I3D + TGM [27]         | 21.50   | 4446.66| 27.00     |
| I3D + super-events + TGM [27]| 22.30 | 4446.75| 28.28     |
| I3D + STGCN [10]       | 19.09   | 4450.94| 29.18     |
| I3D + biGRU + VS-ST-MPNN [23]| 23.70 | 2223.03| 12.45     |
| X3D-L [6] + TGM [27]   | 20.01   | 38.26  | 4.35      |
| SlowFast$_{det}$ (with X3D) | 22.31 | 54.51  | 7.41      |
| Fine-Fine (ours)       | 24.43   | 94.80  | 7.80      |
| Coarse-Fine (ours)     | 25.10   | 73.37  | 7.82      |

We report the performance (mAP), compute requirement to process a clip of $128 \times 10$ frames (GFLOPs) and the number of parameters (M).

We are able to confirm that our Coarse-Fine Network performs better than all previous methods, establishing the new state-of-the-art of 25.10% mAP on Charades localization. The Coarse-Fine Network, which only uses RGB, is not only superior to the previous RGB models but also is better than the methods using additional inputs modalities (i.e. optical-flow and object detection). It shows a relative improvement of +1.4% mAP compared to the previous best performing method [23], which benefits from additional training data for its object detector training and additional input modality (objects).

We also note that the Coarse-Fine Network is extremely computationally efficient. Compare to the previous models, it often requires only $\sim 1/75$ of computations (e.g., 73 vs. 4446 GFLOPS). Further, this computation is without considering the overhead for optical flow computation or object detection in prior works. The significant computation efficiency of the Coarse-Fine Networks is due to the better utilization of the RGB features, which discards the need for processing additional modalities, as well as an efficient usage of X3D modules with our temporal Grid Pooling and Multi-stage Fusion, which we confirm further with our ablations in the next section.

We further report a version of our method: Fine-Fine Networks, in which the Grid Pool layer is removed from the Coarse stream, to highlight the importance of the Coarse features. Fine-Fine Networks still benefit from our Multi-stage Fusion. The Grid Pool operation dynamically sample important frames to generating a coarse temporal resolution, which gives the Coarse-Fine Networks an relative performance gain of +0.67% mAP and 23% computation reduction. We also evaluated the baseline extension of X3D as a two-stream network (with different temporal resolutions) in a form similar to [7], which we name SlowFast$_{det}$. This does not have our Grid Pool layer or the Multi-stage Fusion mechanism. The result shows the benefits of the components, giving a relative improvement of 2.79%mAP. A larger version of X3D (i.e., X3D-L) shows that the performance improvement of Coarse-Fine compared to X3D is not merely due to the increased compute.

It is important to also note that all previous methods work on pre-extracted features from a frozen backbone, essentially making them late-modeling techniques, either using graph-based methods [10, 23] or abstracting long-term temporal information [25, 27]. In contrast, our method allows end-to-end training of feature fusions at intermediate locations of the network, enabling it to learn better representations using only RGB information.

Fig. 2 further highlights the benefit of Coarse-Fine Networks compared to previous state-of-the-arts. We show the performance/complexity trade here, with the x-axis (complexity in GFLOPs) in log-scale. Our method shows comparable performance with the previous best performing method which outperforms all previous state-of-the art methods, while being extremely efficient in design.

## Ablations

Here, we discuss multiple ablation experiments validating our design decisions, specifically on our Grid Pool layer and Multi-stage Fusion. We use the Charades dataset (with the localization setting as above).

In our ablation experiments, we take advantage of more robust evaluation metric to compare our approach and the baselines. We make the model to generate a multi-class activity annotation at every time step, and compare it with the ground truth to measure the mAP. This is very similar to the official ‘Charades_v1_localize setting’ used above, except that (i) it is evaluated for $\times 25$ more frames for the completeness and that (ii) we measure mAP per activity class and average them.

### Fusion location:

First, we explore which locations in our two stream architecture would be ideal to implement the fusion connections. We consider the late fusion as a baseline, in which, the only fusion happens just before the logits layer. This is similar to the previous methods in [25, 27]. Next, We extend this fusion to multiple intermediate levels, specifically, after each res block, in which we fuse the two streams at only equivalent abstraction levels, i.e., at the same depth. This is similar to the fusion in SlowFast [7]. Finally, we consider multiple abstraction-levels of Fine features for the fusion, which gives our Multi-stage Fusion. The results of this ablation is given in Table 3a. Note that here we
evaluate our fusion in a Fine-Fine Network to decouple the effect of Grid Pool from our fusion. Multi-stage Fusion shows an +0.81% mAP improvement compared to only using a late fusion. The improvement of considering multiple abstraction-levels is marginal, at +0.15% mAP, suggesting that features at the same abstraction level can provide the most complementary information.

**Fusion dimensions**: We experiment on the significance of different dimensions in the fusion features. Either having only channel dimension (C) similar to [25], channel-spatial (CHW) dimensions or all channel-temporal-spatial (CHWT) dimensions similar to [7] is considered here. The results of this experiment are reported in Table 3b. Note that the dimensions which are not available in any of the above scenarios are average pooled before the fusion. We see that having all CHWT dimensions in the fusion feature has a large improvement compared to the baseline, specifically +4.54% mAP. Introduction of the temporal dimension (T) shows the most improvement, which is +2.79% mAP. This is in fact mainly due to the temporal Gaussians in our Fusion that calibrate the features based on the location, without which, we can not see such improvement (i.e., +0.61% mAP over a single stream, when naively selecting corresponding temporal locations in the two streams for fusion w/o Gaussians).

**Fusion mask**: Here, we evaluate how important it is to filter the Fine features at the input of fusion, results of which, is shown in Table 3c. In the Multi-stage Fusion setting, having a self-attention mask to adaptively weight each Feature point gives an improvement of +1.23% mAP compared to feeding the Fine feature directly.

**Pooling type**: Next, we explore the performance gain caused by the proposed (temporal) Grid Pool layer. We compare against conventional temporal pooling operations such as max pooling, average pooling and even simple temporal striding. Here, we report the numbers for a Coarse-only network to decouple the effect of Grid Pool from that of Multi-stage Fusion. In these experiments, we set \( \alpha = 1/4 \), which essentially means a \( 4 \times 4 \) temporal downsampling, and perform the downsampling after the \( \text{res}_2 \) block. Max pooling, average pooling use a similar setting of kernel size of 4 and a stride of 4. Grid pooling gives a consistent improvement over other methods, specifically +1.91% mAP and +1.48% mAP over commonly used max pooling and average pooling, respectively. We also note that a simple temporal striding can outperform max pooling and average pooling by +1.28% mAP and +0.85% mAP, respectively. We suspect that the inferior performance of max/average pooling, the stride is 40). In such a large window, pooling tends to blur most of the temporal information.

**Grid Pool configuration**: We try different configurations of Grid Pooling to evaluate its performance and compute requirement. Similar to above, we use the Coarse-only network. We consider an input of temporal length \( T = 128 \) at a stride of 10 or \( T = 256 \) at a stride of 5, to cover entire duration of Charades videos [33]. We try temporally downsampling each of these with \( \alpha = 1/4 \times 4 \) downsampling) or \( \alpha = 1/8 \times 8 \) downsampling). The performance of these configurations is given in Table 3e. \( 8 \times 8 \) downsampling shows a significantly lower performance indicating.
that it looses too much information with such an aggressive stride, i.e., more frames are important and need to be sampled by the Grid Pool layer. Moreover, increasing the number of input frames does not necessarily improve the performance (only +0.02% mAP) with $\alpha = 1/4$. Among these configurations, $T = 128$ with $\alpha = 1/4$ shows the best performance/compute trade-off.

**Grid Pool and Multi-stage Fusion combined:** Finally, we evaluate the combined performance of Grid Pool and Multi-stage Fusion. To do this, we consider a two-stream baseline without these components, which we call SlowFastdet. This performs $\times 4$ temporal downsampling in the Coarse stream based on striding, and use direct frame correspondences between Fine and Coarse streams for fusion, similar to SlowFast [7] while still using X3D modules like ours. The results of this study is given in Table 3f. When compared with this baseline, introduction of either Grid Pool or Multi-stage Fusion provides consistent improvements of +0.21% mAP and +2.18% mAP respectively. Our Coarse-Fine Networks outperform this baseline by a margin of +2.63% mAP.

**Trade-off with X3D:** Coarse-Fine network is designed to use a similar amount of computation as a two-stream version of X3D-M. Another way to use the extra compute is by increasing the number of layers. To understand if the increased compute is meaningful, we test X3D-L, a larger version of X3D (Table 2). X3D-L shows 20.03% mAP with a compute of 147.04 GFLOPS. Coarse-Fine Networks outperform this in both accuracy and speed with 25.10% mAP at 73.37 GFLOPS.

**4.3. MultiTHUMOS**

**Dataset:** MultiTHUMOS [46] is an extension of the THUMOS [16] dataset with the untrimmed videos densely annotated for 65 different action classes. It provides frame-level action annotations for 30 hours of video across 413 videos, split as 200 for training and 213 for validation. On average, it contains 1.5 labels per frame and 10.5 action classes per video. It contains significantly smaller number of videos compared to Charades [33] and each video has a larger temporal length, which make the training difficult. We created a segmented version of this data, where each clip is limited to a maximum of 1280 frames, which gives a similar evaluation setting to Charades. For the computational efficiency, we use the temporal striding of 10.

**Training:** We follow a training process similar to what we did for Charades: We follow two stages of training with our Coarse and Fine streams pretrained on Kinetics-400 [18], i.e., separately and combined. We use the same hyperparameter settings and training schedules as in Charades (refer subsection 4.1). We use a dropout rate of 0.5 before the logits layer. The logits go through a sigmoid to make multi-label predictions for each frame. We use an average of classification and localization loss for training.

| MultiTHUMOS mAP (%) | X3D | X3D + TGM | Coarse-Fine + TGM | I3D | I3D + super-events | I3D + super-events + TGM | I3D + TGM | rgb | flow |
|---------------------|-----|-----------|------------------|-----|--------------------|-------------------------|-----------|-----|------|
|                     | 02% | 03%       | +2%              | 10% | 14%                | +4%                     | 4%        | 0   | 4   |

Figure 5. Performance/complexity trade-off of state-of-the-art methods for activity detection on MultiTHUMOS [46]. Our Coarse-Fine Networks /w TGM show a comparable performance with the state-of-the-arts with ~75x speed, without using additional input modalities.

**Inference:** At inference, we make predictions for every frame. Even though our input is sampled at a stride of 10, we consider the labels for all frames (at a stride of 1) and interpolate the logits to fit the original temporal length. Each input is center-cropped to 224×224. We report the performance (mAP), compute requirement to process an input clip of 1024×10 frames as TFLOPs ($\times 10^9$) and the number of learnable parameters(M). The length of 1024×10 frames is only considered as a reference for reporting the complexity values, and there are longer clips in the dataset with up to $\times 5$ frames. We process these frames fully convolutionally.

**Results:** We show the performance (mAP) of Coarse-Fine Networks on MultiTHUMOS [46] activity detection with the corresponding compute requirement (TFLOPs, i.e., $\times 10^{12}$) in Fig. 5. We observe an improvement of +4.63% from X3D [6] to Coarse-Fine. While our Coarse-Fine network is almost 75 times faster than the previous model (0.49 TFLOPs (Coarse-Fine) vs. 35.57 TFLOPs (I3D + TGM)), it still achieves comparable performance to the previous state-of-the-art. Models using the X3D backbone including ours lose motion details due to the aggressive 1/10 striding compared to I3D [2] that doesn’t do striding, making them less effective when combined with other temporal modeling methods (e.g., TGM [27]). Still, our Coarse-Fine Networks were able to overcome such limitation and perform comparably. Coarse-Fine /w TGM shows a further improvement of +2.21% mAP.

**5. Conclusion**

We presented Coarse-Fine Networks, which is a two-stream architecture to combine temporally Coarse representations with Fine representations. We introduced the approach of temporal Grid Pooling that learns to differentiably select informative frames while discarding the other, to obtain Coarse representations. We also introduced the Multi-stage Fusion to best combine the Coarse stream with the Fine stream. We confirmed that the Coarse-Fine networks obtain the best known performance on Charades localization, while spending much less computation time.
A. Appendix

A.1. Pretraining details

We initialize our models from scratch and pretrain for classification in Kinetics-400 [18] prior to finetuning for activity detection. Kinetics-400 is commonly used as a pretraining dataset for video models. It contained 240k training and 20k validation videos at release, but due to availability, our version contains ∼220k training and ∼17k validation videos. Each video contains a single action out of 400 human action categories.

Similar to [6], we do not pretrain on ImageNet [4] prior to Kinetics-400 pretraining. We follow the recently-proposed multigrid training [41] recipe for efficient training of our baseline. We train for 200k iterations with a base batch size of 128 on 4 Titan RTX GPUs. The learning rate is initialized at 0.2 and decreased by a factor of 10 at 80k, 130k and 170k iterations. We fix the batch size considered for batch normalization to be 8 in-spite of the varying batch size in multigrid training, following an observation in [11]. Each clip is sampled at a stride of 10 frames at the base multigrid configuration. During training, each input is randomly sampled in [256, 320] pixels, spatially cropped to 224×224 and applied a random horizontal flip similar to [7, 35, 40]. We use a dropout rate of 0.5 before the last fully-connected layer.

In our pretraining setting, the baseline model achieves 62.62% Top-1 accuracy (with 3-view testing) on Kinetics-400. This is with a partial training dataset and a ×4 shorter schedule compared to [6]. We also tried directly using X3D checkpoint (from Facebook) pretrained in the same setting, since both our Coarse and Fine network are composed of the X3D modules with the same number of modules/layers. The difference in this case is that the checkpoint is trained with 240k Kinetics-400 training videos and a longer schedule (without multigrid training [41]), which gives 71.48% Top-1 accuracy on Kinetics-400.

A.2. Self-supervision for Grid Pool

We further explored multiple self-supervised losses to constrain the learned grid locations similar to [9]. Since the input itself can provide cues (i.e., motion or static) on important frame locations, such self-supervision can potentially benefit learning grid locations. In other words, learning grid locations does not necessarily need to rely on supervised losses such as classification or localization.

A flow-based loss was used to minimize the flow within a grid cell, while maximizing the flow in-between cells, forcing the cell boundaries to lie in the regions of maximum flow change. A reconstruction loss was used to minimize the L1 distance between the input and an upsampled output of the Grid Pool layer to force a faithful downsampling operation. Our overall loss function can be given as,

$$\mathcal{L} = 0.5 \mathcal{L}_{\text{class}} + 0.5 \mathcal{L}_{\text{loc}} + 0.1 \mathcal{L}_{\text{aux}},$$

where $\mathcal{L}_{\text{class}}$ and $\mathcal{L}_{\text{loc}}$ represent classification and localization losses respectively. $\mathcal{L}_{\text{aux}}$ is either the reconstruction loss $\mathcal{L}_{\text{rec}}$ or flow-based loss $\mathcal{L}_{\text{flow}}$. The results of this experiment is shown in Table A.1. None of these auxiliary losses improved the performance compared to using only the classification/localization supervision. The reconstruction loss may not be ideal as it can force the sampled frame in each grid closer to an average of all frames within the grid. The flow-based loss is more intuitive in comparison, as the motion information can prevent the sampling of redundant frames. However, since we perform aggressive striding at the input of X3D (usually a stride of 10), optical-flow computation can be noisy. Moreover, the online flow computation method that we use (Representation Flow [26]) can be computationally heavy to accommodate more iterations to converge, which adds to the noise in the computed flow. These limitations may have affected the performance in the self-supervised setting.

References

[1] Fabien Baradel, Natalia Neverova, Christian Wolf, Julien Mille, and Greg Mori. Object level visual reasoning in videos. In Proceedings of the European Conference on Computer Vision (ECCV), pages 105–121, 2018. 2
[2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6299–6308, 2017. 2, 4, 6, 8
[3] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In Proceedings of the IEEE international conference on computer vision, pages 764–773, 2017. 2
[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pages 248–255. Ieee, 2009. 9
[5] Victor Escorcia, Fabian Caba Heilbron, Juan Carlos Niebles, and Bernard Ghanem. Daps: Deep action proposals for action understanding. In European Conference on Computer Vision, pages 768–784. Springer, 2016. 2
[6] Christoph Feichtenhofer. X3D: Expanding architectures for efficient video recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 203–213, 2020. 2, 4, 6, 8, 9
[7] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In Proceedings of
Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1933–1941, 2016.

Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Learning fixed grid: Learning geometric image representation with a deformable grid. arXiv preprint arXiv:2008.09269, 2020.

Pallabi Ghosh, Yi Yao, Larry Davis, and Ajay Divakaran. Stacked spatio-temporal graph convolutional networks for action segmentation. In The IEEE Winter Conference on Applications of Computer Vision, pages 576–585, 2020.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677, 2017.

Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Learning spatio-temporal features with 3d residual networks for action recognition. In Proceedings of the IEEE International Conference on Computer Vision Workshops, pages 3154–3160, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2017–2025, 2015.

Jingwei Ji, Ranjay Krishna, Li Fei-Fei, and Juan Carlos Niebles. Action genome: Actions as compositions of spatio-temporal scene graphs. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10236–10247, 2020.

Yu-Gang Jiang, Jingen Liu, A Roshan Zamir, George Toderici, Ivan Laptev, Mubarak Shah, and Rahul Sukthankar. Thumos challenge: Action recognition with a large number of classes, 2014.

Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pages 1725–1732, 2014.

Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. arXiv preprint arXiv:1705.06950, 2017.

Bruno Korbar, Du Tran, and Lorenzo Torresani. Scsampler: Sampling salient clips from video for efficient action recognition. In Proceedings of the IEEE International Conference on Computer Vision, pages 6232–6242, 2019.

Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 156–165, 2017.

Lanlan Liu and Jia Deng. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. arXiv preprint arXiv:1701.00299, 2017.

Chih-Yao Ma, Asim Kadav, Iain Melvin, Zsolt Kira, Ghassan AlRegib, and Hans Peter Graf. Attend and interact: Higher-order object interactions for video understanding. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6790–6800, 2018.

Effrosyni Mavroudi, Benjamin Béjar Haro, and René Vidal. Representation learning on visual-symbolic graphs for video understanding. In European Conference on Computer Vision, pages 71–90. Springer, 2020.

Yue Meng, Chung-Ching Lin, Rameswar Panda, Prasanna Sattigeri, Leonid Karlinsky, Aude Oliva, Kate Saenko, and Rogerio Feris. Ar-net: Adaptive frame resolution for efficient action recognition. arXiv preprint arXiv:2007.15796, 2020.

AJ Piergiovanni and Michael S Ryoo. Learning latent super-events to detect multiple activities in videos. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5304–5313, 2018.

AJ Piergiovanni and Michael S Ryoo. Representation flow for action recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 9945–9953, 2019.

AJ Piergiovanni and Michael S. Ryoo. Temporal gaussian mixture layer for videos. In International Conference on Machine learning, pages 5152–5161, 2019.

Adria Recasens, Petr Kellnhofer, Simon Stent, Wojciech Matusik, and Antonio Torralba. Learning to zoom: a saliency-based sampling layer for neural networks. In Proceedings of the European Conference on Computer Vision (ECCV), pages 51–66, 2018.

Michael Ryoo, AJ Piergiovanni, Mingxing Tan, and Anelia Angelova. AssembleNet: Searching for multi-stream neural connectivity in video architectures. In International Conference on Learning Representations (ICLR), 2020.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. IEEE Transactions on Neural Networks, 20(1):61–80, 2008.

Zheng Shou, Jonathan Chan, Alireza Zareian, Kazuyuki Miyazawa, and Shih-Fu Chang. Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 5734–5743, 2017.

Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1049–1058, 2016.

Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In European Conference on Computer Vision, pages 510–526. Springer, 2016.
[34] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014. 2

[35] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 9

[36] Du Tran, Lubomir D Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. C3d: generic features for video analysis. *CoRR, abs/1412.0767*, 2(7):8, 2014. 2

[37] Du Tran, Jamie Ray, Zheng Shou, Shih-Fu Chang, and Manohar Paluri. Convnet architecture search for spatiotemporal feature learning. *arXiv preprint arXiv:1708.05038*, 2017. 2

[38] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018. 6

[39] Gül Varol, Ivan Laptev, and Cordelia Schmid. Long-term temporal convolutions for action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1510–1517, 2017. 2

[40] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 9

[41] Chao-Yuan Wu, Ross Girshick, Kaiming He, Christoph Feichtenhofer, and Philipp Krahenbühl. A multigrid method for efficiently training video models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 153–162, 2020. 9

[42] Zuxuan Wu, Caiming Xiong, Yu-Gang Jiang, and Larry S Davis. Liteeval: A coarse-to-fine framework for resource efficient video recognition. In *Advances in Neural Information Processing Systems*, pages 7780–7789, 2019. 2

[43] Zuxuan Wu, Caiming Xiong, Chih-Yao Ma, Richard Socher, and Larry S Davis. Adaframe: Adaptive frame selection for fast video recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1278–1287, 2019. 2

[44] Zhang Xinyi and Lihui Chen. Capsule graph neural network. In *International conference on learning representations*, 2018. 2

[45] Huijuan Xu, Abir Das, and Kate Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. In *Proceedings of the IEEE international conference on computer vision*, pages 5783–5792, 2017. 2

[46] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *International Journal of Computer Vision*, 126(2-4):375–389, 2018. 2, 5, 8

[47] Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2678–2687, 2016. 2

[48] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015. 2

[49] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaou Tang, and Dahua Lin. Temporal action detection with structured segment networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2914–2923, 2017. 2

[50] Luowei Zhou, Yannis Kalantidis, Xinlei Chen, Jason J Corso, and Marcus Rohrbach. Grounded video description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6578–6587, 2019. 2

[51] Mohammadreza Zolfaghari, Kamaljeet Singh, and Thomas Brox. Eco: Efficient convolutional network for online video understanding. In *Proceedings of the European conference on computer vision (ECCV)*, pages 695–712, 2018. 2