Applying High Performance Computing to Early Fusion Video Action Recognition

by

Matthew S. Hutchinson

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2020

Author .....
Department of Electrical Engineering and Computer Science
May 12, 2020

Certified by .....
Charles E. Leiserson
Edwin Sibley Webster Professor of Computer Science and Engineering
Thesis Supervisor
May 12, 2020

Certified by .....
Vijay Gadepally
MIT Lincoln Laboratory Senior Scientist
Thesis Supervisor
May 12, 2020

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Applying High Performance Computing to Early Fusion Video Action Recognition

by

Matthew S. Hutchinson

Submitted to the Department of Electrical Engineering and Computer Science
on May 12, 2020, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Over the past few years, there has been significant interest in video action recognition systems and models. However, direct comparison of accuracy and computational performance results remain clouded by differing training environments, hardware specifications, hyperparameters, pipelines, and inference methods. Additionally, the literature demonstrates a fixedness on late fusion approaches to audio-video multimodal problems. This project provides a side-by-side comparison of several 2-Dimensional Convolutional Neural Network (2D-CNN) video action recognition approaches and investigates the effectiveness and efficiency of new audio-video early fusion, slicing, and sampling methods. Model accuracy is evaluated using standard Top-1 and Top-5 metrics in addition to novel p-ROC metrics, and this project demonstrates the usefulness of the latter. Computational performance is measured via total training time and training time per epoch on a variety of high-performance computing (HPC) training configurations.

Thesis Supervisor: Charles E. Leiserson
Title: Edwin Sibley Webster Professor of Computer Science and Engineering

Thesis Supervisor: Vijay Gadepally
Title: MIT Lincoln Laboratory Senior Scientist

Acknowledgments

These are wild times, and I never expected my M.Eng. conclusion to be like this. There are many people to thank for everything that has made this research and my education possible even in these trying times.

I would first like to thank my parents, Scott and Lynn Hutchinson for everything they have done to help me and sustain me. They were instrumental in helping push my education forward through high school and onto MIT. While at MIT, they continued to support me in many ways–financially, logistically, and emotionally.

Second, I would like to thank my supervisor, Dr. Vijay Gadepally. Dr. Gadepally helped me find an interesting project and avenue of research. He has always been helpful in refining problems, bouncing ideas around, and keeping me on track. I appreciate how he finds time to meet with me even amid his busy schedule.

Third, I would like to thank my thesis advisor, Professor Charles Leiserson. Professor Leiserson oversaw my research and allowed me conduct much of it through MIT Lincoln Laboratory Supercomputing Center (LLSC) via the VI-A Program.

I would also like to thank everyone else who made this research and education possible. The entire LLSC team helped answering my questions, providing awesome computational resources, making the research process enjoyable, and letting me camp out in their training room often. I will miss Wednesday afternoon tea. My friends also helped me get through these five years. MIT is a tough place with seemingly endless work, but it's my friends that constantly found ways to make it fun. Additionally, countless MIT professors, staff, and administrators made the MIT experience unforgettable. I will wear my Brass Rat proud.

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the Under Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Under Secretary of Defense for Research and Engineering.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the last decade, advances in computing hardware and data availability have yielded significant progress in machine learning and artificial intelligence [20]. For example, applications such as object recognition in images have essentially reached, and in some cases surpassed, the accuracy of human object recognition (e.g., 29 of 38 teams in the 2017 ImageNet [15] Challenge achieving error rates $<5\%$) [23]. However, other applications such as the classification of actions in trimmed and untrimmed videos remain a challenge due to the volume and complexity of analyzing video streams. This project dived into the trimmed action recognition problem with two goals:

1. Presenting a comparison of existing approaches.

2. Testing novel early fusion methods.

Both aspects of the project were enabled by high performance computing (HPC) resources and the availability of curated action recognition datasets.

1.1 Video Action Recognition

Action (or activity) recognition is the computer vision task of identifying what is occurring in a video. Figure 1-1 displays several examples of video frames with their corresponding action (verb) labels. An action recognition model takes a video as an input and produces softmax probability predictions for the action class labels.

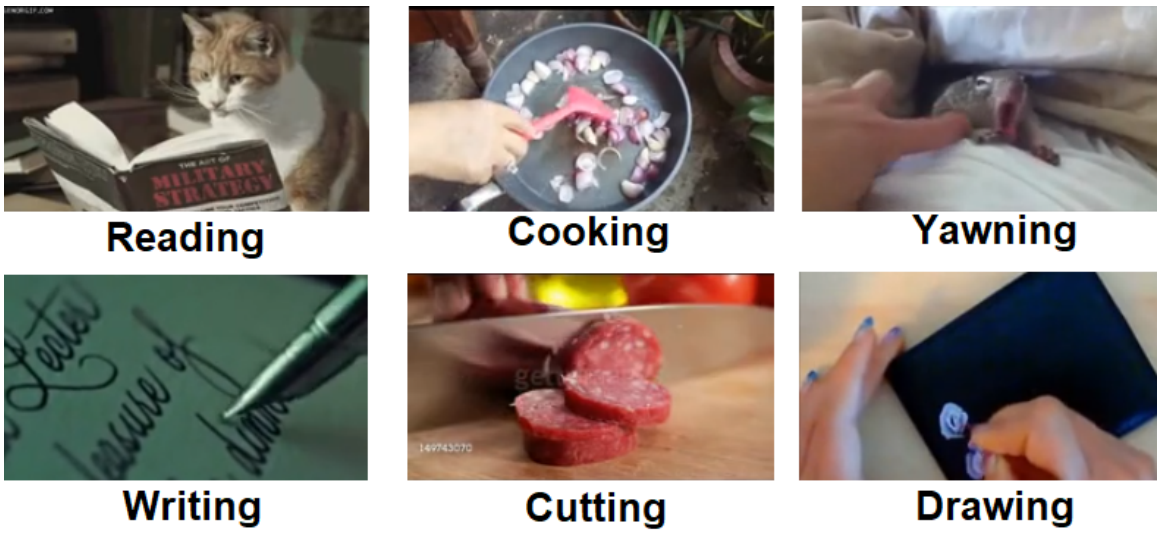

Figure 1-1: Example action recognition categories and video screenshots from the Moments in Time dataset [54].

Early approaches to action recognition utilized Hidden Markov Models (HMM) [92] and Dynamic Bayesian Networks (DBN) [91]. Later, Support Vector Machines (SVM) [77] used visual bag-of-words with many hand-crafted spatial, temporal, and spatio-temporal features: Histogram of Oriented Gradient (HOG), HOG3D [41], Histogram of Optical Flow (HOF) [10], Motion Boundary Histogram (MBH) [14], Speeded-Up Robust Features (SURF) [5], KLT Trajectories [51], SIFT Trajectories [71], Dense Trajectories (DT) [79], and Improved Dense Trajectories (iDT) [80]. However, since approximately 2014, Deep Neural Network (DNN) approaches have eclipsed these traditional methods and yielded better accuracy results on a variety of datasets. DNNs benefit from learning appropriate feature representations from raw data rather than requiring carefully hand-crafted inputs. The successes of deep learning in other computer vision problems such as image recognition, object detection, and scene segmentation, beg the question: can DNNs (and Deep Convolutional Neural Networks in particular) follow a similar path with video?

In order to test action recognition approaches, large video datasets have been collected, organized, and labeled. One such dataset is Moments in Time, a collection of approximately one million 3-second videos labeled by their actions [54]. The

Moments in Time dataset creators in collaboration with CVPR 2018 released the Moments in Time Challenge which encouraged teams to develop models that yield high Top-1 and Top-5 accuracies on the dataset. The top performing team produced an ensemble model with a Top-1 accuracy of 38.64% and a Top-5 accuracy of 67.19%. Clearly, there is significant room for improvement remaining in video action recognition, particularly with the classification problem posed by the Moments in Time dataset.

1.2 Modality Fusion

Action recognition is inherently a multi-modal problem with spatial, temporal, and often auditory domain components. The spatial domain is commonly captured in a visual Red-Green-Blue (RGB) modality. The temporal domain is commonly dealt with relationing between spatial data or via video transformations such as to optical flow. Auditory domain information can be encoded in a raw (1D) form or via transformations such as into a (2D) spectrogram.

DNN approaches to learning with multi-modal features can be generally categorized as "early fusion" or "late fusion." In early fusion, multi-modal features are fused prior to learning as shown in Figure 1-2. In late fusion, individual mode features are learned in separate networks and then fused via an ensemble learner as shown in Figure 1-3. Early fusion requires only a single session of learning while late fusion requires two (one session for the individual mode features and one to ensemble).

In the video action recognition literature, essentially all best-performing current models use late fusion rather than early fusion for several reasons. First, late fusion benefits significantly from research and testing already completed on the individual modality networks which can be pretrained on similar datasets. For example, a spatial model which trains on video frames can be pretrained on images from ImageNet [15]. Similarly, an audio model which trains on the video's audio channels can be pretrained on audio clips from AudioSet [22]. Pretraining can help these models generalize beyond an individual dataset's training data and can speed up training.

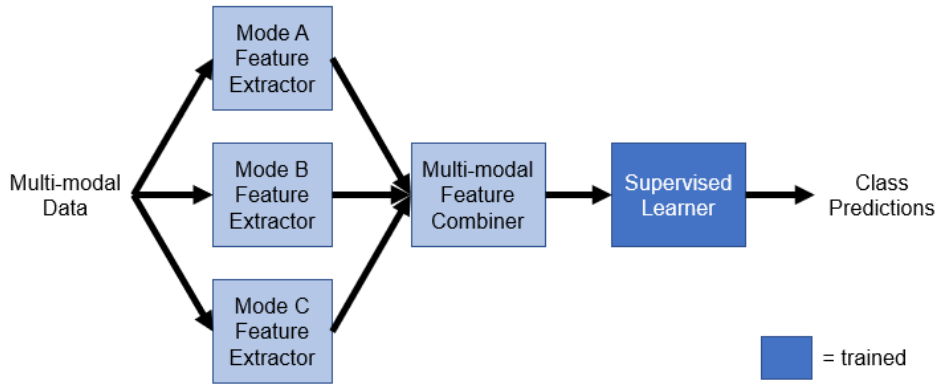

Figure 1-2: Generic early fusion architecture (adapted from [68]). Multi-modal features are extracted and fused prior to being used as input to a supervised learner.

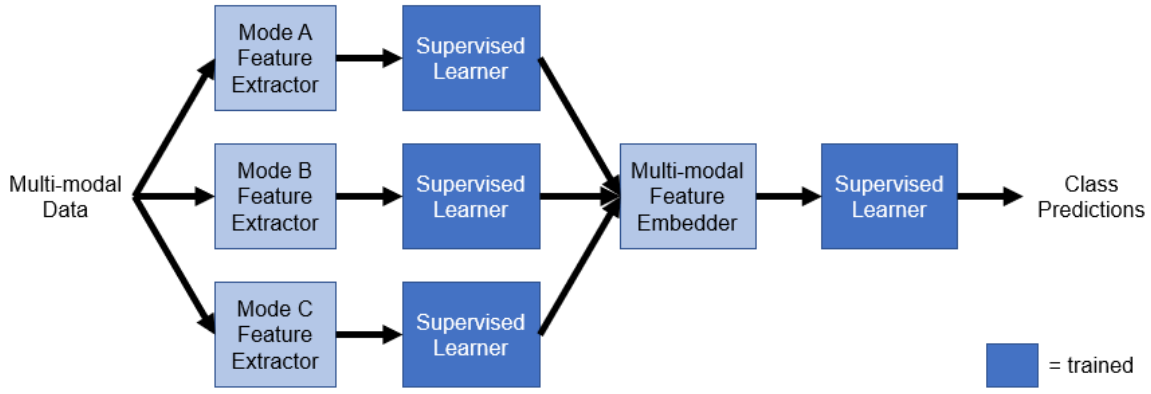

Figure 1-3: Generic late fusion architecture (adapted from [68]). Multi-modal features are extracted and fed as inputs to separate supervised learned. The outputs of those learners are fused and fed as inputs to another supervised learner which typically consists of only a few dense layers to do classification and end with softmax output.

Second, data parsing, wrangling, and fusing necessary for early feature fusion on large datasets is computationally costly. Without sufficient computational resources for these data augmentation steps (often in the form of a distributed computing system) or money to access one (such as Amazon Web Services), late fusion is more appealing as it can often be done on smaller systems.

Third, it can be argued that late fusion research is more tried and true. With early fusion, researchers must often take a "gamble" when designing new multi-modal feature vectors with little to no guarantees of improved performance over late fusion

Table 1.1: MIT Supercloud TX-E1/GAIA Specifications (2019–spring 2020). [59]

	Intel Xeon E5-2650	Intel Xeon E5-2683	AMD Opteron	Intel Xeon E5-2680	Intel Xeon G6-6248
Number of nodes	25	7	20	4	224
CPU cores/node	16	2 x 14	2 x 16	2 x 14	2 x 20
GPUs/node	0	0	0	2 or 4	2
GPU type	N/A	N/A	N/A	Volta V100	Volta V100
RAM (GB)	64	256	192	500	384
Local Disk (TB)	16	12	8	2	3.8

methods. Experimentation can be the only way of truly determining the effectiveness of an early fusion approach. Meanwhile late fusion can take the best existing model or models for each individual modality. Therefore, late fusing well-studied single-modality models is the default that most action recognition research uses and this is reflected in the Moments in Time Challenge 2018 results.

1.3 High Performance Computing

High performance computing (HPC) refers to employing aggregated computing power to achieve performance not possible through normal workstation computing. HPC was critical in this project because of the computational demands of working with videos. Video data, consisting of dozens of frames and additional audio channels, is orders of magnitude more data dense than image data. For example, ImageNet is approximately 150 GB of raw data [65] while Moments in Time is 42 TB of data after cropping to 224x224 frame size and parsing into NumPy [56] arrays.

Throughout this project, the HPC resources of TX-GREEN, TX-E1, and TX-GAIA (Supercloud) via MIT Lincoln Laboratory were instrumental in speeding up all aspects of the parse-wrangle-train-analyze pipeline. Parsing and wrangling were easily parallelized by action class via a mapping function. Training and validation were parallelized across compute nodes and/or GPUs via OpenMPI and Horovod [63].

An overview of the MIT Supercloud (TX-E1) infrastructure is provided in Table 1.1 and a detailed description can be found in [59].

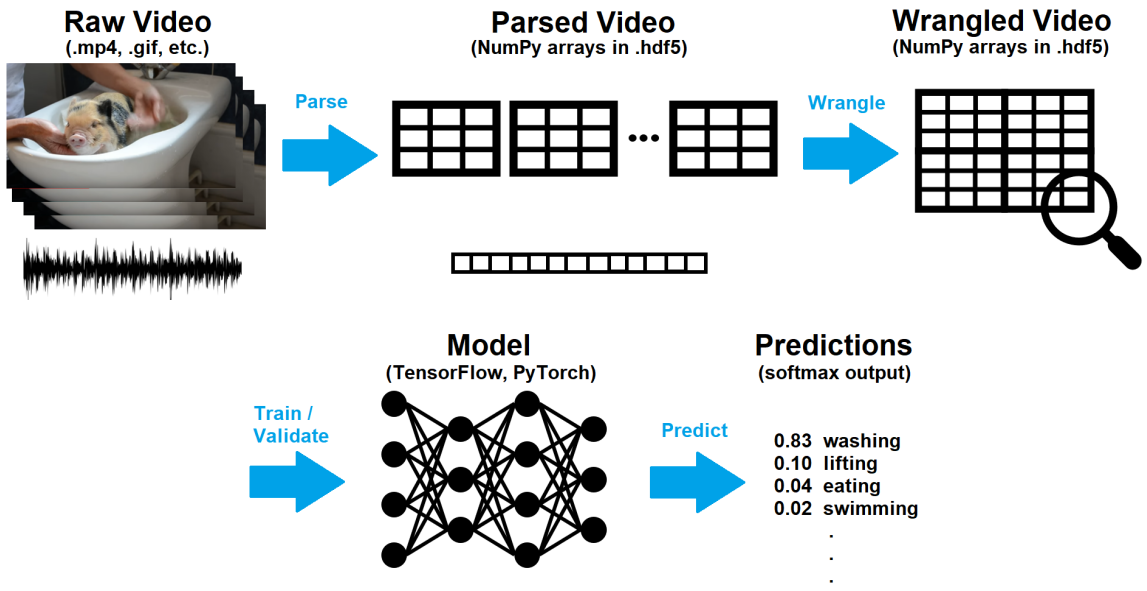

Figure 1-4: Overview of the Training Pipeline.

1.4 Training Pipeline

The first phase of this project involved constructing a pipeline for working with video data on the HPC systems described in section 1.3. Figure 1-4 describes the steps in that parse-wrangle-train-analyze pipeline. Some prior, yet incomplete work had been done on FFmpeg extraction of raw video frames by a previous LLSC intern, but the bulk of the pipeline was created during this project.

Moments in Time dataset raw video frames (stored in .mp4, .gif, and other common video file formats) were extracted at 30 frames per second (fps) using FFMPEG and converted into three-channel (RGB) 224x224x3 pixel tensors and stored as NumPy [56] arrays in HDF5 files (one per action class). Audio was parsed at both sample rates of 37.9KHz and 15.2KHz, transformed into log-mel spectrograms with 224 mels and a max frequency of 8192 Hz, and saved as one-channel images. Further audio parsing details can be found in section 5.1. Runtime statistics for parsing and transforming the training and validation data can be found in Appendix B, Table B.1. If those operations were performed serially on an Intel Xeon-e5 core, parsing the videos would take over 55 days, but was reduced to less than one day by parallelizing

the job across 60 cores using MIT Supercloud HPC resources.

An optional wrangling stage occured when only particular aspects of the data are required. This includes additional pre-processing (see Chapter 4 for examples). Parsed/wrangled videos were then fed into a model which outputs softmax predictions across the class options. Models were trained on a training set and evaluated on a validation set.

1.5 Project Overview

At a high level, this project had two goals. The first was to present a comparison of computational performance and accuracy of existing models using the Moments in Time training and validation sets. The second was to attempt to approach or surpass current performance and/or accuracy using novel features and architectures. The project therefore involved (1) constructing a training pipeline for using the Moments in Time dataset, (2) exploring feature engineering with early fusion of video and audio modalities, (3) investigating video action recognition architectures and hyperparameter selection, and (4) applying high performance computing methods to engineered features and models.

Chapter 2 presents a background on action recognition datasets and machine learning approaches. It also highlights research decisions made when designing these studies including why 2D models are the centerpiece of this research and how the action recognition approaches will be evaluated. Chapter 3 describes the comparison study implemented and conducted to baseline accuracy performance and computational performance of some action recognition models. Chapter 4 describes the slicing and sampling study implemented and conducted to investigate 2D spatial and spatio-temporal features. Chapter 5 describes the early fusion study implemented and conducted to investigate audio-video fusion using results obtained from the comparison study and the slicing and sampling study. Chapter 6 presents conclusions from this project, both from the literature review and the experiments, as well as recommendations for avenues to continue and expand upon this research.

Chapter 2

Background, Related Work, and Model Assessment

This chapter presents a background on video datasets and machine learning approaches to the action recognition problem. It will then explain why the Moments in Time dataset is used throughout these experiments and what model assessment techniques are appropriate for adequately comparing accuracy performance and computational performance.

2.1 Action Recognition Datasets

Datasets for video action recognition are defined by a set of qualities: source, preprocessing, point-of-view, number of videos, length of each video, number of action classes, classes per video (single-label or multi-label), annotation style, and purpose. A myriad of datasets have been crafted and curated to span this spectrum of qualities.

The vast majority of these video datasets focus exclusively on human actions for the obvious reason that human actions are extremely relevant to all aspects of everyday life. Early datasets, KTH [62], Weizmann [6], GTEA [18], GTEA GAZE [17], and GTEA GAZE+ [17], were created by research groups focused on daily human activities. Spurred by the growth of online video, UCF101 [69] and HMDB51 [44], with 13,000 and 7,000 videos, respectively, quickly became foundational benchmarks

Figure 2-1: Action Recognition Dataset Zoo. Two screenshots are taken from separate videos in each of the labeled datasets, giving a glimpse into the quality and focus of each of each.

in human action recognition. Thumos [36] and ActivityNet [29] had similar goals but did not gain the same level of popularity in the literature.

While human actions and human activities continue to dominate the field of action recognition datasets, slowly other purposes of these datasets emerged. Among them, YouTube-8M [2] and Micro-Videos (MV) [55] focused on human and non-human actions and visual entities. The Something-Something [24] dataset looks at low-level action captions for intuitive physics and semantics.

Among the most current iterations of these datasets, only a few have the breadth (action classes) and depth (videos per action class) that are comparable to ImageNet and other object recognition datasets. Kinetics-600 [8] and VLOG [19] achieve this for human actions and daily interactions. Moments in Time [54], which will be described in further detail in section 2.3, also achieves this quality. For a more detailed timeline of the creation of these datasets as well of information about dataset size and number of classes, see Appendix C, Table C.2.

2.2 Machine Learning for Action Recognition

Deep CNNs for computer vision proved their worth in 2012 with the application of AlexNet [43] to the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC). Table 2.2 provides a timeline of major CNN architecture breakthroughs in computer vision specifically emphasizing those useful in common action recognition approaches. With these developments, action recognition model architectures can be broadly categorized into two groups: 2D approaches and 3D approaches.

2D action recognition models are categorized as 2D not because they ignore the temporal domain aspect, but because they use 2-Dimensional convolutional kernels generally in the form of 2D CNN model backbones. These approaches include traditional 2D Convolutional Neural Networks (C2D), Temporal Segment Networks (TSN) [87], Long-term Recurrent Convolutional Neural Networks (LRCN) [16] sometimes referred to as CNN+LSTMs, and Temporal Shift Modules (TSM) [49]. C2D carries over directly from image recognition. A frame is extracted from the video and used

Table 2.1: A Brief History of Neural Network Architectures.

Architecture	*Year*	*Description*	*Accuracy (%)* *UCF*	*HMDB*	*Kinetics*
LeNet [45]	1998	pioneered CNNs			
AlexNet [43]	2012	introduced ReLU & max pool			
VGG [67]	2014	deeper net with smaller kernels			
Inception [74, 75] (GoogLeNet)	2014	inception cell convolves at multiple scales then aggregates			
2-Stream CNN [66]	2014	late fusion of a spatial and optical-flow based temporal nets	88.0	59.4	
ResNet [28, 89]	2015	residual block learns a residual function of the input			
C3D [76]	2015	extended 2D convolution to 3D	90.4		
$F_{ST}CN$ [72]	2015	factorized 3D kernel as 2D spatial followed by 1D temporal	88.1	59.1	
TDD [81]	2015	combined hand-crafted & deep-learned features	91.5	65.9	
CNN+LSTM [16] (LRCN)	2015	CNN followed by an LSTM for sequence-based actions	82.7		
SqueezeNet [31]	2016	decreased parameter-space with AlexNet-level results			
TSN [87]	2016	learns temporal structure via segment/two-stream/consensus	94.2	69.4	
ResNeXt [86]	2017	replaced residual blocks with a "split-transform-merge" block			
Xception [12]	2017	improved inception cell by performing 1x1 convolution first then channel-wise spatial			
Two-Stream I3D [9]	2017	two-3D convolutional streams on dense RGB and optical flow	97.9	80.2	
DenseNet [32]	2017	connected every layer to every successive network layer			
SENet [31]	2018	modified residual block for channel interdependencies			
NL [82]	2018	introduced a non-local block for long dependencies			77.7
TRN [93]	2018	used a temporal relations pool instead of TSN's average pool	83.8		63.2
MKAF [50]	2018	multi-modal keyless attention fusion for fast LSTM			77.0

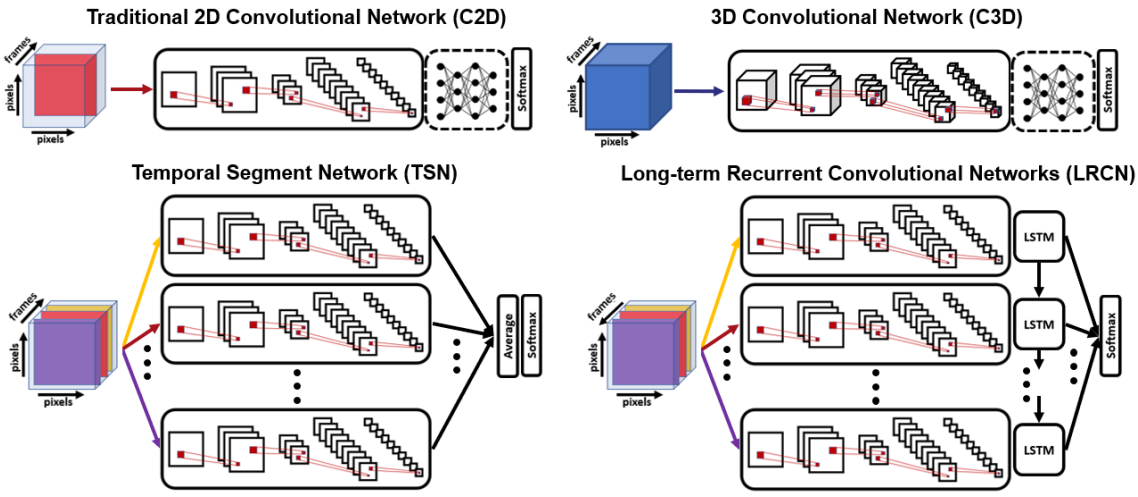

Figure 2-2: Four most common video action recognition approaches. Note that these are simplified diagrams where icons for 2D-ConvNets, 3D-ConvNets, dense classification networks, LSTM modules, and averaging/softmax layers are used only as visual interpretation. Their actual design can vary significantly. Similarly, the majority of action recognition approaches have 2-stream variants for RGB+Optical Flow.

as input to a 2D-ConvNet. After several convolution and pooling layers, the logits are fed into one or more fully-connected layers which can produce a softmax output predictions over the classes. TSN segments a video along its temporal dimension and extracts a frame from each segment. The frames are used as inputs to a 2D-ConvNets that share weights. The predictions from each segment are then averaged before the softmax output. Variants and additions to a TSN baseline include Temporal Relations Networks (TRN) [93] which perform multi-scale segmentation and relationing. LRCN similarly segments a video, extracts a frame from each segment and feeds those frames into 2D-ConvNets. However, the ConvNet outputs are used as inputs to a Long-Short Term Memory (LSTM) network. The LSTM's output is used for softmax predictions.

3D action recognition approaches include the temporal aspect in the convolution via the use of 3-Dimensional convolutional kernels. C3D models were designed as the 3D analogy to 2D CNNs because of the widespread success of 2D CNNs [76]. However, because of the difficulties of working with videos and the long-term dependencies aspects of actions, C3D models often have had less success on action recognition than

their 2D counterparts on object recognition. To attempt to bridge the gap between 2D and 3D models, Inflated 3D (I3D) models were created by ”inflating” pretrained 2D kernels into 3D kernels [9]. This allows I3D models to benefit from pretraining on 2D image datasets like ImageNet. Some believe that, while still in their early days, 3D approaches will be able to retrace the successful history of their 2D siblings [27]. Both C3D and I3D work by using either the entire or a selected portion (often 16, 32, or 64 frames) as an input to a 3D-ConvNet. Similar to C2D, the 3D-ConvNet's output is then fed into a one or more layer classification network before outputting softmax predictions across classes.

2.3 Moments in Time

Among action recognition datasets, Moments in Time uniquely offers high inter-class and intra-class variation [54]. Inter-class variation refers to a large semantic separation between the 339 action verb classes. Intra-class variation refers to various levels of abstraction within each action verb class. For example, the "opening" class can include videos of opening doors, drawers, curtains, flower petals, and more. The actions can be performed by various agents: humans, animals, animations, or nature.

Each 3-second video in the dataset has a single action verb label from one of the 339 action classes. The training set consists of 802,264 videos with between 500 and 5,000 videos per class. The validation set consists of 33,900 videos with 100 videos per class. High-level dataset statistics can be found in Appendix C, Table C.1.

The Moments in Time dataset creators tested a variety of existing action recognition model architectures on their dataset (see Appendix C, Table C.3) as well as ran two challenges as a part of CVPR'18 and ICCV'19. The 2018 challenge tasked teams to develop state-of-the-art methods for achieving top-1 and top-5 accuracies on the dataset. Essentially all top-performing teams used late fusion approaches by training a series of individual models and then ensembling their results. Table 2.2 highlights the best-performing individual models used by teams in the challenge as described in their reports. Table 2.3 highlights the types of models and modalities used by each

Table 2.2: Moments in Time Challenge 2018 top-performing single model validation accuracies as described in optional report submissions by competition teams.

Type	*Backbone*	*Accuracy (%) Top-1*	*Top-5*	*report*
Video				
C2D	SENet-152	**33.7**	**61.3**	[47]
	SEResNeXt	30.0	60.2	[47]
	Xception	31.8	59.2	[47]
	ResNet-50	28.3	53.2	[47]
TSN	ResNet-152	**33.0**	n/a	[85]
	DPN-107	31.1	n/a	[85]
	ResNet-50	27.4	53.2	[47]
TRN	SENet-154	**31.9**	**58.8**	[11]
	Inception-v3	29.7	55.7	[11]
	InceptionResNet-v2	29.3	55.6	[11]
I3D	ResNet-50	**34.2**	**61.4**	[47]
	ResNet-101-NL	33.7	n/a	[85]
	Inception-v3	27.6	53.9	[11]
C3D	InceptionResNet-v2	**35.1**	**63.3**	[46]
	ResNet-101	33.6	61.2	[46]
Audio				
C2D	VGGish	**17.1**	n/a	[85]
	SENet-50	16.8	n/a	[85]
	M34-res	14.8	27.4	[46]
	ResNet-34	13.8	23.6	[46]
	EnvNet+ResNet	13.2	25.9	[46]
	NetVLAD	9.0	19.5	[11]
	SoundNet	7.6	18.0	[48]

of the top-performing teams. While model architectures such as C3D, I3D and TRN may intuitively be expected to provide significant accuracy performance advantages, when applied to the Moments in Time dataset, these architectures barely outperform, and in some cases underperform, conventional and less complex 2D CNN models. Additionally, while the literature on action recognition lacks an adequate discussion of computational performance and complexity of training these models, it is sometimes hinted that 3D approaches requires significantly more computational resources and time to train.

The Multi-Moments in Time Challenge 2019, conducted through ICCV'19, tasked

Table 2.3: Moments in Time Challenge 2018 top-performing teams approaches as described in optional report submissions by competition teams.

	Team	*Test Accuracy (%)*		*Fusion*	*Modalities*	*# Models*
		Top-1	*Top-5*	*Type*	*Used*	*Ensembled*
1	DEEP [46]	38.6	67.2	late	V+A	6+3
2	Megvii [47]	37.5	65.0	late	V+F+A	7+1+1
3	Qiniu [85]	36.4	63.7	late	V+F+A	12+6+3
4	Alibaba-Venus [11]	35.5	63.7	late	V+F+A	9+1+2
5	Xtract AI [30]	32.0	57.6	late	V+A	5+1
6	SSS [94]	32.0	57.6	late	V+F	6+3
7	CM-AML [34]	31.0	58.4	late	V+F+A	6+1+1
8	UNSW-DS [48]	30.4	54.9	late	V+A	2+1
9	Fengwuxuan	28.6	54.9	n/a*	n/a	n/a
10	SYSU [26]	27.3	53.9	late	V+A	5+1

**Team Fengwuxuan did not submit a report*

V = Visual (RGB)
F = Optical Flow
A = Audio

teams to develop state-of-the-art methods for detecting multiple event labels from videos. Of the reports that teams submitted, few deviated from 2D CNN approaches likely because they drew the same conclusions as described above in addition to the difficulties that come with working on more complex models. For these reasons, the studies conducted in this project also focus heavily on 2D CNN action recognition methods.

The results of these challenges (demonstrating significant room for improvement), the uniqueness of the dataset, and the potential for 2D approach improvements made Moments in Time the focus of this project.

2.4 Model Assessment Techniques

Video action recognition models are primarily assessed along accuracy performance and secondarily assessed along computational performance. Accuracy performance refers to how effective a trained model is at the action recognition task. Computational performance refers to the compute required to perform the training. Both are required to fully gauge a model's effectiveness because of the common trade-off

between them. A model with high accuracy performance that takes hundreds of years to train is not an effective model. Obviously, neither is a model that trains quickly but has poor accuracy results. Therefore, while often overlooked in the literature, including both aspects of the assessment together is instrumental in comparing video action recognition approaches.

Canonically, Top-k accuracy is used to measure the effectiveness of action recognition models. The model's softmax output yields a probability for each of the $|C|$ possible classes where C is the set of action classes. If the correct class label is within the k highest probability predicted classes, the model has successfully classified the video. Top-1 accuracy is intuitively useful because it describes the percentage of validation data cases in which the model's top predicted classification is correct. However, Top-5 is arbitrarily chosen and has unfortunately become a default in the literature.

This project demonstrates that plotting what will be referred to as a psuedo-Receiver Operator Characteristic (p-ROC) curve is a better method of representing accuracy performance. In this p-ROC, the Top-k accuracy is plotted against k analogous to plotting the true positive rate against the false positive rate for our classifier. Even though it has been noted that the ROC area under the curve (AUC) and the maximum Youden index (J_{max}), the curve height about the chance line, provide desirable properties as a classification metric [7, 33], the practice has not become standard. These benefits easily transfer to our p-ROC curve and allow a user to quickly and more intuitively select a model with accuracy characteristics that they desire. One key difference between the p-ROC curve and a traditional ROC curve is that the horizontal axis is discrete, not continuous. Hence the Youden index, sometimes referred to as Youden's J-statistic, is only defined at these discrete values of k. An additional advantage p-ROC AUC is that it can be normalized via dividing by the number of classes $|C|$. This allows model accuracy comparison across datasets with different numbers of classes (which is common among the datasets mentioned in Section 2.1). Equations 2.1, 2.2, and 2.3 show how to calculate p-ROC AUC, AUC_{norm}, and J_{max} where $acc(k)$ refers to a function computing Top-k accuracy for a given k.

$$AUC = \sum_{k=0}^{|C|-1} \frac{acc(k+1) - acc(k)}{2} \tag{2.1}$$

$$AUC_{norm} = \frac{AUC}{|C|} \tag{2.2}$$

$$J_{max} = \max_{k \in \{0,1,\ldots,|C|\}} acc(k) - \frac{k}{|C|} \tag{2.3}$$

The example 15-class classification problem shown in Figure 2-3 is intended to illustrate the usefulness of p-ROC curves in action recognition model accuracy performance analysis. Shown in the figure are three models (A, B, and C) as well as what would be expected from a random chance guess (i.e. the $k/15$ line). Table 2.4 shows summary statistics for comparing these models along traditional Top-1 and Top-5 accuracies as well as with p-ROC AUC and J_{max}. The best-to-worst ordering of these models along Top-1 accuracy is C, A, B. Along Top-5 accuracy, the ordering is A, B, C. Therefore, without using p-ROC AUC, one might naively conclude that Model B is worse at this classification task than models A and C. However, Model B holds the highest p-ROC AUC among the three models. Clearly, determining the "best" model is no longer straightforward. Because different applications of action recognition problems might require different accuracy performance properties, reporting a p-ROC curve instead of simply the canonical Top-1/Top-5 accuracies is beneficial to the model user. Importantly, we claim that higher Top-5 accuracy does not always correlate with higher p-ROC metrics. This project not only demonstrates that this claim is sometimes true but rather it is fairly common.

Table 2.4: p-ROC curve statistics for the 15-class problem plotted in Figure 2-3.

Model	*Top-1 Accuracy*	*Top-5 Accuracy*	*p-ROC AUC*	J_{max}
Chance	0.067	0.333	7.50	0.0
A	0.300	**0.800**	11.72	**0.47** ($k = 5$)
B	0.250	0.780	**11.82**	0.46 ($k = 6$)
C	**0.400**	0.760	11.56	0.43 ($k = 5$)

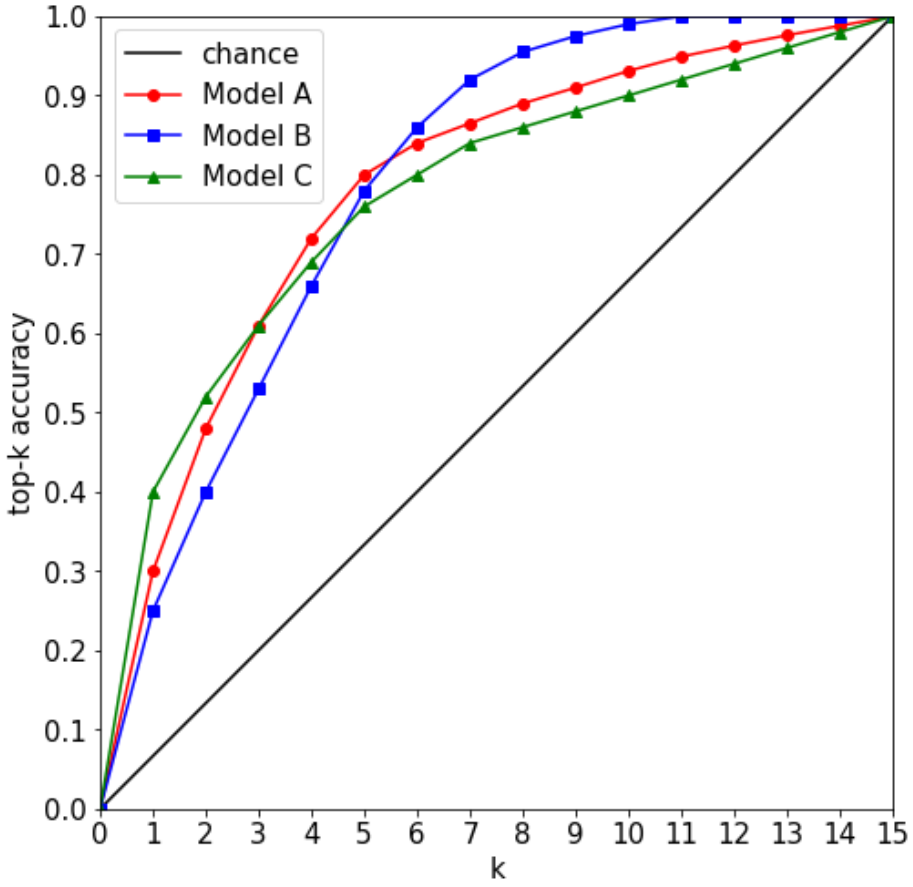

Figure 2-3: Example p-ROC curve for a 15-class problem. Three model Top-K values are plotted (as well as a random chance line that represents an uninformed guesser.

Because of the challenges with dealing with video as opposed to simpler uni-modal classification problems such as image object recognition, computational performance of training is a second important method of assessing models. Unfortunately the literature on action recognition model computational performance is extremely lacking in detail. Almost all emphasis has been place on Top-1/Top-5 accuracy performance. Throughout this project, computational performance is measured directly by training time and training time per epoch. Attention was also paid to how varying the compute resources affects training (i.e. yields speedup curves).

Therefore, throughout this project, the accuracy performance and computational performance metrics described above are used to assess models and training techniques applied to the Moments in Time dataset.

Chapter 3

Comparison Study

Unlike the field of image classification, video action recognition lacks a thorough discussion of model and algorithm comparison. For example, it is difficult to compare accuracy metrics of various algorithms which are often developed and tested on heterogenous datasets or lack sufficient details regarding model architectures. Further, the race for higher Top-1 accuracies has sidelined discussions of the equally relevant aspect of computational performance. Therefore, this set of experiments catalogs a subset of state-of-the-art video action recognition models. The goal is to provide a side-by-side comparison of these "off-the-shelf" models using the performance metrics outlined in section 2.4.

Specifically, this study (1) utilized the Moments in Time dataset for action recognition model comparison, (2) trained and evaluated a set of action recognition models under similar hyperparameters, training methods, and hardware, and (3) discussed both their accuracy performance and computational performance.

3.1 Initial Comparison

A first set of comparisons was between five “off-the-shelf” TensorFlow [1] C2D models: ResNet34 [28], ResNet50 [86], Inception-v3 [74], Inception-ResNet-v2 [73], and Xception [12]. Table 2 shows a comparison of the complexity of these models, and they are listed in increasing order of trainable parameters.

Table 3.1: Complexity of "Off-the-Shelf" C2D Model Backbones

Model	*Layers*	*Trainable Parameters*
ResNet34	34	21,471,379
Xception	71	21,501,563
Inception-v3	48	22,462,963
ResNet50	50	24,229,203
Inception-ResNet-v2	164	54,797,235

3.1.1 Experimental Design

Python 3.6.5 scripts trained and validated these off-the-shelf models in a distributed fashion using Horovod 0.16.1 [63] and OpenMPI. Key package versions used were NumPy 1.14.1 [56], H5py 2.7.1 [13], SciPy 1.1.0 [78], TensorFlow 1.13.1 [1], PyTorch 1.0.1 [57], Pillow 5.1.0 and FFmpeg 3.3.7.

To compare computational performance and the effects of batch size, three types of distributed training situations were tested. The first used 8 NVIDA Tesla K40 GPU accelerators across two nodes. The second used 16 Intel Xeon-e5 nodes with 28 CPU cores per node. The third used 32 Intel Xeon-64c nodes with 64 cores per node. The infrastructure used is described in detail in [59].

The Moments in Time pre-processed 30 frames per second (fps) videos resized to 224x224 cropped 3-channel frames were used as inputs. Videos were parsed at 15 fps in addition to the 30 fps parsing described in section 1.4. This was completed for training and validation sets that were defined by the Moments in Time creators.

With the exception of ResNet34, which was initialized with random weights, models were initialized with ImageNet pretrained weights. A dense classification layer was added to the top of each of these CNNs. Each network was trained by randomly sampling one frame from each 15 fps parsed video. A Horovod-wrapped distributed stochastic gradient descent (SGD) optimizer and categorical cross-entropy loss metric updated network weights. The learning rate started at 0.1 and decayed by a factor of 10 at 30 epochs. It was also adjusted during the first 5 (warmup) epochs, increased by a factor of the total number of processes launched. Standard momentum of 0.5 was used.

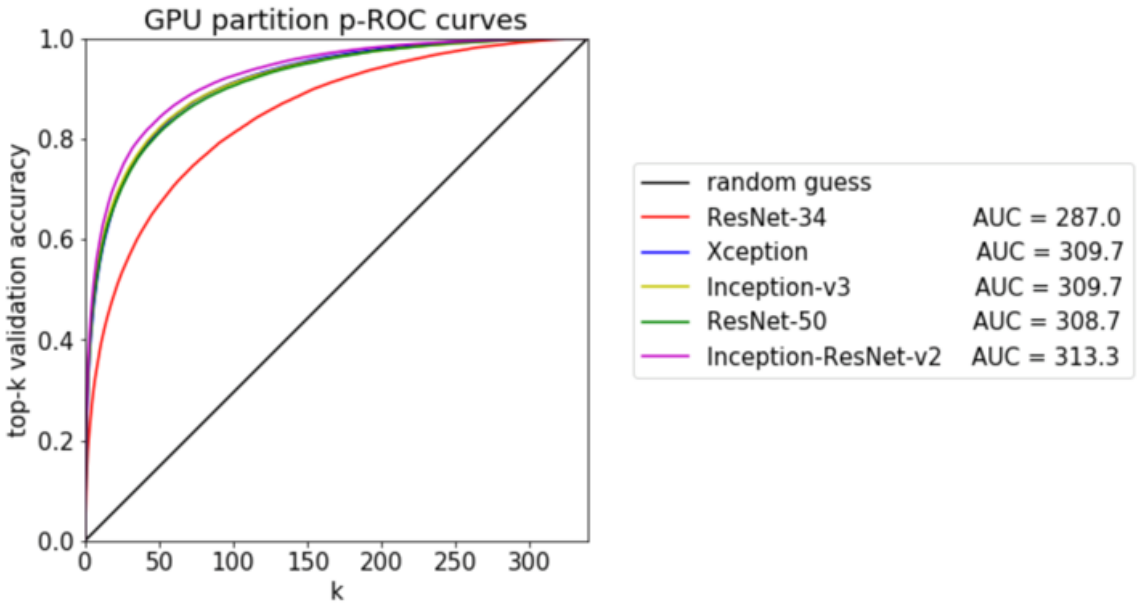

Figure 3-1: p-ROC curves for GPU-partition trained models. See Appendix A Figure A-1 for p-ROC in the other training configurations.

For proper comparison, other hyperparameters were held consistent across different model training sessions. Each model was trained for 50 epochs with a batch size of 32 per training process. One process was launched per Tesla K40 GPU yielding an effective batch size of 256 on the GPU nodes partition. On each of the CPU node partitions, one process was launched per node yielding effective batch sizes of 512 and 1024 on Xeon-e5 node partitions and a Xeon-64c node partitions, respectively.

Trained C2Ds were expanded into TSNs for validation inference. Video level inference was therefore the averaged prediction across 6 evenly spaced frames from the 90 frame (30 fps) video.

3.1.2 Accuracy Performance Results

After 50 epochs, models with pretraining averaged 22.1% Top-1 and 45.7% Top-5 accuracy using the smallest effective batch size corresponding to the GPU nodes partition. The ResNet34 model that did not benefit from pretraining performed worse. The Inception-ResNet-v2 model, which has significantly higher complexity than the

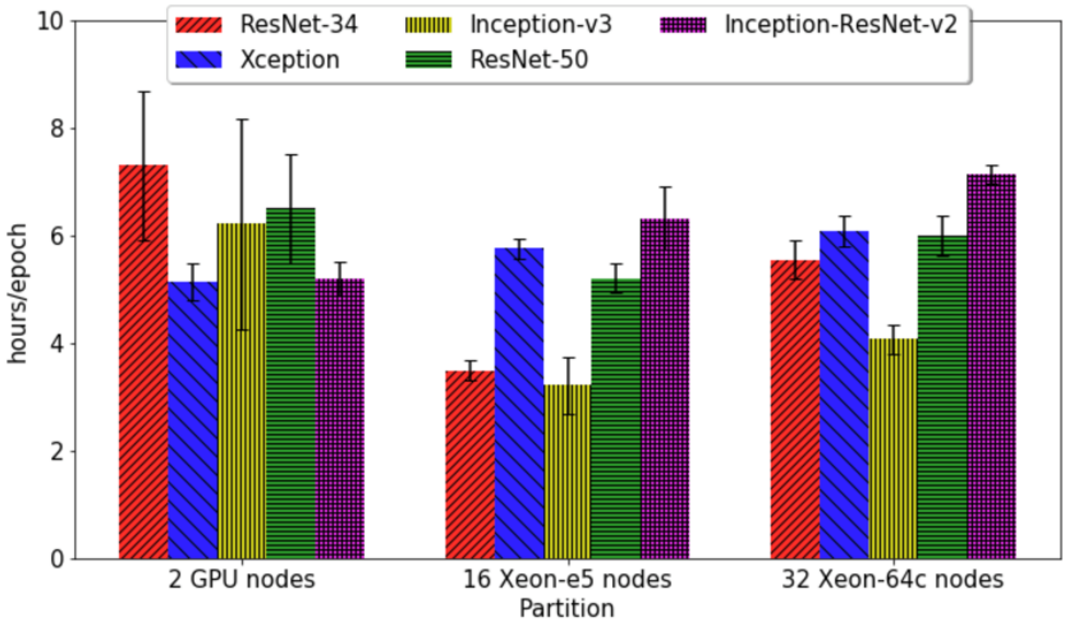

Figure 3-2: Training time per epoch on three types of distributed training partitions. Error bars indicate standard deviation.

other models, had the greatest Top-1 accuracy, Top-5 accuracy, p-ROC AUC, and J_{max}. The p-ROC curves are shown in Figure 3-1 and full validation results are shown in Appendix A Table A.1.

3.1.3 Computational Performance Results

The computational performance of models averaged around 6 hours per epoch on a 2-node GPU partition, 5 hours per epoch on a 16-node Xeon-e5 partition, and 6 hours per epoch on a 32 node Xeon-64c partition. On GPU nodes, model training time fluctuated between 4 and 9 hours per epoch and saw higher variation than models trained on Xeon-e5 or Xeon-64c nodes. As shown in Figure 3-2, the best performing model in accuracy (with an Inception-ResNet-v2 backbone) had similar computational costs to other models across partition types despite having twice as many parameters and two to three times as many layers.

Across these five models, no significant correlations between number of layers nor trainable parameters exists with training time per epoch. Across most training runs,

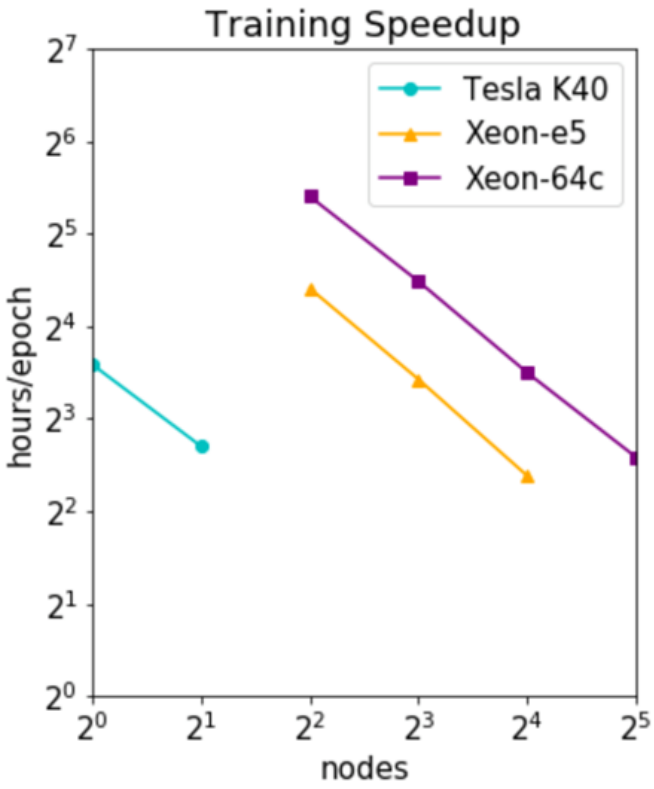

Figure 3-3: Training time speedup curve for ResNet-50 backbone model.

the standard deviation of training time per epoch varied by no more than 1.5 hours. These results further show, as has been noted in the literature [38], that the number of floating-point operations does not always directly correspond to computational costs due to other training time activities. Further research is needed to identify which aspects of training are contributing the most to the computational performance costs. However, these results are still a valuable start to researchers working with C2D models for action recognition who may not initially know what computational resources and time are required for training models. This is particularly relevant because of the monetary costs involved in using cloud-based resources like AWS.

Figure 3-3 shows the speedup curve for training a ResNet50 model. Other C2D models should have analogous speedup curves. Across the three partitions, a 2x increase in nodes directly yielded a 2x reduction in training time.

Table 3.2: Complexity of Models in Expanded Comparison Study

Model Type	*Model Backbone*	*Layers*	*Trainable Parameters*
C2D	VGG19	19	20,198,291
	MobileNet (M)	28	3,554,451
	Inception-v3 (Iv3)	48	22,462,963
	ResNet50 (R50)	50	24,229,203
	MobileNet-v2 (Mv2)	53	2,658,131
	Xception (X)	71	21,501,563
	Inception-ResNet-v2 (IRv2)	164	54,797,235
	DenseNet169 (D169)	169	13,048,915
	DenseNet201 (D201)	201	18,744,147
LRCN	n/a (16f)	38	9,788,915
C3D	n/a (16f)	18	148,590,675
	n/a (32f)	18	456,872,019
I3D	Inception-v1 (Iv1) (16f)	27	12,279,984
	Inception-v1 (Iv1) (64f)	27	12,279,984

3.2 Expanded Comparison

Due to increased computational resource availability later in this research, the comparison study was able to be expanded to include more TensorFlow action recognition models and greater breath of computational performance testing. Table 3.2 lists details of the models compared.

3.2.1 Experimental Design

Python 3.6.5 scripts trained and validated these off-the-shelf models in a distributed fashion using Horovod 0.18.2 [63] and OpenMPI 4.0. Key package versions used were NumPy 1.16.5 [56], H5py 2.9.0 [13], SciPy 1.3.2 [78], and TensorFlow 1.14.0 [1].

C2D and I3D models were initialized with ImageNet pretrained weights while C3D and LRCN models were initialized with random weights. On each pass through the dataset during training, C2D inputs were randomly sampled frames from each video. LRCN, C3D (f16), and I3D (f16) had inputs of 16 evenly spaced frames from the 30 fps videos. C3D (32f) and I3D (64f) randomly sampled 32 and 64 continuous frames, respectively.

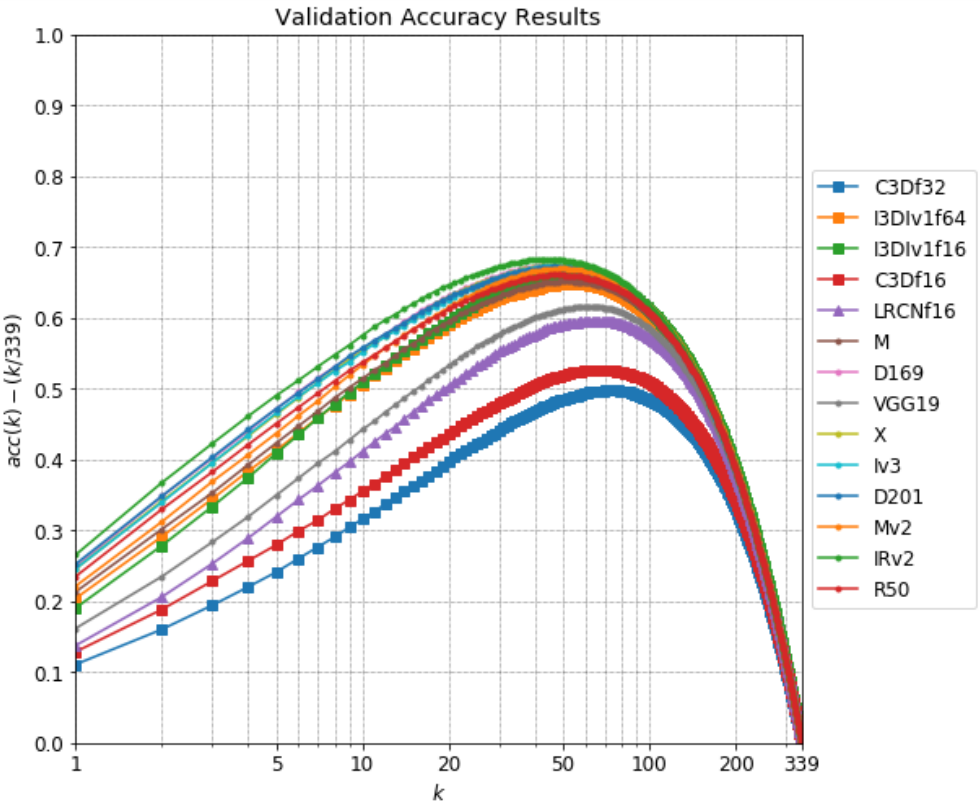

Figure 3-4: p-ROC curve log-scaled with $k/339$ subtracted out for each value of k to more easily show the peak J-statistic.

For proper comparison, other hyperparameters were held consistent across different model training sessions. A Horovod-wrapped distributed ADADELTA [90] optimizer and categorical cross-entropy loss metric updated network weights. Five warmup epochs slowly raised the learning rate to 1.0 which was subsequently decayed at 20, 35, and 50 epochs. Each model was trained for 65 epochs.

For validation, LRCN, C3D, and I3D model inference was performing the same as training. C2D model inference was performed in a TSN-style averaging across 6 evenly spaced frames for the 90 frame (30 fps) video.

3.2.2 Accuracy Performance Results

Validation accuracy results can be found in Figure 3-4 and in detail in Appendix A, Table A.2. By our p-ROC, the top three performing models were all C2Ds: Inception-

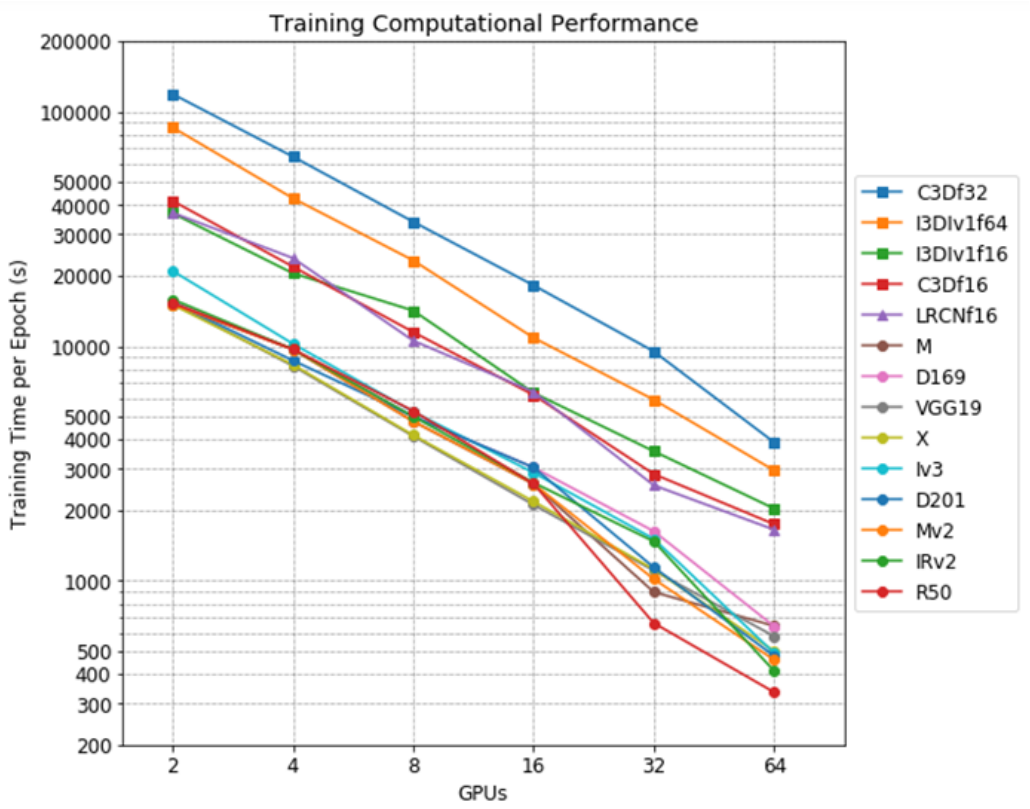

Figure 3-5: Training Time per Epoch across training configurations of 1, 2, 4, 8, 16, and 32 nodes where each node as 2 Volta V100 GPUs.

ResNet-v2, DenseNet169, and Xception with p-ROC *AUC*s of 310.99, 310.32, and 310.02, respectively.

3.2.3 Computational Performance Results

Computational performance results can be found in Figure 3-5 and in detail in Appendix B, Table B.3. As expected, LRCN, C3D, and I3D models had significantly greater training times compared to the much simpler C2D models. When training on 64 GPUs for 65 epochs, the quickest model was ResNet50 which had an average training time of 335.4 seconds per epoch. Therefore, the model successfully trained in just over six hours.

As expected, essentially all models approximately halved their training times when trained on twice as many GPUs. Individual node differences and network lag fluctua-

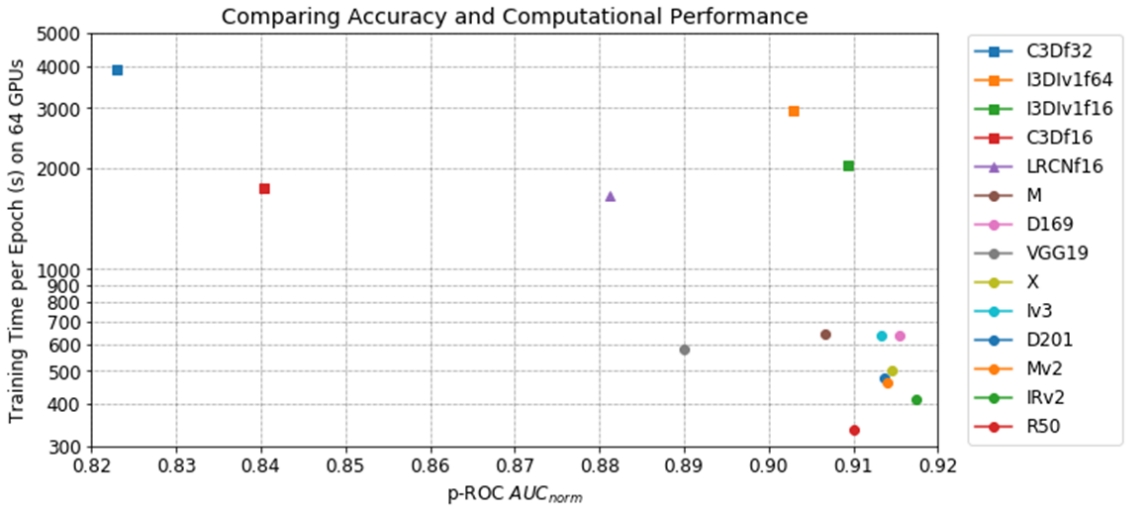

Figure 3-6: Plotting training time per epoch (in seconds) against p-ROC AUC_{norm} to show accuracy-computational performance trade-offs The best performing models are in the bottom right: Inception-ResNet-v2, ResNet50, MobileNet-v2, Xception, and DenseNet201.

tions on the TX-GAIA system became more apparent in the larger (16 and 32 node) training runs as evidenced by the increased variation on the right of Figure 3-5.

3.3 Discussion and Conclusions

Two comparison experiments comprised this study. The first looked at five off-the-shelf C2D models trained in both GPU and CPU environments. The second expanded the number of C2D models analyzed and incorporated more complex LRCN, C3D, and I3D models. When combining accuracy and computational performance into a single plot, as shown in Figure 3-6, the best performing models are in the bottom right. Those correspond to high p-ROC AUC_{norm} and low trainings times per epoch. Among the models tested, it is apparent that C2Ds are better performers than their more complex counterparts. Among the C2Ds, those with greatest model depth stand out. This hints that deeper is the direction to go with future model architecture research.

Chapter 4

Exploration of Video Slicing and Sampling

The use of 2D convolutional kernels in video action recognition models remains widespread, yet the literature shows a functional fixedness on using frame-wise training methods. This study explored horizontal and vertical video cube slicing methods as well as several sampling techniques for determining where along a given video axis to "slice." In this study, frame slicing refers to extracting a frame ($pixels$ x $pixels$ x $channels$) from a video ($frames$ x $pixels$ x $pixels$ x $channels$) and then training the 2D kernel on that frame. The frame is a spatial representation for a portion of that video. Horizontal and vertical slicing methods extract a spatio-temporal feature from a video. Essentially, by taking a particular row or column of pixels across the entire temporal domain (i.e. that row/column in every frame), a spatio-temporal "image" is extracted from the video "cube" with a shape ($frames$ x $pixels$ x $channels$). The video cube slicing methods are shown in Figure 4-1. Within each slicing method, it is also possible to vary the sampling technique—how to select a given row/column/frame from those available. Figure 4-2 shows example sampling techniques along a slicing axis. One might presume that the center of the video is more likely to include the action occurring. When a camera operator films an action, it is likely that they are centering the action in the shot rather than intentionally keeping it on the edge of the frame (especially in internet-sourced low-quality videos such as those found in the

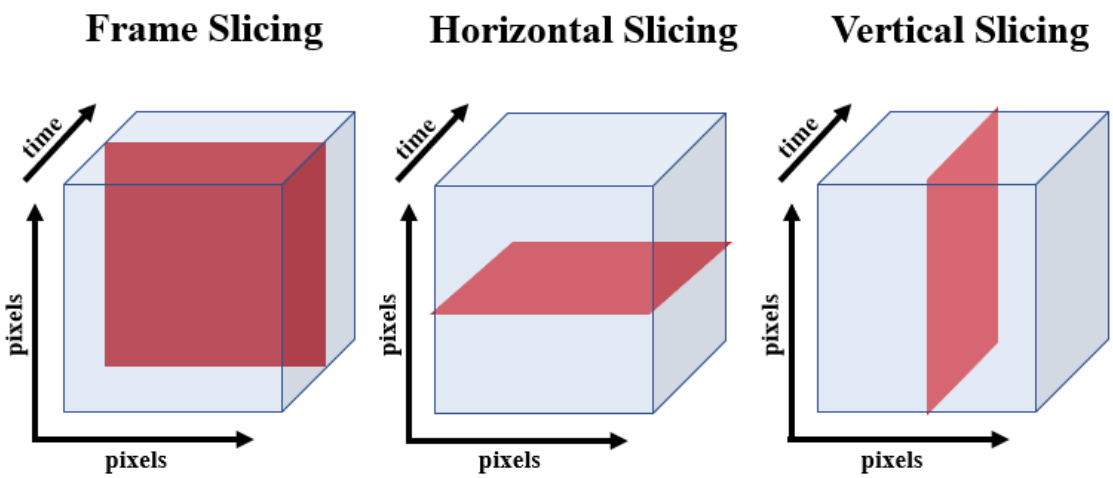

Figure 4-1: Video "cubes" are comprised of densely stacked frames. "Slices" can be taken (1) frame-wise by selecting a particular frame, (2) horizontally by selecting a particular row of pixels across all frames, or (3) vertically by selecting a particular column of pixels across all frames.

Moments in Time dataset).

Specifically, this study (1) compared horizontal and vertical cube slicing methods to the traditional frame-wise slicing in 2D CNN action recognition models and (2) compared six sampling techniques for determining where to slice the cube in each of the methods.

4.1 Experimental Design

Training was conducted using 2 NVIDIA Volta 100 GPU accelerators on a single compute node connected by PCIe. The node uses 2x14 Intel Xeon E5-2690v4 CPU cores with 512 GB of RAM and 2 TB of local disk storage. Each validation run was conducted on a single compute node using 2x16 AMD Opteron CPU cores with a total of 128 GB of RAM and 8TB of local disk storage. Scripts were written in Python 3.6.5 and distributed training used Horovod 0.16.1 [63]. Key package versions used were NumPy 1.14.3 [56], H5Py 2.7.1 [13], SciPy 1.1.0 [78], TensorFlow 1.13.1 [1], and Pillow 5.1.0.

Due to the limited computational resources available during this study, only 10

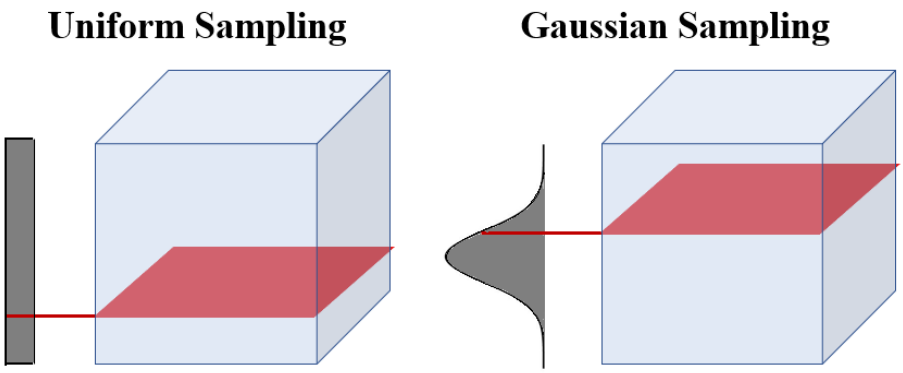

Figure 4-2: Examples of slice sampling techniques that can be applied along any axis (slicing method) of the video cube. The left shows uniformly sampling across the axis while the right shows preferentially sampling the center of the axis using a Gaussian distribution.

classes (applauding, baking, crashing, descending, eating, flooding, guarding, hitting, inflating, jumping) of the 339 in the dataset were used. Therefore, this project made the simplifying assumption that the results obtained from these slicing and sampling comparisons will correlate with performances attained on the entire dataset. This assumption is reasonable because of the high interclass variance in the Moments in Time dataset [54]. Given this simplification, the training set includes 27,494 videos and the validation set includes 1,000 videos. In their parsed form, the videos from these 10 classes are over 1.4 Terabytes of data.

A feature embedding for each video frame slice, horizontal slice, and vertical slice was created by passing each slice through an ImageNet pretrained VGG networks with the top dense layers removed. This results in a (1 x 7068) vector for each horizontal or vertical slice and a (1 x 25088) vector for each frame.

Two model architectures were used. The first was the standard VGG-top layers (two 4096 unit fully-connected layers followed by a 10 unit fully connected layer with softmax output). The second was the same VGG but with two dropout layers inserted ($p = 0.5$) after each 4096 unit fully-connected layer. Both models were initialized with random weights. This model is displayed in Figure 4-3.

For each model and slicing technique, the network was trained by sampling a slice from the video. Six sampling techniques were tried for each model and slicing

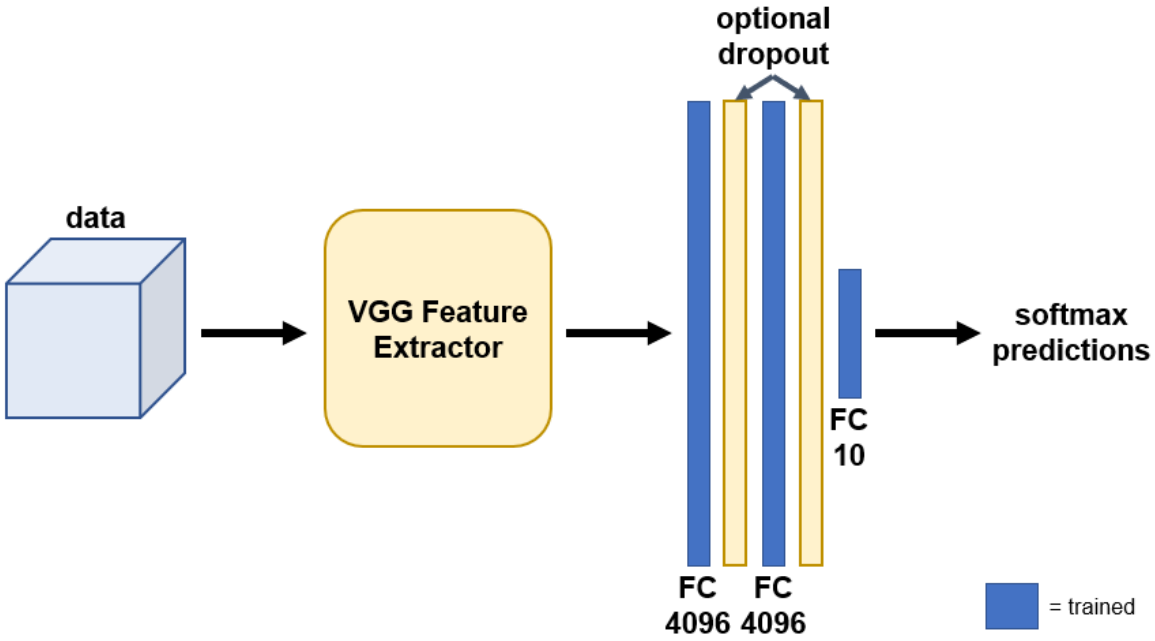

Figure 4-3: Video slices are first passed through an ImageNet-pretrained VGG Feature Extractor which creates embedded feature vectors used as inputs to a three layer network consisting of two 4096-unit fully connected layers followed by a 10-unit fully connected layer that outputs softmax predictions. The second version of the model includes two dropout ($p = 0.5$) layers in between the fully connected layers.

method. Slices were sampled at random from either a Uniform distribution over the pixels rows/cols or frames (depending on slicing method) or from a Gaussian distribution centered on the center-row/col or center frame with a standard deviation (σ) of 5, 10, 20, 30, or 40. These distributions when applied to horizontal/vertical slicing are shown in Figure 4-4.

A Horovod wrapped distributed stochastic gradient descent (SGD) optimizer and categorical cross-entropy loss metric update the network weights. The learning rate starts at 0.01 and is decayed by a factor of 10 at 30 epochs. It is also adjusted during the first five (warmup) epochs with respect to the number of processes launches. Standard momentum of 0.5 is used. Each model is trained for 50 epochs (or for 12 hours, whichever occurs first) with a batch size of 32 per process yielded an effective batch size of 64. Of the 36 training jobs, only three terminated before completing 50 epoch (at the 49 epoch point). Validation video level inference is the averaged prediction of

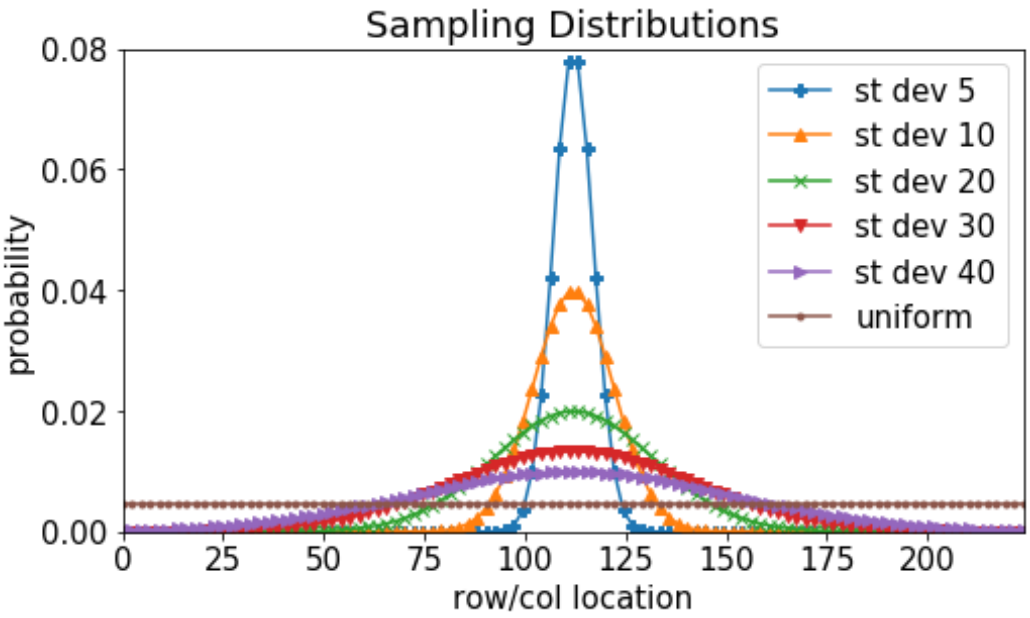

Figure 4-4: Visualization of sampling distributions tested in this study. Here, they are centered on 112 which is the center row/column on the video cube when sliced horizontally or vertically.

6 slices sampling using the same sampling technique employed for training.

4.2 Accuracy Performance Results

The standard VGG-top model outperformed the modified model with dropout across all slicing methods in both best and average sampling accuracy performance. This indicates the models are not overfitting the training data, and the addition of dropout for regularization is unnecessary in this context. Therefore, the results displayed in the p-ROC curves in Figure 4-5 as well as expanded in Appendix A Table A.3 are for training conducted in the model without dropout.

Across all accuracy metrics (top-1, top-5, p-ROC AUC, and J_{max}), frame slicing methods outperformed horizontal and vertical slicing. This is to be expected for two reasons. First, the VGG net that was used to extract the embedded features was pretrained on images, not other spatio-temporal video slices. Second, each frame slice has (224 x 224) = 50176 pixels while each horizontal or vertical slice has (224

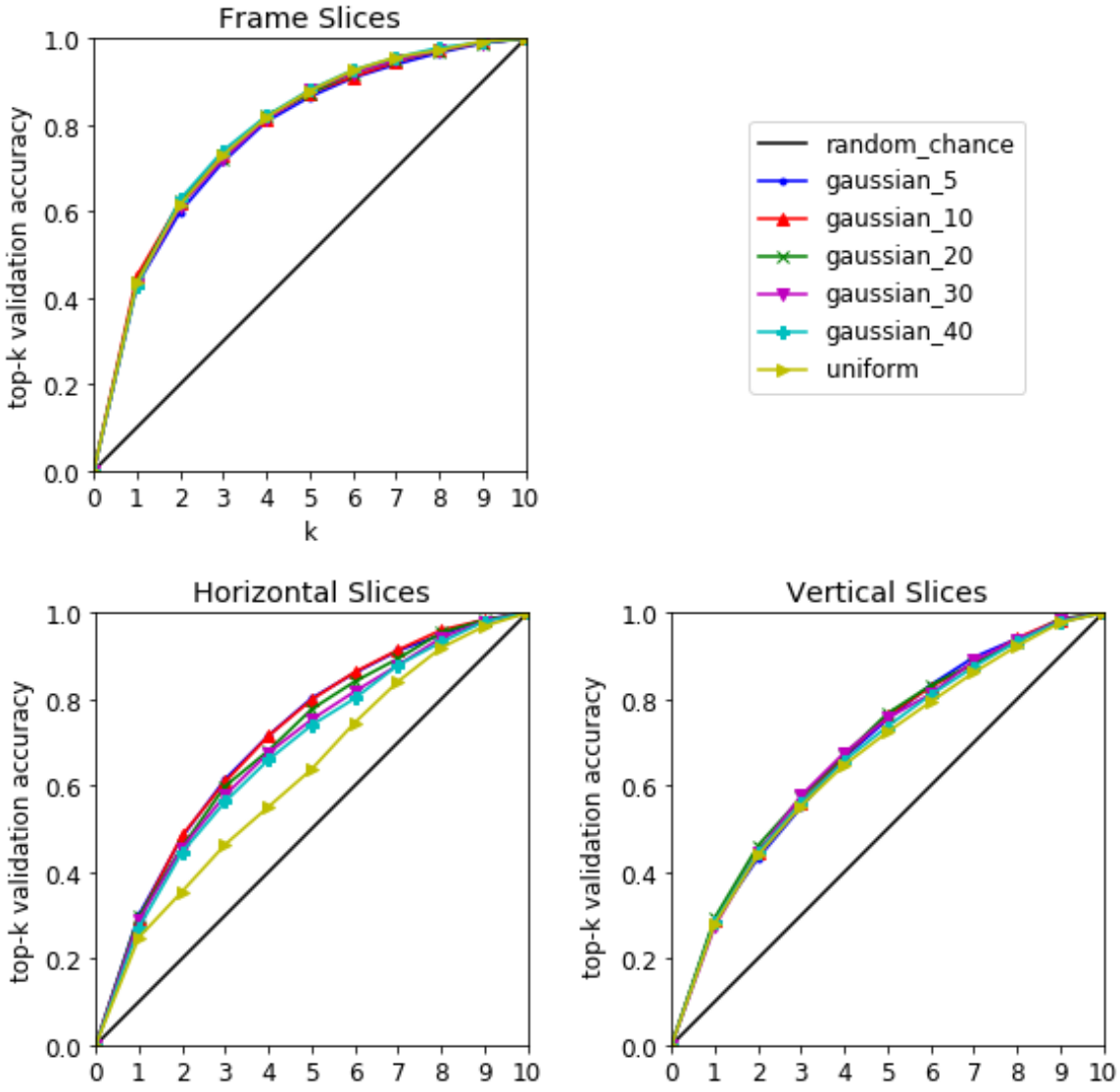

Figure 4-5: p-ROC validation curves for slicing and sampling on the best performing architecture. With the frame slicing method, there was little to no difference between sampling techniques. Horizontal slicing method shows the most significant difference between sampling techniques.

x 90) = 20160 pixels. Frames simply hold more data. However, the reasonable performance of learning on horizontal and vertical slices relative to frame slices shows that powerful transferability of the convolutional filters of the original VGG feature extractor. Additionally, horizontal slicing marginally outperformed vertical slicing with the best p-ROC AUCs of 7.12 and 6.90, respectively.

For frame slicing, wide Gaussians and uniform sampling performed the best. For horizontal slicing, narrow and centrally located Gaussians performed the best. For vertical slicing, Gaussian sampling methods outperformed uniform sampling methods, but only marginally. Therefore, sampling method matters the most when training on horizontal slices, and the center of the video is the most relevant area to select these slices. This is to be expected if the videos are captured with the person shooting the video keeping the action near the center of the frame.

4.3 Computational Performance Results

As shown in Figure 4-6, frame, horizontal, and vertical slicing training jobs averaged 14.28, 5.63, and 5.60 minutes per epoch, respectively. The frame slicing method also showed the largest variation in training time per epoch. In order to achieve such small training times, the embedded feature vectors for all frame, vertical, and horizontal slices had to be precomputed which required approximately one week on 60 Xeon-e5-2650 cores operating in parallel.

4.4 Discussion and Conclusions

The results of this experiment demonstrated that frame-wise video slicing for action recognition is not the only viable way to approach spatio-temporal learning using 2D convolution. However, the sampling method matters significantly more for horizontal and vertical slices than frame slices so careful consideration should go into this model training aspect.

Also, this study empirically shows that the best Top-1 accuracy and Top-5 accu-

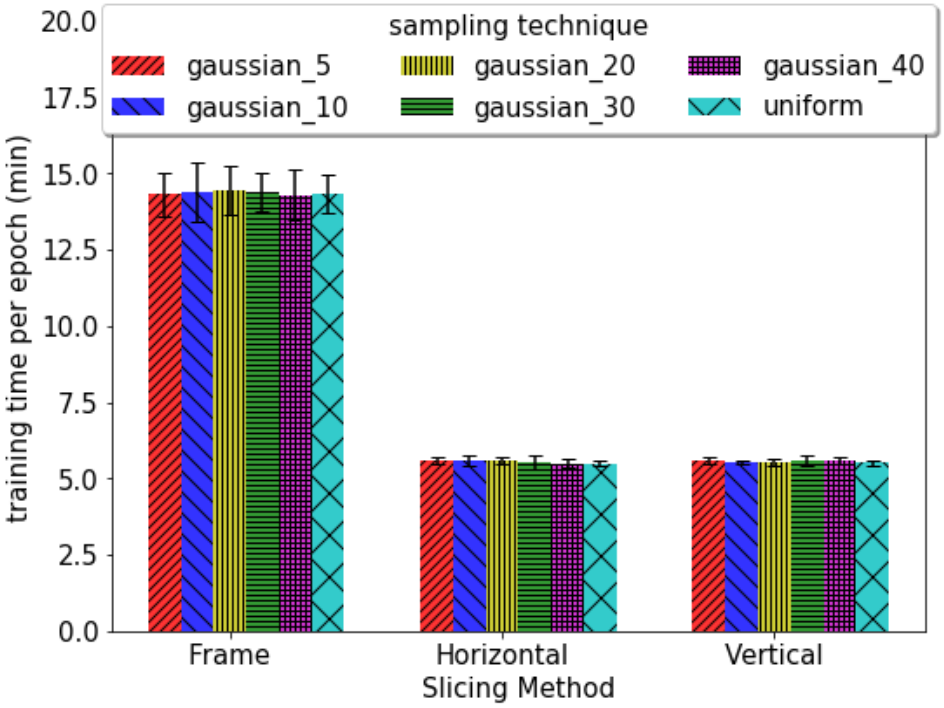

Figure 4-6: Average training time per epoch (in minutes) for each slicing method and sampling technique. Error bars indicate standard deviation across epochs. Training times shown are using 2 NVIDIA Volta V100 GPUs with PCIe connection.

racy are not always aligned with the best p-ROC AUC and J_{max}. Therefore, p-ROC curves and metrics that can be derived from it have validity to their usefulness in assessing action recognition model accuracy performance.

Chapter 5

Experimentation with Early Fusion

After creating a parse-wrangle-train-analyze pipeline, conducting an initial benchmarking study, and exploring the potential for alternate video slicing and sampling methods, the foundation was set for incorporating audio to make the models multimodal. This set of studies explored two audio-video early fusion techniques applied in a variety of ways.

5.1 Audio Representations

Intuitively, one way to bridge the gap between the effectiveness of 2D convolution with spatial or spatio-temporal images and the inclusion of audio in these videos is to convert the audio into an image. The method used in these studies was to convert the raw audio waveform input into a Mel-Frequency Spectrogram. This is a four step process nicely explained in [21] and summarized below:

1. Load the raw audio waveform.

2. Compute the fast fourier transform (FFT) on a rolling window to convert from the time domain to the frequency domain for each window.

3. Transform the frequency axis into a mel axis.

4. Decompose the magnitude into the individual mels and log scale them.

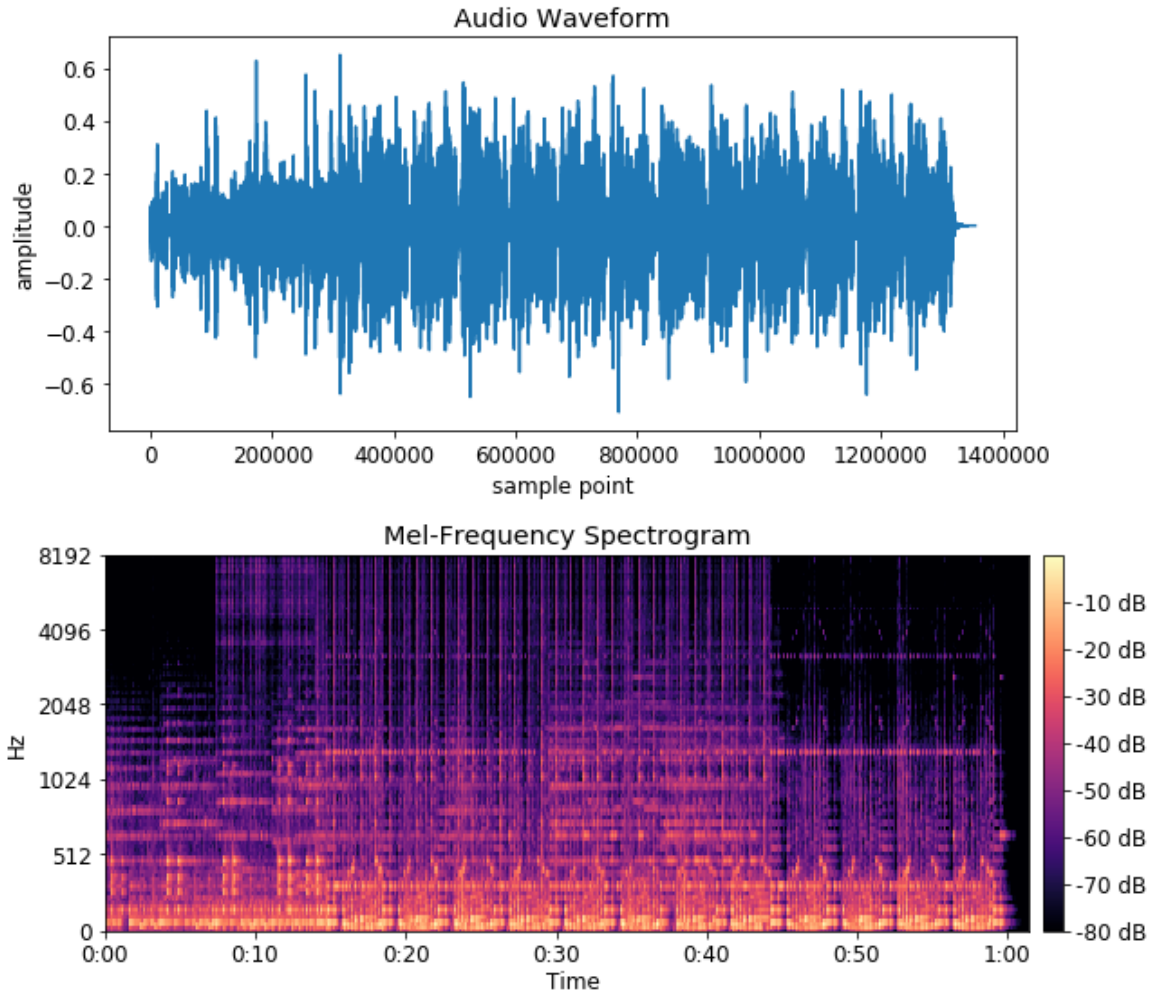

Figure 5-1: The top diagram shows an example audio waveform with a sample rate of 22050. The bottom diagram shows the transformation to a Mel-Frequency Spectrogram with a log power scale, 128 mels, and a maximum frequency of 8192 Hz.

The mel scale is a non-linear transformation on frequency to account for human perception of the difference between frequencies [70]. 1000 mels corresponds to 1000 Hz as the reference point. Equal differences in perceived pitch correspond to equal mel distances. An example audio transformation from waveform to Mel-Frequency Spectrogram is shown in Figure 5-1. Mel-Frequency Spectrograms in these studies were created using Librosa 0.7.1 [53]. Note that while it is displayed in color to highlight the power spectrum difference, these spectrograms are effectively single channel images.

The most common other audio transform undertaken to convert the audio into an image is the mel-frequency cepstral coefficients (MFCC) which has one addition step after creating the Mel-Frequency Spectrogram. It requires taking the discrete cosine transform (DCT) of the mel log powers. This project uses Mel-Frequency Spectogram instead of MFCC because the literature indicates that they lead to better learning with CNNs used for classification [35].

5.2 Early Fusion Methods

These studies then fused the audio spectrogram images in one of two ways which will be referred to as stitching or stacking. Not including the audio was used as the baseline to compare the stitching and stacking early fusion methods.

At a high level, stitching refers to concatenating images side-by-side. Given an RGB video slice ($pixels_0$ x $pixels_1$ x 3), the audio spectrogram (dim_0 x dim_1 x 1) is then duplicated three times to ensure both are three-channel images. The two images are then concatenated into a larger image with dimensions (max($pixels_0, dim_0$) x ($pixels_1$+dim_1) x 3). By carefully crafting the spectrogram, it is possible to ensure that $pixels_0 = dim_0$; however, when they are not equivalent sizes, the smaller image can be stretched for the stitching. The resulting image is larger than either original image. A visual description of training via stacking audio-video early fusion can be found in Figure 5-2.

Stacking refers to concatenating the 3-channel RGB images with the 1-channel spectrogram along the channels dimension. In essence, the spectrogram is made into the fourth "color" channel. Given an RGB video slice ($pixels_0$ x $pixels_1$ x 3) and an audio spectrogram (dim_0 x dim_1 x 1), the resulting early fusion stack has dimensions ($pixels_0$ x $pixels_1$ x 4). Note that this requires $pixels_0 = dim_0$ and $pixels_1 = dim_1$. The resulting image is only larger along the channels axis. A visual description of training via stacking audio-video early fusion can be found in Figure 5-3.

An advantage of the stitching fusion method over stacking fusion is that models can benefit from pretraining, such as on ImageNet. CNN model weights are not

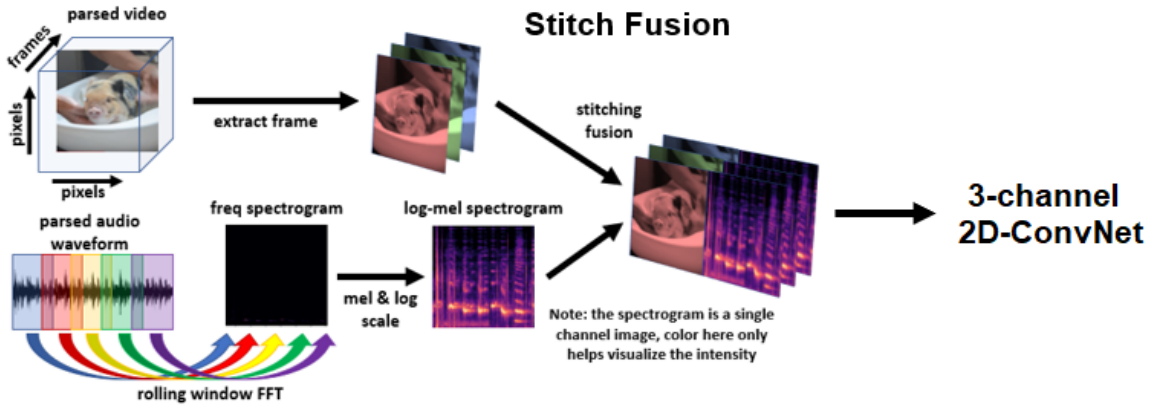

Figure 5-2: Process of stitching early audio-video fusion between an RGB frame and the log-mel spectrogram. The input to the ConvNet is a wider 3-channel image.

dependent on the size of the the input image, but rather on the number of channels. Therefore when using the stacking fusion method, models weights must be randomly initialized. An advantage to stacking is the ability to align domains. With horizontal and vertical slices, the RGB images are spatio-temporal features. The time axis of one of these slices can be aligned to the time axis of the spectrogram which could lead to training benefits as audio and video.

5.3 10-Class Experiment

This study was conducted with only 10 classes (applauding, baking, crashing, descending, eating, flooding, guarding, hitting, inflating, jumping) in order to test the greatest spread of models, early fusion methods, and hyperparameters with the available computational resources in a reasonable amount of time. Therefore, as in the slicing and sampling study described in Chapter 4, the training set used for this study includes 27,494 videos and the validation set includes 1,000 videos.

5.3.1 Experimental Design

Training and validation were both conducted using 8 NVIDIA Volta 100 GPU accelerators across compute nodes. Scripts were written in Python 3.6.9 and distributed

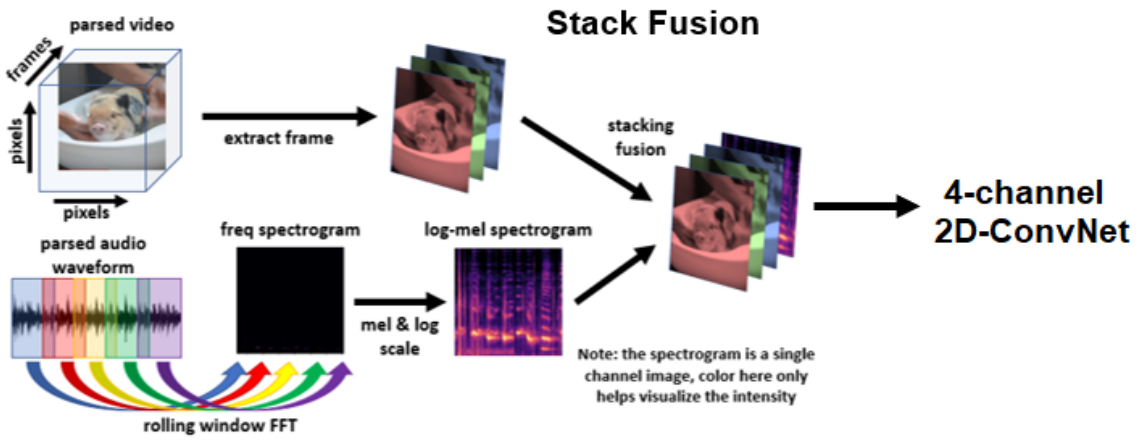

Figure 5-3: Process of stacking early audio-video fusion between an RGB frame and the log-mel spectrogram. The input to the ConvNet is a 4-channel image with the same pixel height and width as the original frame.

training used Horovod 0.18.2 [63]. Key package versions used were NumPy 1.16.5 [56], H5Py 2.9.0 [13], SciPy 1.3.2 [78], and TensorFlow 1.14.0 [1].

Two model architectures (Inception-ResNet-v2, Inception-v3), three fusion methods (none, stitching, stacking), three slicing methods (frame, vertical, horizontal), and two sampling techniques (uniform, Gaussian ($\sigma = 30$)) were tested. Therefore, there were 36 training combinations. Models were initialized with ImageNet pretrained weights when the model input shape had only three channels (i.e. without fusion or with stitching fusion). Otherwise, model weights were randomly initialized.

For each model and fusion method, the network was trained by sampling a slice from the video and fusing as specified. Validation is performed in the same manner.

A Horovod wrapped distributed stochastic gradient descent (SGD) optimizer and categorical cross-entropy loss metric updated the network weights. The learning rate was set at 0.01 and was adjusted during the first five (warmup) epochs with respect to the eight processes launches. Nesterov momentum was used. Each model was trained for 30 epochs with a batch size of 32 per process yielding an effective batch size of 256.

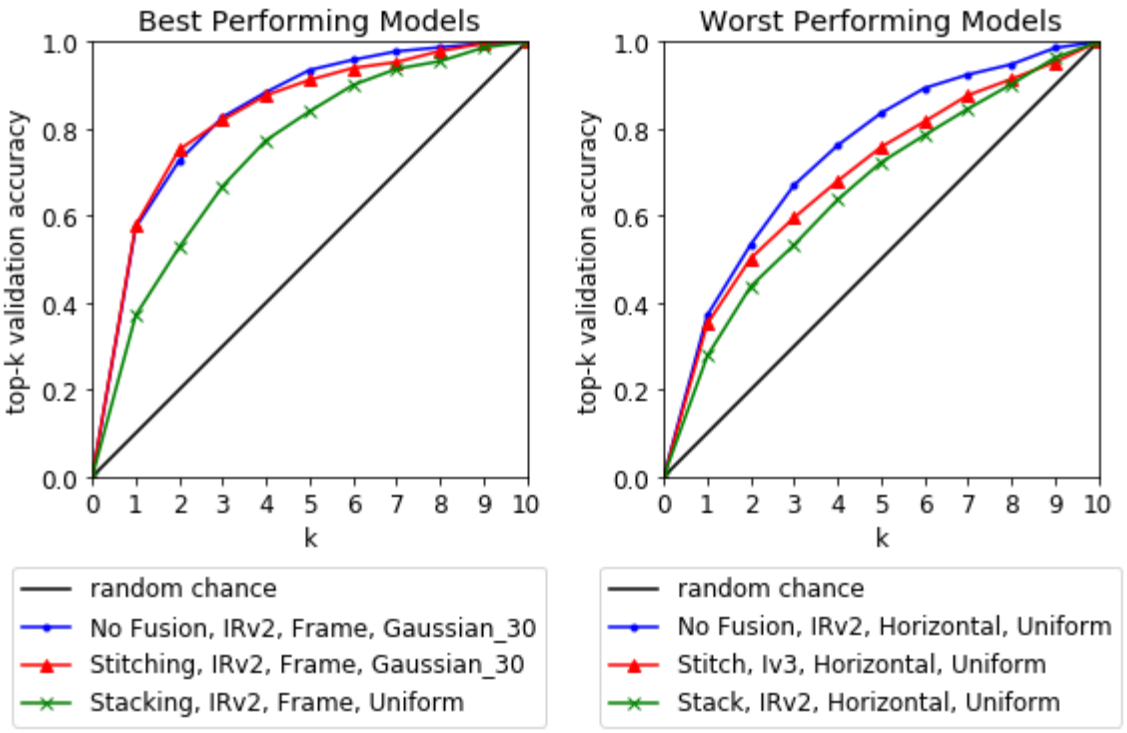

Figure 5-4: p-ROC curve for best and worst performing models using no fusion, stitching fusion, and stacking fusion. "Best" and "Worst" in this figure are referring only to the highest and lowest p-ROC *AUC* values.

5.3.2 Accuracy Performance Results

Figure 5-4 displays the p-ROC curves for the best and worst performing combinations of backbone models and hyperparameters for each of the early fusion methods. Full validation accuracy performance results can be found in Appendix A, Table A.4. Stitching fusion outperformed stacking fusion across almost all training configurations; however, it failed to show significant improvement over the no fusion baseline. The highest top-1 accuracy (made by stitching fusion with an Inception-ResNet-v2 backbone, frame slicing, and uniform sampling) was 62.7%. By comparison, the highest no fusion top-1 accuracy achieved was 57.3%. However, that improvement is less clear when looking at p-ROC *AUC*s and J_{max} values (of which the best values are split between stitching fusion and no fusion).

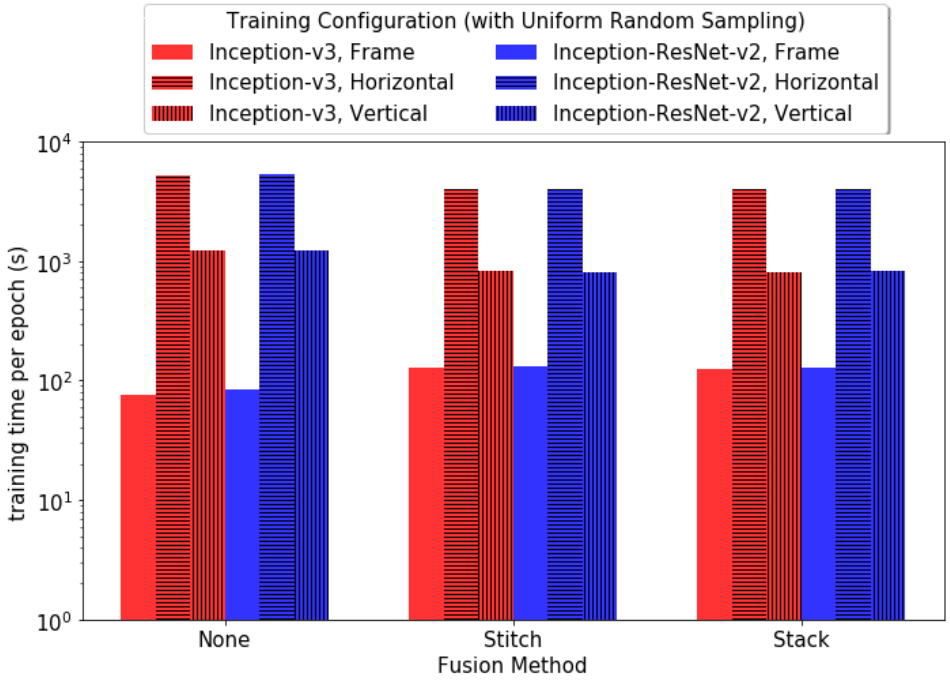

Figure 5-5: 10-Class Fusion Experiment Computational Performance Results when training on 8 NVIDIA Volta 100 GPUs. Note that training time per epoch is plotted on a log scale as horizontal and vertical slicing took significantly longer than frame slicing. Results with Gaussian ($\sigma = 30$) sampling are analogous and the plot can be found in Appendix B, Table B-1.

5.3.3 Computational Performance Results

Full training time and training time per epoch results can be found in Appendix B, Table B.5. Overall, stitching and stacking fusion methods had little effect on computational performance because the video cube slicing became a bottleneck. Because of this bottleneck, training times indicated that frame slicing was occurring quicker than vertical slicing and significantly quicker than horizontal slicing. Because the main training time bottleneck was on video slicing, almost no difference was observed between model backbones either as they would wait on batches to be created.

5.4 339-Class Experiment

This study was conducted with the full 339 Moments in Time classes in order to verify the results hinted at in the 10-Class experiment and test new and larger multi-slice

model architectures. Because of the benefits of pretraining obtained when looking at the frame slicing over horizontal slicing, frame slices with uniform sampling were the primary features used in this fusion experiment.

5.4.1 Experimental Design

Training and validation were both conducted using 64 NVIDIA Volta 100 GPU accelerators across compute nodes. Language and package versions used in this study were the same as described in Section 5.3.1.

Eleven models were also tested with stitching fusion and stacking fusion. The first four have ResNet50, Xception, Inception-v3, and Inception-ResNet-v2 backbones and fuse the uniformly sampled frame slices with the audio mel spectrograms. The fifth model, named Cross-v1-X in this project, samples uniformly randomly both a horizontal and vertical slice, fuses the mel spectrogram to both, runs each through an Xception backbone model, and concatenates their logits as an input to a 339 neuron fully-connected layer that yields a softmax output. The sixth model, named Cross-v2-X-IRv2 in this project, is similar to Cross-v1-X; however, it also uniformly randomly samples a frame slice, passes that through an Inception-ResNet-v2 backbone model, and concatenates the logits to the Xception models logits prior to the input to the fully-connected layer. Cross-v1-X and Cross-v2-X-IRv2 are described further in Figures 5-6 and 5-7. The last five models were ResNet50, Xception, Inception-ResNet-v2, MobileNet-v2, and DenseNet201 which had slightly different training sessions than the first four models listed here. They were also converted into 6-frame TSNs for validation video-level inference.

The first six models trained with a Horovod-wrapped distributed stochastic gradient descent (SGD) optimizer and categorical cross-entropy loss metric that updated the network weights. The learning rate was set at 0.01 and was adjusted during the first five (warmup) epochs with respect to the eight processes launches. Nesterov momentum was used. Each model was trained for 50 epochs with a batch size of 32. The five models to be used for TSN 6-frame averaging inference used a Horovod-wrapped ADADELTA optimizer and trained for 65 epochs instead of 50.

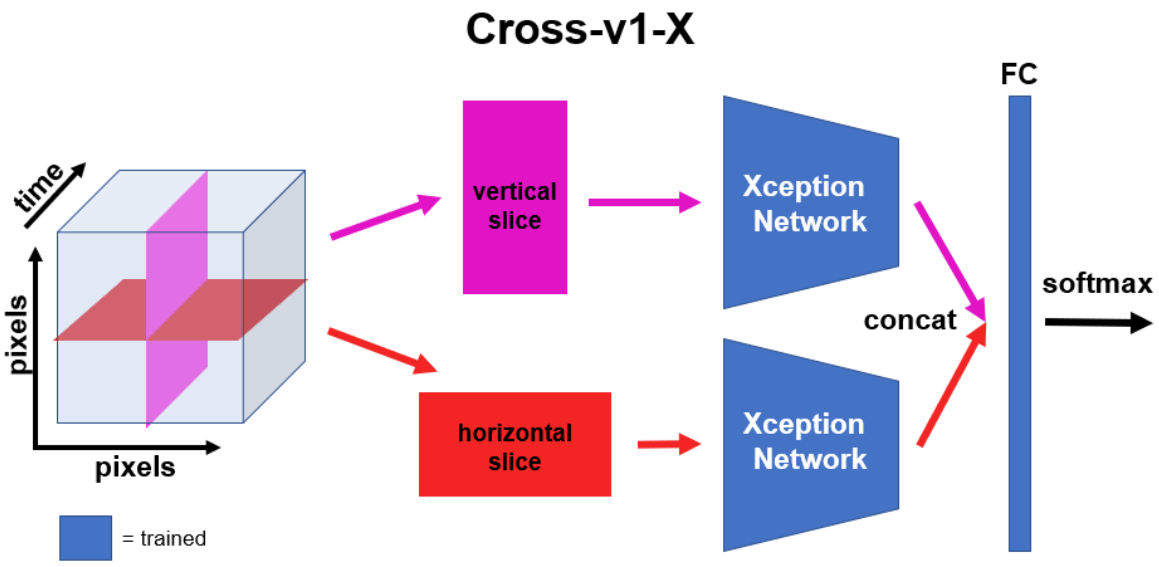

Figure 5-6: The Cross-v1 model architecture that utilizes vertical and horizontal video slices.

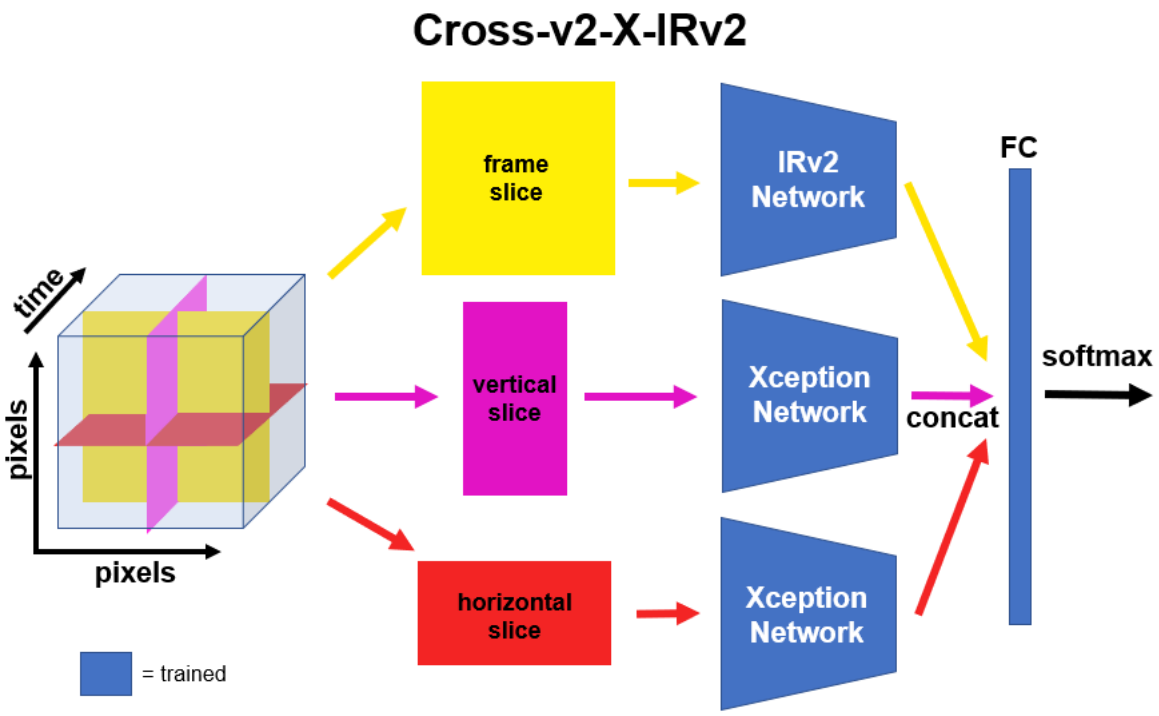

Figure 5-7: The Cross-v2 model architecture that utilizes frame, vertical, and horizontal video slices.

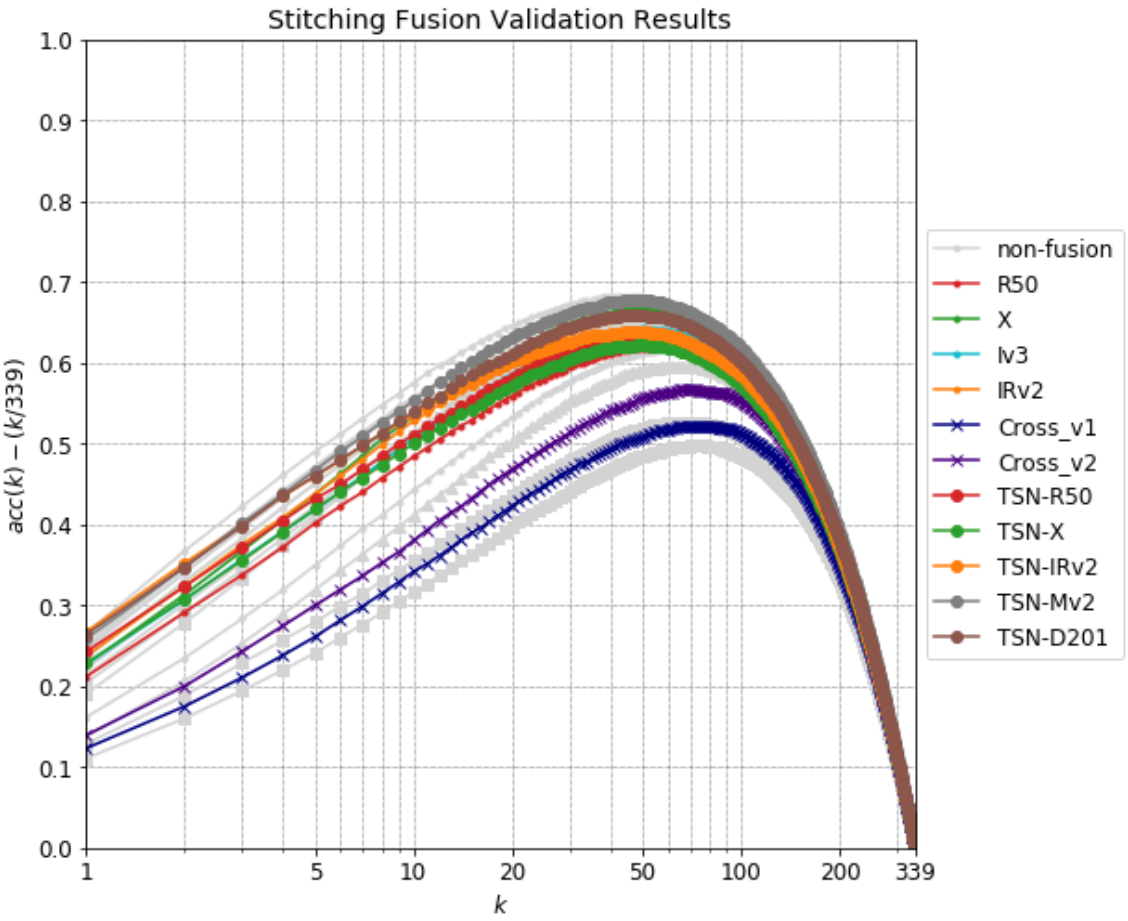

Figure 5-8: 339-class experiment stitching fusion validation results. The colored markers, as labeled in the legend to the right, indicate the various stitching early fusion models. Grayed-out models dots respond to the non-fusion baselines described in Chapter 3.

5.4.2 Accuracy Performance Results

Figures 5-8 and 5-9 display the modified p-ROC curves for stitching and stacking fusion methods across the variety of models trained. Full validation accuracy performance results can be found in Appendix A, Table A.5.

Overall, it is clear that stitching fusion is outperforming stacking fusion likely due to the benefits of pretraining. As expected, across both stitching and stacking, TSN validation-level inference improved accuracies by approximately 3-4%. While Top-1 accuracies showed some minor improvements via stitching over a non-fusion baseline, p-ROC AUC_{norm} and J_{max} clearly indicated that audio-video fusion did

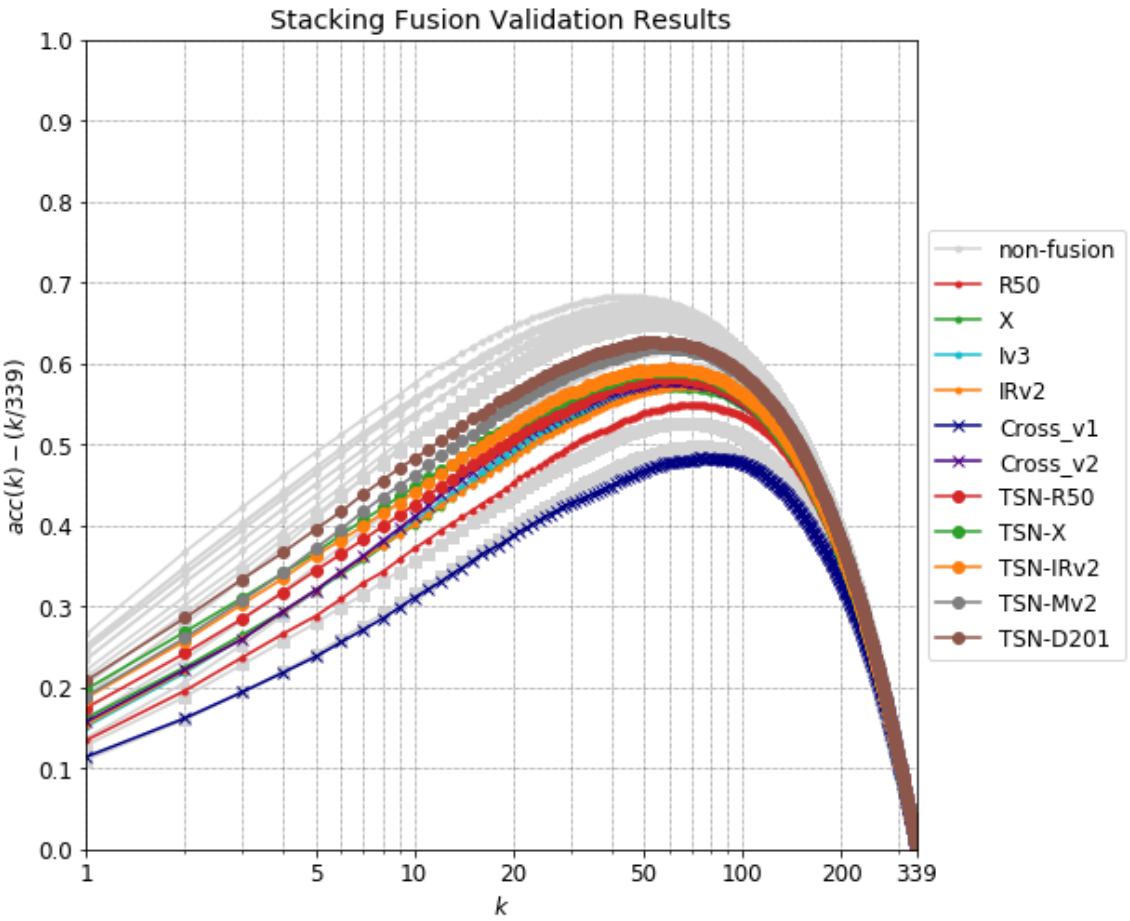

Figure 5-9: 339-class experiment stacking fusion validation results. The colored markers, as labeled in the legend to the right, indicate the various stitching early fusion models. Grayed-out models dots respond to the non-fusion baselines described in Chapter 3.

not produce better models. The top performing stitching fusion model was the TSN-enabled MobileNet-v2 with a p-ROC AUC_{norm} of 0.917. The top performing stacking fusion model was the TSN-enabled DenseNet201 with a p-ROC AUC_{norm} of 0.895. Both Cross-v1 and Cross-v2 underperformed the 2D frame slicing models across all accuracy metrics.

5.4.3 Computational Performance Results

Computational performance via training time per epoch is shown in Figures 5-10 and 5-11, and detailed results can be found in Appendix B, Table B.6. All stitching and

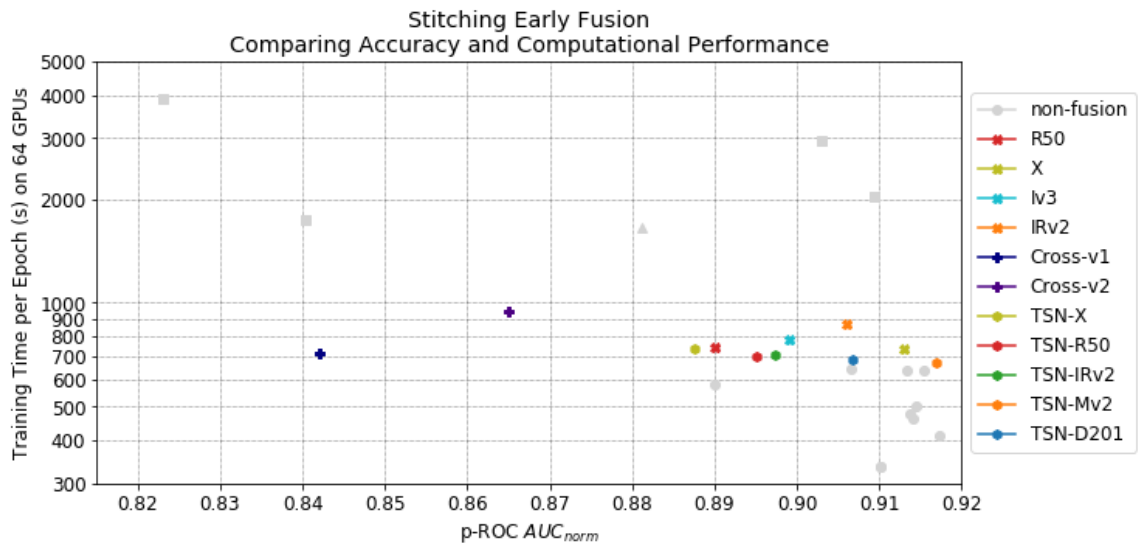

Figure 5-10: A comparison of stitching early fusion accuracy and computational performance when trained on 64 Volta V100 GPUs (2 per node). The colored markers, as labeled in the legend to the right, indicate the various stitching early fusion models. Grayed-out models dots respond to the non-fusion baselines described in Chapter 3.

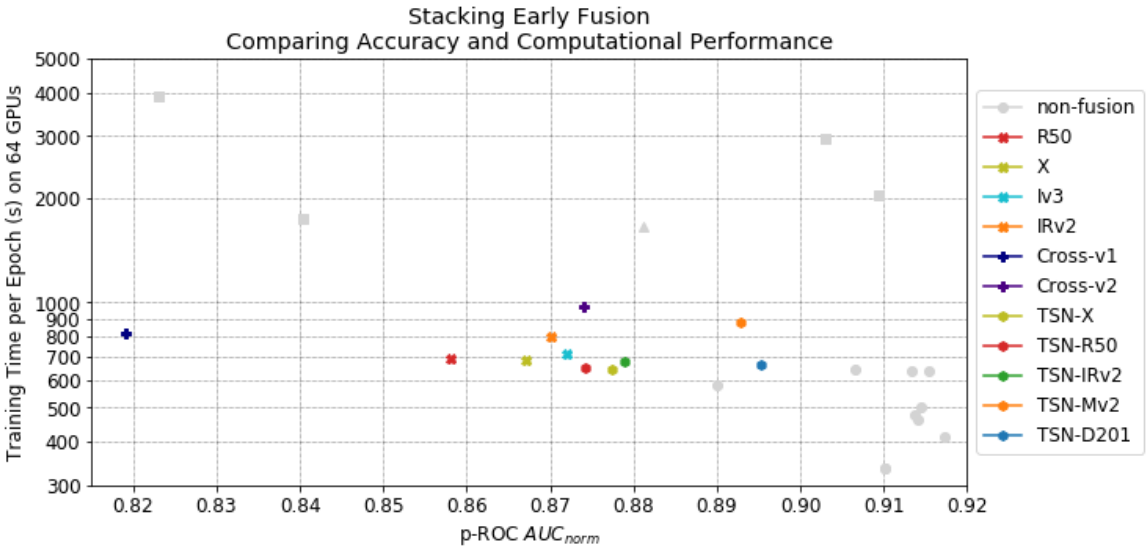

Figure 5-11: A comparison of stacking early fusion accuracy and computational performance when trained on 64 Volta V100 GPUs (2 per node). The colored markers, as labeled in the legend to the right, indicate the various stitching early fusion models. Grayed-out models dots respond to the non-fusion baselines described in Chapter 3.

stacking fusion models had training times per epoch between 600 and 1000 seconds and averaged around 50% longer to train than their non-fusion baselines. This com-

putational cost was expected due to the burdens of performing the early fusion prior to model input as well as the increased size of the model inputs. Across both fusion methods, Cross-v2, the most complex of models trained in this experiment, took the longest to train.

5.5 Discussion and Conclusions

Unfortunately, early fusion did not produce any significant benefits over non-fusion or late fusion methods. However, there are several important takeaways from these two early fusion studies:

1. The application of stitching fusion may appear to yield moderate accuracy performance improvements over similar non-fusion methods when viewing Top-1 accuracy; however, other accuracy metrics with a broader view of the model, such as p-ROC AUC_{norm}, demonstrate no improvement. Stacking fusion yields worse results than non-fusion likely due to the lost benefit of ImageNet-pretrained weights.

2. Among 2D model backbones, Inception-ResNet-v2 continues to yield the highest top-1 validation accuracies; however, accuracy performance comparison along p-ROC AUC and J_{max} indicate that DenseNet and MobileNet-v2 may actually be higher performing models. This also leads further credence to the claim that Top-1 and Top-5 accuracy should not be the defining accuracy performance metrics in video action recognition.

3. More complex models such as Cross-v1 and Cross-v2 are not effective at overcoming the challenges faced by vertical and horizontal slicing methods. With significantly more parameters, they are likely overfitting the training data leading to decreased accuracy performance. Model depth, not feature input width, is the key to action recognition success.

4. By utilizing a large number of GPUs for each training session, total training times were reduced from days and weeks to mere hours. Therefore, HPC re-

sources will continue to be necessary to perform these large comparison tests to make incremental progress in the action recognition field.

Chapter 6

Conclusion

This project has presented a comparison of some existing action recognition approaches (Chapters 2 and 3) and tested two novel video-audio early fusion methods (stitching and stacking) across a variety of avenues: slicing methods, sampling techniques, model backbones, etc. (Chapters 4 and 5).

Additionally, this project has addressed several issues within the current action recognition literature. Mainly, the current literature generally lacks:

1. adequate descriptions of important training details including both hardware and hyperparameter selections necessary for independent verification of state-of-the-art results. Papers and reports focus heavily on model architecture often to the detriment of these other important details.

2. any comparison of computational performance. Due to the data challenges associated with video, computational performance should be a necessary component of any action recognition model evaluation.

3. creativity to use accuracy metrics beyond Top-1/Top-5 which have become a de facto standard in the field.

4. a true "apples-to-apples" comparison. Few papers and reports show more than one or only a few model backbones tested with the same training settings.

5. any attempts at video-audio early fusion with the Moments in Time dataset. More broadly, late-fusion (ensemble) approaches dominate the field.

This project has addressed these gaps and issues in the literature by:

1. carefully noting key training details for each experiment. These include hardware specifications, software package versions, hyperparameter selections, and training/validation details.

2. introducing training time and training time per epoch as a key comparison metric of model computation performance in each experiment. As elaborated throughout thesis project, computational performance should be a necessary component of evaluating any video action recognition model.

3. introducing the novel p-ROC curves with AUC, AUC_{norm}, and J_{max} as alternative accuracy metrics. These experiments have demonstrated their potential including examples when p-ROC AUC does not directly correlate with the traditional Top-1/Top-5 accuracies.

4. applying a variety of "off-the-shelf" model backbones under very similar training and validation settings to make "apples-to-apples" comparisons. The HPC resources allowed a significant breadth of comparison.

5. experimenting with two early fusion approaches with a variety of training techniques. While neither showed significant improvements over non-fusion methods, it demonstrates that deeper model architectures are needed. Early fusion with current architectures is unlikely to produce any significant improvements.

Future work can extend this research in several ways. One such way to extend this research would be to utilize other video datasets to conduct additional pretraining. This would be particularly helpful for models used in these studies that do not have existing pretrained weight sets such as those needed for the 4-channel stacking fusion. Based on the conclusions listed above, another way to extend this research, particularly relevant to the early fusion, would be develop deeper new network architectures.

It is apparent that existing models, albeit useful, will not be able to tackle action recognition problem posed by the Moments in Time dataset.

Appendix A

Accuracy Performance Tables and Additional Figures

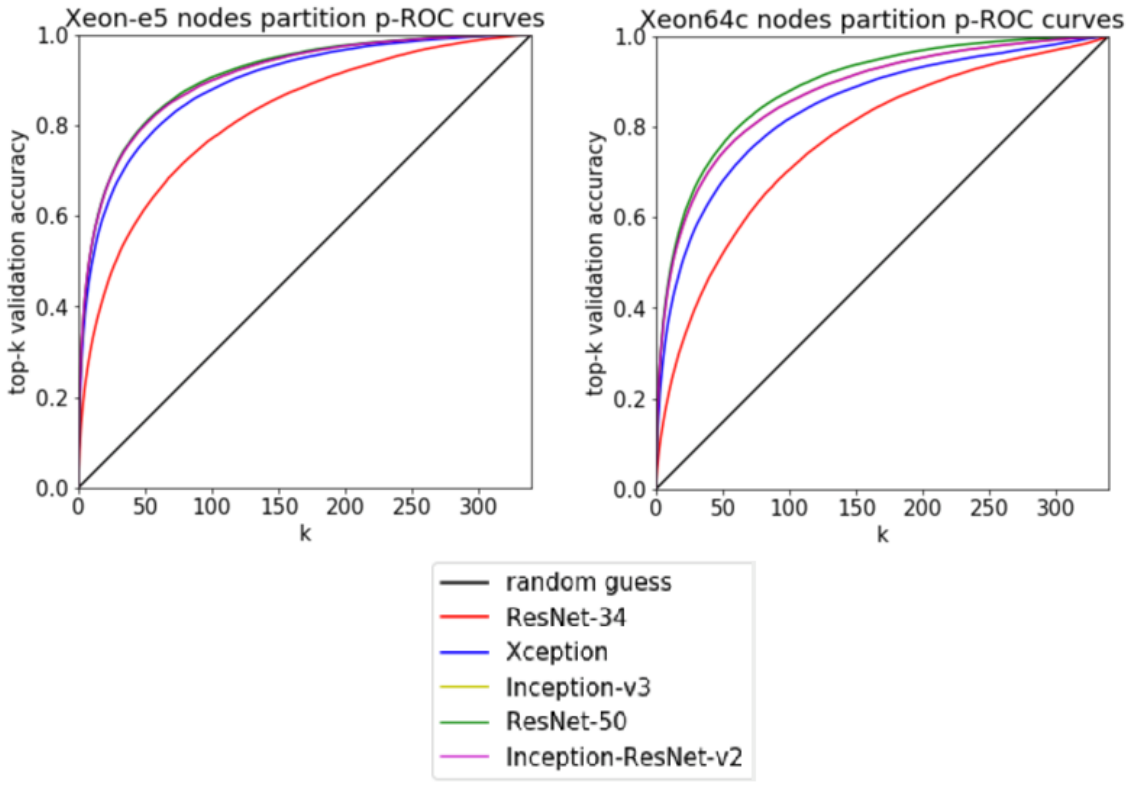

Figure A-1: p-ROC curves for CPU partitions trained models.

Table A.1: 2D Model Comparison of Accuracy Performance.

	Val Acc (%)		*p-ROC*		
Model	*Top-1*	*Top-5*	*AUC*	AUC_{norm}	J_{max}
random chance	*0.29*	*1.47*	*169.5*	*0.5*	*0.0*
effective batch size 256					
ResNet34	11.8	28.0	287.0	0.847	0.532 *(k=71)*
Xception	20.8	43.8	309.7	0.914	0.670 *(k=52)*
Inception-v3	22.0	45.8	309.7	0.914	0.674 *(k=50)*
ResNet50	21.6	45.0	308.7	0.911	0.664 *(k=51)*
Inception-ResNet-v2	**23.9**	**48.3**	**313.3**	**0.924**	**0.694** *(k=42)*
effective batch size 512					
ResNet34	8.5	22.5	277.2	0.818	0.485 *(k=72)*
Xception	17.0	38.2	302.1	0.891	0.621 *(k=57)*
Inception-v3	19.3	42.1	307.1	0.906	0.653 *(k=50)*
ResNet50	18.7	41.3	**307.9**	**0.908**	**0.658** *(k=52)*
Inception-ResNet-v2	**20.0**	**43.4**	307.1	0.906	0.653 *(k=50)*
effective batch size 1024					
ResNet34	4.4	13.9	260.7	0.769	0.412 *(k=92)*
Xception	10.2	27.5	285.9	0.843	0.542 *(k=69)*
Inception-v3	14.5	34.6	296.4	0.874	0.596 *(k=59)*
ResNet50	14.3	**41.3**	**301.5**	**0.889**	**0.617** *(k=59)*
Inception-ResNet-v2	**14.9**	35.0	296.4	0.874	0.596 *(k=59)*

Table A.2: Expanded Comparison - Accuracy Performance

		Val Acc (%)		*p-ROC*		
Type	*Backbone*	*Top-1*	*Top-5*	*AUC*	AUC_{norm}	J_{max}
random chance		0.29	1.47	169.5	0.5	0.0
C2D	VGG19	16.45	36.41	301.74	0.891	0.616 ($k = 59$)
	M	21.61	43.79	307.35	0.908	0.652 ($k = 51$)
	Iv3	24.87	48.20	309.61	0.915	0.673 ($k = 48$)
	R50	23.77	46.54	308.52	0.911	0.661 ($k = 47$)
	Mv2	22.42	45.16	309.86	0.915	0.670 ($k = 50$)
	X	24.84	47.58	310.02	0.919	0.683 ($k = 41$)
	IRv2	**26.83**	**50.50**	**310.99**	**0.919**	**0.683** ($k = 41$)
	D169	25.13	48.63	310.32	0.917	0.674 ($k = 45$)
	D201	25.52	48.62	309.76	0.915	0.672 ($k = 46$)
LRCN	n/a (16f)	14.04	33.40	298.75	0.883	0.596 ($k = 63$)
C3D	n/a (16f)	13.15	29.41	284.90	0.842	0.499 ($k = 74$)
	n/a (32f)	11.36	25.58	278.98	0.824	0.499 ($k = 74$)
I3D	Iv1 (16f)	19.33	42.36	308.26	0.911	0.661 ($k = 52$)
	Iv1 (64f)	20.69	42.74	306.10	0.904	0.649 ($k = 57$)

Legend:

C2D = Traditional 2D ConvNet
LRCN = Long-term Recurrent CNN
C3D = 3D ConvNet
I3D = Inflated 3D ConvNet

16f = 16 frame inputs
32f = 32 frame inputs
64f = 64 frame inputs

M = MobileNet
Iv3 = Inception-v3
R50 = ResNet50
Mv2 = MobileNetv2
IRv2 = Inception-ResNet-v2
D169 = DenseNet169
D201 = DenseNet201
Iv1 = Inception-v1

Table A.3: Exploration of Video Slicing and Sampling: validation results each slicing method (frame, horizontal, and vertical).

	Sampling Technique	*Val Acc (%)*		*p-ROC*		
		Top-1	*Top-5*	*AUC*	AUC_{norm}	J_{max}
	random chance	*10.0*	*50.0*	*5.0*	*0.5*	*0.0*
Frame	Gaussian, $\sigma = 5$	43.0	86.5	7.72	0.772	0.415 *(k=3)*
	Gaussian, $\sigma = 10$	**45.1**	87.4	7.81	0.781	0.431 *(k=3)*
	Gaussian, $\sigma = 20$	43.8	87.3	7.81	0.781	0.432 *(k=2)*
	Gaussian, $\sigma = 30$	43.1	88.0	7.79	0.779	0.421 *(k=3)*
	Gaussian, $\sigma = 40$	42.7	**88.1**	**7.85**	**0.785**	**0.441** *(k=3)*
	Uniform	43.6	87.9	7.83	0.783	0.430 *(k=3)*
Horizontal	Gaussian, $\sigma = 5$	30.1	**80.2**	**7.12**	**0.712**	**0.316** *(k=4)*
	Gaussian, $\sigma = 10$	29.3	79.9	**7.12**	**0.712**	0.315 *(k=4)*
	Gaussian, $\sigma = 20$	**29.8**	77.7	6.98	0.698	0.299 *(k=3)*
	Gaussian, $\sigma = 30$	28.9	75.3	6.87	0.687	0.278 *(k=3)*
	Gaussian, $\sigma = 40$	27.2	74.0	6.77	0.677	0.263 *(k=3)*
	Uniform	24.9	63.8	6.23	0.623	0.164 *(k=3)*
Vertical	Gaussian, $\sigma = 5$	28.7	75.3	6.83	0.683	0.265 *(k=4)*
	Gaussian, $\sigma = 10$	28.7	76.2	6.86	0.686	0.269 *(k=4)*
	Gaussian, $\sigma = 20$	**29.4**	**76.7**	**6.90**	**0.690**	0.272 *(k=3)*
	Gaussian, $\sigma = 30$	27.4	75.6	6.85	0.685	**0.277** *(k=3)*
	Gaussian, $\sigma = 40$	28.5	73.7	6.78	0.678	0.260 *(k=3)*
	Uniform	28.0	72.4	6.70	0.670	0.253 *(k=3)*

Table A.4: 10-Class Early Fusion Study Validation Results.

	Model	*Slicing Method*	*Sampling Technique*	*Val Acc (%) Top-1*	*Val Acc (%) Top-5*	*p-ROC AUC_{norm}*	J_{max}
	random chance			*10.0*	*50.0*	*0.5*	*0.0*
No Fusion	Iv3	Frame	Uniform	50.1	87.0	0.768	0.461 *(k=2)*
			Gaussian	54.4	88.9	0.784	0.495 *(k=2)*
		Horizontal	Uniform	25.3	70.1	0.639	0.229 *(k=3)*
			Gaussian	31.5	78.4	0.688	0.295 *(k=4)*
		Vertical	Uniform	33.5	78.3	0.695	0.328 *(k=3)*
			Gaussian	36.5	81.4	0.710	0.334 *(k=4)*
	IRv2	Frame	Uniform	55.2	93.0	0.806	**0.536** *(k=3)*
			Gaussian	**57.3**	**93.5**	**0.808**	0.527 *(k=3)*
		Horizontal	Uniform	25.3	67.7	0.627	0.227 *(k=3)*
			Gaussian	32.9	79.8	0.702	0.325 *(k=3)*
		Vertical	Uniform	37.1	83.6	0.724	0.371 *(k=3)*
			Gaussian	32.0	82.4	0.715	0.356 *(k=3)*
Stitch Method	Iv3	Frame	Uniform	52.5	88.3	0.781	0.511 *(k=2)*
			Gaussian	52.7	**91.2**	0.798	0.507 *(k=2)*
		Horizontal	Uniform	35.2	75.8	0.677	0.302 *(k=2)*
			Gaussian	38.1	82.4	0.718	0.364 *(k=3)*
		Vertical	Uniform	40.0	80.1	0.713	0.385 *(k=3)*
			Gaussian	34.6	77.0	0.678	0.304 *(k=2)*
	IRv2	Frame	Uniform	**62.7**	89.6	0.794	0.527 *(k=1)*
			Gaussian	57.8	**91.2**	**0.802**	**0.552** *(k=2)*
		Horizontal	Uniform	36.1	75.2	0.677	0.292 *(k=3)*
			Gaussian	36.7	74.6	0.679	0.317 *(k=3)*
		Vertical	Uniform	37.3	79.7	0.699	0.333 *(k=3)*
			Gaussian	43.8	82.0	0.728	0.396 *(k=2)*
Stack Method	Iv3	Frame	Uniform	30.0	78.1	0.681	0.305 *(k=5)*
			Gaussian	29.9	76.7	0.684	0.297 *(k=4)*
		Horizontal	Uniform	30.1	76.8	0.669	0.284 *(k=4)*
			Gaussian	31.2	75.8	0.672	0.280 *(k=2)*
		Vertical	Uniform	36.1	77.9	0.694	0.316 *(k=2)*
			Gaussian	32.4	77.3	0.687	0.296 *(k=2)*
	IRv2	Frame	Uniform	37.1	**84.0**	**0.727**	**0.373** *(k=4)*
			Gaussian	39.4	81.0	0.712	0.354 *(k=3)*
		Horizontal	Uniform	27.7	72.3	0.646	0.238 *(k=2)*
			Gaussian	34.6	77.0	0.686	0.304 *(k=2)*
		Vertical	Uniform	30.9	77.3	0.687	0.299 *(k=4)*
			Gaussian	**40.4**	83.0	0.718	0.360 *(k=4)*

Iv3 = Inception-v3
IRv2 = Inception-ResNet-v2

Table A.5: 339-Class Early Fusion Study Validation Results.

	Model	*Slicing Method*	*Val Acc (%) Top-1*	*Top-5*	*p-ROC AUC*	AUC_{norm}	J_{max}
	random chance		*0.3*	*1.5*	*169.5*	*0.5*	*0.0*
Stitch	R50	F	21.5	41.6	301.7	0.890	0.619 ($k = 55$)
	X	F	22.9	45.2	309.6	0.913	0.667 ($k = 52$)
	Iv3	F	23.0	43.3	304.6	0.899	0.643 ($k = 50$)
	IRv2	F	24.0	45.2	307.3	0.906	0.658 ($k = 51$)
	Cross-v1	H+V	12.6	27.6	285.4	0.842	0.522 ($k = 76$)
	Cross-v2	F+H+V	14.2	31.5	293.2	0.865	0.567 ($k = 70$)
	TSN-R50	6F	24.6	44.7	303.4	0.895	0.634 ($k = 50$)
	TSN-X	6F	23.2	43.4	300.9	0.888	0.622 ($k = 49$)
	TSN-IRv2	6F	**27.0**	47.4	304.2	0.897	0.639 ($k = 48$)
	TSN-Mv2	6F	26.1	**48.2**	**310.8**	**0.917**	**0.678** ($k = 48$)
	TSN-D201	6F	26.7	47.5	307.4	0.907	0.659 ($k = 51$)
Stack	R50	F	13.8	30.3	290.7	0.858	0.549 ($k = 71$)
	X	F	16.5	33.4	293.9	0.867	0.570 ($k = 60$)
	Iv3	F	15.4	33.4	295.6	0.872	0.580 ($k = 72$)
	IRv2	F	15.7	33.5	294.8	0.870	0.573 ($k = 66$)
	Cross-v1	H+V	11.7	25.2	277.8	0.819	0.484 ($k = 79$)
	Cross-v2	F+H+V	16.0	33.5	296.4	0.874	0.580 ($k = 57$)
	TSN-R50	6F	17.8	35.9	296.3	0.874	0.583 ($k = 61$)
	TSN-X	6F	20.1	38.1	297.5	0.877	0.592 ($k = 57$)
	TSN-IRv2	6F	19.0	37.8	298.0	0.879	0.594 ($k = 61$)
	TSN-Mv2	6F	19.2	38.6	302.7	0.893	0.621 ($k = 57$)
	TSN-D201	6F	**21.1**	**41.0**	**303.5**	**0.895**	**0.627** ($k = 53$)

F = Frame
6F = 6 evenly spaced Frames
H = Horizontal
V = Vertical

Appendix B

Computational Performance Tables and Additional Figures

Table B.1: Video and audio parsing computational performance per class (minutes).

		Video 90 fps			*Audio* 15.2KHz sr			*Audio* 37.9KHz sr			
set		*Extract*	*Transform*	*Save*	*Extract*	*Transform*	*Save*	*Extract*	*Transform*	*Save*	***Total***
trn	avg	39.5	159.6	10.8	10.9	0.7	3.2	11.4	1.0	3.4	240.5
	st dev	19.4	95.8	5.4	6.9	0.5	1.9	7.1	0.7	1.8	131.2
val	avg	0.64	0.55	0.12	0.50	0.03	0.13	0.50	0.04	0.13	2.63
	st dev	0.12	0.47	0.04	0.15	0.01	0.06	0.15	0.02	0.07	0.56

Table B.2: 2D Model Comparison of Computational Performance.

Model	*Total Training Time (s)*	*Per Epoch (s)* *AVG*	*SD*
effective batch size 256			
ResNet34	1315136	26302.72	4955.08
Xception	923615	18472.30	1241.91
Inception-v3	1118826	22376.52	6944.31
ResNet50	1169589	23391.78	3585.26
Inception-ResNet-v2	934996	18719.92	1100.65
effective batch size 512			
ResNet34	627129	12542.58	655.62
Xception	1036649	20731.98	693.79
Inception-v3	578917	11578.34	1881.93
ResNet50	937007	18740.14	998.07
Inception-ResNet-v2	1136617	22732.34	2105.14
effective batch size 1024			
ResNet34	491343	9826.86	1311.86
Xception	1119828	22396.56	1039.20
Inception-v3	793462	15869.24	969.98
ResNet50	1077449	21548.98	329.75
Inception-ResNet-v2	1285862	25717.24	612.71

Table B.3: Expanded Comparison - Computational Performance

		Training Time per Epoch (s) on g Volta V100 GPUs					
Type	*Backbone*	$g = 2$	$g = 4$	$g = 8$	$g = 16$	$g = 32$	$g = 64$
C2D	VGG19	15256.1	8235.5	4135.0	2109.6	1106.4	580.8
	M	15103.4	9693.4	4783.7	2585.2	895.6	642.5
	Iv3	20920.4	10238.6	5253.3	2890.2	1509.8	493.0
	R50	15261.6	9694.5	5298.6	2594.9	660.8	335.4
	Mv2	15103.4	9687.1	4783.7	2592.3	1017.8	460.7
	X	14997.3	8342.6	4177.8	2179.3	1120.7	502.1
	IRv2	15831.6	9694.6	5019.1	2619.3	1473.0	413.5
	D169	15399.2	9690.7	5019.1	3041.9	1627.4	666.6
	D201	15399.4	8687.1	5019.1	3041.9	1141.9	478.9
LRCN	n/a (16f)	37009.4	23740.9	10553.6	6388.7	2553.8	1835.8
C3D	n/a (16f)	41622.7	21823.9	11485.8	6195.0	2851.1	2107.2
	n/a (32f)	118738.2	64250.2	33911.3	18177.1	9505.5	5688
I3D	Iv1 (16f)	36838.8	20456.2	14182.7	6331.1	3567.3	2303
	Iv1 (64f)	85565.7	42697.6	23209.8	10864.0	5916.8	2991.8

Legend:
C2D = Traditional 2D ConvNet
LRCN = Long-term Recurrent CNN
C3D = 3D ConvNet
I3D = Inflated 3D ConvNet

16f = 16 frame inputs
32f = 32 frame inputs
64f = 64 frame inputs

M = MobileNet
Iv3 = Inception-v3
R50 = ResNet50
Mv2 = MobileNetv2
IRv2 = Inception-ResNet-v2
D169 = DenseNet169
D201 = DenseNet201
Iv1 = Inception-v1

Table B.4: Exploration of Video Slicing and Sampling: computational performance results for each slicing method (frame, horizontal, and vertical).

	Sampling Technique	*Total Training Time (s)*	*Per Epoch (s)* *AVG*	*SD*
Frame	Gaussian, $\sigma = 5$	42961	852.22	43.22
	Gaussian, $\sigma = 10$	43140	862.80	58.63
	Gaussian, $\sigma = 20$	42495	867.24	47.69
	Gaussian, $\sigma = 30$	43167	863.34	37.49
	Gaussian, $\sigma = 40$	42900	858.06	48.68
	Uniform	43006	860.12	36.50
Horizontal	Gaussian, $\sigma = 5$	16793	335.86	6.82
	Gaussian, $\sigma = 10$	16718	334.56	10.08
	Gaussian, $\sigma = 20$	16793	335.86	8.19
	Gaussian, $\sigma = 30$	16618	332.26	12.42
	Gaussian, $\sigma = 40$	16493	329.86	8.80
	Uniform	16509	330.18	3.95
Vertical	Gaussian, $\sigma = 5$	16754	335.08	6.56
	Gaussian, $\sigma = 10$	16600	332.00	4.19
	Gaussian, $\sigma = 20$	16654	333.08	6.75
	Gaussian, $\sigma = 30$	16844	336.88	10.30
	Gaussian, $\sigma = 40$	16812	336.24	6.06
	Uniform	16560	331.20	4.69

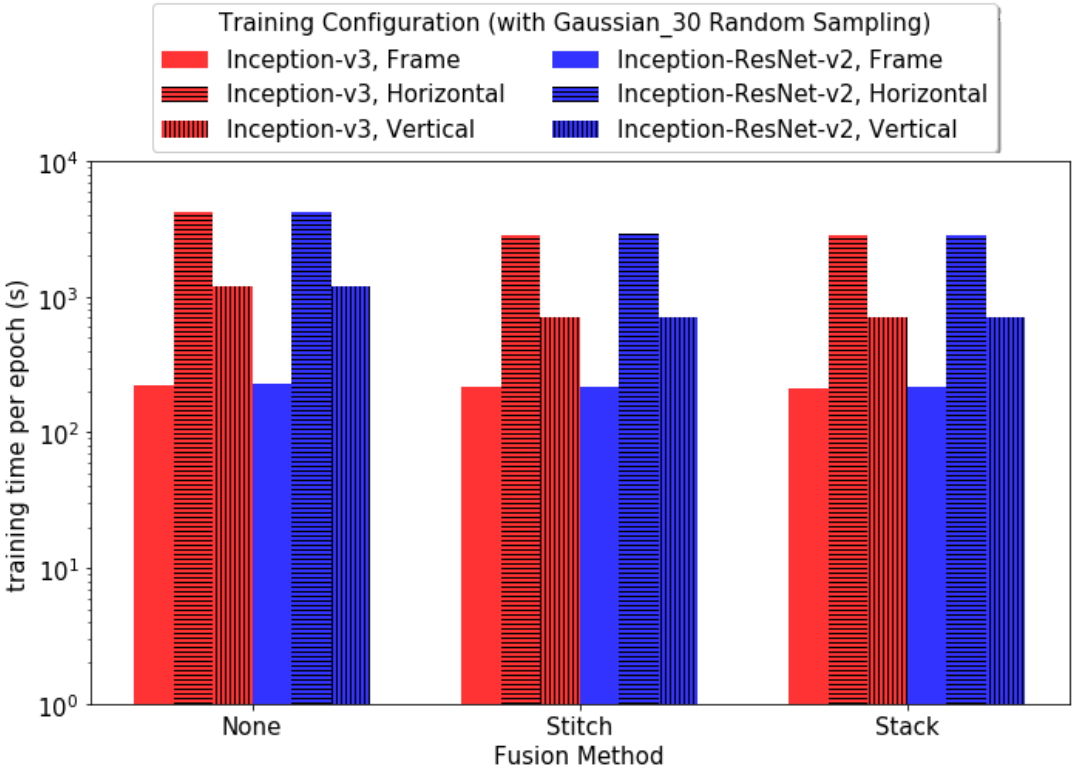

Figure B-1: 10-Class Fusion Experiment Computational Performance Results (with Gaussian $\sigma = 30$ sampling). Note that training time per epoch is plotted on a log scale as horizontal and vertical slicing took significantly longer than frame slicing.

Table B.5: 10-Class Early Fusion Study Computational Performance Results.

	Model	*Slicing Method*	*Sampling Technique*	*Total Training Time (s)*	*Training Time Per Epoch (s)*
No Fusion	Iv3	Frame	Uniform	2255.92	75.20
			Gaussian	6612.31	220.41
		Horizontal	Uniform	157684.87	5256.16
			Gaussian	125018.97	4167.30
		Vertical	Uniform	36696.07	1223.20
			Gaussian	35900.35	1196.68
	IRv2	Frame	Uniform	2533.86	84.46
			Gaussian	6839.01	227.97
		Horizontal	Uniform	158155.13	5271.84
			Gaussian	125072.40	4169.08
		Vertical	Uniform	36603.13	1220.10
			Gaussian	35989.75	1199.66
Stitch Method	Iv3	Frame	Uniform	3797.94	126.60
			Gaussian	6467.93	215.60
		Horizontal	Uniform	119238.54	3974.62
			Gaussian	85951.04	2865.03
		Vertical	Uniform	24670.63	822.35
			Gaussian	21221.88	707.40
	IRv2	Frame	Uniform	3921.33	130.71
			Gaussian	6580.51	219.35
		Horizontal	Uniform	119851.21	3995.04
			Gaussian	86454.05	2881.80
		Vertical	Uniform	24376.57	812.55
			Gaussian	21388.96	712.97
Stack Method	Iv3	Frame	Uniform	3719.47	123.98
			Gaussian	6374.31	212.48
		Horizontal	Uniform	119407.69	3980.26
			Gaussian	86075.06	2869.17
		Vertical	Uniform	24318.30	810.61
			Gaussian	20907.78	696.93
	IRv2	Frame	Uniform	3825.68	127.52
			Gaussian	6464.28	215.48
		Horizontal	Uniform	119521.00	3984.03
			Gaussian	86241.92	2874.73
		Vertical	Uniform	24511.26	817.04
			Gaussian	21185.52	706.18

Iv3 = Inception-v3
IRv2 = Inception-ResNet-v2

Table B.6: 339-Class Early Fusion Study Computational Performance Results (when trained on 64 Volta V100 GPUs, 2 per node). Note that the last five models in each category are trained as simple C2Ds but then used as 6-frame TSN models for validation video-level inference as described in Chapter 5. Those models were trained for 65 epochs rather than 50 for the other models which is why their total training times are generally higher.

	Model	*Slicing Method*	*Total Training Time (s)*	*Training Time Per Epoch (s)*
Stitch	R50	F	37039.60	740.79
	X	F	36780.93	735.62
	Iv3	F	39177.22	783.54
	IRv2	F	43377.81	867.56
	Cross-v1	H+V	35875.44	717.51
	Cross-v2	F+H+V	47070.10	941.40
	R50	F	45701.00	703.09
	X	F	47708.73	733.98
	IRv2	F	45957.36	707.04
	Mv2	F	43823.06	674.20
	D201	F	44566.39	685.64
Stack	R50	F	34786.50	695.73
	X	F	34102.44	682.05
	Iv3	F	35834.68	716.69
	IRv2	F	80038.35	800.38
	Cross-v1	H+V	40984.56	819.69
	Cross-v2	F+H+V	48761.20	975.22
	R50	F	42101.96	647.72
	X	F	41672.64	641.12
	IRv2	F	44201.03	680.02
	Mv2	F	57374.13	882.68
	D201	F	43358.09	667.05

F = Frame
H = Horizontal
V = Vertical

Appendix C

Dataset Details

Table C.1: Moments in Time Dataset Statistics (at time of download).

	Number of Videos	*Portion of Videos with Audio*	*Videos Per Class*		*Videos Per Class with Audio*	
			Avg	*SD*	*Avg*	*SD*
training set	802,224	61.84%	2366.5	973.8	1463.4	897.9
validation set	33,900	64.35%	100	0	64.35	17.83

Table C.2: Overview of Video Action Recognition Datasets.

Dataset	*Year*	*Videos*	*Classes*	*Purpose*
KTH [62]	2004	2,391	0	human actions
Weizmann [6]	2005	90	0	human actions
Hollywood2 [52]	2009	3,669	0	human actions in movies
GTEA [18]	2011	4	71	1st person actions
GTEA GAZE [17]	2012	14	40	actions w/ eyetracking
GTEA GAZE+ [17]	2012	6	44	actions w/ eyetracking
UCF101 [69]	2012	13,000	101	human actions
ADL [58]	2012	20	18	actions w/ object tracks
HMDB51 [44]	2012	7,000	51	human actions
CAD-120 [42]	2013	120	20	object affordances
JPL Interaction [61]	2013	57	7	actions at the observer
Sports-1M [39]	2014	1,000,000	487	sports
Thumos [36]	2014	20,700	101	untrimmed actions
MPII-MD [60]	2015	68,000	*N/A	movie audio descriptions
Watch-n-Patch [84]	2015	458	21	human actions
ActivityNet [29]	2015	27,000	203	human actions
Instructions [3]	2016	150	5	instruction videos
YouTube-8M [2]	2016	8,00,000	*4,800	visual entities
MV [55]	2016	260,000	*58,000	visual entities & actions
Charades [64]	2016	9,848	157	human actions
Kinetics [40]	2017	306,245	400	human actions
DALY [83]	2017	510	10	daily human activities
MultiTHUMOS [88]	2017	400	65	multiple human actions
Something-Something [24]	2017	220,847	174	intuitive physics
VLOG [19]	2017	114,000	*N/A	hand-object interactions
AVA [25]	2018	437	80	atomic visual actions
Moments in Time [54]	2018	1,000,000	339	human & non-human

*tags visual entities, not actions

Table C.3: Moments in Time Developers Validation Results [54].

Model	*Pretraining*	*Domains*	*Accuracy (%)*	
			Top-1	*Top-5*
ResNet50 [28]	None	Spatial	23.65	46.73
ResNet50 [28]	Places	Spatial	26.44	50.56
ResNet50 [28]	ImageNet	Spatial	27.16	51.68
TSN [87]	None	Spatial	24.11	49.10
BNInception [37]	None	Temporal	11.60	27.40
TSN [87]	None	Temporal	15.71	34.65
TSN [87]	None	Temporal	15.71	34.65
2-Stream TSN [87]	None	Spatial+Temporal	25.32	50.10
TRN-Multiscale [93]	None	Spatial+Temporal	28.27	53.87
I3D-ResNet50 [9]	ImageNet	Spatial+Temporal	29.51	56.06
SoundNet [4]	Flickr	Auditory	7.60	18.00

BNInception = Batch-Normalized Inception-v1
TSN = 6 frame input Temporal Segment Network
TRN = Temporal Relations Network

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

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

[3] Jean-Baptiste Alayrac, Piotr Bojanowski, Nishant Agrawal, Josef Sivic, Ivan Laptev, and Simon Lacoste-Julien. Unsupervised learning from narrated instruction videos. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4575–4583, June 2016.

[4] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. Soundnet: Learning sound representations from unlabeled video. *arXiv preprint arXiv:1610.09001*, 2016.

[5] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, June 2008.

[6] M. Blank, L. Gorelick, E. Shectman, M. Irani, and R. Basri. Actions as space-time shapes. *2005 IEEE International Conference on Computer Vision (ICCV)*, 1:1395–1402, 2005.

[7] Andrew P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, July 1997.

[8] João Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about kinetics-600. *arXiv preprint arXiv:1808.01340*, 2018.

[9] João Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[10] Rizwan Chaudhry, Avinash Ravichandran, Gregory Hager, and René Vidal. Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions. *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), CVPR Workshops 2009*, pages 1932–1939, 2009.

[11] Chen Chen, Xueyong Wei, Xiaowei Zhao, and Yang Liu. Alibaba-venus at activitynet challenge 2018 – task C trimmed event recognition (moments in time). *http://moments.csail.mit.edu/challenge2018/Alibaba_ Venus.pdf*, 2018.

[12] François Chollet. Xception: Deep learning with depthwise separable convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[13] Andrew Collette. *Python and HDF5*. O'Reilly Media, November 2013.

[14] Navneet Dalal, Bill Triggs, and Cordelia Schmid. Human detection using oriented histograms of flow and appearance. *2016 European Conference on Computer Vision (ECCV)*, 3952:428–441, 2006.

[15] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

[16] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[17] A. Fathi, Y. Li, and J. M. Rehg. Learning to recognize daily actions using Gaze. *2012 European Conference on Computer Vision (ECCV)*, pages 314–327, 2012.

[18] A. Fathi, X. Ren, and J. M. Rehg. Learning to recognize objects in egocentric activities. *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3281–3288, 2011.

[19] D. F. Fouhey, W. Kuo, A. Efros, and J. Malik. From lifestyle vlogs to everyday interactions. *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4991–5000, 2018.

[20] Vijay Gadepally, Justin Goodwin, Jeremy Kepner, Albert Reuther, Hayley Reynolds, Siddharth Samsi, Jonathan Su, and David Martinez. AI enabling technologies: a survey. *arXiv preprint axXiv:1905.03592*, 2019.

[21] D. Gartzman. Getting to know the mel spectrogram. towardsdatascience. *https://towardsdatascience.com/getting-to-know-the-mel-spectrogram-31bca3e2d9d0*, August 2019.

[22] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. *2017 IEEE International Conference of Acoustics, Speech, and Signal Processing (ICASSP)*, pages 776–780, 2017.

[23] D. Gershgorn. The Quartz guide to artificial intelligence: What is it, why is it important, and should we be afraid? *https://qz.com/1046350/the-quartz-guide-to-artificial-intelligence-what-is-it-why-is-it-important-and-should-we-be-afraid/*, September 2017.

[24] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fründ, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "Something Something" video database for learning and evaluating visual common sense. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5843–5851, 2017.

[25] Chunhui Gu, Chen Sun, David Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, Cordelia Schmid, and Jitendra Malik. AVA: A video dataset of spatio-temporally localized atomic visual actions. *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6047–6056, 2018.

[26] S. Guan and H. Li. Team SYSU Submission to Moments in Time Challenge 2018. *http://moments.csail.mit.edu/challenge2018/SYSU_isee.pdf*, 2018.

[27] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet? *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6546–6555, 2018.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[29] F C Heilbron, V Escorcia, B Ghanem, and J C Niebeles. ActivityNet: A large-scale video benchmark for human activity understanding. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–970, 2015.

[30] E. Holtham, M. R. Tora, K. Lensink, D. Begert, L. Meng, M. Holtham, E. Haber, L. Horesh, and R. Horesh. Team XtractAI submission to Moments in Time challenge 2018. *http://moments.csail.mit.edu/challenge2018/Xtract_AI.pdf*, 2018.

[31] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7132–7141, 2018.

[32] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Weinberger. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017.

[33] Jin Huang and Charles Ling. Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 17:299–310, March 2005.

[34] P. Y. Huang, X. Chang, and A. G. Hauptmann. Team CMU-AML submission to Moments in Time challenge 2018. *http://moments.csail.mit.edu/challenge2018/CMU_AML.pdf*, 2018.

[35] Muhammad Huzaifah. Comparison of time-frequency representations for environmental sound classification using convolutional neural networks. *arXiv preprint arXiv:1706.07156*, June 2017.

[36] Haroon Idrees, Amir Roshan Zamir, Yu-Gang Jiang, Alex Gorban, Ivan Laptev, Rahul Sukthankar, and Mubarak Shah. The THUMOS challenge on action recognition for videos "in the wild". *Computer Vision and Image Understanding (CVIU)*, 155, April 2016.

[37] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *2015 International Conference on Machine Learning (ICML)*, 37:448–456, 2015.

[38] Daniel Justus, John Brennan, Stephen Bonner, and Andrew Stephen McGough. Predicting the computational cost of deep learning models. *2018 IEEE International Conference on Big Data (Big Data)*, pages 3873–3882, 2018.

[39] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Fei Fei Li. Large-scale video classification with convolutional neural networks. *2014 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732, 2014.

[40] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The Kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[41] Alexander Kläser, Marcin Marszalek, and Cordelia Schmid. A spatio-temporal descriptor based on 3D-gradients. *2008 British Machine Vision Conference (BMVC)*, 2008.

[42] Hema Koppula, Rudhir Gupta, and Ashutosh Saxena. Learning human activities and object affordances from RGB-D videos. *The International Journal of Robotics Research*, 32(8), July 2013.

[43] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS) 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[44] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A large video database for human motion recognition. *2011 International Conference on Computer Vision (ICCV)*, pages 2256–2563, 2011.

[45] Yann Lecun, Leon Bottou, Y. Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[46] C. Li, Z. Hou, J. Chen, Y. Bu, J. Zhou, Q. Zhong, D. Xie, and S. Pu. Team DEEP-HRI Moments in Time challenge 2018 technical report. *http://moments.csail.mit.edu/challenge2018/DEEP_HRI.pdf*, 2018.

[47] Y. Li, Z. Xu, Q. Wu, Y. Cao, S. Zhang, L. Song, J. Jiang, C. Gan, G. Yu, and C. Zhang. Team Megvii submission to Moments in Time challenge 2018. *http://moments.csail.mit.edu/challenge2018/Megvii.pdf*, 2018.

[48] Z. Li and L. Yao. Team UNSW-Data Science submission to the Moments in Time challenge 2018. *http://moments.csail.mit.edu/challenge2018/UNSW_Data_Science.pdf*, 2018.

[49] Ji Lin, Chuang Gan, and Song Han. Temporal shift module for efficient video understanding. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7082–7092, 2019.

[50] Xiang Long, Chuang Gan, Gerard Melo, Xiao Liu, Yandong Li, Fu Li, and Shilei Wen. Multimodal keyless attention fusion for video classification. *2018 AAAI Conference on Artificial Intelligence*, pages 7202–7209, 2018.

[51] Bruce Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. *1981 International Joint Conference on Artificial Intelligence (IJCAI)*, 81:121–130, 04 1981.

[52] M. Marszalek, Ivan Laptev, and Cordelia Schmid. Actions in context. *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2929–2936, 2009.

[53] Brian McFee, Vincent Lostanlen, Matt McVicar, Alexandros Metsai, Stefan Balke, Carl Thomé, Colin Raffel, Dana Lee, Frank Zalkow, Kyungyun Lee, Oriol Nieto, Jack Mason, Dan Ellis, Ryuichi Yamamoto, Eric Battenberg, B M, Rachel

Bittner, Keunwoo Choi, Josh Moore, Ziyao Wei, Scott Seyfarth, nullmightybofo, Pius Friesch, Fabian-Robert Stöter, Darío Hereñú, Thassilo, Taewoon Kim, Matt Vollrath, Adam Weiss, and Adam Weiss. librosa/librosa: 0.7.1. *https://doi.org/10.5281/zenodo.3478579*, October 2019.

[54] Mathew Monfort, Bolei Zhou, Sarah Adel Bargal, Alex Andonian, Tom Yan, Kandan Ramakrishnan, Lisa M. Brown, Quanfu Fan, Dan Gutfreund, Carl Vondrick, and Aude Oliva. Moments in Time dataset: one million videos for event understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 42(2):502–508, February 2020.

[55] Phuc Xuan Nguyen, Grégory Rogez, Charless C. Fowlkes, and Deva Ramanan. The open world of micro-videos. *arXiv preprint arXiv:1603.09439*, 2016.

[56] Travis Oliphant. *Guide to NumPy*, 2nd edition. CreateSpace Independent Publishing Platform:. September 2015.

[57] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlche Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NIPS) 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[58] Hamed Pirsiavash and Deva Ramanan. Detecting activities of daily living in first-person camera views. *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2847–2854, 2012.

[59] A. Reuther, J. Kepner, C. Byun, S. Samsi, W. Arcand, D. Bestor, B. Bergeron, V. Gadepally, M. Houle, M. Hubbell, M. Jones, A. Klein, L. Milechin, J. Mullen, A. Prout, A. Rosa, C. Yee, and P. Michaleas. Interactive supercomputing on 40,000 cores for machine learning and data analysis. *2018 IEEE High Performance extreme Computing Conference (HPEC)*, pages 1–6, 2018.

[60] Anna Rohrbach, Marcus Rohrbach, Niket Tandon, and Bernt Schiele. A dataset for movie description. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3202–3212, 2015.

[61] Michael Ryoo and Larry Matthies. First-person activity recognition: What are they doing to me? *2013 IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 2730–2737, 2013.

[62] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local SVM approach. *2004 International Conference on Pattern Recognition (ICPR)*, 3:32–36, 2004.

[63] Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in TensorFlow. *arXiv preprint arXiv:1802.05799*, 2018.

[64] Gunnar Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in Homes: Crowdsourcing data collection for activity understanding. *2016 European Conference on Computer Vision (ECCV)*, pages 510–526, 2016.

[65] Julien Simon. ImageNet — part 1: going on an adventure. Medium. *https://medium.com/@julsimon/imagenet-part-1-going-on-an-adventure-c0a62976dc72*, 2017.

[66] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *2014 International Conference on Neural Information Processing Systems (NIPS)*, page 568–576, 2014.

[67] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *2015 International Conference on Learning Representations (ICLR)*, *https://arxiv.org/abs/1409.1556*. 2015.

[68] Cees Snoek, Marcel Worring, and Arnold Smeulders. Early versus late fusion in semantic video analysis. *2005 ACM International Conference on Multimedia (MM)*, pages 399–402, 2005.

[69] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[70] S. Stevens, J. Volkmann, and E. Newman. A scale for the measurement of the psychological magnitude pitch. *Journal of the Acoustical Society of America*, 8:185–190, January 1937.

[71] Ju Sun, Xiao Wu, Shuicheng Yan, Loong-Fah Cheong, Tat-Seng Chua, and Jintao Li. Hierarchical spatio-temporal context modeling for action recognition. *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2004–2011, 2009.

[72] Lin Sun, Kui Jia, Dit-Yan Yeung, and Bert Shi. Human action recognition using factorized spatio-temporal convolutional networks (fstcn). *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4597–4605, 2015.

[73] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *2017 AAAI Conference on Artificial Intelligence*, pages 4278–4284, 2017.

[74] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[75] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and ZB Wojna. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[76] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015.

[77] J Uijlings, I C Duta, E Sangineto, and N Sebe. Video classification with densely extracted hog/hof/mbh features: An evaluation of the accuracy/computational efficiency tradeoff. *International Journal of Multimedia Information Retrieval (IJMIR)*, 4(1):33–44, 2015.

[78] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 2020.

[79] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Dense trajectories and motion boundary descriptors for action recognition. *International Journal of Computer Vision*, 103:60–79, March 2013.

[80] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. *2013 IEEE International Conference on Computer Vision (ICCV)*, pages 3551–3558, 2013.

[81] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4305–4314, 2015.

[82] Xiaolong Wang, Ross Girshick, Harikrishna Mulam, and Kaiming He. Non-local neural networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7794–7803, 2018.

[83] Philippe Weinzaepfel, Xavier Martin, and Cordelia Schmid. Human action localization with sparse spatial supervision. *arXiv preprint arXiv:1605.05197*, 2016.

[84] Chenxia Wu, Jiemi Zhang, Silvio Savarese, and Ashutosh Saxena. Watch-n-patch: Unsupervised understanding of actions and relations. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4362–4370, 2015.

[85] Zhang Xiaoteng, Bao Yixin, Zhang Feiyun, Hu Kai, Wang Yicheng, Zhu Liang, He Qinzhu, Lin Yining, Shao Jie, and Peng Yao. Team Qiniu submission to ActivityNet Challenge 2018. *http://moments.csail.mit.edu/challenge2018/Qiniu.pdf*, 2018.

[86] Saining Xie, Ross Girshick, Piotr Dollar, Z. Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5987–5995, 2017.

[87] Y Xiong, Z Wang, Y Qiao, D Lin, X Tang, and L Van Gool. Temporal segment networks: Towards good practices for deep action recognition. *2016 European Conference on Computer Vision (ECCV)*, pages 20–36, 2016.

[88] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Fei Fei Li. Every moment counts: Dense detailed labeling of actions in complex videos. *International Journal of Computer Vision*, 126:375–389, July 2015.

[89] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *2016 British Machine Vision Conference (BMVC)*, pages 87.1–87.12, September 2016.

[90] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[91] Zhi Zeng and Qiang Ji. Knowledge based activity recognition with dynamic bayesian network. *2010 European Conference on Computer Vision (ECCV)*, pages 532–546, 2010.

[92] H. Zhang. Video action recognition based on hidden markov model combined with particle swarm. *International Journal on Computer Science and Information Systems (IADIS)*, 7(2):1–17, 2012.

[93] B. Zhou, A. Andonian, A. Oliva, and A. Torralba. Temporal relation reasoning in videos. *2018 European Conference on Computer Vision (ECCV)*, pages 803–818, 2018.

[94] Y. Zhou, P. Ma, and Y. Lu. Team SSS submission to Moments in Time challenge 2018. *http://moments.csail.mit.edu/challenge2018/SSS.pdf*, 2018.