Confidence Distillation for Efficient Action Recognition

CONFIDENCE DISTILLATION FOR EFFICIENT ACTION RECOGNITION

BY

SHERVIN MANZURI SHALMANI, B.Sc., Computer Engineering

Sharif University of Technology, Tehran, Iran

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTING AND SOFTWARE

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

Master of Science

Master of Science (2020) McMaster University

(Computing and Software) Hamilton, Ontario, Canada

TITLE:	Confidence Distillation for Efficient Action Recognition
AUTHOR:	Shervin Manzuri Shalmani B.Sc., Computer Engineering Sharif University of Technology, Tehran, Iran
SUPERVISOR:	Dr. Rong Zheng, Dr. Fei Chiang
NUMBER OF PAGES:	xiii, 68

Abstract

Modern neural networks are powerful predictive models. However, when it comes to recognizing that they may be wrong about their predictions and measuring the certainty of beliefs, they perform poorly. For one of the most common activation functions, the ReLU and its variants, even a well-calibrated model can produce incorrect but high confidence predictions. In the related task of action recognition, most current classification methods are based on clip-level classifiers that densely sample a given video for non-overlapping, same sized clips and aggregate the results using an aggregation function - typically averaging - to achieve video level predictions. While this approach has shown to be effective, it is sub-optimal in recognition accuracy and has a high computational overhead. To mitigate both these issues, we propose the confidence distillation framework to firstly teach a representation of uncertainty of the teacher to the student and secondly divide the task of full video prediction between the student and the teacher models. We conduct extensive experiments on three action recognition datasets and demonstrate that our framework achieves state-of-the-art results in action recognition accuracy and computational efficiency.

Acknowledgements

I would like to thank the following people, without whom I would not have been able to complete this research, and without whom I would not have made it through my Masters's degree.

I would like to thank the WiSeR lab and the Data Science lab teams at McMaster University, especially to my supervisors Dr. Fei Chiang and Dr. Rong Zheng, whose insight and knowledge around the subject matter and academic conduct steered me through this research. I would also like to thank Dr. Shahram Shirani and Dr. Wenbo He for their thoughtful and useful comments as my examining committee members.

A special thanks to (now) Dr. Yihao Fang for giving me feedback on some riskier ideas that I wanted to test during my experiments. And furthermore, I would like to thank anonymous Redditors who also engaged in meaningful conversations with me when I asked particular optimization questions. I would like to thank my colleagues at McMaster University and outside of it who supported me with more off-topic discussions, giving me a much-needed break to develop my ideas better.

And my biggest thanks goes to my family for all the support you have shown me throughout this research. Specifically, my parents, without the support of whom I would not have been able to get to this point in my profession.

Abbreviations

ANN	Artificial Neural Network
DNN	Deep Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long short-term memory
DRL	Deep Reinforcement Learning
FLOP	Floating Point Operation
KD	Knowledge Distillation
KLD	Kullback-leibler Divergence
MCFP	Monte-carlo forward pass
SGD	Stochastic Gradient Descent
FPS	Frames Per Second
CPS	Clips Per Second
FP	False Positives
TN	True Negatives
TP	True Positives
FN	False Negatives

FPR	False positive rate
FNR	False negative rate

Notation

$\mathcal{V}$	A full length video
$\mathbf{V_i}$	The i-th clip of a video
Ω	A computationally expensive pre-trained clip classifier
Φ	A computationally efficient clip sampler/classifier
T	Arbitrary length for a video
$\mathbf{s}$	Network's output logit vectors
$\mathbf{p}$	Output probability vectors after applying softmax to a network's logit vector
$\tilde{\mathbf{p}}$	Scaled probability vectors after applying temperature scaling
$\hat{\mathbf{p}}$	Modified probability vectors after applying confidence scaling
τ	Temperature hyperparameter for distillation
z	Ground truth scalar confidence score
$\tilde{z}$	Predicted scalar confidence score
λ	Weight of the confidence loss
μ	Weight of the positive component of the confidence loss
α	The learning rate hyperparameter
K	The number of sampled clips
H	Empirical entropy

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Deep Learning

The human brain has 100 billion neurons. Each of these neurons is, on average connected to 10,000 other neurons to exchange information. Some neurons feed other neurons with data (e.g. vision, sound, pain), some other neurons control actions like muscles, and most other neurons are hidden somewhere in between, continually changing their signal strength in order for us to learn. Inspired by this abstract understanding of biology, deep learning aims to model non-linear relationships of the input data through an end-to-end differentiable architecture that can be optimized with gradient-based learning. While neural networks have been around since the 1960s [1], they were later optimized with a popular form of gradient descent based error propagation [2] (although this was independently rediscovered many times). Architectures that comprise the backbone of today's neural networks such as the convolutional neural network (CNN) [3, 4] and the long short-term memory (LSTM)

[5] were later introduced and have been applied to problems that took images, time-series and natural language as input.

As of today, there has been tremendous progress and success made with research on deep neural networks in a wide array of real-world artificial intelligence problems. These algorithms' effloresence has provided us with powerful models to solve problems in computer vision, reinforcement learning and natural language processing, wherein commercial applications focus on supervised learning to make artificial neural networks (ANN) imitate human teachers using human labelled data. In 2014, attention mechanisms [6] allowed for locating information in theoretically infinite sized contexts. In 2015, the ResNet architecture [7] surpassed the human level accuracy in the ImageNet classification challenge by increasing the depth of CNNs and tackling the vanishing gradient problem. In 2016, the AlphaGo [8] reinforcement learning agent beat the then human world champion Lee Sedol in the game of Go. In 2018, human-level parity was achieved in the task of Chinese-to-English translation [9].

These successes can be mostly attributed to: (i) the increased complexity of the models, (ii) the curation of many high-quality, large-scale datasets and (iii) the dramatical increase in computing power. However, this means that using large-scale datasets is crucial to train deep neural networks, which can contain billions upon billions of parameters. For example, the ImageNet [10] dataset, which contains more than a million high-quality labelled images, facilitates the training of very deep CNNs that acquire generic feature representations [11] transferable from the source tasks (classification) to other tasks such as semantic segmentation, image captioning and object detection.

1.1.1 Video Recognition

Like image recognition, video recognition is an active research topic in the computer vision community due to its significant potential for improving accessibility, better video recommendation, video retrieval, and other contexts. The growth of online social platforms and portable video recording devices has lead to an ever-increasing amount of videos being captured, shared and consumed every day, consequently increasing access to large scale datasets.

Training Datasets

Compared to their image recognition datasets, it is more challenging to aquire and label video recognition datasets. Until recently, representative benchmark datasets such as the UCF-101 [12] and HMDB-51 [13] only contained around 10K videos, a number too small to be used for optimizing and training very deep spatiotemporal neural networks from scratch. Before the introduction of larger benchmark video datasets, many video recognition models were pre-trained on ImageNet and then tuned on these video datasets.

In 2015, ActivityNet [14] was produced, which is somewhat larger and allows for untrimmed action classification with a limited number of action instances. More recently, the Kinetics dataset [15] was created and became one of the de facto video dataset standards for video recognition. As of the time of writing, the dataset contains [16] more than 650 K videos for 700 action classes. It has been shown that this dataset is sufficient [17] to train generic feature representations for the task of video recognition in spatiotemporal CNNs. Notwithstanding, even bigger datasets such as Sports-1M [18], YouTube-8M [19] and IG-65M [20] have been produced. While

Table 1.1: Action recognition datasets.

Dataset	Objective	Trimmed	Collection	#Category	#Video (all splits)
UCF-101	Fine-tuning	Yes	YouTube	101	13,320
HMDB-51	Fine-tuning	Yes	Movie	51	7,000
ActivityNet	Fine-tuning	No	YouTube	200	19,994
Kinetics*	Pre-training	Yes*	YouTube	700**	650,000
MiT	Pre-training	Yes	YouTube	339	903,964

All datasets contain a single label per video.
* Videos can be untrimmed depending on the processing scheme.
** The kinetics dataset has several variations, with 400, 600 and 700 classes.

these datasets are larger than Kinetics, noisy and weakly labelled video-level annotations prevents them from providing sufficient generalization capability to models after training. Another dataset of note used in conjunction with Kinetics for pre-training is the moments in time (MiT) dataset [21]. This dataset is larger than Kinetics with close to a million videos. However, its ability to efficiently train a general representation is at best on par with Kinetics [22]. In Table 1.1 we provide a summary of the mentioned datasets.

The datasets, as mentioned earlier, are usually utilized to measure a network's ability to learn. Therefore, they are mostly used to pre-train a spatiotemporal CNN that is later fine-tuned on the specific video recognition task. Of the task-specific datasets in this study's scope, the Something-Something dataset [23] and the Jester dataset [24] are better suited to benchmark temporal-related tasks.

As can be seen, in recent years, considerable effort has been made to address the lack of data for video recognition, laying the foundation for the development of better models for this task.

Architectures

As opposed to images, videos have an additional temporal dimension. This extra dimension requires us to model the temporal dependencies to understand temporal tasks efficiently. In this study, we focus on models that target spatiotemporal visual recognition. In the deep learning era, there are primarily three different categories of models to extract a video representation:

- 2D CNNs, starting from the idea of CNN+LSTM [25] as their baseline. This approach generally also involves a two-stream CNN [26, 27] architecture, one stream for an RGB image and another stream to feed the optical flow to the network.

- 2D CNNs augmented with temporal modelling [28, 29] blocks that aim to efficiently propagate information through time.

- 3D CNNs that directly compute video volumes with spatiotemporal xyt kernels (e.g. C3D [30], CSN [31] or 3D ResNets [17]) and (2+1)D CNNs (e.g. R(2+1)D [32]) that perform similarly to 3D CNNs but separately process the spatial and temporal volume at each stacked block.

The majority of models developed for action recognition focus on building powerful clip-level classifiers that operate on short time-windows spanning several seconds [33, 30, 32, 31, 34, 35, 36]. The classifier is applied to all the clips across a video to recognize actions, and the results are aggregated using an aggregation operator or some more sophisticated spatio-temporal modelling technique such as a recurrent neural network or weighted averaging.

Even though initially 2D CNNs outperformed their 3D counterparts, the most recent trend has shifted to focus on 3D CNNs which have started to significantly outperform RNN and 2D CNN alternatives on video classification benchmarks. More specifically, 3D and (2+1)D methods are said to be the most promising methods for video recognition [17, 22]. Nevertheless, they also suffer from poor computational efficiency and high parameterization, making them prone to overfitting.

Video Classification

Action recognition and video classification models are typically evaluated by two metrics: (1) classification accuracy and (2) number of floating-point operations (FLOPs) per clip. Optimizing the second metric - which is commonly used to determine inference efficiency - is a great challenge for large-scale applications due to the massive number of parameters in deep spatio-temporal CNNs. Most modern action recognition models operate by first classifying individual clips (segments) of fixed temporal length. Full videos are then classified by aggregating the clip-level predictions over the full video. This aggregation can be done by averaging or a more sophisticated approach (e.g. long-term filtering and pooling using temporal strides in [37]). However, the most common and practical approach is simple averaging. Applying a clip classifier over all individual clips of a video stream may be reasonable when the video length is known to be short. In real-world applications, however, such an assumption does not hold. Videos may be more than a few seconds long, providing the classifier with ample opportunity to misclassify, negatively affecting the video-level classification accuracy. When accuracy is not that important, one could simply sample clips at random intervals. Other recent works [38, 29, 39] perform equidistant sampling

instead of sampling all clips or random sampling; that is, instead of averaging predictions over all clips, predictions are averaged for K clips, sampled at equidistant intervals over the video. This approach would improve computational efficiency; however, the naive assumption of equidistant positioning for keyframes will make the models inaccurate. In a real-world setting, aleatoric uncertainty (the uncertainty inherent in the data) of input clips does not necessarily follow an equidistant distribution, and there is still ample opportunity to misclassify by the classifier.

The goal of this thesis is to design an effective clip sampling scheme for methods that utilize clip averaging for video classification.

1.2 Challenges

Given that most architectures for video classification task rely on a form of spatio-temporal aggregating of clip-level results, two significant challenges arise. (1) *Computational efficiency*: applying a clip classifier to the entire video is computationally expensive. For instance, if a video is comprised of 18 clips, this would mean that each clip has to go through a forward pass through an expensive but accurate 3D classifier. In the case of a state-of-the art model such as I3D [40], this requires 11,289,856 Mega FLOPs of computations in total. (2) *Aggregation over incorrect predictions*: ReLU networks (and other variants of the ReLU, basically any network which results in a piecewise affine classifier function) are shown to be biased towards producing high confidence (probability) scores even after calibration [41] (which we will expand upon later in Chapter 2); as such they can often produce high-confidence incorrect predictions. This means that by incorporating these high-confidence incorrect clip-level predictions in the aggregate video-level score, the overall video-level accuracy will be

negatively affected. In summary, we are paying a high computational price for worse results.

A case study is shown in Table 1.2 below using the 3D ResNeXt-101 classifier [33] on the Kinetics-600 dataset (validation set). Here the classifier takes as input clips of shape $112 \times 112 \times 3 \times 16$, i.e. a stack of 16 RGB frames. The aggregation operator is average pooling.

Table 1.2: Dense sampling vs. sampling only the best k clips

	Dense	Oracle			
	k = All	k = 1	k = 5	k = 10	k = 15
ResNeXt-101	68.3	81.27	76.64	71.3	69.1

In Table 1.2, the top-1 accuracy on the Kinetics-600 video classification task is shown. Here accuracy denotes cases where the true class matches with the most probable class predicted by the model. Dense sampling applies the classifier to all frames and averages the results. To calculate an upper bound for classification scores we define an *Oracle* that cheats by looking at the label y and only considers clips that yield the k highest classification scores for y to return an average prediction. The maximum k is 18 (this is similar to how an upper bound was calculated in [42]). The results are aggregated at the video level and averaged over the whole dataset.

Based on this case study, we can posit that by accurately discarding even a few bad clips - clips that are incorrectly classified with high confidence - we can improve the model's classification performance. By taking a closer look at Table 1.2 we see that as K increases, the Oracle's accuracy also decreases. In simple terms, this means that some videos are more challenging than others, where a classifier is only correctly classifying a subset of clips out of all clips for a full video. Furthermore, in terms

of computational efficiency, the detection of these clips should be less costly than densely applying the classifier to all clips.

In order to address the two challenges, we propose a confidence distillation framework. We propose a loss function to train a light-weight classifier using knowledge distillation that also learns to predict which clips will be ambiguous for the teacher. In other words, the student learns to output a score $\tilde{z}$ for its confidence in the teacher's correct classification of that specific input. By delegating the sampling task to the student, we will only perform prediction on the best clips using the teacher model, which leads to a reduction of computational cost. Furthermore, using this loss also allows us to devise simple scheme to divide the prediction task between the student and the teacher, depending on how difficult a clip is to classify. We show in experimental results that using confidence distillation, the accuracy of a given clip-based action recognition can be increased by up to 30% compared to other baselines and up to 5% compared to expensive dense sampling; the mean time per video also decreases by up to 30-70%. By dividing the classification task between the student and the teacher, this reduction in video processing time can be further increased to 35-80% at a negligible cost to accuracy. An overview of this framework is given in Figure 3.1.

1.3 Contributions

In this thesis, we make the following contributions:

1. We propose a confidence distillation framework that can be applied to any ReLU based CNN. We take the "imperfectness" of the teacher in the distillation into account using a custom loss function to learn to classify and estimate confidence

jointly.

2. We propose a distillation loss function to train a classifier that also learns ambiguity of the samples for the teacher. Using this loss also allows us to devise a simple scheme to divide the prediction workload. More specifically, in this loss function, the teacher would dynamically adjust the amount of information it gives to the student based on 1) its confidence and 2) the student's confidence. Contrary to the previous work, we directly tackle learning clip ambiguity instead of using clip redundancy as a proxy.

3. We show that confidence distillation outperforms other baselines through rigorous experiments. It performs up to 30% more accurately than both naive approaches (such as equidistant sampling), and more sophisticated approaches such as model-based sampling [43, 42, 44, 45]. Furthermore, it also outperforms dense sampling in both long and short videos, not explored in previous work.

1.4 Organization of this Thesis

The rest of this thesis is organized as follows: The second chapter will be focused on the related work. The third chapter will present a formulation of the problem and the solution. The fourth will discuss the experimental settings, implementation details and hyperparameter choices, with a final focus on ablation studies. The fifth chapter will be dedicated to discussion of the results, conclusion and the directions for future work.

Chapter 2

Related Work

A bird's eye view of the problem of action recognition/video classification was given in the previous chapter. This chapter will explain the relevant methods in more detail as well as other work that is not related to action recognition but could be used in this setting to solve this problem. More specifically, there are three categories of recent work that are relevant: (1) Works related to video classification and action recognition; (2) works on efficient video analysis and (3) works that inspire our approach in solving this problem, namely a succinct overview of model compression, action localization and uncertainty in deep learning.

2.1 Video Classification

The task of video classification is to map an input video to a set of video classes. This problem has been widely investigated in action recognition, where each video would be mapped to a set of action classes; or, each video can have either one class at the video-level or multiple classes at the timeframe-level. In this study, we focus on the

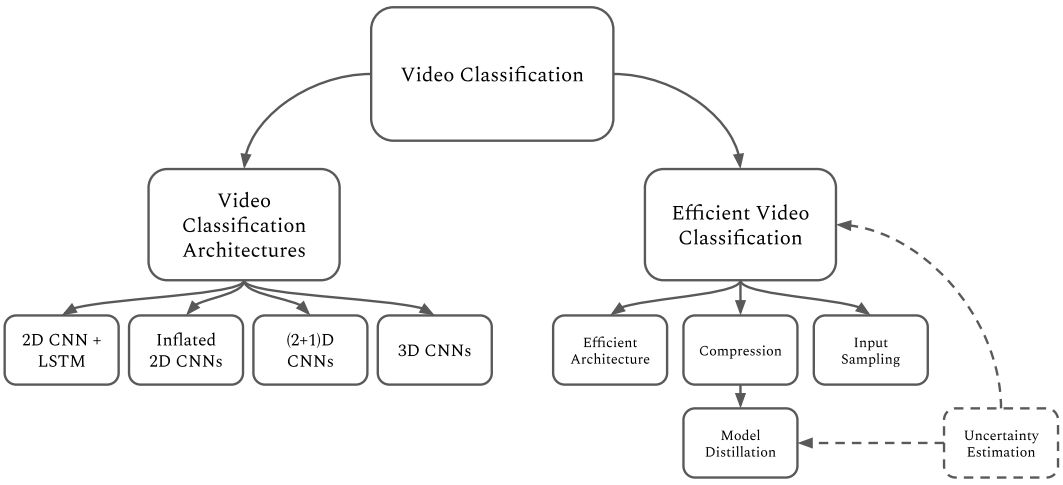


Figure 2.1: A taxonomy of related work.

former.

Recent architectures for accurate action recognition are mostly designed by extending image classifiers with a temporal dimension while preserving the spatial properties of each frame. Among them include directly transforming 2D models [46, 47, 7, 48] such as Inception or ResNet to 3D [40, 17], adding RNNs on top of 2D CNNs [25, 49, 50, 51, 52, 53], using two identical 2D CNNs or using more sophisticated volume based convolutions [34, 54, 55, 30, 31, 32]. A taxonomy of related work is given in Figure 2.1.

2.1.1 2D Convolutional Networks

Convolutions are standard operations in convolutional neural networks. Unlike the convolution operation used in signal processing, the convolution operation used in

deep learning has more resemblance to the cross-correlation operation. Suppose we have an $N \times N$ image $\mathbf{I}$ and a convolution filter F with size $f \times f$. The resulting convolution Z from the image and the filter would be:

$$Z = F * N$$

$$Z[i,j] = \sum_{u=1}^{f} \sum_{v=1}^{f} F[u,v] I[i+u, j+v]$$

More simply, a convolution operation slides a filter across the input incrementally, adding the results into each cell of Z at each step. In CNNs, this Z is fed into a non-linearity function like ReLU to calculate the activations A of the current layer, fed into later layers as inputs. The result of a single convolution filter is an *area*. Typically images have multiple channels, and there are multiple convolutional filters per layer whose weights are learned through back-propagation.

Recurrent 2D Networks

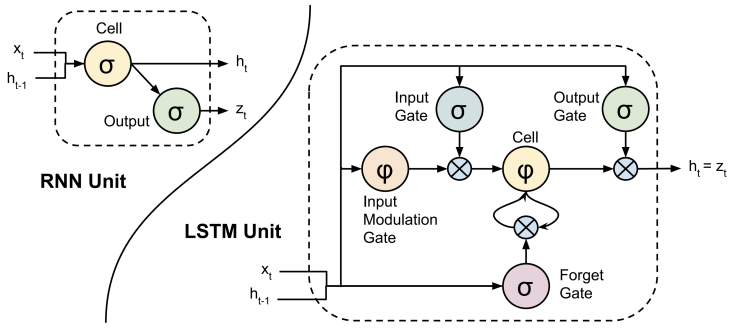


Figure 2.2: RNN and LSTM cells.

To model the temporal relations in a video, one approach is the Recurrent 2D CNN architecture. In Figure 2.2 an overview of the main building blocks of RNNs and

LSTMs are shown. These so-called hidden cells model complex temporal dynamics by mapping input sequences (namely, frames of a video) to a sequence of hidden states and then from hidden states to outputs. LSTMs [5] were designed to deal with the vanishing / exploding gradient problem of RNNs through memory cells.

Models that incorporate this structure [25, 51, 53, 52, 49, 50] for action recognition use the aforementioned building blocks for temporal modelling. For instance, in Figure 2.3, the building architecture of [25] is shown as an example.

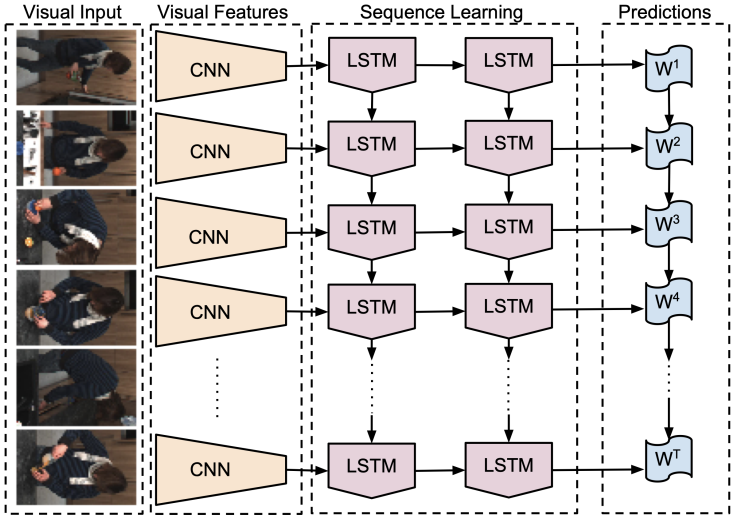


Figure 2.3: A 2D CNN + LSTM for Action Recognition.

Here all frames are passed through a 2D CNN backbone [7, 47, 46, 48] to produce a low-level representation. These representations are then fed as features into an LSTM architecture, and the prediction scores are aggregated in the end through an aggregation function.

Two-stream 2D Networks

A further extension of the above idea would be to incorporate motion information. Typically this is achieved through designing a two-branch *2D CNN* where one branch takes optical flow (which is a handcrafted feature) as input [26, 27, 40, 56] and the other branch is fed RGB frames. In Figure 2.4 the two-stream architecture of [26] is shown.

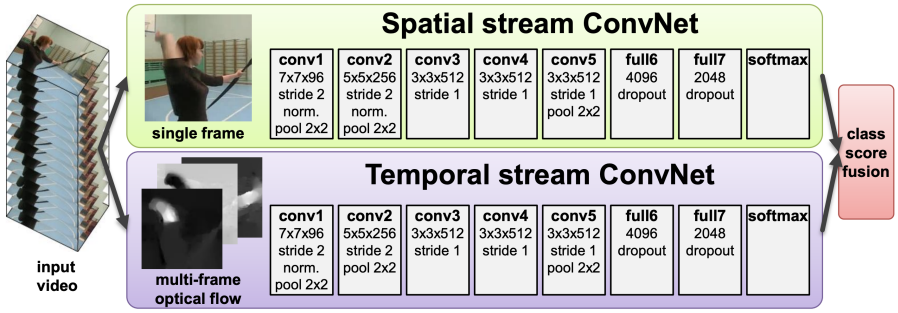


Figure 2.4: The two-stream image and flow architecture.

Here frames for the spatial stream are sampled according to a sampling policy (for instance, randomly), while the temporal stream corresponds to the optical flow over a fixed number of adjacent frames. One issue with this design is from a methodological standpoint, optical flow is a hand-crafted feature, which is undesirable in an end-to-end optimized setting.

2.1.2 3D Convolutional Networks

Tough promising, recurrent 2D CNNs and two-stream CNNs are shown to be inferior in terms of learning capacity at spatio-temporal modelling compared to 3D CNNs [17]; 3D CNNs can outperform their 2D counter-parts as more data becomes available and overfitting is less of an issue.

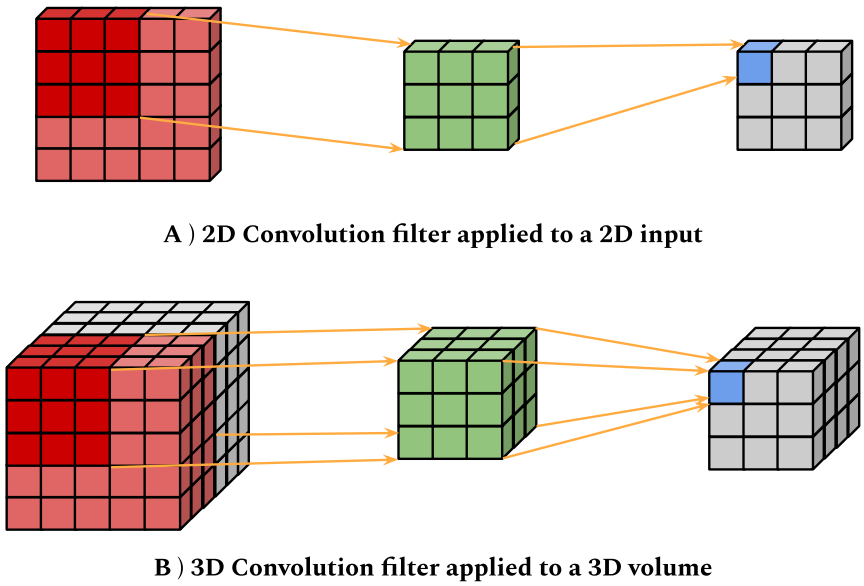


Figure 2.5: 2D Convolutions vs. 3D Convolutions.

As seen in Figure 2.5, the main building block of 3D spatio-temporal neural networks is the 3D convolutional filter. The 3D convolution is an extension of 2D convolutions across time, i.e. instead of a single image, the input to a 3D convolutional network will be a volume of many images.

Inflation and Direct Transformation

This simple transformation allows many powerful 2D CNNs [7, 47, 46, 48] to be simply extended into a 3D one [40, 17, 54, 55, 30]. Although they provide superior spatio-temporal modelling capabilities, before larger datasets such as Kinetics were available, these models were prone to overfitting since a large enough dataset was not available for pre-training.

Instead of training a 3D CNN from scratch, it is possible to pre-train a 2D CNN [7, 47, 46, 48] on an image-based dataset and inflate that CNN into a 3D one [34, 40].

In the I3D model, [40], a given pre-trained 2D architecture is inflated - endowed with an additional temporal dimension - into a 3D model across all convolutional and pooling layers, adding another dimension to enable spatio-temporal modelling on input volumes with convolutional filters. For example, a given $N \times N$ filter would be inflated to $N \times N \times N$.

A downside to this approach is that it makes video architectures inherently biased towards their image-based counterparts.

(2+1)D Convolutional Networks

In R(2+1)D [32], a 3D convolution block is decomposed into a 2D spatial convolution followed by 1D temporal convolution, named (2+1)D convolution which is also closely related to Psuedo-3D networks [54].

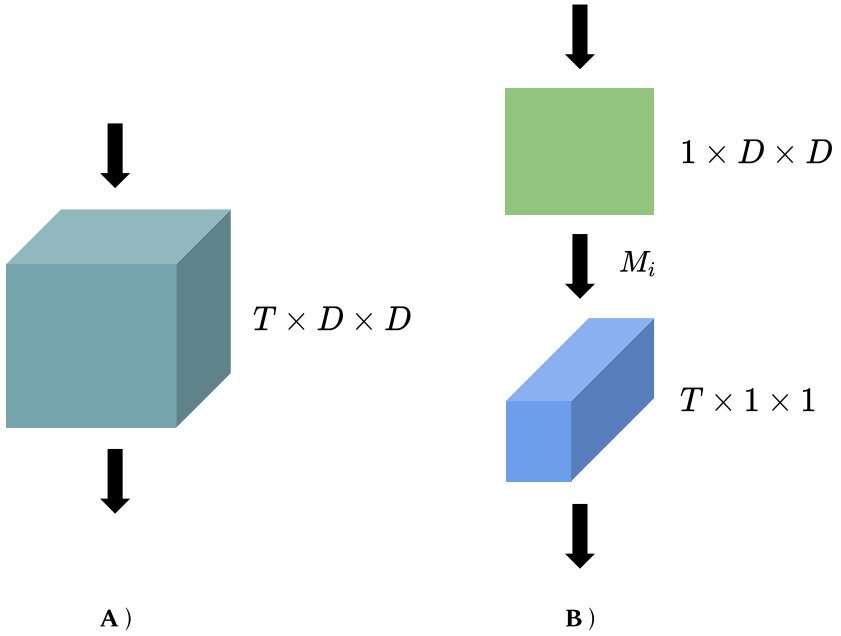


Figure 2.6: The (2+1)D Convolution. A) 3D convolution, B) (2+1)D convolution

In Figure 2.6, an illustration of a simplified setting is given where the input consists of a single feature channel of a spatio-temporal volume. A full 3D convolution (A) is carried out using a filter with shape $T \times D \times D$, where T and D denote temporal and spatial extents respectively. A (2+1)D convolutional block decomposes the computation into a spatial 2D convolution followed by a temporal 1D convolution. The number of 2D filters M_i here is chosen so that the number of parameters the (2+1)D block matches that of the full 3D convolutional block.

Other 3D Convolutional Networks

In SlowFast [35], instead of processing one input volume of videos, an input clip is decomposed into a low frame rate, low temporal resolution Slow pathway and a high frame rate and higher temporal resolution Fast pathway. The Fast pathway uses fewer channels, and the two pathways are fused using lateral connections. In [36] whether the slow pathway is needed given a competitive fast pathway is investigated.

2.2 Efficient Video Classification

2.2.1 Efficient Architectures

There have also been many innovative architectures proposed for efficient video classification [39, 57, 38, 58, 59, 60, 31, 61, 62, 63, 64, 65, 66, 67, 68, 36]. Here the idea is to optimize the architecture itself, while our framework is mostly architecture-independent. For instance, in [36], a given 2D architecture is extended to an efficient spatio-temporal one through a step-wise expansion approach over the key variables such as temporal duration, frame rate, spatial resolution or network width. Similarly,

efficient architectures [69, 70] using depth-wise and channel-wise separable convolutions have recently been applied [39, 31] to video processing. In [29], an efficient temporal shift module (TSM) is introduced to extend a ResNet to capture temporal information using memory shifting operations. These approaches are orthogonal to ours and can be applied in conjunction with the proposed method.

2.2.2 Input Sampling

There is also a line of active research on adaptive frame sampling techniques [71, 42, 72, 73, 74, 75, 44] to reduce the computational cost by not processing redundant parts of the video.

At the clip-level, SCSampler [42] uses compressed features with *softmax confidence* scores to score clips for their visual sampler and selects the top k clips. Similarly, in IMGAUD2VID [44] multiple modalities such as audio and video are used to select less redundant frames. Other state-of-the-art frame selection methods include reinforcement learning based approaches such as AdaFrame [74] which uses a single agent with a global memory, and [73] that uses multiple agents to perform frame selection collaboratively. By contrast, our method requires neither a complex RL policy gradient nor access to audio level features. Furthermore, by focusing on redundancy, the effectiveness of these models diminishes as the size of the dataset becomes shorter and there are less irrelevant sections in the video.

Additionally, many input sampling methods do not explicitly consider the classifier's uncertainty in sampling decisions. For instance, using the *softmax confidence* scores is not a good surrogate for classifier confidence. When deployed in the real world, machine learning models [76, 77] and ReLU networks specifically [78] often fail

"silently" by providing high confidence predictions (giving a high probability to one class) while being lamentably incorrect.

This is in fact the main motivation for designing the confidence distillation framework. We make no such assumption that the highest probability score in the output of the network indicates its confidence. Furthermore instead of complex policy gradients or the need of access to multiple modalities, we sample the same input to the network using the confidence distillation framework.

2.2.3 Other Related Work

Among other related work, adaptive computation aims to achieve decent recognition accuracy while accommodating varying computational budgets. Among the earliest work to save computations are cascaded classifiers [79] that quickly reject easy negative windows for face recognition. Another way to reduce computations related to our work is knowledge distillation. Inspired by how humans learn, knowledge distillation [80, 81, 82] is an approach to address the problem of compressing large models into smaller models, achieving widespread success and recent efforts have been made into understanding why it performs better than learning from raw data [83, 84]. More specifically, in offline knowledge distillation, a pre-trained teacher trains the student with a smaller size and efficient basic operations. Since a compression problem is being addressed, typically, the student is deployed without the teacher. Here we employ both the student and the teacher to address the problem of efficient video analysis cooperatively.

This form of two-step filtering and analysis has been utilized before in the context of *Action Localization*. The goal of action localization is to designate the start and

end of an action in an untrimmed video and then recognize the action class [85, 86, 87, 88, 56]. Most approaches to this problem have a two-step design [86, 87, 88, 56]. First, an action segment proposal method identifies a large number of candidate segments. Subsequently, a sophisticated method validates each candidate and refines the temporal boundaries. Translated to our approach, a less sophisticated student model will designate candidate clips for the more sophisticated teacher model. A key difference between this analysis of action recognition models and action localization is that it is assumed that the class labels of each video belong to a single action class for action recognition. Furthermore, both steps in action localization mechanisms can be sophisticated and expensive. In some cases, sophisticated features like flow [88] are used that are at least as expensive as computing one forward pass through an expensive clip classifier. We aim to take high efficiency into account and, instead, identify a small number of candidate clips using an efficient classifier before invoking the expensive classifier.

Finally, a highly active and related area of research is the task of determining when a network should say *"I do not know."* or provide confidence and uncertainty estimates of a network's outputs. Work in this category includes Bayesian neural networks [89], variational inference [90, 91] and more stable and simpler methods such as Monte Carlo dropout [43] where measures such as *predictive entropy* are used. Some methods learn unsupervised, unbounded [92] or bounded confidence values [45] for classification and regression. In [43, 78], *Predictive entropy* is a measure applied to capture aleatoric uncertainty; the type of uncertainty inherent in the data such as an image of a number 7 that may appear similar to a number 1. In [45], a given untrained classifier is regularized during training to detect out-of-distribution samples without

supervision. We take a different approach by having the student learn confidence estimates of its teacher to maintain computational efficiency in the video analysis task.

Chapter 3

Approach

This chapter will focus on a formal definition of the problem and the learning objective of confidence distillation. Our goal is to perform accurate and efficient action recognition in videos. We will first formally define our problem in Section 3.1; we then introduce how to teach the student the notion of confidence - how confident it is that the teacher will not make a mistake - using the teacher's outputs and its alternatives; finally, we present how this confidence score will be used to skip over ambiguous clips in the video.

3.1 Problem Formulation

Given a video $\mathcal{V}$, the goal of video classification is to map $\mathcal{V} \in \mathbb{R}^{T \times 3 \times H \times W}$ of arbitrary length T into a predefined set of C classes.

Because of the length of $\mathcal{V}$ and memory constraints, it is often intractable to process and stack all the video frames together - which can be hundreds - into a single deep network. As such, a majority of current approaches [26, 30, 40, 32, 93, 35] first

train a clip classifier $\Omega(\cdot)$ to operate on a short fixed-length video clip $\mathbf{V} \in \mathbb{R}^{F\times 3\times H\times W}$ of F frames with spatial resolution $H \times W$, typically spanning less than a second. Then, given a sample video, these methods apply the clip-classifier to all N clips $\{\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_N\}$ where $N = \frac{T}{F}$ and padding is applied to the last clip when not divisible. The final video-level prediction is obtained by aggregating the clip-level predictions of all N clips. The aggregation is usually an average pooling operation, but there have been other sophisticated schemes devised.

As the length of $\mathcal{V}$ grows, so does the computational cost. This makes inference using these methods very costly. This can also result in poor prediction accuracy since every incorrect clip prediction across $\{\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_N\}$ will negatively affect the overall prediction for $\mathcal{V}$. Given a pre-trained clip classifier, our goal is to train an efficient classifier $\Phi(\cdot)$ that can predict whether $\Omega(\cdot)$ will misclassify a given clip $\mathbf{V}_i$.

First, to address the problem of efficiency, we choose *shufflenet-v2* as the backbone for $\Phi(\cdot)$. We specifically use the version with 3D convolutions for their superior performance in short time-frame spatio-temporal modelling, high speed, and modest learning capacity [39]. Next, we will address the learning objective of $\Phi(\cdot)$.

3.1.1 Learning Objective

We consider three learning objectives for $\Phi(\cdot)$ and experimentally compare them in later chapters.

Training the sampler separately

A baseline and naive choice would be to train the sampler $\Phi(\cdot)$ separately from the classifier $\Omega(\cdot)$ on the same training set $\mathcal{D}$ with one-hot labels. We have two options

to train a sampler separately from $\Omega(\cdot)$:

1. Training an efficient classifier $\Phi_{st-ent}(\cdot)$ and calculating the predictive entropy of the output as a substitute for confidence scores. Given a clip $\mathbf{V}$ as an input, $\Phi_{st-ent}(\mathbf{V})$ yields a vector of prediction logits, the softmax function is applied to this vector and finally, the predictive entropy of this vector is calculated as a psuedo confidence score. We call this variant of Φ as the ST loss (Separate Training) on entropy (ST-Ent).

2. Since no ground-truth values for confidence are available in this case, separately train an efficient classifier $\Phi_{st-conf}(\cdot)$ using the method in [45] to obtain unsupervised confidence estimates. Using this method, the output of the network will have two branches, one for classification and one for confidence estimation. Given a clip $\mathbf{V}$ as an input, $\Phi_{st-conf}(\mathbf{V})$ yields a vector of prediction logits and a confidence logit. Then the softmax function is applied to the prediction logits, and the confidence logit is passed through a sigmoid function to obtain a confidence score $\tilde{z} \in [0, 1]$. We call this variant of Φ as the ST loss (Separate Training) on confidence (ST-Conf).

In both cases Φ will be trained as an action classifier that performs sampling.

Distilling only the confidence

Intuitively, training $\Omega(\cdot)$ and $\Phi(\cdot)$ independently implies that without supervision signals from $\Omega(\cdot)$, there is no guarantee that the sampler and the classifier would agree on clip ambiguity. For us, only the clips $\Omega(\cdot)$ finds ambiguous are relevant.

Therefore, to transfer ambiguity supervision signals from $\Omega(\cdot)$ to $\Phi(\cdot)$, we design pseudo-ground-truth binary confidence labels on the training set $\mathcal{D}$ denoted z_i as

follows: Given that the ground-truth classification label for each clip V_i is y, we apply the softmax function to the prediction logits from $\Omega(V_i)$ to obtain the vector of class prediction probabilities $\mathbf{p}$. We then find which class c has the maximum prediction probability p_c in $\mathbf{p}$. The confidence score z_i for each clip V_i are then defined as:

$$z_i = \begin{cases} 1 & \text{if} \quad argmax_{c \in \{1,\ldots,C\}}(\mathbf{p}) = y \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

In other words, a clip has a confidence score of 0 when the pre-trained $\Omega(\cdot)$ misclassifies it. Then, the learning objective for $\Phi(\cdot)$ corresponds to a binary classification task on these confidence labels using binary cross-entropy as the loss function. It outputs a score between 0 and 1 for each clip that indicates whether the teacher is likely to classify the clip correctly. In practice, the positive samples outnumber the negative samples significantly; therefore we downsample the majority class during training. We call this variant of $\Phi(\cdot)$ the BCE-C loss (Binary cross entropy confidence).

Learning confidence scores as part of the classification task

Intuitively, when learning where one is likely to make mistakes, it would also help to learn about the context; here, the context is the video class where some clips belonging to harder classes can be more ambiguous than others. This prior could lead to sparser, and more informative predictions [94]. For instance, to classify activities between abseiling and hiking using binary labels, one would only know that a video belongs to the "abseiling" class, and no information is given about similarities to other classes; when the class distribution is softer - i.e. other than the highest probability

class, other classes also have non-zero probabilities in the categorical distribution - one can see that the video most likely belongs to the "abseiling" class, but it may also be "hiking" with a lower probability. In the context of learning when the teacher is likely to make mistakes, this dark knowledge [81] may be useful.

Inspired by multi-task learning [95], where a shared representation is learned for multiple different tasks to improve the overall accuracy, we contend that learning this shared representation would improve the accuracy of the sampler $\Phi(\cdot)$ for the task of detecting ambiguous clips.

In contrast to the loss variant introduced in Section 3.1.1, the idea behind teacher-student knowledge distillation is that the output softmax or "soft" probabilities of a trained *teacher* network contains a lot more information about a data point than just the class label. For example, if multiple classes are assigned high probabilities for a video clip, the clip may lie close to a decision boundary among those classes. Having the student mimic these probabilities can assimilate some of the teacher's knowledge, providing information beyond the one-hot training labels.

Concretely, given an input video clip $\mathbf{V}$ the teacher network $\Omega(\cdot)$ produces a vector of logits $\mathbf{s}^t(\mathbf{V})$:

$$\mathbf{s}^t(\mathbf{V}) = [s_1^t(\mathbf{V}), s_2^t(\mathbf{V}), \ldots, s_C^t(\mathbf{V})] \tag{3.2}$$

In order to produce "softened", non-peaky and more informative probability distributions from the logit vector $\mathbf{s}^t(\mathbf{V})$, temperature scaling is used alongside the softmax function [81, 41] to produce $\tilde{\mathbf{p}}^t(\mathbf{V})$:

$$\tilde{\mathbf{p}}^t(\mathbf{V}) = [\tilde{p}_1^t(\mathbf{V}), \tilde{p}_2^t(\mathbf{V}), \ldots, \tilde{p}_C^t(\mathbf{V})] \tag{3.3}$$

$$\tilde{p}_k^t(\mathbf{V}) = \frac{e^{s_k^t(\mathbf{V})/\tau}}{\sum_j e^{s_j^t(\mathbf{V})/\tau}} \tag{3.4}$$

where τ is the temperature hyperparameter. The student model $\Phi(\cdot)$ similarly produces a softened class probability distribution, $\tilde{\mathbf{p}}^s(\mathbf{V})$. The student also needs to learn a confidence score $\tilde{z}^s(\mathbf{V}) \in [0, 1]$ using the pseudo-ground-truth binary labels defined in Section 3.1.1. This presents an interesting optimization issue: When the student is learning to classify, we want the overall loss to decrease when it has correctly classified the input; but in cases where the confidence score of a video clip is 0, $\mathbf{p}^t(\mathbf{V})$ as provided by the teacher would be misleading. To mitigate this issue, we modify the loss function similar to [92, 96, 45], and guide the student by incorporating the confidence scores into the distillation loss.

Imagine a student learning in a History class; a teacher is right most of the time and seldom wrong. To learn from the teacher, the student must have confidence that the teacher is right. When the student is confident, learning takes place by mimicking. The teacher then proclaims that "Spartacus won the war against the Romans." Based on prior knowledge, the student may now think that the teacher may be wrong and ask for more information from the teacher; "How did Spartacus win?" the teacher would then be inclined to give more information to the student. Translated to the distillation mechanism, this is akin to increasing τ when the predicted confidence

is low. When $\tau \to \infty$, the pseudo-probability output classes would have a uniform distribution. When the $\tilde{z} = 0$, we would want $\tau \to \infty$ and when $\tilde{z} = 1$, $\tau \to T$ which is a hyperparameter.

Specifically, for a given video clip $\mathbf{V}$ we further modify the teacher's probability scores after applying the softened softmax function to $\mathbf{p}^t(\mathbf{V})$ as:

$$\hat{\mathbf{p}}^t(\mathbf{V}) = \begin{cases} \tilde{z} * \tilde{\mathbf{p}}^t(\mathbf{V}) + (1 - \tilde{z}) * \mathcal{U}([0,1]^C) & \text{if} \quad z(\mathbf{V}) = 1 \\ (1 - \tilde{z}) * \tilde{\mathbf{p}}^t(\mathbf{V}) + \tilde{z} * \mathcal{U}([0,1]^C) & \text{o.w.} \end{cases} \tag{3.5}$$

where $\mathcal{U}([0,1]^C)$ is the uniform distribution between 0 and 1 over the C classes (the same shape as $\tilde{\mathbf{p}}^{\mathbf{t}}$). In the overall loss function, we include a distillation loss $\mathcal{L}_{KD}$ to match the student and teacher outputs. Furthermore, to prevent a naive solution of $\tilde{z} = 0$ from being converged to as the training progresses, we add a binary cross-entropy loss $\mathcal{L}_{conf}$ over the ground-truth confidence labels. The overall is then given by:

$$\mathcal{L} \quad = \mathbf{z}\mathcal{L}_{KD}(\hat{\mathbf{p}}^t, \tilde{\mathbf{p}}^s) + (1 - \mathbf{z})\mathcal{L}_{KD}(\hat{\mathbf{p}}^t, \tilde{\mathbf{p}}^s) + \lambda\mathcal{L}_{conf} \tag{3.6}$$

wherein τ and λ are hyperparameters. In $\mathcal{L}_{KD}(\hat{\mathbf{p}}^t, \tilde{\mathbf{p}}^s)$ the similarity of the two pseudo-probability posterior distribution vectors for input $\mathbf{V}$ of the student and the teacher is measured using the KL divergence metric:

$$\mathcal{L}_{KD}(\hat{\mathbf{p}}^t, \tilde{\mathbf{p}}^s) = \tau^2[-\frac{1}{n}\sum_{i=1}^{n} \hat{p}_i^t(\mathbf{V})\mathbf{log}\frac{\tilde{p}_i^s(\mathbf{V})}{\hat{p}_i^t(\mathbf{V})}] \tag{3.7}$$

which is equal to the difference between the cross entropy of the labels $H(\hat{\mathbf{p}}^t, \tilde{\mathbf{p}}^s)$ and the empirical entropy $H(\hat{\mathbf{p}}^t)$:

$$\mathcal{L}_{KD}(\hat{\mathbf{p}}^t, \tilde{\mathbf{p}}^s) = H(\hat{\mathbf{p}}^t, \tilde{\mathbf{p}}^s) - H(\hat{\mathbf{p}}^t) \tag{3.8}$$

In essence, we include two KL distances. One to supervise learning under certainty and one for supervising learning under uncertainty. The two KL terms can be integrated together to update the network parameters using gradient descent optimization.

When the student is wrong about the teacher's confidence, she can ask for more information. When the teacher is wrong, and the student cannot tell, the teacher gives hints by providing more information. This creates an optimization problem where the most undesirable outcome is where the student-teacher confidences differ while the student learns from the teacher. We refer to this loss variant as ConDi-SR (Confidence distillation shared representation). A high-level overview of the framework is shown in Figure 3.1.

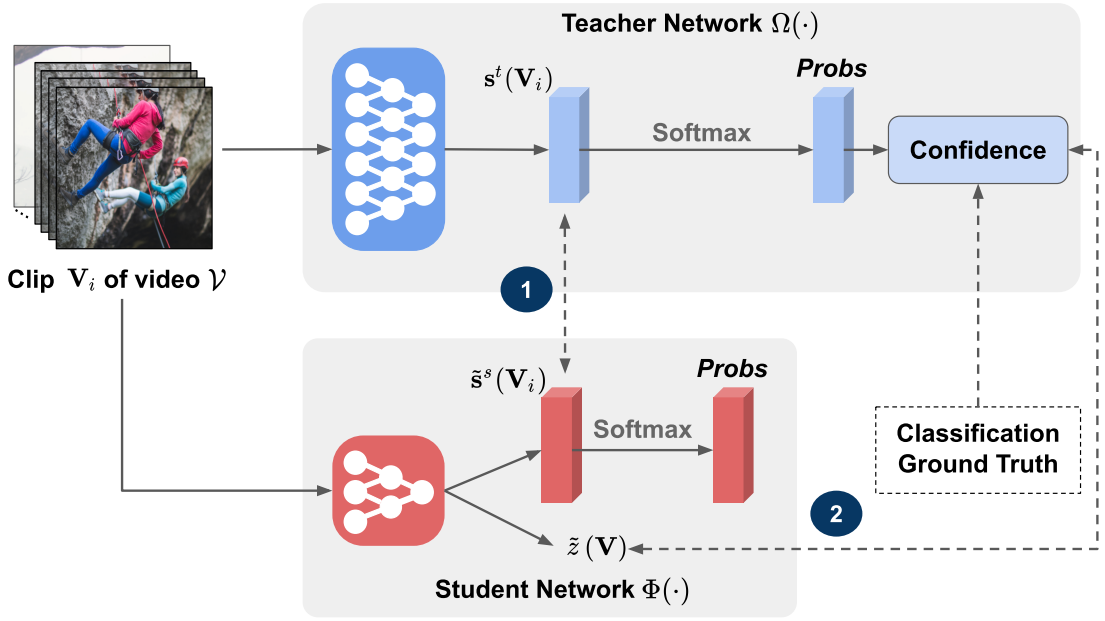


Figure 3.1: An overview of the distillation framework.

3.1.2 Inference

Sampling Algorithm

Using the afore-mentioned loss variants, we can train a sampler that outputs a $\tilde{z}$ sampling score. *ST-Ent* outputs an entropy score, while *ST-Conf*, *BCE-Conf* and *ConDi-SR* all output a confidence score $\tilde{z} \in [0, 1]$ for each clip. After all the clips in a video are scored by the sampler, the outputs are sorted to produce a ranked list in descending order. The top K clips in the list will then be considered for classification. A high-level overview of this algorithm is shown in Figure 3.2.

The yellow student prediction branch in Figure 3.2 indicates that this branch is only used during training to provide supervision signals during backpropagation. It can be safely removed during inference.

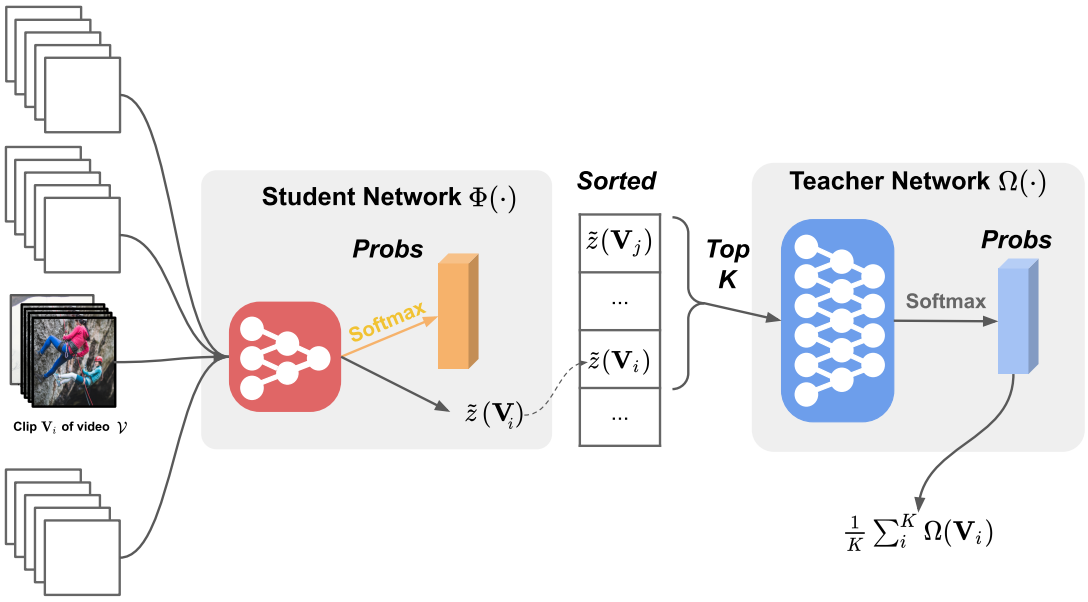


Figure 3.2: Inference using only the teacher's best clips.

Division of Inference Task

In the case of *ST-Ent*, *BCE-Conf* and *ST-Conf* losses we feed all top K clips to Ω to be classified. However, in the case of *ConDi-SR*, we have more options. Since a shared representation is being learned for both classification and confidence estimation in these tasks, ϕ can be utilized to also classify identified good clips alongside Ω, dividing some of the tasks and providing additional computation savings. In this case, two ranked lists would be generated, one representing the uncertainty of the teacher (or in other words, the confidence of the student that the teacher will be right) and one representing the uncertainty of the student, where we can use the predictive entropy of the student's classification output.

To combine these two ranked lists, we use a hyperparameter K_s to denote the number of top clips to be predicted by the student and feed the rest $K_t = K - K_s$ to

the teacher.

Note that while *ST-Conf* learns a shared representation of confidence and classification, it differs from ConDi-SR. In *ST-Conf*, the confidence scores represent the *confidence of the model itself* while in *ConDi-SR* the confidence scores represent the *confidence of the student regarding incorrect classification by the teacher*. This allows *ConDi-SR* to be able to produce two meaningful ranked lists. A high-level overview of this algorithm is shown in Figure 3.3.

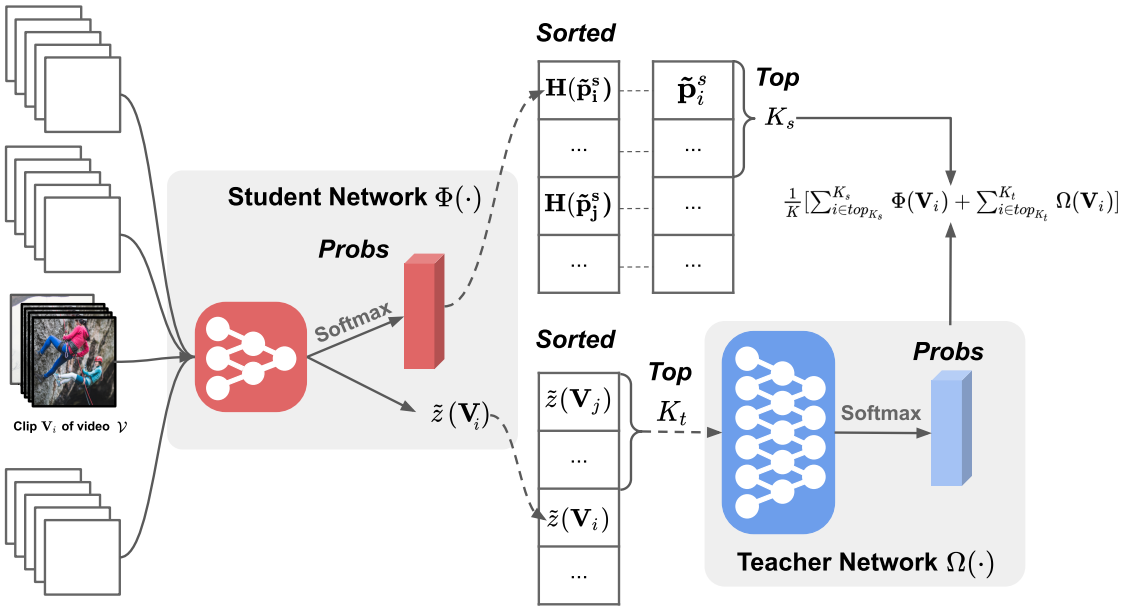


Figure 3.3: Inference by combining the top K_s clips from the student and the top K_t clips from the teacher.

In Figure 3.3, the predictive entropy of the student's predictions is calculated over the video clips of video $\mathcal{V}$. These are then sorted, and a reference to the original prediction vector for each $\mathbf{H}(\mathbf{p}_i^s)$ is kept. Finally, only the top-K_s prediction vectors are used based on their predictive entropy from the student, while the top-K_t prediction vectors from the teacher are used. When the lists contain overlapping clips,

the score for those clips is averaged between the student and the teacher. The overall prediction vector for video $\mathcal{V}$ is calculated using an averaging aggregation operator.

Chapter 4

Experiments

In the experiments, we consider the following: (1) Action recognition datasets that encompass different types of learning challenges. (2) State-of-the-art 3D CNN based action recognition architectures; and (3) Sampling baselines such as random sampling and more sophisticated sampling techniques in literature.

4.1 Datasets

We consider the following datasets: (1) the Kinetics dataset [15, 97, 16], a video recognition dataset generally used to determine a model's capacity to learn. The videos in this dataset are less than 10 seconds long, too short for many current sampling approaches to be directly applied successfully; (2) the UCF-101 dataset [12], one of the most widely used benchmark datasets; and (3) the Something-Something V2 dataset [23], a dataset containing short but complex actions for spatio-temporal reasoning in videos.

In Table 4.5 an overview of all the aforementioned datasets is given.

Table 4.1: Experiment datasets.

Dataset	Instances	Average Length	Classes
UCF-101	13320	7	101
Something-Something V2	220847	2-8	174
Kinetics	Varies	10	400-700

4.2 Models

We consider two baseline models for the teacher: 3D residual networks (3D-ResNeXt) with 101 layers from [33]. We also consider a spatio-temporal network with 2+1D convolutions [32]. For the student architecture we use a 3D ShuffleNetV2 [39]. Details for each model are given in Table 4.2 below.

Table 4.2: Network baselines.

		GFLOPs	Params	Depth	Two-Stream
Baseline Model	3D ResNeXt-101	6.932	48.34M	101	No
	R(2+1)D	6.321	38.76M	34	No
Sampler Model	3D ShuffleNet V2	0.360	6.64M*	26	No

Both teacher models are based on some form of spatio-temporal convolutions. While this dataset is more suited for dense sampling, we show that the accuracy of two state-of-the-art models can be improved compared to dense sampling by using confidence distillation.

4.3 Sampling Baselines

For sampling baselines we consider a few simple and more complex options. All samplers have a hyperparameter K that determines how many clips are sampled per video.

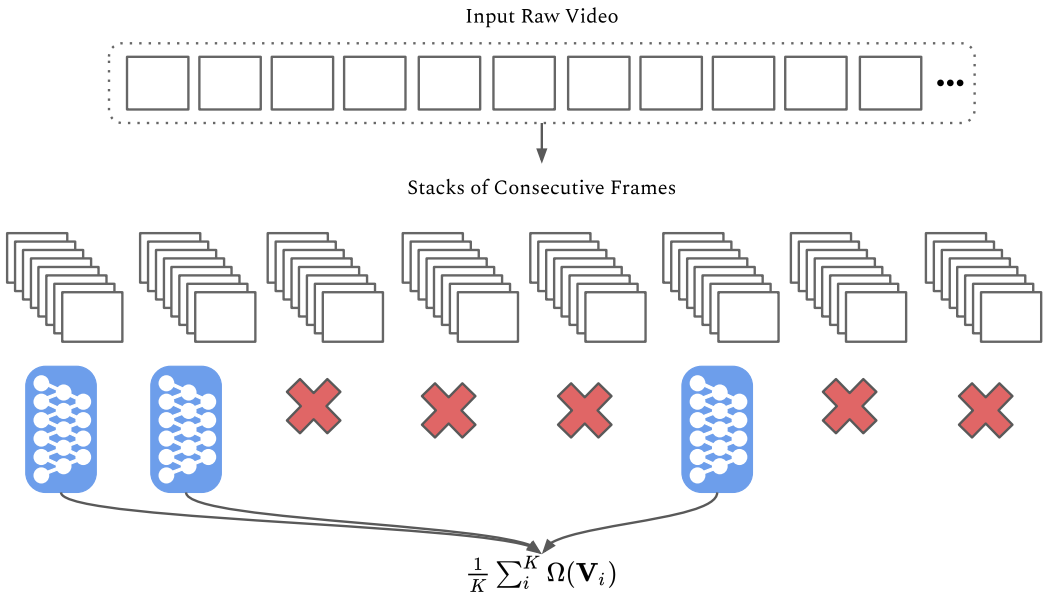


Figure 4.1: Random clip sampling.

Random sampling

Random sampling chooses K clips from a video randomly. The results over the datasets are shown after 3 random passes over the datasets with different seeds. An illustration of random clip sampling is shown in Figure 4.1.

Equidistant sampling

Equidistant sampling chooses K clips at equidistant time intervals. This is similar to i-Frames in compression. An illustration of equidistant sampling is shown in Figure 4.2.

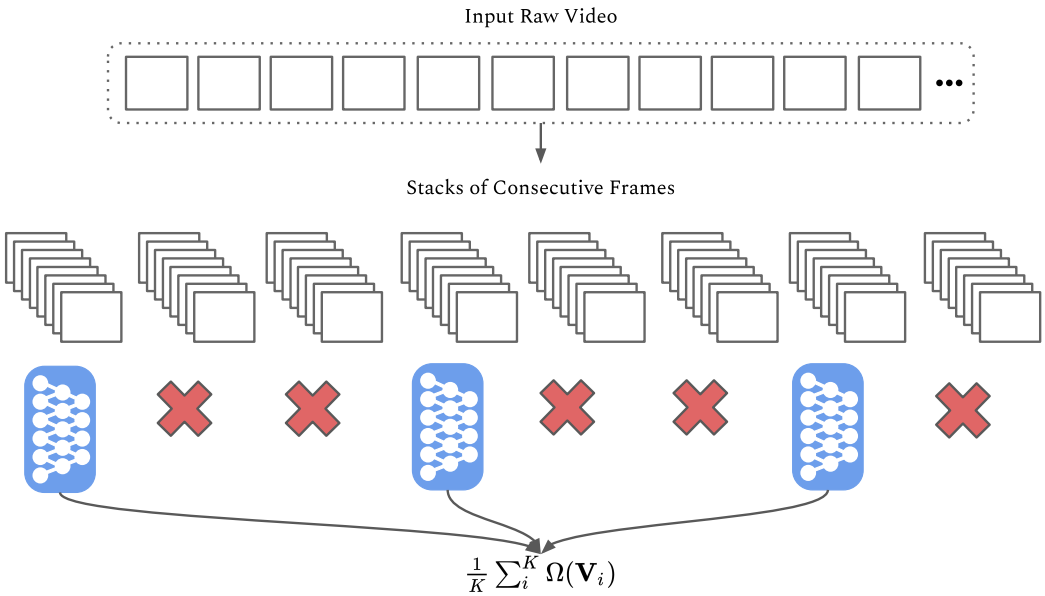


Figure 4.2: Equidistant clip sampling.

Predictive entropy sampling

The entropy [98] of a random variable captures the average amount of information contained in the possible outcomes of the variable. The higher the entropy, the more information the associated outcome has. Given an input V, the predictive entropy of a model trained on dataset $\mathcal{D}_{\textbf{train}}$ is:

$$H(y|\mathbf{V},\mathcal{D}_{\textbf{train}}) = -\sum_{c} p(y=c|\mathbf{V},\mathcal{D}_{\textbf{train}})\textbf{log}p(y=c|\mathbf{V},\mathcal{D}_{\textbf{train}}) \tag{4.1}$$

where the summation is done over all the classes that y can take. The maximum predictive entropy is reached with the uniform distribution and the minimum predictive entropy is reached with one-hot probabilities. This measure was used in [43, 78] to capture aleatoric uncertainty reliably; this uncertainty is the type of uncertainty

inherent in the data, such as an image of a number 7 that may appear similar to a number 1.

To sample with an entropy score, we train the sampler variant from Section 3.1.1 without the confidence branch using a soft-label distillation loss [81]. An illustration of a sampler model is shown in Figure 4.3.

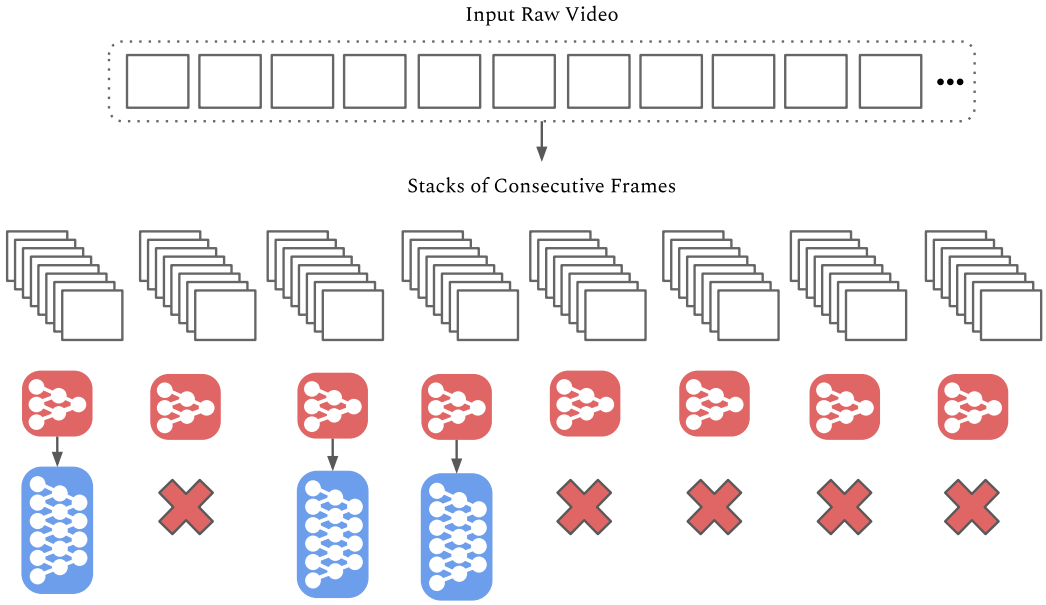


Figure 4.3: Model based clip sampling.

Other model based sampling techniques

Sampling based on predictive entropy requires a trained model. Similarly, other methods train a model to do the video clip or frame sampling [42, 44, 74] which are complementary to our method although they solve a similar problem of sampling a subset of the video frames for classification by the expensive $\Omega(\cdot)$ classifier. For instance, in all methods, it is assumed that the video is very long and the number of sampled clips can be large. Since we don't make an assumption on video length,

[44, 42, 74] can be applied to a long video to reduce the search space and sample a subset of the video frames, and confidence distillation can be used to choose a subset of sampled frames. This forms a stack of samplers. One drawback is that these methods are redundancy based and do not apply to all the datasets, specifically those with shorter videos. We will be comparing the reported results directly where a result has been reported on the target dataset by the original authors.

Oracle Sampler

To calculate an upper bound for classification scores, we construct an *Oracle* sampler similar to [42] which was previously referred to in Chapter 1. *Oracle* cheats by looking at the ground-truth label y and only considers clips that yield the k highest classification scores for y to return an average prediction.

4.3.1 Implementation Details

We have implemented the models using the PyTorch package. Nvidia P100 GPUs have been used for testing the models. Both the teacher and student take stacks of $16 \times 3 \times 112 \times 112$ clip volumes. For distillation hyperparameters, we use $\tau \in \{0.9, 1.0, 2.0\}$. For confidence, we use $\lambda = \{0.5, 1.5, 2.0\}$ and $K \in \{1, 3, 5, 7, 10\}$ where applicable. In knowledge distillation literature, τ is typically chosen to be a large number like 5 or 20 in the beginning and may be annealed. In our case, the reasoning behind using smaller τ values is that the teacher's entropy (information) will be dynamically adjusted depending on the student's confidence. As such, it is intuitive to start with smaller values as the baseline τ.

For other hyperparameters, we start with a learning rate $\alpha = 1e-2$ on UCF-101

and Something-Something V2 datasets. We use a learning rate of $\alpha = 1e-1$ for training on the Kinetics dataset. The learning rate is reduced by a factor of 10 every 15 epochs. The confidence branch's learning rate starts the same as the rest of the network but is reduced by a factor of 10 every epoch after the first epoch.

The models trained on Kinetics are trained from scratch without pre-training similar to [33]. The models trained on UCF-101 and Something-Something V2 are pre-trained on Kinetics and then fine-tuned on the target datasets.

For classification metrics, we consider the Top-1 accuracy, denoting the true class matches with the most probable class predicted by the model.

Class Imbalance

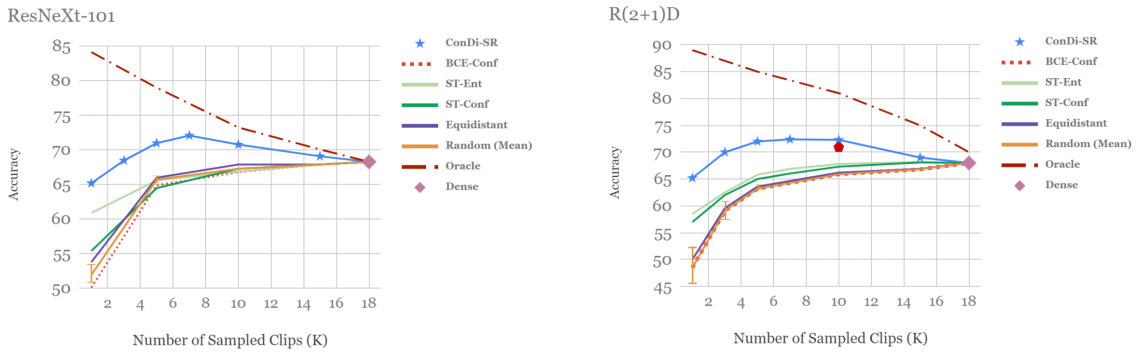
As discussed in 3.1.1, binary confidence labels were generated for the teacher and *Binary Cross Entropy* is used as a regularizer on these labels during training to prevent trivial solutions by the optimizer. The teacher is right more often than it is wrong, and thus there are many more positive classes than negative ones. However, we find that for our case, false positives in the predicted confidence score (in other words, a bad clip being considered a good clip) would always negatively affect accuracy. Therefore during training, we want to minimize the number of false positives predicted by the student. To guide the optimizer in the correct direction, a higher weight μ is placed on the positive component of $\mathcal{L}_{conf}$. Here we experiment with $\mu \in 1.0, 1.5, 2$.

4.4 Experimental Results

4.4.1 Classification Accuracy

Kinetics

The Kinetics dataset is a widely used, large dataset for action recognition. The dataset is big enough to train models from scratch. However, there are different variants, and a percentage of video samples may not be available or accessible. Nevertheless, we train a ShuffleNet-V2 based sampler on this dataset and report the results. Please note that due to the lack of available models, the ResNeXt-101 is trained on the Kinetics-600 dataset, and the R(2+1)D is trained on the Kinetics-400 dataset as teacher models. Testing is performed on the same dataset.



(a) Results on the Kinetics-600 dataset with ResNeXt-101.

(b) Results on the Kinetics-400 dataset with R(2+1)D-34

Figure 4.4: Results for the two teachers on the kinetics dataset. Yellow error bars are shown for random sampling.

Yellow error bars in Figure 4.4 show values within a 1 standard deviation interval for the 10 passes of random sampling. From Figure 4.4, we see that *ConDi-SR* and *ST-Ent* perform the best while the rest of the baselines achieve poorer but comparable results amongst themselves. As K increases, all methods converge to the same

point representing dense sampling. The oracle has a steep curve as K increases. This suggests that there exists label noise in the dataset [43]. Additionally, we see that confidence distillation has up to 10% improvement compared to other baselines, achieving the highest accuracy at $K = 7$. In Table 4.3 more detailed results at this K is shown.

Table 4.3: Detailed results for $K = 7$ for Kinetics.

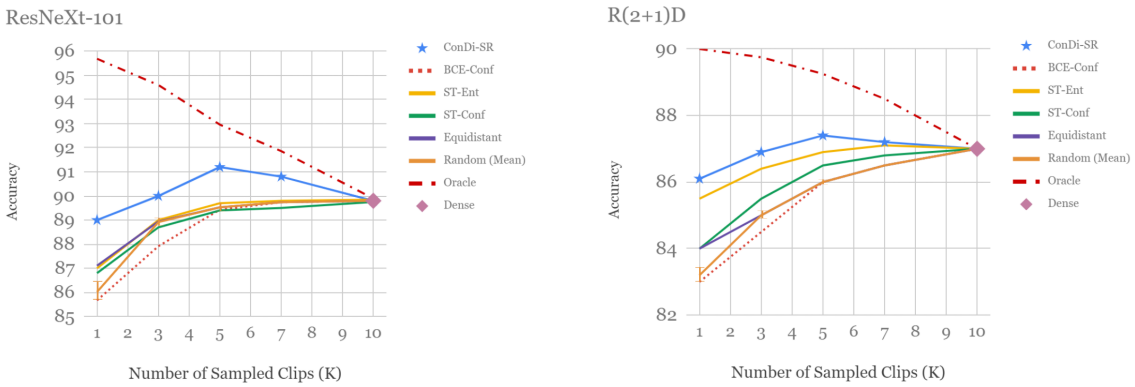
Model	Random (Mean)	Equidistant	ConDi-SR	ST-Conf	ST-Ent	Dense	Oracle
ResNeXt-101	66.7	67	**72.1**	66.1	67.4	68.3	76.14
R(2+1)D-34	63.4	64.03	**72.4**	66.02	68.1	68	84.3

In [42], an accuracy of 70.9% has been achieved on the same task with the R(2+1)D model. Here we can achieve a higher accuracy of 72.4% using confidence distillation without the need for utilizing multiple modalities.

UCF-101

Recall the statistics of the UCF-101 dataset in Table 4.5. The challenge with this dataset is two-fold. Firstly the size of the dataset is too small to allow model training from scratch, and therefore the student and teacher models need to be pre-trained on the Kinetics dataset (here, we use Kinetics-600 for ResNeXt and Kinetics-400 for R(2+1)D). secondly, the videos are too short for sampling methods such as SCSampler [42] or IMGAUD2VID [44] to work effectively. In Figure 4.5, the performance of applicable baselines is shown. The results shown are the mean values of test results over the three official validation folds of the UCF-101 dataset for reproducibility.

In Figure 4.5, error bar lines have also been drawn for random sampling similar to Figure 4.4. As shown in Figure 4.5, confidence distillation consistently outperforms



(a) With ResNeXt-101 as the teacher.

(b) With R(2+1)D-34 as the teacher

Figure 4.5: Results for the two teachers on the UCF-101 dataset. Yellow error bars are shown for random sampling.

all baselines. We can also see an "elbow" shape appear at an optimal K. As we increase K to the maximum for the dataset, all methods converge to *dense* sampling. We see that the Oracle has a close to linear steepness, suggesting less label noise than the kinetics dataset. Furthermore, we see that the *BCE-Conf* loss variant performs close to or worse than random. This may be a result of a lack of supervision signals in the labels used for that loss. Finally we also see that equidistant sampling performs comparably to a naive sampler with *ST-Ent* and *ST-Conf* losses.

In Table 4.4 more detailed results are given for $K = 5$.

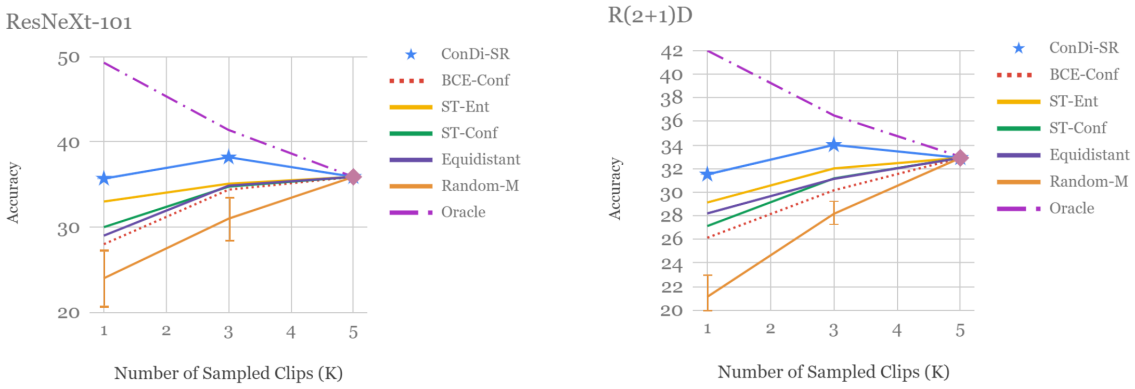
Table 4.4: Detailed results for $K = 5$ for UCF-101.

Model	Random (Mean)	Equidistant	ConDi-SR	ST-Conf	ST-Ent	Dense	Oracle
ResNeXt-101	89.53	89.53	**91.1**	89.5	89.7	89.74	92.96
R(2+1)D-34	86.07	86.12	**87.4**	87.01	87.08	87	89.25

While both models are based on spatio-temporal convolutions and the dataset is more suited for dense sampling, we can improve the accuracy of both state-of-the-art models by $2-3\%$, averaged over the three official folds of the UCF-101 dataset.

Something-Something V2

This dataset is a collection of trimmed video clips that show humans performing pre-defined actions with everyday objects [8]. It allows for the development of models capturing a fine-grained understanding of basic actions. A characteristic of this dataset is that even though each video belongs to a single class, that class can be a combination of two other classes semantically. The dataset consists of 220847 videos of 174 classes, split into training, validation and test sets containing 168913, 24777 and 27157 videos, respectively. For the experiments, the validation set is used as the test set labels are not available. Similar to the previous dataset, We pre-train the models on the Kinetics dataset and perform fine-tuning using the training data of the Something-something V2 dataset.



(a) With ResNeXt-101 as the teacher.

(b) With R(2+1)D-34 as the teacher

Figure 4.6: Results for the two teachers on the Something-something V2 dataset. Yellow error bars are shown for random sampling.

Figure 4.6 shows the task is harder and using confidence distillation improves the results only marginally compared to dense sampling. We also see that the error bars for random sampling are larger, indicating that the key frames have a more

non-uniform distribution. The optimal $K = 3$.

Table 4.5: Detailed results for $K = 3$ for Something-Something V2.

Model	Random (Mean)	Equidistant	ConDi-SR	ST-Conf	ST-Ent	Dense	Oracle
ResNeXt-101	34.94	34.87	**38.4**	34.76	34.9	38	41.4
R(2+1)D-34	28.16	31.12	**34**	31.16	32.01	32.9	36.3

More detailed results are shown in Table 4.5. Here while a small improvement is made to dense sampling, the overall results indicate that more sophisticated spatio-temporal modelling is needed to model this dataset's correlations.

4.4.2 Computational Efficiency

As seen in Table 4.2, the implementation of the sampler's architecture uses at least an order of magnitude less floating-point operations than that of the teacher. Additionally, it has close to an order of magnitude fewer parameters. As we increase K, the overall computational gains are reduced. Table 4.6 shows average time/video on the UCF-101 and Kinetics datasets. We omit the Something-something V2 dataset from this comparison as the average video length is too small.

Here we can see that for even a small dataset such as UCF-101, we can improve the prediction accuracy and the computation time. When the dataset becomes larger, as in the case of kinetics, this improvement becomes more prominent.

4.4.3 Dividing the Prediction Workload

In Section 3.1.2, we devised a simple approach to divide the workload between the teacher and the student during inference. This is done by selecting a K_t and K_s for

Table 4.6: Computational Savings @ different K values for ResNeXt-101 Teacher model.

		UCF-101		Kinetics	
Method	K	Accuracy	Mean Time/Video(s)	Accuracy	Mean Time/Video(s)
ConDi-SR	1	89	1.24	65.2	1.31
	3	90.02	1.9	67.3	1.89
	5	**91.2**	**2.31**	69.4	2.28
	7	90.5	3.1	**71.5**	**3.05**
Random/Equidistant	1	86.02/87.11	0.29	1	0.29
	3	88.916/88.94	0.84	3	0.84
	5	89.54/89.54	1.42	5	1.42
	7	89.68/89.68	2.1	7	2.1
Dense	All	89.8	2.9	68.3	5.22

the teacher and student to infer respectively where $K_t + K_s = K$. We show the results of this division of labour in Table 4.7. We show the results for the Kinetics and UCF-101 datasets. We omit the Something-something V2 dataset from this comparison as the average video length is too small to show a significant difference.

Table 4.7: Dividing the workload between the teacher and student at different K_s values for the ResNeXt-101 teacher model.

Dataset (K)	K_s	Accuracy	Mean Time/Video (s)
UCF-101 (5)	0	91.2	2.31
	1	90.65	2.16
	3	90.105	1.96
	5	89.03	1.7
Kinetics (7)	0	71.5	3.05
	1	70.99	2.9
	4	69.51	2.6
	7	66.89	2.17

While this approach will lead to a further increase of computational efficiency at a small accuracy cost for the best K_s, it comes at an additional expense of time (sorting the $H(\mathbf{p}_i^s)$ scores) and space (keeping both the $H_{\mathbf{p}_i^s}$ and $\mathbf{p}_i^s$ vectors in memory).

We posit that the choice of dividing the workload will be application dependent. Intuitively, this design choice is more pronounced in cases where a large number of clips are being sampled and need to be classified (i.e. K is large). We similarly see in Table 4.7 that the decrease in mean processing time per video is more pronounced in the larger Kinetics dataset where K also tends to be larger for optimal values.

We can also compare the results to Table 4.6, where we see in some cases such as at $K_s = 1$ for Kinetics a 5% increase in speed is achieved at the cost of less than 1 percent in accuracy. However we see that the division is more of a design choice that is made possible through confidence distillation, while other approaches such as [42, 74, 44] would not have this option.

4.5 Ablation Studies

In this section we explore the choices made to design the loss function and hyperparameters.

4.5.1 Hyperparameters

We find through grid search the optimal choices of learning rate α, positive weight multiplier μ and loss parameter λ for *ConDi-SR* and finally the distillation temperature τ. Here we examine in-depth the rationales behind not using default parameters for each choice.

Positive Component Weight

For the choice of μ, we vary it from $\mu = 1$, the default value for most binary classification tasks to $\mu = 1.5$. The metric we compare is the false positive rate: $\frac{FP}{FP+TN}$. As discussed in Section 1.2, even a single false positive will hurt performance.

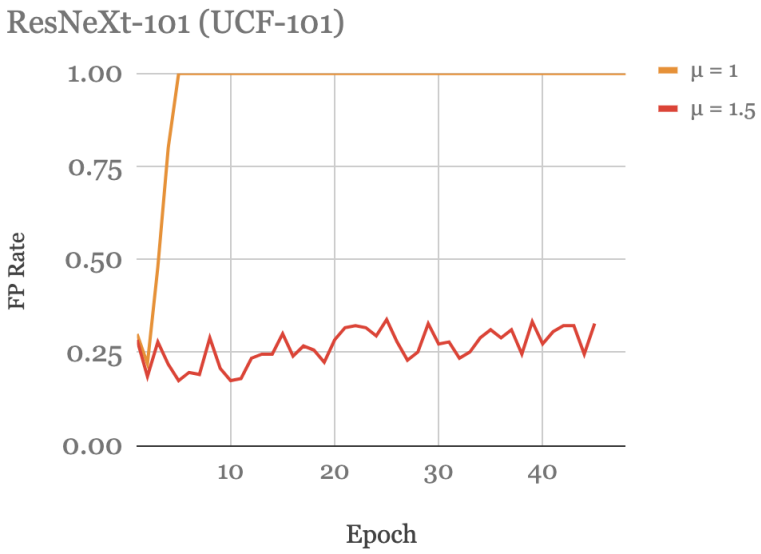


Figure 4.7: False positive rate for different μ values.

As shown in Figure 4.7, we see that by using the default weight of $\mu = 1$, the student will learn a trivial solution to confidence estimation. It will output uniform scores for all inputs, which will reduce the overall loss since there are many more positive examples. Since the worst outcome is false positives (i.e. a clip that will lead to misclassification but is determined otherwise), this trivial solution will reduce accuracy. With $\mu = 1.5$, this problem can be mitigated, and more acceptable false-positive rates are achieved over the validation set. We also find that using a higher weight for μ beyond 1.5 will prevent the distillation loss $\mathcal{L}_{KD}$ from decreasing at a

reasonable rate.

Learning Rate

Typically, for classification tasks, a network's learning rate $\alpha \in [1e-1, 1e-6]$ is used. We find that for fine-tuning tasks (UCF-101 and Something-Something v2), $\alpha = 0.01$ works best and for training from scratch (Kinetics) $\alpha = 0.1$ works better. This also holds when training the network with a confidence branch.

For the confidence score branch in *ConDi-SR*, we start with the same learning rate as the rest of the network; starting with a higher learning rate ($\alpha \geq 0.1$) encourages exploration of the loss landscape and escaping from sub-optimal local minimas. However, we find that the precision and recall over the validation set's binary confidence labels, diverges more quickly if the learning rate is dropped at the same rate as the learning rate of the classification task branch. Therefore we reduce the learning rate by a factor of 10 at every epoch but *only* for the confidence branch.

Temperature

In distillation literature [81] a value of $\tau > 1$ is typically used to encourage more information sharing between the student and the teacher by increasing the entropy of the soft output labels. However, this is offset by a classification loss, which is kept at $\tau = 1$. In confidence distillation we omit the classification loss and only use the $\mathcal{L}_{KD}$ component of soft label distillation. We indirectly increase τ and the entropy of the teacher towards infinity. The lower the student's confidence, the higher τ will be if she is right. As such, it is intuitive to start with a $\tau \leq 1$. We test with both $\tau = 1$ and $\tau = 0.9$. We find that there is not much difference between the two in terms of

the overall loss. However, when $\tau > 1$, the training fails to converge, indicating that by using $\tau \leq 1$ which results in peakier output distributions when the teacher and the student are both confident, we can provide better supervision signals in this loss. A more peaky output means that the distribution is closer to the one-hot ground-truth labels.

4.5.2 Loss Function Components

A naive approach to confidence distillation would be to use a loss function like below:

$$\mathcal{L} = \beta\mathcal{L}_{cls} + (1-\beta)\mathcal{L}_{KD} + \lambda\mathcal{L}_{conf} \tag{4.2}$$

Here the loss is simply a linear combination of distillation loss [81] with a cross-entropy regularizer to prevent trivial confidence scores. We discussed in Chapter 3 how this loss on its own would misdirect the student. In other words, the student will not be able to decrease its overall loss when it correctly predicts the teacher's confidence score. Furthermore, $\mathcal{L}_{cls}$ will allow the student to cheat, reducing its overall loss by learning to classify without learning a good representation of the confidence scores. To illustrate this, consider a case study over the UCF-101 dataset.

Table 4.8: Different assumptions about loss function design.

		ConDi-SR	Naive	Random
Method	K	Accuracy	Accuracy	Accuracy
ResNeXt-101	1	**89**	85.8	86.02
	3	**90.02**	88.93	88.92
	5	**91.2**	88.99	89.53
	7	**90.5**	89.65	89.64

For confidence distillation we use the optimal hyperparameters ($\tau = 0.9$, $\alpha = 0.01$,

$\mu = 1.5$), for the naive loss we use the same hyperparameters, except we vary $\tau = 5$ [81] and $\beta = 0.9$. We can see in Table 4.8 that the naive loss barely performs better than random sampling and even in some cases, hinders performance slightly. This is because it learns a trivial representation of confidence since there are not enough supervision signals to guide the student on discriminating between good and bad clips. As shown, We cannot expect a simple regularization term to do the job.

4.5.3 Summary

To summarise, in this chapter we showed that using confidence distillation, the efficiency and accuracy of classifying actions using 3D neural networks can be increased. We saw a 5% increase in accuracy as well as a 70% reduction in mean processing time spent per video compared to dense sampling. We also saw that this reduction in processing time can be increased to 80% when the workload is divided between the student and the teacher at a small cost of accuracy. As the dataset sizes become smaller, the efficiency and accuracy gains also become smaller as information becomes more compact. However, even in these cases and in datasets such as UCF-101, we can still gain 2% in accuracy and reduce mean processing time per video by 25%. Finally, we showed that the problem requires a non-trivial solution, and trivial solutions such as ST-Ent, ST-Conf or the loss function shown in Section 4.5.2, at best perform on par with random sampling.

Chapter 5

Conclusion and Future Work

In this thesis, we proposed a confidence distillation framework and demonstrated its effectiveness in improving the prediction accuracy and computational efficiency of 3D convolutional neural networks. We explored different choices of hyperparameters and their effects on performance. We empirically evaluated the effect of this proposed form of distillation on sampling for video recognition and found that the improvements are noticeable across different datasets compared to baseline methods, with higher gains when the dataset size is larger. We also showed that unlike other state-of-the-art sampling models, our method learns a better estimate of confidence and is scalable to small and large datasets, short and long videos. Here we have shown confidence distillation to be effective when it is applied to the image modality. It would be of interest to see if a similar improvement can be achieved when the student has been trained using other modalities such as audio or optical flow.

While we have shown that confidence distillation is advantageous in efficient action recognition, extending this method to other types of machine learning tasks is another direction for future work. Additionally, different datasets have different characteristics

related to label noise, out-of-distribution outliers, and adversarial attacks. It will be interesting to investigate if confidence distillation provides any benefit to these tasks. Furthermore, more experiments and analysis are needed to gain insights on why confidence distillation works, other than acting as an indirect regularizer.

Finally, currently confidence distillation uses video clips (stacks of frames) in sampling. Another promising direction for future work is to examine the effect of frame-level sampling instead of clip-level in the accuracy/computational cost trade-off.

Bibliography

[1] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[3] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE transactions on acoustics, speech, and signal processing*, vol. 37, no. 3, pp. 328–339, 1989.

[4] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[8] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[9] H. Hassan, A. Aue, C. Chen, V. Chowdhary, J. Clark, C. Federmann, X. Huang, M. Junczys-Dowmunt, W. Lewis, M. Li *et al.*, "Achieving human parity on automatic chinese to english news translation," *arXiv preprint arXiv:1803.05567*, 2018.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[11] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 843–852.

[12] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[13] 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*. IEEE, 2011, pp. 2556–2563.

[14] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles, "Activitynet: A large-scale video benchmark for human activity understanding," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2015, pp. 961–970.

[15] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev *et al.*, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.

[16] J. Carreira, E. Noland, C. Hillier, and A. Zisserman, "A short note on the kinetics-700 human action dataset," *arXiv preprint arXiv:1907.06987*, 2019.

[17] K. Hara, H. Kataoka, and Y. Satoh, "Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?" in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 6546–6555.

[18] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[19] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan, "Youtube-8m: A large-scale video classification benchmark," *arXiv preprint arXiv:1609.08675*, 2016.

[20] D. Ghadiyaram, D. Tran, and D. Mahajan, "Large-scale weakly-supervised pre-training for video action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 046–12 055.

[21] M. Monfort, A. Andonian, B. Zhou, K. Ramakrishnan, S. A. Bargal, T. Yan, L. Brown, Q. Fan, D. Gutfruend, C. Vondrick *et al.*, "Moments in time dataset: one million videos for event understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–8, 2019.

[22] H. Kataoka, T. Wakamiya, K. Hara, and Y. Satoh, "Would mega-scale datasets further enhance spatiotemporal 3d cnns?" *arXiv preprint arXiv:2004.04968*, 2020.

[23] R. Goyal, S. E. Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag *et al.*, "The" something something" video database for learning and evaluating visual common sense," in *The IEEE International Conference on Computer Vision (ICCV)*, vol. 1, no. 2, 2017, p. 3.

[24] "The 20bn-jester dataset v1," https://20bn.com/datasets/jester.

[25] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2625–2634.

[26] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in neural information processing systems*, 2014, pp. 568–576.

[27] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1933–1941.

[28] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *European conference on computer vision*. Springer, 2016, pp. 20–36.

[29] J. Lin, C. Gan, and S. Han, "Tsm: Temporal shift module for efficient video understanding," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 7083–7093.

[30] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4489–4497.

[31] D. Tran, H. Wang, L. Torresani, and M. Feiszli, "Video classification with channel-separated convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 5552–5561.

[32] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 6450–6459.

[33] K. Hara, H. Kataoka, and Y. Satoh, "Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?" in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[34] C. Feichtenhofer, A. Pinz, and R. P. Wildes, "Spatiotemporal multiplier networks for video action recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4768–4777.

[35] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," in *Proceedings of the IEEE international conference on computer vision*, 2019, pp. 6202–6211.

[36] C. Feichtenhofer, "X3d: Expanding architectures for efficient video recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 203–213.

[37] G. Varol, I. Laptev, and C. Schmid, "Long-term temporal convolutions for action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1510–1517, 2017.

[38] M. Zolfaghari, K. Singh, and T. Brox, "Eco: Efficient convolutional network for online video understanding," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 695–712.

[39] O. Kopuklu, N. Kose, A. Gunduz, and G. Rigoll, "Resource efficient 3d convolutional neural networks," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[40] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6299–6308.

[41] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," *arXiv preprint arXiv:1706.04599*, 2017.

[42] B. Korbar, D. Tran, and L. Torresani, "Scsampler: Sampling salient clips from video for efficient action recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6232–6242.

[43] Y. Gal, "Uncertainty in deep learning," *University of Cambridge*, vol. 1, no. 3, 2016.

[44] R. Gao, T.-H. Oh, K. Grauman, and L. Torresani, "Listen to look: Action recognition by previewing audio," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 457–10 467.

[45] T. DeVries and G. W. Taylor, "Learning confidence for out-of-distribution detection in neural networks," *arXiv preprint arXiv:1802.04865*, 2018.

[46] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[48] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[49] D. Li, Z. Qiu, Q. Dai, T. Yao, and T. Mei, "Recurrent tubelet proposal and recognition networks for action detection," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 303–318.

[50] Z. Li, K. Gavrilyuk, E. Gavves, M. Jain, and C. G. Snoek, "Videolstm convolves, attends and flows for action recognition," *Computer Vision and Image Understanding*, vol. 166, pp. 41–50, 2018.

[51] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4694–4702.

[52] L. Sun, K. Jia, K. Chen, D.-Y. Yeung, B. E. Shi, and S. Savarese, "Lattice long short-term memory for human action recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2147–2156.

[53] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," *arXiv preprint arXiv:1506.06579*, 2015.

[54] Z. Qiu, T. Yao, and T. Mei, "Learning spatio-temporal representation with pseudo-3d residual networks," in *proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5533–5541.

[55] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, "Convolutional learning of spatio-temporal features," in *European conference on computer vision*. Springer, 2010, pp. 140–153.

[56] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks for action recognition in videos," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 11, pp. 2740–2755, 2018.

[57] L. Zhu, D. Tran, L. Sevilla-Lara, Y. Yang, M. Feiszli, and H. Wang, "Faster recurrent networks for efficient video classification." in *AAAI*, 2020, pp. 13 098–13 105.

[58] C.-Y. Wu, M. Zaheer, H. Hu, R. Manmatha, A. J. Smola, and P. Krähenbühl, "Compressed video action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6026–6035.

[59] L. Wang, W. Li, W. Li, and L. Van Gool, "Appearance-and-relation networks for video classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1430–1439.

[60] H. Wang, D. Tran, L. Torresani, and M. Feiszli, "Video modeling with correlation networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 352–361.

[61] S. Sun, Z. Kuang, L. Sheng, W. Ouyang, and W. Zhang, "Optical flow guided feature: A fast and robust motion representation for video action recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1390–1399.

[62] A. Piergiovanni and M. S. Ryoo, "Representation flow for action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9945–9953.

[63] C. Luo and A. L. Yuille, "Grouped spatial-temporal aggregation for efficient action recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 5512–5521.

[64] M. Lee, S. Lee, S. Son, G. Park, and N. Kwak, "Motion feature network: Fixed motion filter for action recognition," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 387–403.

[65] N. Hussein, E. Gavves, and A. W. Smeulders, "Timeception for complex action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 254–263.

[66] B. Fernando, E. Gavves, J. M. Oramas, A. Ghodrati, and T. Tuytelaars, "Modeling video evolution for action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5378–5387.

[67] L. Fan, W. Huang, C. Gan, S. Ermon, B. Gong, and J. Huang, "End-to-end learning of motion representation for video understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6016–6025.

[68] A. Diba, M. Fayyaz, V. Sharma, M. Mahdi Arzani, R. Yousefzadeh, J. Gall, and L. Van Gool, "Spatio-temporal channel correlation networks for action classification," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 284–299.

[69] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[70] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 116–131.

[71] H. Alwassel, F. Caba Heilbron, and B. Ghanem, "Action search: Spotting actions in videos and its application to temporal action localization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 251–266.

[72] Y.-C. Su and K. Grauman, "Leaving some stones unturned: dynamic feature prioritization for activity detection in streaming video," in *European Conference on Computer Vision*. Springer, 2016, pp. 783–800.

[73] W. Wu, D. He, X. Tan, S. Chen, and S. Wen, "Multi-agent reinforcement learning based frame sampling for effective untrimmed video recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6222–6231.

[74] Z. Wu, C. Xiong, C.-Y. Ma, R. Socher, and L. S. Davis, "Adaframe: Adaptive frame selection for fast video recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1278–1287.

[75] S. Yeung, O. Russakovsky, G. Mori, and L. 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*, 2016, pp. 2678–2687.

[76] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[77] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.

[78] M. Hein, M. Andriushchenko, and J. Bitterwolf, "Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 41–50.

[79] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

[80] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 535–541.

[81] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[82] G. Urban, K. J. Geras, S. E. Kahou, O. Aslan, S. Wang, R. Caruana, A. Mohamed, M. Philipose, and M. Richardson, "Do deep convolutional nets really need to be deep and convolutional?" *arXiv preprint arXiv:1603.05691*, 2016.

[83] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, "Born again neural networks," *arXiv preprint arXiv:1805.04770*, 2018.

[84] X. Cheng, Z. Rao, Y. Chen, and Q. Zhang, "Explaining knowledge distillation by quantifying the knowledge," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 925–12 935.

[85] M. Jain, J. Van Gemert, H. Jégou, P. Bouthemy, and C. G. Snoek, "Action localization with tubelets from motion," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 740–747.

[86] F. Caba Heilbron, J. Carlos Niebles, and B. Ghanem, "Fast temporal activity proposals for efficient detection of human actions in untrimmed videos," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1914–1923.

[87] H. Xu, A. Das, and K. Saenko, "R-c3d: Region convolutional 3d network for temporal activity detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5783–5792.

[88] T. Lin, X. Zhao, H. Su, C. Wang, and M. Yang, "Bsn: Boundary sensitive network for temporal action proposal generation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19.

[89] D. J. MacKay, "Bayesian methods for adaptive models," Ph.D. dissertation, California Institute of Technology, 1992.

[90] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," *arXiv preprint arXiv:1505.05424*, 2015.

[91] S. Farquhar, M. A. Osborne, and Y. Gal, "Radial bayesian neural networks: Beyond discrete support in large-scale bayesian deep learning," *stat*, vol. 1050, p. 7, 2020.

[92] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *Advances in neural information processing systems*, 2017, pp. 5574–5584.

[93] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7794–7803.

[94] B. R. Paredes, A. Argyriou, N. Berthouze, and M. Pontil, "Exploiting unrelated tasks in multi-task learning," in *Artificial intelligence and statistics*. PMLR, 2012, pp. 951–959.

[95] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[96] P. Gurevich and H. Stuke, "Pairing an arbitrary regressor with an artificial neural network estimating aleatoric uncertainty," *Neurocomputing*, vol. 350, pp. 291–306, 2019.

[97] J. Carreira, E. Noland, A. Banki-Horvath, C. Hillier, and A. Zisserman, "A short note about kinetics-600," *arXiv preprint arXiv:1808.01340*, 2018.

[98] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.