Efficient Video Classification Using Fewer Frames

Shweta Bhardwaj*

shweta@cse.iitm.ac.in

Mukundhan Srinivasan

NVIDIA Bangalore

msrinivasan@nvidia.com

Mitesh M. Khapra*

miteshk@cse.iitm.ac.in

Abstract

Recently, there has been a lot of interest in building compact models for video classification which have a small memory footprint (< 1 GB) [16]. While these models are compact, they typically operate by repeated application of a small weight matrix to all the frames in a video. For example, recurrent neural network based methods compute a hidden state for every frame of the video using a recurrent weight matrix. Similarly, cluster-and-aggregate based methods such as NetVLAD have a learnable clustering matrix which is used to assign soft-clusters to every frame in the video. Since these models look at every frame in the video, the number of floating point operations (FLOPs) is still large even though the memory footprint is small. In this work, we focus on building compute-efficient video classification models which process fewer frames and hence have less number of FLOPs. Similar to memory efficient models, we use the idea of distillation albeit in a different setting. Specifically, in our case, a compute-heavy teacher which looks at all the frames in the video is used to train a compute-efficient student which looks at only a small fraction of frames in the video. This is in contrast to a typical memory efficient Teacher-Student setting, wherein both the teacher and the student look at all the frames in the video but the student has fewer parameters. Our work thus complements the research on memory efficient video classification. We do an extensive evaluation with three types of models for video classification, viz., (i) recurrent models (ii) cluster-and-aggregate models and (iii) memory-efficient cluster-and-aggregate models and show that in each of these cases, a see-it-all teacher can be used to train a compute efficient see-very-little student. Overall, we show that the proposed student network can reduce the inference time by 30% and the number of FLOPs by approximately 90% with a negligible drop in the performance.

1. Introduction

Today video content has become extremely prevalent on the internet influencing all aspects of our life such as education, entertainment, communication *etc.* This has led to an increasing interest in automatic video processing with the aim of identifying activities [32, 40], generating textual descriptions [9], generating summaries [11, 27], answering questions [15] and so on. On one hand, with the availability of large-scale datasets [34, 36, 18, 1, 39] for various video processing tasks, it has now become possible to train increasingly complex models which have high memory and computational needs but on the other hand there is a demand for running these models on low power devices such as mobile phones and tablets with stringent constraints on latency, memory and computational cost. It is important to balance the two and design models which can learn from large amounts of data but still be computationally cheap at inference time.

In this context, the recently concluded ECCV workshop on YouTube-8M Large-Scale Video Understanding (2018) [16] focused on building memory efficient models which use less than 1GB of memory. The main motivation was to discourage the use of ensemble based methods and instead focus on memory efficient single models. One of the main ideas explored by several participants [24, 28, 30] in this workshop was to use knowledge distillation to build more compact student models. More specifically, they first train a teacher network which has a large number of parameters and then use this network to guide a much smaller student network which has limited memory requirements and can thus be employed at inference time. Of course, in addition to requiring less memory, such a model would also require fewer FLOPs as the size of weight matrices, hidden representations, *etc.* would be smaller. However, there is scope for reducing the FLOPs further because existing models process all frames in the video which may be redundant.

Based on the results of the ECCV workshop [16], we found that the two most popular paradigms for video classification are (i) *recurrent neural network* based methods and (ii) *cluster-and-aggregate* based methods. Not surprisingly, the third type of approaches based on C3D (3D convolu-

---
*Indian Institute of Technology Madras and Robert Bosch Centre for Data Science and AI (RBC-DSAI)

tions) [3] were not so popular because they are expensive in terms of their memory and compute requirements. For example, the popular I3D model [3] is trained using 64 GPUs as mentioned in the original paper. Hence, in this paper, we focus only on the first two paradigms. We first observe that, *RNN* based methods [28, 8, 31] compute a hidden representation for *every* frame in the video and then compute a final representation for the video based on these frame representations. Hence, even if the model is compact due to smaller weight matrix and/or hidden representations, the number of FLOPs would still be large because this computation needs to be done for every frame in the video. Similarly, *cluster-and-aggregate* based methods [23, 24, 30, 35, 17] have a learnable clustering matrix which is used for assigning soft clusters to *every* frame in the video. Even if the model is made compact by reducing the size of the clustering matrix and/or hidden representations, the number of FLOPs would still be large. To alleviate this problem, in this work, we focus on building models which have fewer FLOPs and are thus computationally efficient. Our work thus complements existing work on memory efficient models for video classification.

We propose to achieve this by again using the idea of distillation wherein we train a computationally expensive *teacher* network which computes a representation for the video by processing all frames in the video. We then train a relatively inexpensive *student* network whose objective is to process only a few frames of the video and produce a representation which is very similar to the representation computed by the teacher. This is achieved by minimizing (i) the squared error loss between the representations of the student network and the teacher network and/or (ii) by minimizing the difference between the output distributions (class probabilities) predicted by the two networks. Figure 1 illustrates this idea where the teacher sees every frame of the video but the student sees fewer frames, *i.e.*, every j-th frame of the video. At inference time, we then use the student network for classification thereby reducing the time required for processing the video.

We experiment with two different methods of training the *Teacher-Student* network. In the first method (which we call *Serial* Training), the teacher is trained independently and then the student is trained to match the teacher with or without an appropriate regularizer to account for the classification loss. In the second method (which we call *Parallel* Training), the teacher and student are trained jointly using the classification loss as well as the matching loss. This *parallel* training method is similar to on-the-fly knowledge distillation from a dynamic teacher as mentioned in [19]. We experiment with different students, *viz.*, (i) a hierarchical *RNN* based model (ii) *NetVLAD* and (iii) *NeXtVLAD* which is a memory efficient version of *NetVLAD* and was the best single model in the ECCV'18 workshop. We experiment

with the YouTube-8M dataset and show that the smaller student network reduces the inference time by upto 30% while still achieving a classification performance which is very close to that of the expensive teacher network.

2. Related Work

Since we focus on the task of video classification in the context of the YouTube-8M dataset [1], we first review some recent work on video classification and then some relevant work on model compression.

2.1. Video Classification

One of the popular datasets for video classification is the YouTube-8M dataset which contains videos having an average length of 200 seconds. We use this dataset in all our experiments. The authors of this dataset proposed a simple baseline model which treats the entire video as a sequence of one-second frames and uses a Long Short-Term Memory network (LSTM) to encode this sequence. Apart from this, they also propose some simple baseline models like Deep Bag of Frames (DBoF) and Logistic Regression [1]. Various other classification models [23, 37, 20, 5, 33] have been proposed and evaluated on this dataset (2017 version) which explore different methods of: (i) feature aggregation in videos (temporal as well as spatial) [5, 23], (ii) capturing the interactions between labels [37] and (iii) learning new non-linear units to model the interdependencies among the activations of the network [23]. The state-of-the-art model on the 2017 version of the Youtube-8M dataset uses *NetVLAD* pooling [23] to aggregate information from all the frames of a video.

In the recently concluded competition (2018 version), many methods [24, 28, 30, 35, 8, 21, 31, 16] were proposed to compress the models such that they fit in 1GB of memory. As mentioned by [16], the major motivation behind this competition was to avoid the late-stage model ensemble techniques and focus mainly on single model architectures at inference time [31, 30, 8]. One of the top performing systems in this competition was *NeXtVLAD* [30], which modifies *NetVLAD* [23] to squeeze the dimensionality of modules (embeddings). However, this model still processes all the frames of the video and hence has a large number of FLOPs. In this work, we take this compact *NeXtVLAD* model and make it compute efficient by using the idea of distillation. One clear message from this workshop was that the emphasis should be on single model architectures and not ensembles. Hence, in this work, we focus on single model-based solutions.

2.2. Model Compression

Recently, there has been a lot of work on model compression ([7, 26, 12, 25]) in the context of image classification. We refer the reader to survey paper by [7] for a thor-

ough review of the field. For brevity, here we refer to only those papers which use the idea of distillation. For example, [2, 13, 22, 4] use *Knowledge Distillation* to learn a more compact *student* network from a computationally expensive *teacher* network. The key idea is to train a shallow student network using soft targets (or class probabilities) generated by the teacher instead of the hard targets present in the training data. There are several other variants of this technique such as, [29] extend this idea to train a student model which not only learns from the outputs of the teacher but also uses the intermediate representations learned by the teacher as additional *hints*. This idea of *Knowledge Distillation* has also been tried in the context of pruning networks for multiple object detection [4], speech recognition [38] and reading comprehension [14].

In the context of video classification, there is some work [24, 21] on using *Quantization* for model compression. Some work on video-based action recognition [41] tries to accelerate processing in a two-stream *CNN* architecture by transferring knowledge from motion modality to optical modality. In some very recent work on video classification [10] and video captioning [6] the authors use a reinforcement learning agent to select which frames to process. We do not focus on the problem of frame selection but instead focus on distilling knowledge from the teacher once fewer frames have been selected for the student. While in this work we simply select frames uniformly, the same ideas can also be used on top of an RL agent which selects the best frames but we leave this as a future work.

3. Video Classification Models

Given a fixed set of m classes $y_1, y_2, y_3, ..., y_m \in \mathscr{Y}$ and a video $\mathbf{V}$ containing N frames $(F_0, F_1, \ldots, F_{N-1})$, the goal of video classification is to identify all the classes to which the video belongs. In other words, for each of the m classes we are interested in predicting the probability $P(y_i|\mathbf{V})$. This probability can be parameterized using a neural network f which looks at all the frames in the video to predict:

$$P(y_i|\mathbf{V}) = f(F_0, F_1, \ldots, F_{N-1})$$

Given the setup, we now briefly discuss the two state-of-the-art models that we have considered as teacher/student for our experiments.

3.1. Recurrent Network Based Models

We consider the Hierarchical Recurrent Neural Network (*H-RNN*) based model which assumes that each video contains a sequence of b equal sized blocks. Each of these blocks in turn is a sequence of l frames thereby making the entire video a sequence of sequences. In the case of the YouTube-8M dataset, these frames are one-second shots of the video and each block b is a collection of l such one-second frames. The model contains a lower level *RNN* to encode each block (sequence of frames) and a higher level *RNN* to encode the video (sequence of blocks).

3.2. Cluster and Aggregate Based Models

We consider the *NetVLAD* model with Context Gating (CG) as proposed in [23]. This model does not treat the video as a sequence of frames but simply as a bag of frames. For every frame in this bag, it first assigns a soft cluster to the frame which results is a $M \times k$ dimensional representation for the frame where k is the number of clusters considered and M is the size of the initial representation of the frame (say, obtained from a CNN). Instead of using a standard clustering algorithm such as k-means, the authors introduce a learnable clustering matrix which is trained along with all the parameters of the network. The cluster assignments are thus a function of a parameter which is learned during training. The video representation is then computed by aggregating all the frame representations obtained after clustering. This video representation is then fed to multiple fully connected layers with Context Gating (CG) which help to model interdependencies among network activations. We also experiment with *NeXtVLAD* [30] which is a compact version of *NetVLAD* wherein the $M \times k$ dimensional representation is downsampled by grouping which effectively reduces the total number of parameters in the network.

Note that all the models described above look at all the frames in the video.

4. Proposed Approach

The focus of this work is to design a simpler network g which looks at only a fraction of the N frames at inference time while still allowing it to leverage the information from all the N frames at training time. To achieve this, we propose a *Teacher-Student* network as described below wherein the teacher has access to more frames than the student.

TEACHER: The teacher network can be any state-of-the-art model described above (*H-RNN*, *NetVLAD*, *NeXtVLAD*). This teacher network looks at all the N frames of video $(F_0, F_1, \ldots, F_{N-1})$ and computes an encoding $\mathcal{E}_T$ of the video, which is then fed to a simple feedforward neural network with a multi-class output layer containing a sigmoid neuron for each of the $\mathscr{Y}$ classes. The parameters of the teacher network are learnt using a standard multi-label classification loss $\mathcal{L}_{CE}$, which is a sum of the cross-entropy loss for each of the $\mathscr{Y}$ classes. We refer to this loss as $\mathcal{L}_{CE}$ where the subscript CE stands for cross entropy between the true labels y and predictions $\hat{y}$.

$$\mathcal{L}_{CE} = -\sum_{i=1}^{|\mathscr{Y}|} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \qquad (1)$$
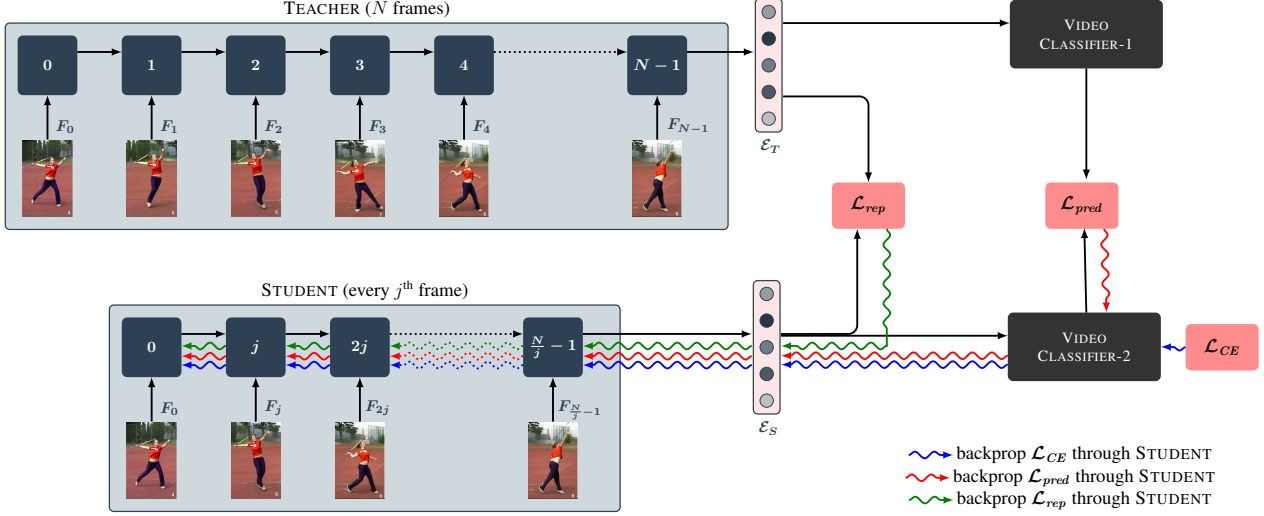
Figure 1: Architecture of TEACHER-STUDENT network for video classification

STUDENT: In addition to this teacher network, we introduce a student network which only processes every j^{th} frame $(F_0, F_j, F_{2j}, \ldots, F_{\frac{N}{j}-1})$ of the video and computes a representation $\mathcal{E}_S$ of the video from these $\frac{N}{j}$ frames. We use only same family distillation wherein both the teacher and the student have the same architecture. For example, Figure 1 shows the setup when the teacher is *H-RNN* and the student is also *H-RNN*. Further, the parameters of the output layer are shared between the teacher and the student. The student is trained to minimize the squared error loss between the representation computed by the student network and the representation computed by the teacher. We refer to this loss as $\mathcal{L}_{rep}$ where the subscript *rep* stands for representations.

$$\mathcal{L}_{rep} = ||\mathcal{E}_T - \mathcal{E}_S||^2 \qquad (2)$$

We also try a simple variant of the model, where in addition to ensuring that the final representations $\mathcal{E}_S$ and $\mathcal{E}_T$ are similar, we also ensure that the intermediate representations ($\mathcal{I}_S$ and $\mathcal{I}_T$) of the models are similar. In particular, we ensure that the representation of the frames $j, 2j$ and so on computed by the teacher and student network are very similar by minimizing the squared error distance between the corresponding intermediate representations. We refer to this loss as $\mathcal{L}_{rep}^{\mathcal{I}}$ where the superscript $\mathcal{I}$ stands for intermediate.

$$\mathcal{L}_{rep}^{\mathcal{I}} = \sum_{i=j,2j,..}^{\frac{N}{j}-1} ||\mathcal{I}_T^i - \mathcal{I}_S^i||^2 \qquad (3)$$

Alternately, the student can also be trained to minimize the difference between the class probabilities predicted by the teacher and the student. We refer to this loss as $\mathcal{L}_{pred}$ where the subscript $pred$ stands for *predicted probabilities*. More specifically if $\mathcal{P}_T = \{p_T^1, p_T^2, ..., p_T^m\}$ and $\mathcal{P}_S = \{p_S^1, p_S^2, ..., p_S^m\}$ are the probabilities predicted for the m classes by the teacher and the student respectively, then

$$\mathcal{L}_{pred} = d(\mathcal{P}_T, \mathcal{P}_S) \qquad (4)$$

where d is any suitable distance metric such as *KL* divergence or squared error loss.

TRAINING: Intuitively, it makes sense to train the teacher first and then use this trained teacher to guide the student. We refer to this as the *Serial* mode of training as the student is trained after the teacher as opposed to jointly. For the sake of analysis, we use different combinations of loss function to train the student as described below:

(a) $\mathcal{L}_{rep}$: Here, we operate in two stages. In the first stage, we train the student network to minimize the $\mathcal{L}_{rep}$ as defined above, *i.e.*, we train the parameters of the student network to produce representations which are very similar to the teacher network. The idea is to let the student learn by only mimicking the teacher and not worry about the final classification loss. In the second stage, we then plug in the classifier trained along with the teacher (see Equation 1) and fine-tune all the parameters of the student and the classifier using the cross entropy loss, $\mathcal{L}_{CE}$. In practice, we found that the fine-tuning done in the second stage helps to improve the performance of the student.

(b) $\mathcal{L}_{rep} + \mathcal{L}_{CE}$: Here, we train the student to jointly minimize the representation loss as well as the classification loss. The motivation behind this was to ensure that

while mimicking the teacher, the student also keeps an eye on the final classification loss from the beginning (instead of being fine-tuned later as in the case above).

(c) $\mathcal{L}_{pred}$: Here, we train the student to only minimize the difference between the class probabilities predicted by the teacher and the student.

(d) $\mathcal{L}_{pred} + \mathcal{L}_{CE}$: Here, in addition to mimicking the probabilities predicted by the teacher, the student is also trained to minimize the cross entropy loss.

(e) $\mathcal{L}_{rep} + \mathcal{L}_{CE} + \mathcal{L}_{pred}$: Finally, we combine all the 3 loss functions. Figure 1 illustrates the process of training the student with different loss functions.

For the sake of completeness, we also tried an alternate mode in which we train the teacher and student in parallel such that the objective of the teacher is to minimize $\mathcal{L}_{CE}$ and the objective of the student is to minimize one of the 3 combinations of loss functions described above. We refer to this as *Parallel* training.

5. Experimental Setup

In this section, we describe the dataset used for our experiments, the hyperparameters that we considered, the baseline models that we compared with and the effect of different loss functions and training methods.

1. Dataset: The YouTube-8M dataset (2017 version) [1] contains 8 million videos with multiple classes associated with each video. The average length of a video is $200s$ and the maximum length of a video is $300s$. The authors of the dataset have provided pre-extracted audio and visual features for every video such that every second of the video is encoded as a single frame. The original dataset consists of 5,786,881 training (70%), 1,652,167 validation (20%) and 825,602 test examples (10%). Since the authors did not release the test set, we used the original validation set as test set and report results on it. In turn, we randomly sampled 48,163 examples from the training data and used these as validation data for tuning the hyperparameters of the model. We trained our models using the remaining 5,738,718 training examples.

2. Hyperparameters: For all our experiments, we used Adam Optimizer with the initial learning rate set to 0.001 and then decreased it exponentially with 0.95 decay rate. We used a batch size of 256. For both the student and teacher networks we used a 2-layered MultiLSTM Cell with cell size of 1024 for both the layers of the hierarchical model. For regularization, we used dropout (0.5) and $\mathbf{L}_2$ regularization penalty of 2 for all the parameters. We trained all the models for 5 epochs and then picked the best model-based on validation performance. We did not see any

benefit of training beyond 5 epochs. For the teacher network we chose the value of l (number of frames per block) to be 20 and for the student network, we set the value of l to 5 or 3 depending on the reduced number of frames considered by the student.

In the training of *NetVLAD* model, we have used the standard hyperparameter settings as mentioned in [23]. We consider 256 clusters and 1024 dimensional hidden layers. Similarly, in the case of *NeXtVLAD*, we have considered the hyperparameters of the single best model as reported by [30]. In this network, we are working with a cluster size of 128 with hidden size as 2048. The input is reshaped and downsampled using 8 groups in the cluster as done in the original paper. For all these networks, we have worked with a batch size of 80 and an initial learning rate of 0.0002 exponentially decayed at the rate of 0.8. Additionally, we have applied dropout of 0.5 on the output of *NeXtVLAD* layer which helps for better regularization.

3. Evaluation Metrics: We used the following metrics as proposed in [1] for evaluating the performance of different models :

- GAP (Global Average Precision): is defined as

$$GAP = \sum_{i=1}^{P} p(i) \nabla r(i)$$

where $p(i)$ is the precision at prediction i, $\nabla r(i)$ is the change in recall at prediction i and P is the number of top predictions that we consider. Following the original YouTube-8M Kaggle competition we use the value of P as 20.
- mAP (Mean Average Precision) : The mean average precision is computed as the unweighted mean of all the per-class average precisions.

4. Models Compared: We compare our *Teacher-Student* network with the following models which helps us to better contextualize our results.

a) Teacher-Skyline: The original teacher model which processes all the frames of the video. This, in some sense, acts as the upper bound on the performance.

b) Baseline Methods: As baseline, we consider a model (*H-RNN*, *NetVLAD* or *NeXtVLAD*) which is trained from scratch but uses only k frames of the video. However, unlike the student model this model is not guided by a teacher. These k frames can be (i) separated by a constant interval and are thus equally spaced (Uniform-k) or (ii) sampled randomly from the video (Random-k) or (iii) the first k frames of the video (First-k) or (iv) the middle k frames of the video (Middle-k) or (v) the last k frames of the video (Last-k) or (i) the first $\frac{k}{3}$, middle $\frac{k}{3}$ and last $\frac{k}{3}$ frames of the video (First-Middle-Last-k).

MODEL		k=6		k=10		k=15		k=20		k=30	
		GAP	mAP	GAP	mAP	GAP	mAP	GAP	mAP	GAP	mAP
Model with k frames		BASELINE METHODS									
Uniform-k		0.715	0.266	0.759	0.324	0.777	0.35	0.785	0.363	0.795	0.378
Random-k		0.679	0.246	0.681	0.254	0.717	0.268	0.763	0.329	0.774	0.339
First-k		0.478	0.133	0.539	0.163	0.595	0.199	0.632	0.223	0.676	0.258
Middle-k		0.577	0.178	0.600	0.198	0.620	0.214	0.638	0.229	0.665	0.25
Last-k		0.255	0.062	0.267	0.067	0.282	0.077	0.294	0.083	0.317	0.094
First-Middle-Last-k		0.640	0.215	0.671	0.242	0.680	0.249	0.698	0.268	0.721	0.287
Training	*Student-Loss*	TEACHER-STUDENT METHODS									
Parallel	L_{rep}	0.724	0.280	0.762	0.331	0.785	0.365	0.794	0.380	0.803	0.392
Parallel	L_{rep}, L_{CE}	0.726	0.285	0.766	0.334	0.785	0.362	0.795	0.381	0.804	0.396
Parallel	$L_{rep}, L_{pred}, L_{CE}$	0.729	0.292	0.770	0.337	**0.789**	0.371	0.796	0.388	**0.806**	0.404
Serial	L_{rep}	0.727	0.288	0.768	0.339	0.786	0.365	0.795	0.381	0.802	0.394
Serial	L_{pred}	0.722	0.287	0.766	0.341	0.784	0.367	0.793	0.383	0.798	0.390
Serial	L_{rep}, L_{CE}	0.728	0.291	0.769	0.341	0.786	0.368	0.794	0.383	0.803	0.399
Serial	L_{pred}, L_{CE}	0.724	0.289	0.763	0.341	0.785	0.369	0.795	0.386	0.799	0.391
Serial	$L_{rep}, L_{pred}, L_{CE}$	**0.731**	**0.297**	**0.771**	**0.349**	**0.789**	**0.375**	0.798	**0.390**	**0.806**	**0.405**

Table 1: Performance comparison of proposed *Teacher-Student* models using different **Student-Loss** variants, with their corresponding baselines using k frames. **Teacher-Skyline** model refers to the default model which process all the frames in a video and achieves GAP and mAP score of **0.811** and **0.414** respectively.

We report results with different values of k : 6, 10, 15, 20, 30.

6. Discussion And Results

Since we have 3 different base models (*H-RNN*, *NetVLAD*, *NeXtVLAD*), 5 different combinations of loss functions (see section 4), 2 different training paradigms (Serial and Parallel) and 5 different baselines for each base model, the total number of experiments that we needed to run to report all these results was very large. To reduce the number of experiments we first consider only the *H-RNN* model to identify the (a) best baseline (Uniform-k, Random-k, First-k, Middle-k, Last-k, First-Middle-Last-k) (b) best training paradigm (Serial v/s Parallel) and (c) best combination of loss function. We then run the experiments on *NetVLAD* and *NeXtVLAD* using only the best baseline, best training paradigm and best loss function thus identified. The results of our experiments using the *H-RNN* model are summarized in Table 1 to Table 3 and are discussed first followed by a discussion of the results using *NetVLAD* and *NeXtVLAD* as summarized in Tables 5 and 6:

1. Comparisons of different baselines: First, we simply compare the performance of different baselines listed in the top half of Table 1. As is evident, the Uniform-k baseline which looks at equally spaced k frames performs better than all the other baselines. The performance gap between Uniform-k and the other baselines is even more significant when the value of k is small. The main purpose of this experiment was to decide the right way of selecting frames for the student network. Based on these results, we ensured that for all our experiments, we fed equally spaced k frames to

Model	Metric	%age of training data		
		10%	25%	50%
Serial	GAP	0.774	0.788	0.796
	mAP	0.345	0.369	0.373
Uniform	GAP	0.718	0.756	0.776
	mAP	0.220	0.301	0.349

Table 2: Effect of amount of training data on performance of Serial and Uniform models using 30 frames

the student.

2. Comparing Teacher-Student Network with Uniform-k Baseline: As mentioned above, the Uniform-k baseline is a simple but effective way of reducing the number of frames to be processed. We observe that all the teacher-student models outperform this strong baseline. Further, in a separate experiment as reported in Table 2 we observe that when we reduce the number of training examples seen by the teacher and the student, then the performance of the Uniform-k baseline drops and is much lower than that of the corresponding *Teacher-Student* network. This suggests that the *Teacher-Student* network can be even more useful when the amount of training data is limited.

3. Serial Versus Parallel Training of Teacher-Student: While the best results in Table 1 are obtained using *Serial* training, if we compare the corresponding rows of *Serial* and *Parallel* training we observe that there is not much difference between the two. We found this to be surprising and investigated this further. In particular, we compared the

(a) Training with L_{rep} (b) Training with L_{rep} and L_{CE} (c) Training: $L_{rep}, L_{CE}, L_{pred}$
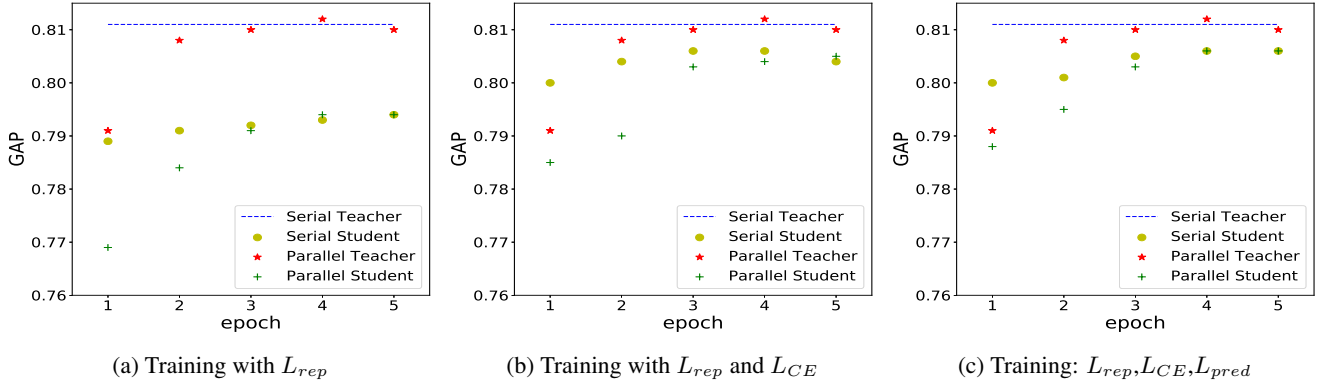
Figure 2: Performance comparison (GAP score) of different variants of *Serial* and *Parallel* methods in *Teacher-Student* training.

performance of the teacher after different epochs in the *Parallel* training setup with the performance of a static teacher trained independently (*Serial*). We plotted this performance in Figure 2 and observed that after 3-4 epochs of training, the *Parallel* teacher is able to perform at par with *Serial* teacher (the constant blue line). As a result, the *Parallel* student now learns from this trained teacher for a few more epochs and is almost able to match the performance of the *Serial* student. This trend is same across the different combinations of loss functions that we used.

4. Visualization of Teacher and Student Representations: Apart from evaluating the final performance of the model in terms of mAP and GAP, we also wanted to check if the representations learned by the teacher and student are indeed similar. To do this, we chose top-5 classes (*class1: Vehicle, class2: Concert, class3: Association football, class4: Animal, class5: Food*) in the Youtube-8M dataset and visualized the TSNE-embeddings of the representations computed by the student and the teacher for the same video (see Figure 3). We use the darker shade of a color to represent teacher embeddings of a video and a lighter shade of the same color to represent the student embeddings of the same video. We observe that the dark shades and the light shades of the same color almost completely overlap showing that the student and teacher representations are indeed very close to each other. This shows that introducing the $\mathcal{L}_{rep}$ indeed brings the teacher and student representations close to each other.

5. Matching Intermediate v/s Final representations: Intuitively, it seemed that the student should benefit more if we train it to match the intermediate representations of the teacher at different timesteps as opposed to only the final representation at the last time step. However, as reported in Table 4, we did not see any benefit of matching intermediate representations.

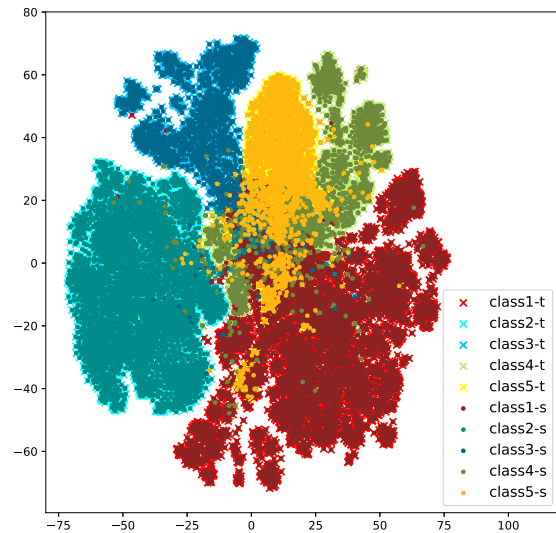**6. Computation time of different models:** The main aim

Figure 3: TSNE-Embedding of teacher and student representations. Here, class c refers to the cluster representation obtained corresponding to c^{th} class, whereas t and s denote teacher and student embedding respectively.

Model	Time (hrs.)	FLOPs (Billion)
Teacher-Skyline	13.00	5.058
k= 10	7.61	0.167
k= 20	8.20	0.268
k= 30	**9.11**	0.520

Table 3: Comparison of FLOPs and evaluation time of models using k frames with *Skyline* model on original validation set using Tesla k80s GPU

of this work was to ensure that the computational cost and time is minimized at inference time. The computational

MODEL		Intermediate		Final	
		GAP	mAP	GAP	mAP
Parallel	L_{rep}	0.803	0.393	0.803	0.392
Parallel	$L_{rep} + L_{CE}$	0.803	0.396	0.804	0.396
Parallel	$L_{rep} + L_{pred}$	0.804	0.400	0.806	0.404
Serial	L_{rep}	0.804	0.395	0.802	0.394
Serial	$L_{rep} + L_{CE}$	0.803	0.397	0.803	0.399
Serial	$L_{rep} + L_{pred}$	0.806	0.405	0.806	0.405

Table 4: Comparison of *Final* and *Intermediate* representation matching by *Student* network using k=30 frames

Model: *NetVLAD*	k=10		k=30	
	mAP	GAP	mAP	GAP
Skyline			**0.462**	**0.823**
Uniform	0.364	0.773	0.421	0.803
Student	0.383	0.784	0.436	0.812

Table 5: Performance of *NetVLAD* model with k= 10, 30 frames in proposed Teacher-Student Framework

Model: *NeXtVLAD*	k=30		FLOPs
	mAP	GAP	(in Billion)
Skyline	0.464	0.831	1.337
Uniform	0.424	0.812	0.134
Student	0.439	0.818	0.134

Table 6: Performance and FLOPs comparison in *NeXtVLAD* model with k=30 frames in proposed Teacher-Student Framework

cost can be measured in terms of the number of FLOPs. As shown in Table 3 when k=30, the inference time drops by 30% and the number of FLOPs reduces by approximately 90%, but the performance of the model is not affected. In particular, as seen in Table 1, when $k = 30$, the GAP and mAP drop by 0.5-0.9% and 0.9-2% respectively as compared to the teacher skyline.

7. Performance using *NetVLAD* models: In Table 5 we summarize the results obtained using *NetVLAD* as the base model in the *Teacher-Student* network. Here the student network was trained using the best loss function ($L_{rep}, L_{pred}, L_{CE}$) and the best training paradigm (*Serial*) as identified from the experiments done using the *H-RNN* model. Further, we consider only the Uniform-k baseline as that was the best baseline as observed in our previous experiments. Here again we observe that the student network does better than the Uniform-k baseline.

8. Combining with memory efficient models: Lastly, we experiment with the compact *NeXtVLAD* model and show that the student network performs slightly better than the Uniform-k baseline in terms of mAP but not so much in terms of GAP (note that mAP gives equal importance to all classes but GAP is influenced more by the most frequent classes in the dataset). Once again, there is a significant reduction in the number of FLOPs (approximately 89%).

7. Conclusion and Future Work

We proposed a method to reduce the computation time for video classification using the idea of distillation. Specifically, we first train a teacher network which computes a representation of the video using all the frames in the video. We then train a student network which only processes k frames of the video. We use different combinations of loss functions which ensures that (i) the final representation produced by the student is the similar as that produced by the teacher and (ii) the output probability distributions produced by the student are similar to those produced by the teacher. We compare the proposed models with a strong baseline and skyline and show that the proposed model outperforms the baseline and gives a significant reduction in terms of computational time and cost when compared to the skyline. In particular, we evaluate our model on the YouTube-8M dataset and show that the computationally less expensive student network can reduce the computation time by 30% while giving an approximately similar performance as the teacher network.

As future work, we would like to evaluate our model on other video processing tasks such as summarization, question answering and captioning. We would also like to train a student with an ensemble of teachers (preferably from different families). Lastly, we would like to train a reinforcement learning agent to first select the most favorable k (or even fewer) frames in the video and use these as opposed to simply using equally spaced k frames.

References

[1] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Apostol (Paul) Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. In *arXiv:1609.08675*, 2016. 1, 2, 5

[2] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in Neural Information Processing Systems 27*, pages 2654–2662, 2014. 3

[3] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[4] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. In I. Guyon, U. V. Luxburg,

S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 742–751. Curran Associates, Inc., 2017. 3

[5] Shaoxiang Chen, Xi Wang, Yongyi Tang, Xinpeng Chen, Zuxuan Wu, and Yu-Gang Jiang. Aggregating frame-level features for large-scale video classification. *CoRR*, abs/1707.00803, 2017. 2

[6] Yangyu Chen, Shuhui Wang, Weigang Zhang, and Qingming Huang. Less is more: Picking informative frames for video captioning. In *The European Conference on Computer Vision (ECCV)*, September 2018. 3

[7] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *CoRR*, abs/1710.09282, 2017. 2

[8] Hyun-Dong Lee Choi and Byoung-Tak Zhang. Temporal attention mechanism with conditional inference for large-scale multi-label video classification. 2

[9] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 1

[10] Hehe Fan, Zhongwen Xu, Linchao Zhu, Chenggang Yan, Jianjun Ge, and Yi Yang. Watching a small portion could be as good as watching all: Towards efficient video classification. In *International Joint Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, July 13-19, 2018*, pages 705–711, 2018. 3

[11] Boqing Gong, Wei-Lun Chao, Kristen Grauman, and Fei Sha. Diverse sequential subset selection for supervised video summarization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2069–2077. Curran Associates, Inc., 2014. 1

[12] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2

[13] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. 3

[14] Minghao Hu, Yuxing Peng, Furu Wei, Zhen Huang, Dongsheng Li, Nan Yang, and Ming Zhou. Attention-guided answer distillation for machine reading comprehension. *CoRR*, abs/1808.07644, 2018. 3

[15] Yunseok Jang, Yale Song, Youngjae Yu, Youngjin Kim, and Gunhee Kim. Tgif-qa: Toward spatio-temporal reasoning in visual question answering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1

[16] Lee Joonseok, Natsev Apostol (Paul), ReadeWalter, Sukthankar Rahul, and Toderici George. The 2nd youtube-8m large-scale video understanding challenge. *In: Proc. of the 2nd Workshop on YouTube-8M Large-Scale Video Understanding (ECCV 2018)*, 2018. 1, 2

[17] Sebastian Kmiec, Juhan Bae, and Ruijian An. Learnable pooling methods for video classification. *arXiv preprint arXiv:1810.00530*, 2018. 2

[18] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: A large video database for human motion recognition. In *2011 International Conference on Computer Vision*, pages 2556–2563, Nov 2011. 1

[19] Gong Shaogang Lan Xu, Zhu Xiatian. Knowledge distillation by on-the-fly native ensemble. *To Appear In Proceedings of Thirty-second Annual Conference on Neural Information Processing Systems NIPS*, 2018. 2

[20] Fu Li, Chuang Gan, Xiao Liu, Yunlong Bian, Xiang Long, Yandong Li, Zhichao Li, Jie Zhou, and Shilei Wen. Temporal modeling approaches for large-scale youtube-8m video understanding. *CoRR*, abs/1707.04555, 2017. 2

[21] Liu Bo Liu Tianqi. Constrained-size tensorflow models for youtube-8m video understanding challenge. *In: Proc. of the 2nd Workshop on YouTube-8M Large-Scale Video Understanding (2018)*, 2018. 2, 3

[22] D. Lopez-Paz, B. Schölkopf, L. Bottou, and V. Vapnik. Unifying distillation and privileged information. In *International Conference on Learning Representations*, 2016. 3

[23] Antoine Miech, Ivan Laptev, and Josef Sivic. Learnable pooling with context gating for video classification. *CoRR*, abs/1706.06905, 2017. 2, 3, 5

[24] Skalic1 Miha and Austin David. Building a size constrained predictive model for video classification. *In: Proc. of the 2nd Workshop on YouTube-8M Large-Scale Video Understanding (2018)*, 2018. 1, 2, 3

[25] Deepak Mittal, Shweta Bhardwaj, Mitesh M. Khapra, and Balaraman Ravindran. Recovering from random pruning: On the plasticity of deep convolutional neural networks. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, March 2018. 2

[26] Sharan Narang, Gregory F. Diamos, Shubho Sengupta, and Erich Elsen. Exploring sparsity in recurrent neural networks. April 2017. 2

[27] P. Pan, Z. Xu, Y. Yang, F. Wu, and Y. Zhuang. Hierarchical recurrent neural encoder for video representation with application to captioning. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1029–1038, June 2016. 1

[28] Ostyakov Pavel, Logacheva Elizaveta, Suvorov Roman, Aliev Vladimir, Sterkin Gleb, Khomenko Oleg, and I. Nikolenko Sergey. Label denoising with large ensembles of heterogeneous neural networks. *In: Proc. of the 2nd Workshop on YouTube-8M Large-Scale Video Understanding (2018)*, 2018. 1, 2

[29] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *In Proceedings of ICLR*, 2015. 3

[30] Lin Rongcheng, Xiao Jing, and Jianping Fan. Nextvlad: An efficient neural network to aggregate frame-level features for large-scale video classification. *In: Proc. of the 2nd Workshop on YouTube-8M Large-Scale Video Understanding (2018)*, 2018. 1, 2, 3, 5

[31] Garg Shivam. Learning video features for multi-label classification. *In: Proc. of the 2nd Workshop on YouTube-8M Large-Scale Video Understanding (2018)*, 2018. 2

[32] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In Z.

Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 568–576. Curran Associates, Inc., 2014. 1

[33] Miha Skalic, Marcin Pekalski, and Xingguo E. Pan. Deep learning methods for efficient large scale video labeling. *CoRR*, abs/1706.04572, 2017. 2

[34] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012. 1

[35] Yongyi Tang, Xing Zhang, Jingwen Wang, Shaoxiang Chen, Lin Ma, and Yu-Gang Jiang. Non-local netvlad encoding for video classification. *arXiv preprint arXiv:1810.00207*, 2018. 2

[36] Haiqiang Wang, Ioannis Katsavounidis, Jiantong Zhou, Jeonghoon Park, Shawmin Lei, Xin Zhou, Man-On Pun, Xin Jin, Ronggang Wang, Xu Wang, Yun Zhang, Jiwu Huang, Sam Kwong, and C.-C. Jay Kuo. Videoset: A large-scale compressed video quality dataset based on jnd measurement. *Journal of Visual Communication and Image Representation*, 46, 2017. 1

[37] He-Da Wang, Teng Zhang, and Ji Wu. The monkeytyping solution to the youtube-8m video understanding challenge. *CoRR*, abs/1706.05150, 2017. 2

[38] Jeremy H. M. Wong and Mark J. F. Gales. Sequence student-teacher training of deep neural networks. In *Interspeech 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA, September 8-12, 2016*, pages 2761–2765, 2016. 3

[39] Jianxiong Xiao, Krista A. Ehinger, James Hays, Antonio Torralba, and Aude Oliva. Sun database: Exploring a large collection of scene categories. *Int. J. Comput. Vision*, 119(1):3–22, Aug. 2016. 1

[40] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 1

[41] Bowen Zhang, Limin Wang, Zhe Wang, Yu Qiao, and Hanli Wang. Real-time action recognition with enhanced motion vector cnns. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 3