Human action classification using temporal slicing for deep convolutional neural networks

Nathan Henderson

HUMAN ACTION CLASSIFICATION USING TEMPORAL SLICING FOR DEEP CONVOLUTIONAL NEURAL NETWORKS

by

NATHAN HENDERSON

A THESIS

Advisor: Dr. Ramazan Aygun

Submitted in partial fulfillment of the requirements
for the degree of Master of Science
in
The Department of Computer Science
to
The School of Graduate Studies
of
The University of Alabama in Huntsville

HUNTSVILLE, ALABAMA

2017

In presenting this thesis in partial fulfillment of the requirements for a master's degree from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department or the Dean of the School of Graduate Studies. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

[illegible] 10/9/2017

Nathan Henderson (date)

THESIS APPROVAL FORM

Submitted by Nathan Henderson in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and accepted on behalf of the Faculty of the School of Graduate Studies by the thesis committee.

We, the undersigned members of the Graduate Faculty of The University of Alabama in Huntsville, certify that we have advised and/or supervised the candidate of the work described in this thesis. We further certify that we have reviewed the thesis manuscript and approve it in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

[illegible] *Dr. Ramazan Aygun*	9/29/2017 (Date)	Committee Chair
[illegible] *Dr. Daniel Rochowiak*	10/3/2017 (Date)	
[illegible] *Dr. Huaming Zhang*	9/29/2017 (Date)	
Heggere S. Ranganath *Dr. Heggere Ranganath*	10-3-2017 (Date)	Department Chair
[illegible] *Dr. Sundar Christopher*	10-5-2017 (Date)	College Dean
[illegible] *Dr. David Berkowitz*	10/10/17 (Date)	Graduate Dean

ABSTRACT

School of Graduate Studies
The University of Alabama in Huntsville

Degree Masters of Science College/Dept. Science/Computer Science

Name of Candidate Nathan Henderson

Title Human Action Classification Using Temporal Slicing for Deep Convolutional Neural Networks

Artificial Neural Networks are a widely used computing system implemented for a wide variety of tasks and problems. A common application of such networks is classification problems. However, a significant amount of this research focuses on one and two-dimensional information, such as vectorized data and images. There is limited research performed on three-dimensional media such as video clips. This can be attributed to a lack of adequate resources, available training datasets, hardware constraints, and appropriate frameworks for implementing such networks.

This thesis attempts to provide an alternate methodology of feeding three-dimensional video data by preprocessing instead of directly inputting to a deep convolutional neural network. By taking sequential segments from multiple frames of a single video clip and combining them into a single image, the temporal dimension of the video can be encoded as a two-dimensional image. This process is called as temporal slicing and repeated for the entire spatial dimension of the video. The end result is spatio-temporal data encoded in a spatial format, which is then propagated through a convolutional neural network as image data. This method is less resource-intensive and is remarkably faster than pre-existing three-dimensional convolutional methods, while achieving significantly higher accuracy compared to the aforementioned network architectures.

Abstract Approval:	Committee Chair	[illegible] *Dr. Ramazan Aygun*
	Department Chair	Heggere S. Ranganath *Dr. Heggere Ranganath*
	Graduate Dean	[illegible] *Dr. David Berkowitz*

ACKNOWLEDGMENTS

Research is almost never an individual endevour, and this thesis is no exception. As such, there are many people I wish to thank who, through their support and contributions, have made this thesis possible.

First, I would like to thank Dr. Aygun for his excellent guidance as my thesis advisor and mentor for the past year. His ingenuity and expertise in the field of machine learning has been extremely beneficial to me in my academic career. I am also thankful for the people at ITSC at UAH for their support during my time as a Graduate Research Assistant, as well as their material support in providing me with the computational resources necessary for my experimentation during the writing of this thesis. Dr. Susan Bridges, also with ITSC, has been a great support to me during my time as a GRA, and during the writing of this thesis. I would like to thank her for her support during this time. Without Dr. Bridges' guidance, my knowledge in the field of machine learning and neural networks would not be where it is today. I would especially like to thank my wife, Ashley, for her undying support for me during this time, and for being understanding of the effort and time a thesis such as this requires. I could not have done this without her by my side every step of the way. Finally, I thank my Lord and Savior for bringing me through this journey. May He receive all the glory.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Applications of Convolutional Neural Networks (CNNs) are becoming more predominant in the fields of computer vision and machine learning, due to the robustness of artificial neural networks and the networks' capability to extract features from multidimensional data. Present efforts involve these networks being applied to spatio-temporal data such as video clips and motion data. This thesis attempts to demonstrate an innovative approach to video classification, specifically classification of human action video data, while using Convolutional Neural Networks.

1.1 Motivation

With the progression of hardware and processing capabilities of computers, an increase in applications of Artificial Neural Networks (ANNs) is seen in the field of machine learning. ANNs are capable of modeling complex, non-linear data and are able to approximate virtually any function. This capability makes the classification as a predominant use of neural network technology. By training a neural network on a certain dataset, it is intuitively and relatively simple to use a neural network to predict the class or label of other specified data samples. With the evolution of

ANNs, comes the Convolutional Neural Network, which applies the same principles of network connectivity as ANNs, while also being tolerant to translation invariance in the input data, through its use of spatial convolution windows for subsampling.

Frequently, the type of datasets used to train a neural network consist of numerical data formatted in a one or two-dimensional structures. Convolutional Neural Networks are commonly used for data processing involving spatial or spatio-temporal information, such as image recognition or natural language processing [4]. This specific network architecture is especially adept at spatial feature extraction. Due to the fact that CNNs are able to somewhat maintain the spatial integrity of the input data, the networks are able to adapt to shifts or translations in the input samples [5]. This proves essential in the application of CNNs to input data that involves such movement or shifting, such as that which is found in video data depicting various human actions or movements. By using filters to adapt to certain recurring features in each video, the CNN is able to learn and track certain motions across the time domain.

Human action recognition is a complex problem due to the multiple variables and parameters involved, such as an infinite possible ways of movements and positions, dynamic camera positions and perspectives, background noise and movements, and appearance of the video subjects, such as clothing and accessories. However, the ability to accurately and reliably classify data depicting human activity and movement is of utmost importance in research fields such as human-computer interaction, computer vision, video retrieval, and video surveillance, as well as recognition problems such as human facial expression or emotion predictions.

In comparison to image data that is most commonly used by CNNs, video datasets are extremely large and can easily be terabytes in size. Common video datasets such as UCF-101 [6] contain over 13,000 videos and 101 classes, and the Sports 1M Dataset [2] contains over 1.1 million videos and 487 classes. Classifying datasets of this size leads to a series of obstacles that must be considered when developing an approach to classify such large amounts of data. These obstacles are as follows:

1. Hardware Constraints

   The neural network must be able to process at least a single training video sample at a time, meaning that there must be an adequate amount of local memory available to handle the mini-batches of training data. Often, a CPU (Central Processing Unit) does not contain enough memory, nor does it provide the speed necessary to train an entire network in a reasonable amount of time. With this in mind, often a GPU (Graphics Processing Unit) is utilized to handle such large image or video datasets, as demonstrated in [7]. Adequately training a deep CNN can take days or weeks at a time; thus, it is imperative that the hardware is able to handle a batch size large enough so that the network training can be completed in a reasonable amount of time.

2. Network Depth

   Often, CNNs used for classifying video data contain several convolutional and pooling layers, as well as fully-connected layers. The convolutional layers can contain hundreds or thousands of filters used for feature extraction, and this greatly contributes to any time or hardware constraints that might exist. It is important that the network is able to extract and learn certain features from each video, therefore the network architecture must not be too small or too large.

3. Computational Time

   As previously stated, the time needed to adequately train a deep CNN can be days or even weeks. The networks that have been used in [2], [3], [8], [9], and other experiments each consist of numerous convolutional layers, with each layer containing a certain number of filters. To help minimize the computational cost of training a network, various measures may be taken in the preprocessing stage, such as selecting the first n number of frames from a video, similar to [10], cropping all frames to certain dimensions, or using different sub-sampling methods. However, such methods result in a loss of spatial information, which can lead to decreased accuracy.

1.2 Our Approach

The temporal aspect of video data is one of the reasons for the increased training time, as well as the addition of background noise and dynamic camera perspectives that negatively impact classification accuracy. In this thesis, we propose a different approach that both improves network accuracy and greatly reduces training time. During the video preprocessing stage, we take a number of frames from each video and extract a subset of sequential pixel "slices" from each frame, combining them into a single data sample, or "segment". This process is then repeated until the entire spatial area of each frame is included in the data sample. The result is a series of sequential segments, comprising a single "strip". Each strip represents an entire video clip. By extracting a fixed number of frames per video clip, we are able to keep the length of each frame strip the same. Through this process, we are able to successfully compress the temporal information, while keeping it intact enough for the convolutional network to adapt to and learn the features of each video sample. Because the convolutional network is now processing two-dimensional (2D) information instead of three-dimensional (3D) information, the computational workload is significantly reduced. The dataset of frame strips is then used to train a CNN called the Deep Slicing Network ("DeepSlice"). Other video clips, previously unseen by the network, are used to test the network's performance.

1.3 Thesis Organization

This thesis is organized as follows: Chapter 2 provides a basic conceptual overview of Convolutional Neural Networks and their respective attributes and parameters, as well as their application to this specific problem domain. Chapter 3 contains previous related work in this field. This includes convolutional neural networks, human action classification, and spatio-temporal processing through slicing. Chapter 4 describes our own approach to the problem, including the dataset used, the preprocessing techniques employed, and the CNN architecture as well. Chapter 5 shows our results of the DeepSlice network, concerning both the classification accuracy and the computational performance. These results are then compared to other benchmark results of previous experiments on similar or identical datasets. Finally, Chapter 6 is a summary of the thesis, specifically our approach and the results, and potential future work in this research field as well.

CHAPTER 2

OVERVIEW OF CONVOLUTIONAL NEURAL NETWORKS

This chapter provides a basic overview of convolutional neural networks, and their various architectures and parameters. This includes the various types of layers that are commonly founded in CNNs and our DeepSlice network, and the hyperparameters involved. Finally, we include a summary of different techniques that are used to improve initial classification results, such as dropout and weight decay.

2.1 CNN Layers

2.1.1 Convolution Layer

Convolutional neural networks are an advanced variation of multilayer perceptron networks, but are designed to maintain the integrity of the spatial dimension. This makes the CNN optimal for video and image processing, and other spatial classification tasks, as well as natural language processing. The primary component of a CNN is the convolution layer. Each convolution produces as output the total weighted input from a local receptive field, determined by the size of the convolutional window. The convolutional window is then translated according to a predetermined "stride". Through this process, the output is contained in a hidden layer. Because this architecture also implements shared weights (weights that are used across multiple receptive fields), the spatial data is preserved [1]. This is further illustrated in Figure 2.1.

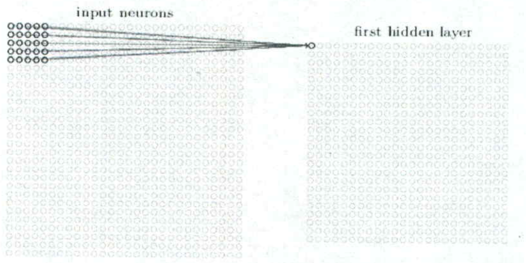

Figure 2.1: Convolution Across a Single Spatial Data Sample [1]

The output of a certain layer l at position (i, j) given a bias B and filter size (FxF), shared weights array w, and a being the input activation to the node, is shown in Equation 2.1. For 3D convolution such as [3], this principle is applied in three dimensions instead of two.

$$x_{j,k} = B + \sum_{l=0}^{F-1} \sum_{m=0}^{F-1} w_{l,m} a_{j+l,k+m} \tag{2.1}$$

This process is continued for a set number of kernels (or "filters"). The output of a filter applied to an input layer is referred to as a "feature map". This process of convolution is repeated for a defined number of filters, thus producing an output with a depth dimension. These feature maps determine the activation of the following layer in the network architecture.

2.1.2 Pooling Layer

A common layer found to follow convolutional layers is called a pooling layer. The purpose of these layers is to preserve the spatial dimensionality of the preceding layer's output, while also downsampling the data to reduce the amount of local data and to avoid overfitting. The pooling layers in DeepSlice implement the "Max Pooling" algorithm which simply returns the highest activation value within the pooling window. We experiment with different pooling windows in this thesis. Typically, due to the unorthodox dimensions of our data, the pooling windows are often rectangular for our data, instead of commonly-used square windows. This technique is used for the purpose of downsampling individual dimensions of the data, to ensure that enough data was present for the duration of the forward-propagation.

2.1.3 Fully Connected Layer

A fully-connected (or "affine") layer contains a single matrix of nodes that are all connected to the input of each of the nodes of the previous layer. An example of a multi-layer perceptron network (constructed entirely of fully-connected layers) is shown in Figure 2.2.

2.1.4 Rectified Linear Unit

The activation function used for the layers in DeepSlice are called Rectified Linear Units. This activation function follows the equation shown in Equation 2.2.

$$f(x) = max(0, x) \tag{2.2}$$

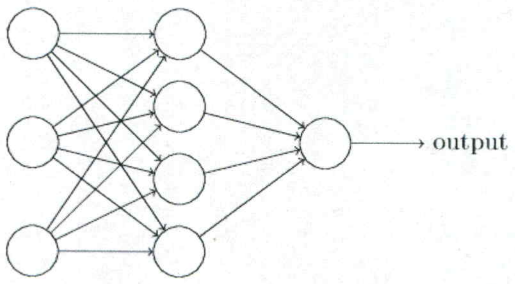

Figure 2.2: Diagram of Multilayer Perceptron Network [1]

2.1.5 Softmax Layer

The output layer of our network is a softmax layer. This layer contains a vector of output of the number of classes. For the UCF101 dataset, the number of output nodes is 101. This network outputs a vector of probabilities that, when summed, equal 1. The probability of class y in input vector x for K classes is shown in Equation 2.3. This layer also calculates the loss of the output of the function. The primary purpose of the network is to minimize this cost function as it approaches the global minimum.

$$P(y|x) = \frac{e^{x^T w_j}}{\sum\limits_{k=1}^{K} e^{x^T w_k}} \tag{2.3}$$

In the literature, results often report a "top-k" accuracy, indicating that the class was correctly predicted in the top "k" group. For example, the top-1 accuracy indicates that the correct class was within the top 1 softmax predictions when sorted by increasing probability. A top-5 accuracy indicates that the network predicted the correct class within the top 5 probabilities.

2.2 Optimization

2.2.1 Stochastic Gradient Descent

The optimization, or learning algorithm, for this network is Stochastic Gradient Descent (SGD) [11]. SGD is an iterative optimization function that uses the gradient of the loss function to gradually approach the global minimum of the loss function. The weights are adjusted during backpropagation using a specified learning rate α and the loss function gradient $\bigtriangledown Q_i$. The momentum coefficient μ determines how much the previous weight adjustment affects the current weight adjustment. The equation for the weight adjustment using SGD with momentum is shown in Equation 2.4. SGD has proven to be a reliable and robust optimization technique [12] and continues to be widely used in many neural network architectures.

$$w := w - \mu \bigtriangledown Q_i(w) + \alpha \bigtriangleup w \tag{2.4}$$

2.2.2 Dropout

A regularization method used in DeepSlice is dropout. This is a technique used to prevent overfitting, especially within fully-connected layers, due to the large amount of parameters and connectivity within these layers. During each training iteration, a randomized selection of nodes within a layer are removed. This allows for a certain amount of variation within the training phase, and is similar to training multiple similar but not identical networks on a single sample (bagging) [13] [14]. In

DeepSlice, the final fully-connected layer implements dropout with a probability of 0.5.

2.3 Summary

This chapter serves as a basic introduction to the basic fundamentals of CNNs, and the benefit that certain optimization and learning algorithms offer to the training process. This is by no means an exhaustive list of possible CNN features and algorithms, and the addition of certain features such as normalization or another learning algorithm might provide an additional increase to the classification of DeepSlice.

CHAPTER 3

RELATED WORK

This chapter describes some of the previous research relevant to our work. This includes the work related to convolutional neural networks in general, their application to motion data, including human action and pose estimation, as well as experimentation with temporal slicing with image and video data. With the availability of video data sets and ground-truths, it is now possible to compare the performance of various techniques fairly.

3.1 Convolutional Neural Networks

With the increase in hardware capabilities supporting Convolutional Neural Networks, much work has been done in the area of image processing, such as [15], [16], and [17]. The authors of [15] introduce a large image dataset called "ImageNet", consisting of over 1.2 million high-resolution images sorted into 1000 classes. The CNN used in their experiments consists of five convolutional layers, three fully-connected layers, and a final softmax output layer. This architecture has been the foundation for several other works, including the C3D Network found in [3]. [16] furthers the work on the ImageNet dataset by exploring the correlation of network depth with classification accuracy. Through their increase in convolutional layer depth and filter sizes, they were able to improve upon the standard baseline accuracy. [17] focuses on the utilization of the computational resources while allowing for increases in the network size and depth. This allows for a wider variety of network architectures to be tested on the Imagenet dataset. [8], [18], and [2] all expand this technology by incorporating temporal data into their respective datasets. The work done by [8] uses CNNs to process videos frame by frame, while the other works attempt to use the temporal context of the data through 3D convolution windows and 3D pooling layers. A combination of three-dimensional convolutional neural networks and recurrent neural networks are used in [18] to implement a feature-extraction mechanism to aid the classification of two different video datasets [19]. [2] introduces the Sports1M dataset, consisting of over 1.2 million videos split into 487 classes, which has led to a number of experiments further exploring spatio-temporal classification using CNNs.

3.2 Human Action Classification

Convolutional neural network architectures have been applied to human action or motion recognition domains in several different approaches. For video classification, Karpathy et al. [2] use single-frame analysis as well as temporal convolution in slow, early, and late fusion models. This allows for certain network architectures to perform feature extraction in parallel, while maintaining spatial and temporal integrity of the data samples. Their paper experiments with various types of networks that fuse temporal information from the neural network at certain times during forward-propagation. The three networks used in this experiment are Late Fusion, Early

Fusion, and Slow Fusion. Each of these networks takes in a sequential grouping of video frames at a time, in order to preserve the temporal order. These results are compared to a network trained on each video frame individually. Their diagram of such processes is shown in Figure 3.1. In this work, their network architecture is described as "C(96, 11, 3)-N-P-C(256, 5, 1)-N-P-C(384, 3, 1)- C(384, 3, 1)-C(256, 3, 1)-P-F C(4096)-F C(4096)" [2], where C is a convolutional layer, P is a pooling layer, N is a normalization layer, and FC is a fully connected layer. The pooling layers use a 2x2 window. Each convolutional layer is shown using *C(d, f, s)*, with d filters of dimensions *fxf*, using a stride of s.

Another experiment attempted in their paper is splitting each input stream into two separate input streams. The first is a low-resolution context stream, and the second is a high-resolution "fovea" stream. This stream focuses on the central region of the frames that contain the most spatial information about the action, instead of the entire high-resolution frame. This is shown to be beneficial in improving the performance of the network. The network is trained on the Sports1M dataset, and the Slow Fusion network obtains the highest accuracy level, a 41.9% top-1 clip accuracy. This network is then fine-tuned on the UCF101 dataset to achieve a 3-fold accuracy of 65.4%, as a result of fine-tuning the top 3 layers. As a baseline, their network is trained from scratch on UCF101, achieving an accuracy of 41.3%.

Du Tran et al. [3] use the C3D network, a network architecture reliant on 3D convolution and 3D pooling throughout. C3D is a caffe-based implementation of an ANN framework that allows for pooling and convolutional layers to perform temporal convolution, and utilizes techniques such as SVMs and fine-tuning of the

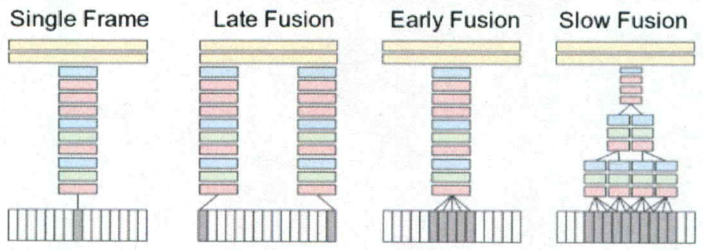

Figure 3.1: Temporal Fusion Networks Tested in [2] (2014 IEEE)

final layers to improve upon their initial accuracy results. C3D implements true three-dimensional convolution and pooling in their network architecture, maintaining spatial and temporal integrity throughout the entire forward-propagation process, and continues the principle discovered by the Slow Fusion model in [2], that is, pure 3D convolution and pooling within the network's convolutional layers achieves higher results. A portion of this experimentation is to discover the best kernel size for convolution, and it is discovered that a 3x3 convolution kernel provides the best results. This theory is also confirmed by our own experiments.

Further experimentation was conducted to determine the best temporal dimension size for the kernels. Several depths were tested, such as 1, 3, 5, and 7. In addition, networks with increasing depths such as 3-3-5-5-7 and decreasing depths such as 7-5-5-3-3 were also tested. These experiments were run on the UCF101 dataset, using the training and test split 1. The constant depth of 3 performed the highest, achieving approximately 44% accuracy when trained from scratch. These previous experiments determined the final architecture of the C3D network, which is shown in Figure 3.2

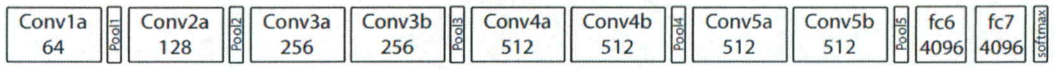

Figure 3.2: C3D Architecture [3] (2015 IEEE)

The convolution window in each layer is 3x3x3, with a stride of 1. Each pooling layer uses a kernel of size 2x2x2, with a stride of 1. However, the first pooling contains a temporal depth of 1, to preserve the temporal dimension for later convolution. This technique was also implemented in our own experiments and networks. C3D was later trained on Sports1M, and tested using the UCF101 dataset, with a top-1 accuracy of 44.9% when trained from scratch.

Ji et al. [20] use the same type of 3D convolution, while also incorporating optical flow information into the dataset. Their work also explores the optimal size of the kernels to produce improved results. Ng et al. employ a recurrent neural network in [8] called LSTM (Long Short Term Memory) that stores video clip information continuously, thus allowing the network to learn more features over longer durations. In this experiment, the authors deviate from the common practice of using sequential video frames, due to the computational workload. Instead, the authors prefer to process each frame individually, and obtain the motion information separately using optical flow techniques. The authors of this paper explore various pooling techniques, such as implementing the pooling layers after the convolutional layers, at the end of the network (after the fully-connected layers), after both the convolutional and fully-connected layers, and after time-domain convolution. The output of this network is then used as input to the LSTM network for the classification. This experiment is first carried out on the Sports1M dataset and then tested on the UCF101 dataset. This paper claims to have achieved a 3-fold accuracy of 82.6% for convolutional pooling of 120 frames.

Specific to human action analysis, Jayabalan et al. [21] use CNN networks to process joint movement data. Through their use of joint data representations, they are able to work with lower-dimensionality data compared to data such as images or videos. This is one of the first implementations of CNNs for this task, as previous human action analysis work was primary performed using Recurrent Neural Networks. Grinciunaite et al. [22] use similar joint regression information in 3D space to predict certain movements and poses. Their work also utilizes 3D convolution in their CNN.

Finally, Tachhetti et al. [10] make comparisons to the biological processes behind visual identification of actions and explores certain aspects of data such as video translation, and camera invariance. It is worth noting that [23], [24], [25], and [8] use information such as optical flow to fine-tune their networks. Many others such as [26] and [3] use other feature extraction methods such as Support Vector Machines (SVMs) to further improve their results.

3.3 Temporal Slicing

The concept of temporal segmentation has been tested for clustering, and retrieval of clustered video shots has been demonstrated by [27], and this concept was further explored in [28]. [27] introduces the concept of temporal slicing of video frames for the extraction of motion patterns and recognition of different color patterns and motion analysis. The segmented data was then used for clustering for a two-class classification problem. This technique was applied to discover various kinds of camera breaks in [28], in order to break a video clip into several individual segments. In our work, we used the temporal slicing technique in order to reduce the computational burden of the neural network. However, the results of our application of temporal slicing is then used for a convolutional neural network, not for a clustering algorithm. In this way, this slicing approach is applied to a new problem domain.

3.4 Summary

Although there have been several noted attempts at human action or motion classification using convolutional neural networks, currently we have not discovered any that utilize the temporal slicing technique to form a different dataset to be used as input to the neural network. Although many of the previous works discussed use optical flow or other external motion data as input into the CNNs, we prefer to use the raw video frames as input, and apply our temporal slicing technique to the original video frames, as this is what makes our approach innovative and unique.

CHAPTER 4

METHODOLOGY

This chapter provide information on the dataset used to test DeepSlice's classification accuracy, UCF-101. The preprocessing of video data using temporal slicing is an important feature of DeepSlice. The experiments to test the functionality and performance of DeepSlice on temporally sliced data are also described here, including the experimentation used to determine the best orientation to be used for the slicing. Finally, the architecture of the DeepSlice network itself is described and illustrated in this chapter, as well as the various hyper-parameters that were shown to provide the most optimal results.

4.1 Dataset

In this thesis, we use the UCF-101 video dataset, as described in [6]. This dataset is comprised of 13,320 video clips depicting 101 human actions. These classes include labels such as Applying Eye Makeup, Brushing Teeth, Hammering, Mopping Floor, Bowling, and Writing On Board. Each class contains over 100 video clips. The average video clip length is 7.21 seconds, with the range of lengths being 1.06 to 71.04 seconds. The videos are recorded with a frame rate of 25 frames per second, at a resolution of 320x240 pixels. The training and test datasets contain separate "groups" of videos, with each group containing individual video clips. Such "clips" contain different actors or other differentiable context, but are taken from the same long video clip. For example, two separate video clips from the same group in the "baseball" class might contain two different batters, but would contain the same background, and the same camera angle. The train and test datasets were separated by groups to avoid similar clips in both sets, leading to inflated accuracy as a result. A list of the individual classes in UCF-101 can be found in Appendix A.

4.2 Temporal Slicing

In order to reduce the training time of the network, while also preserving the spatial and temporal information in some regard, we take a different approach to the data presentation to the network. Projects that use 3D Convolution take in a sample of sequential video frames which keeps the spatial information intact and also maintains the temporal order of the data. However, in our approach, we sample exactly 30 frames from each video clip. Each frame was cropped to 128x171 pixels. A strip of spatial data was taken from the same position in each frame and was sequentially placed in order to create a segment of slices that contain all image data for a single localized area across all sampled frames, thus encoding the temporal dimension as well. This process was repeated until each frame was completely covered. For frames of size h x w pixels, the width of the resulting image strip is f x w pixels, with f being the number of frames taken from each clip.

This process is completed a second time, this time with row extraction, instead of column extraction. Both types of preprocessing were completed with the same dataset, using 128x171 pixel frames, and 30 frames per video. This concept is further illustrated in Figure 4.1 and Figure 4.2, which shows a single image strip for a series of frames from a single video clip, containing n different segments. It is worth noting that in row-wise slicing, the resulting frame strip has an extended vertical dimension, while the column-wise slicing produces a frame strip with an extended horizontal dimension, more closely resembling an actual "filmstrip".

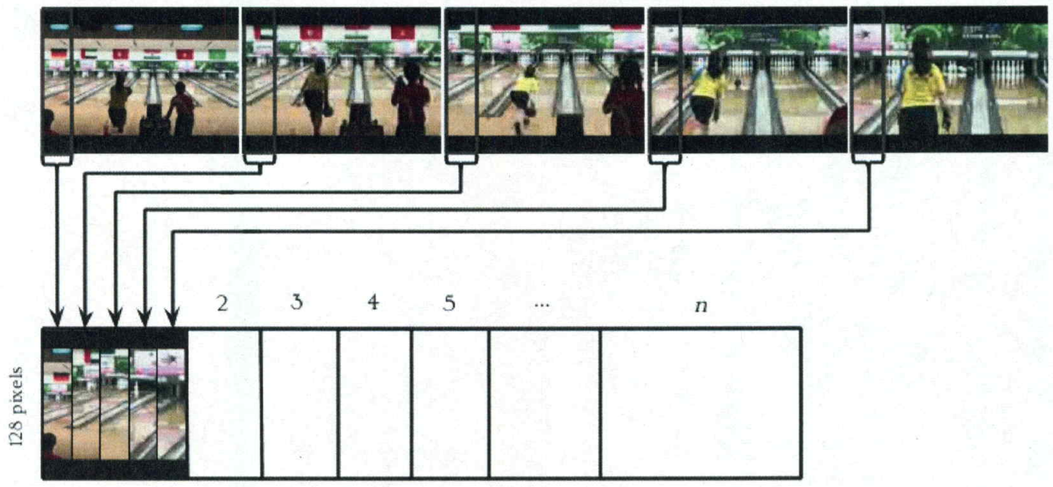

Figure 4.1: Localized Column Temporal Slicing

Figures 4.5 through 4.20 contains examples of video clips after the preprocessing stage. This includes the frame extraction from each clip, and the temporal slicing. All videos are cropped to 128x171 pixel dimensions. Note that for the sake of this thesis, the figures display a single frame strip divided into 4-6 segments placed

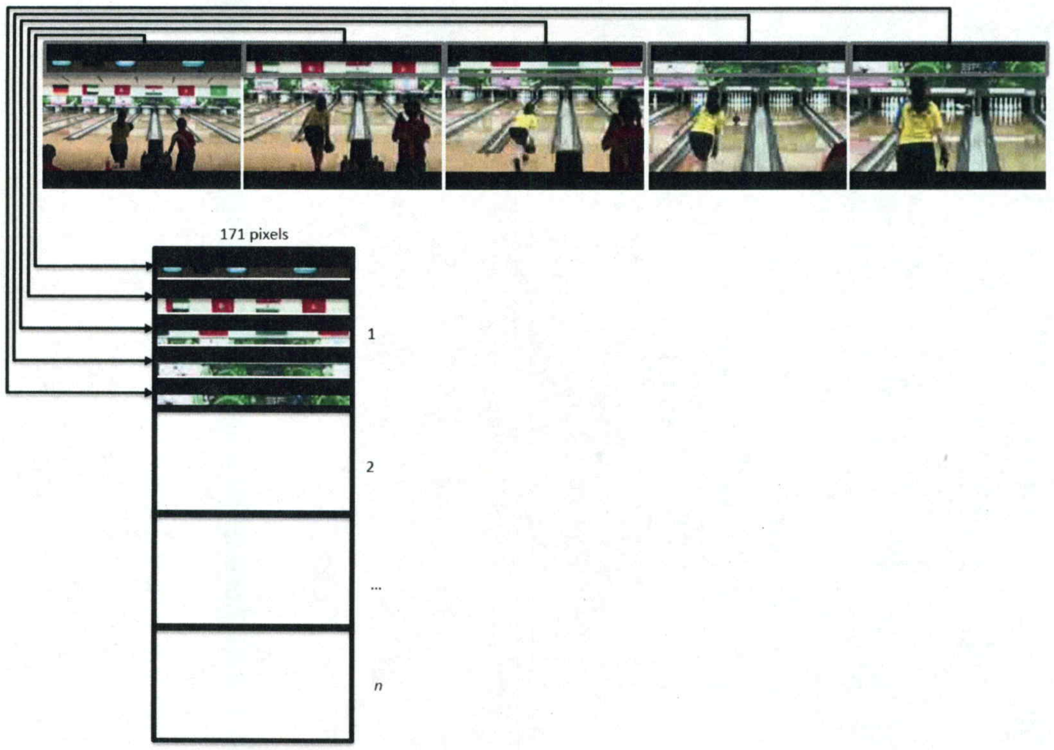

Figure 4.2: Localized Row Temporal Slicing

in parallel, due to sizing constraints. The actual images are several thousand pixels long, and are handled by the CNN as such.

4.2.1 Original Frames

The following sections contain visual examples of the preprocessing that occurs for the DeepSlice network. Two different video clips are taken from two separate classes, "Golf Swing" and "High Jump". This is intended to visualize the effect that various slicing widths and orientations has on the image strip. Each frame is 128x171 pixels, and exactly 30 frames are taken from each video using iterative extraction so that the entire duration of the video clip is covered.

The first example shown in Figure 4.3 is an example from the class "Golf Swing". This video clip depicts a golfer performing a normal golf swing, hitting a golf ball off of a tee. The camera remains stationary during this shot, which eliminates a significant source of background noise and translation.

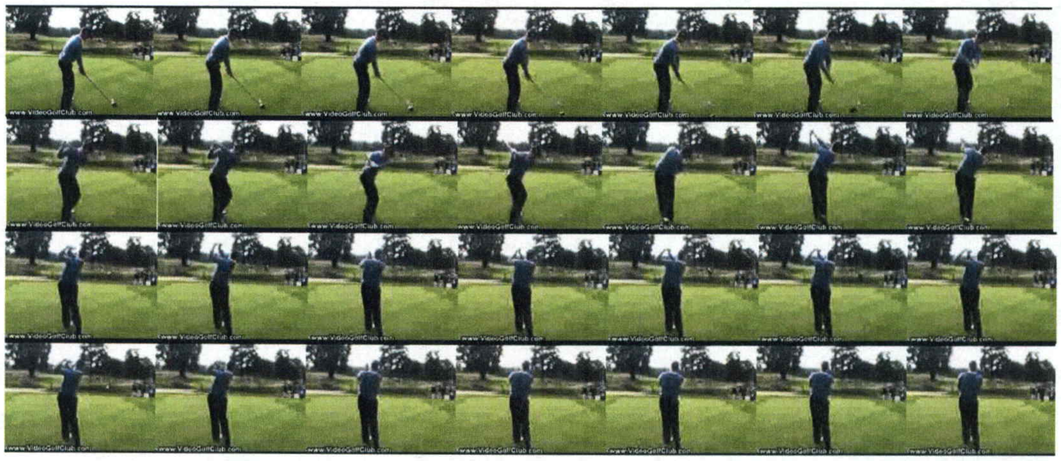

Figure 4.3: Sampled Frames From "Golf Swing" Video Clip

The second example, shown in Figure 4.4, is from the "High Jump" class. This video depicts an athlete performing the high jump over a crossbar inside of a gymnasium. However, in contrast to the previous clip, this camera follows the movement of the athlete, and shifts horizontally during the video, which is visible from the extracted clips.

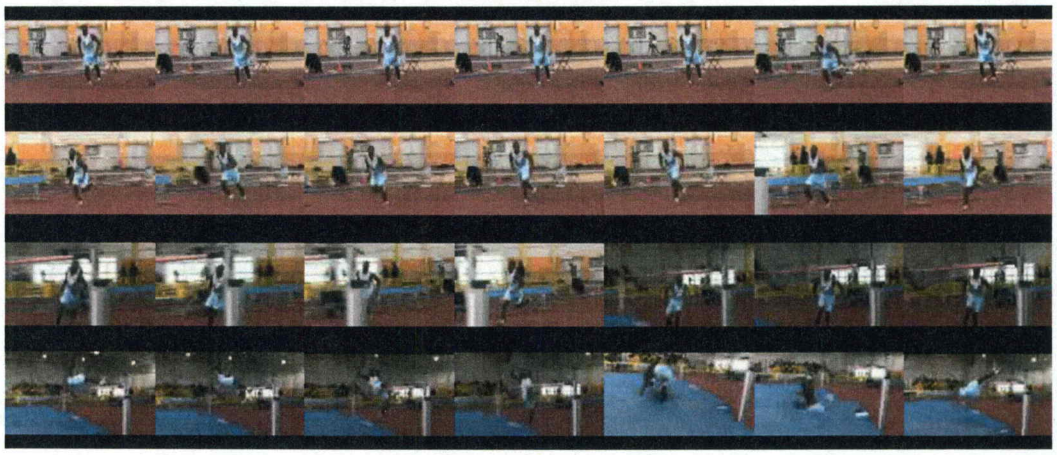

Figure 4.4: Sampled Frames From "High Jump" Video Clip

4.2.2 Column-wise Slicing

This section displays the resulting frame strips of the temporal slicing preprocessing technique. Each of the 30 frames that were extracted previously have a select slice removed, and are then combined in a contiguous manner. This is then repeated until the entirety of the spatial dimension of each frame is covered. This process is shown for slice widths of 3, 6, 12, and 24 pixels. Each of the images are shown with segments of the 128x5040 image in parallel, due to sizing constraints.

Shown in Figures 4.5 through 4.8 are the results of the column-wise temporal slicing for the "Golf Swing" video clip for various slice widths, whose extracted frames are shown in Figure 4.3.

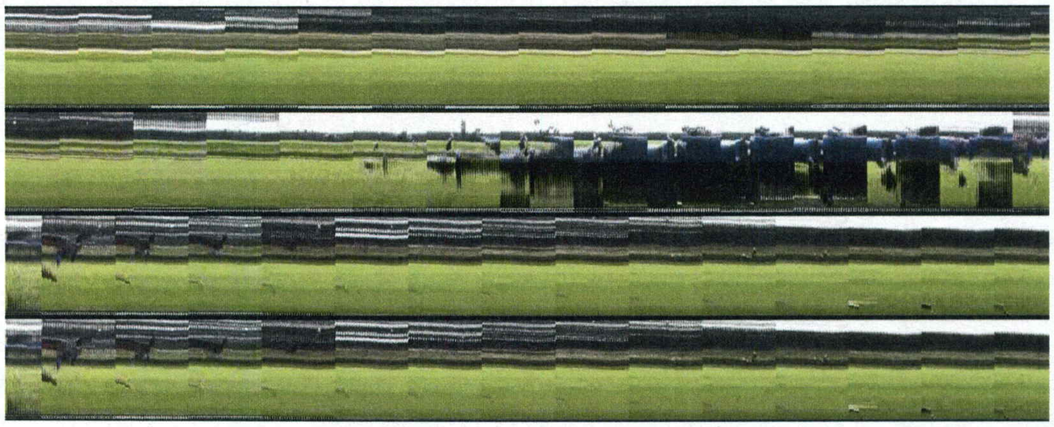

Figure 4.5: "Golf Swing" Clip Using Column Slice Width of 3 pixels

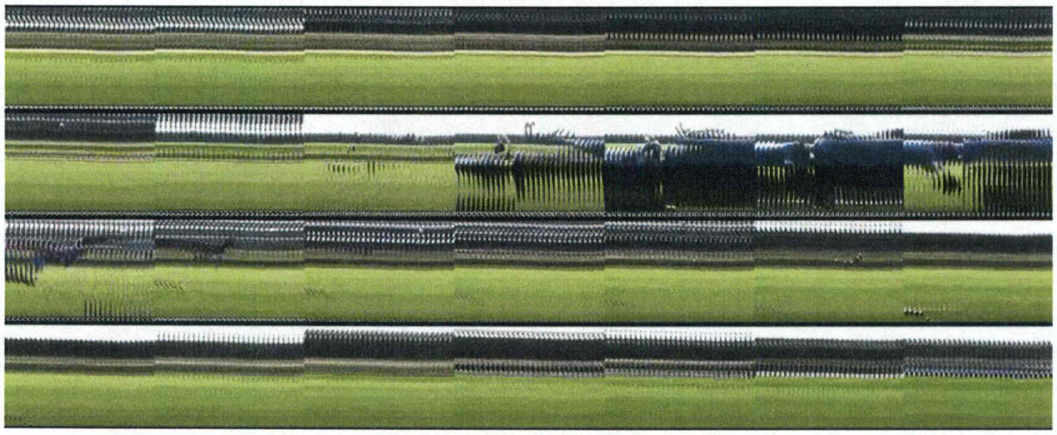

Figure 4.6: "Golf Swing" Clip Using Column Slice Width of 6 pixels

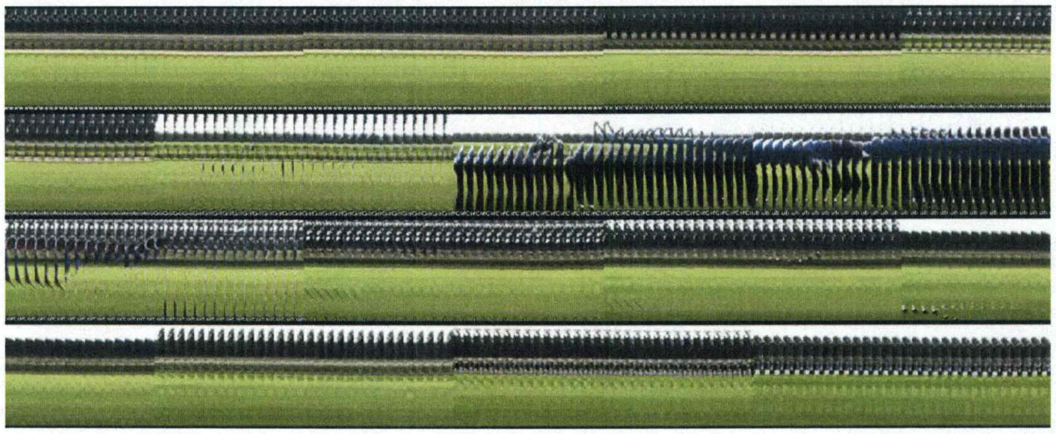

Figure 4.7: "Golf Swing" Clip Using Column Slice Width of 12 pixels

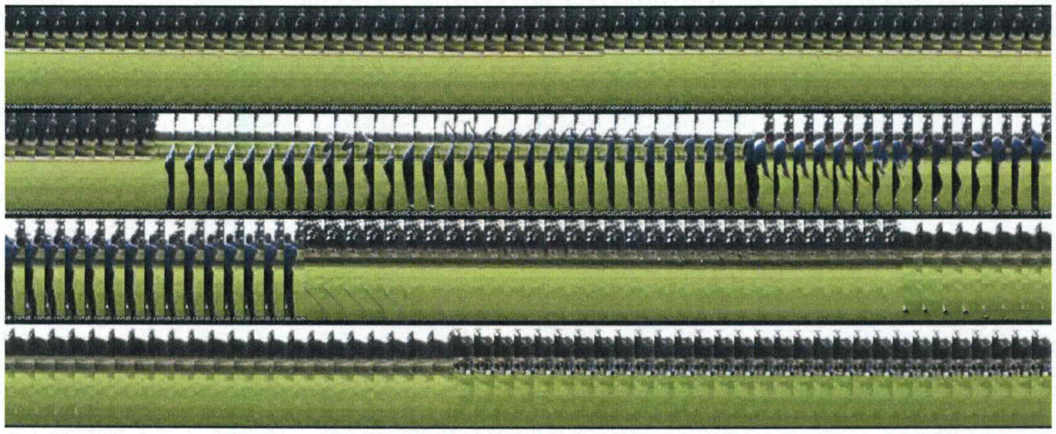

Figure 4.8: "Golf Swing" Clip Using Column Slice Width of 24 pixels

Shown in Figures 4.9 through 4.12 are the results of the column-wise temporal slicing for the "High Jump" video clip for various slice widths, whose extracted frames are shown in Figure 4.4. When slice width is increased, the content can be interpreted better by a human. It is difficult to grasp the content of a video for images using low slice width.

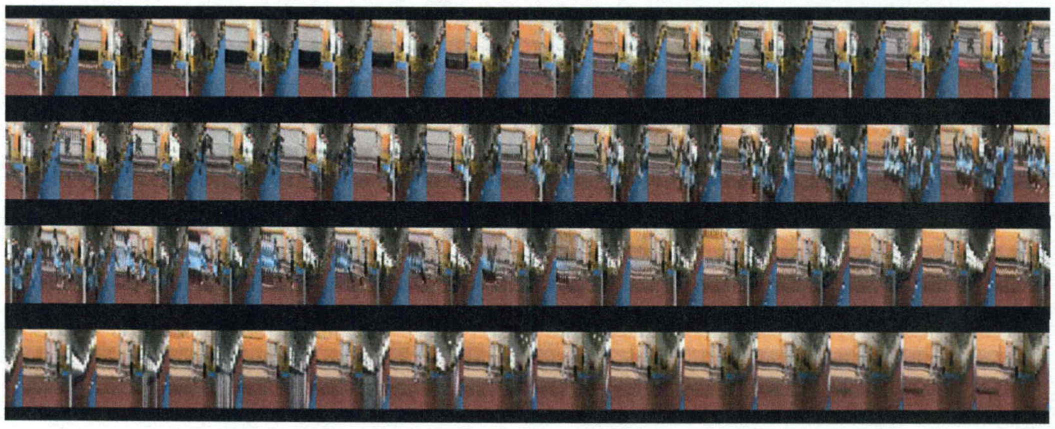

Figure 4.9: "High Jump" Clip Using Column Slice Width of 3 pixels

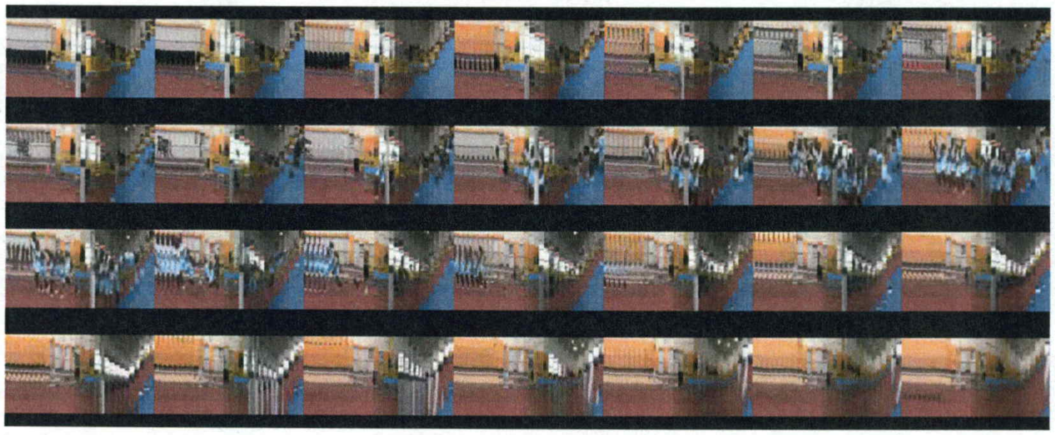

Figure 4.10: "High Jump" Clip Using Column Slice Width of 6 pixels

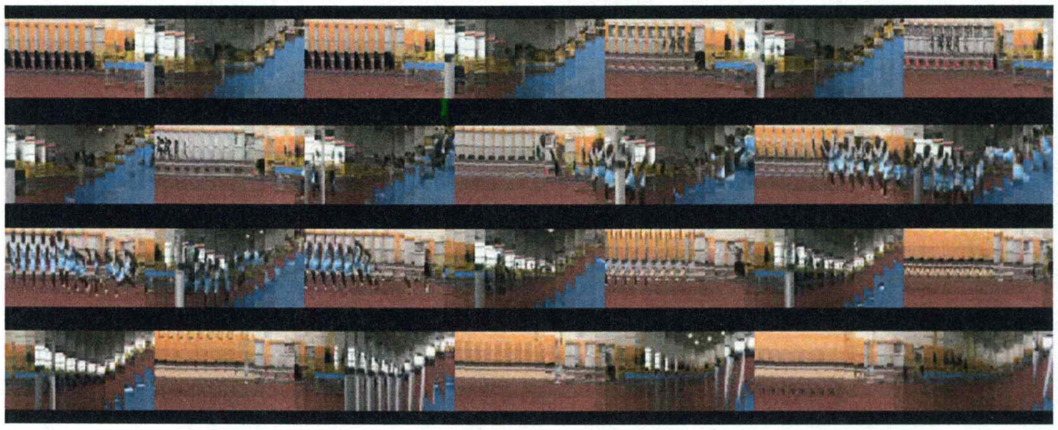

Figure 4.11: "High Jump" Clip Using Column Slice Width of 12 pixels

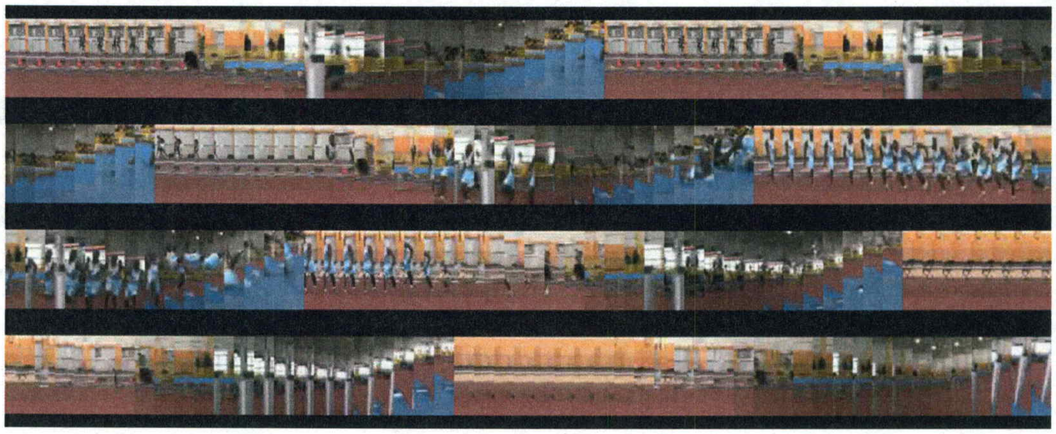

Figure 4.12: "High Jump" Clip Using Column Slice Width of 24 pixels

4.2.3 Rowwise Slicing

The same temporal slicing process is repeated on the original frames, this time with a horizontal orientation. This results in a 3780x171 strip. Once again, several segments of the same strip are shown side-by-side due to sizing constraints of this thesis. This process contains the results of row widths of 3, 6, 12, and 24 pixels.

Figures 4.13 through 4.16 show the rowwise temporal slicing for the Golf Swing clip for various slice widths.

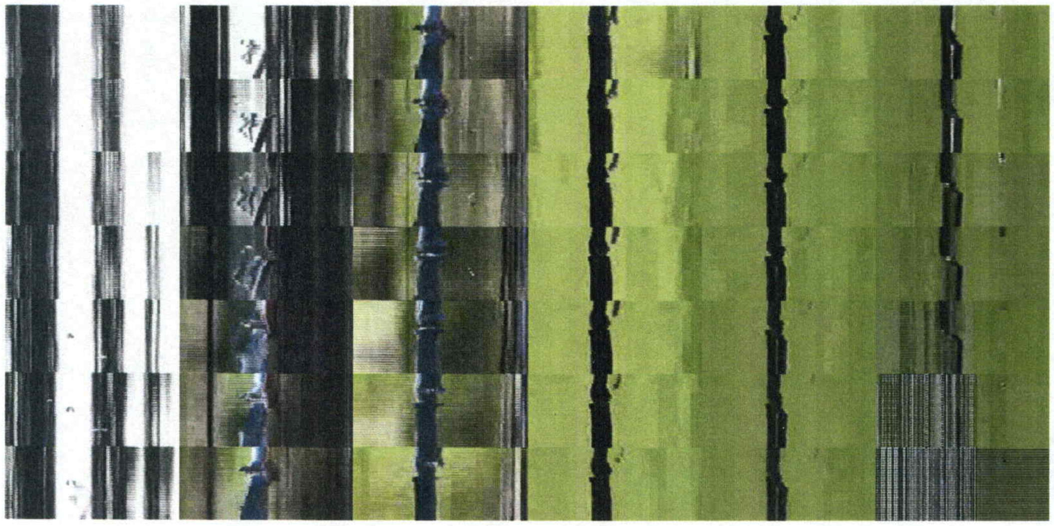

Figure 4.13: "Golf Swing" Clip Using Row Slice Width of 3 pixels

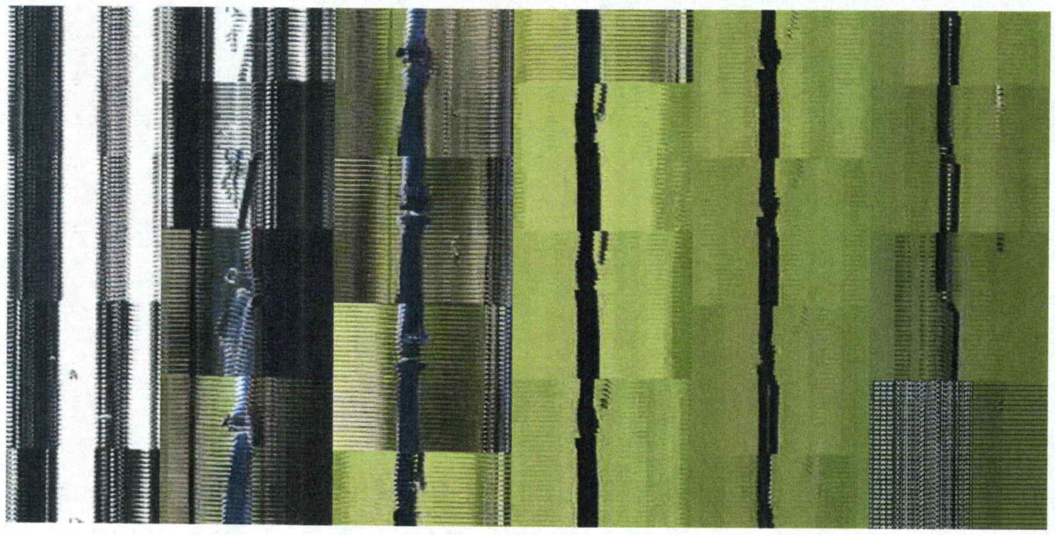

Figure 4.14: "Golf Swing" Clip Using Row Slice Width of 6 pixels

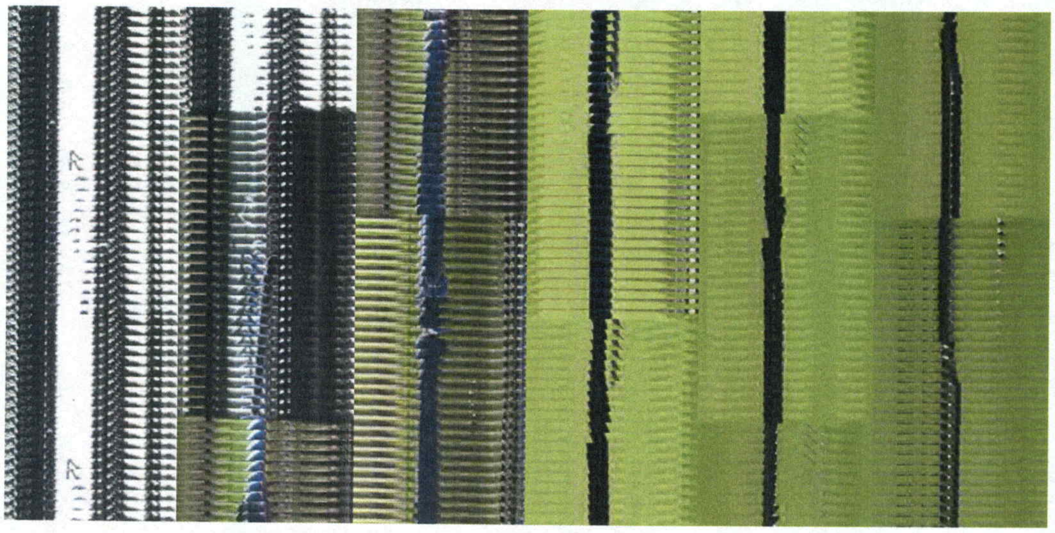

Figure 4.15: "Golf Swing" Clip Using Row Slice Width of 12 pixels

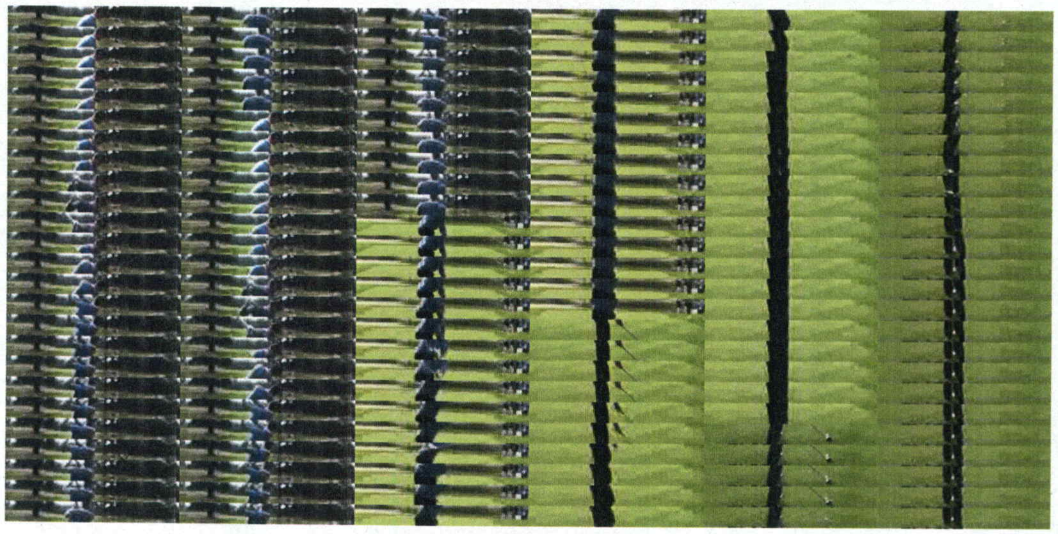

Figure 4.16: "Golf Swing" Clip Using Row Slice Width of 24 pixels

Figure 4.17 through 4.20 show the results of the rowwise temporal slicing for the High Jump video clip for various slice widths. Note that a significant amount of blank space is shown due to the internal cropping of the camera viewer in this video clip, and this information is also encoded into the frame strips.

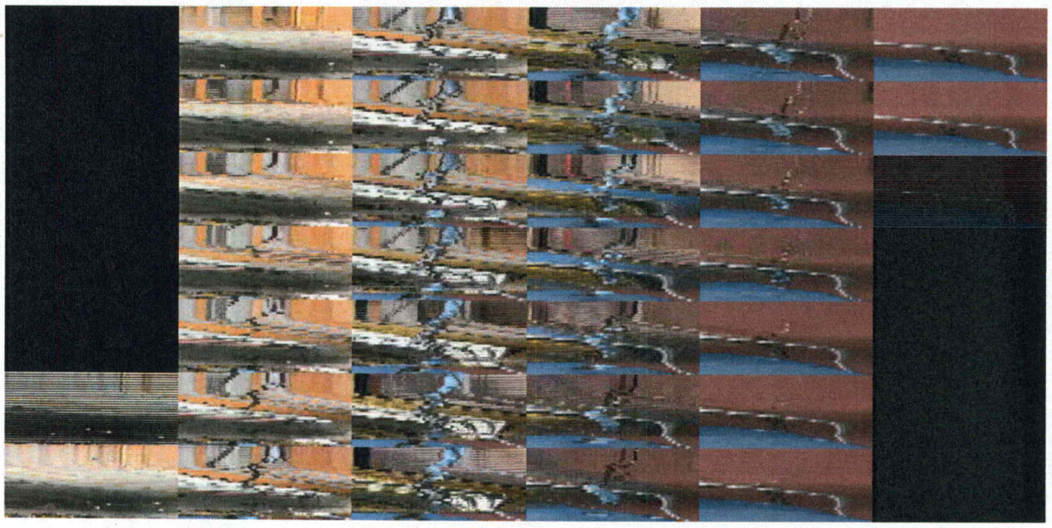

Figure 4.17: "High Jump" Clip Using Row Slice Width of 3 pixels

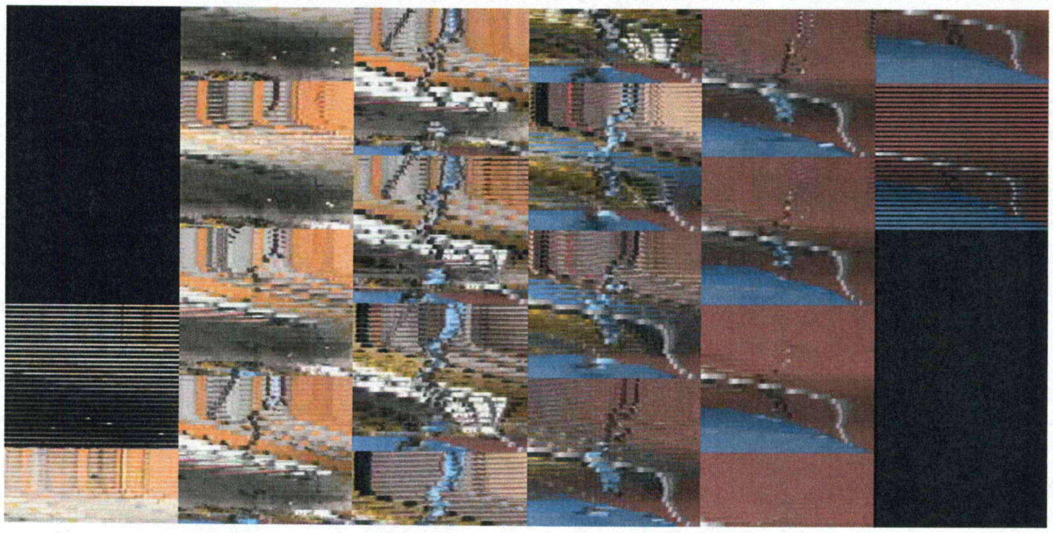

Figure 4.18: "High Jump" Clip Using Row Slice Width of 6 pixels

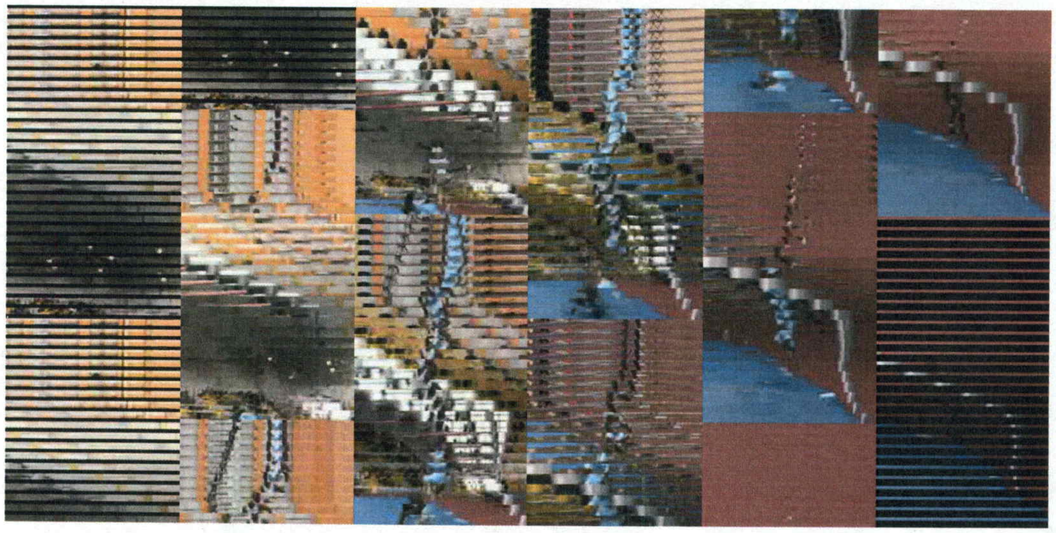

Figure 4.19: "High Jump" Clip Using Row Slice Width of 12 pixels

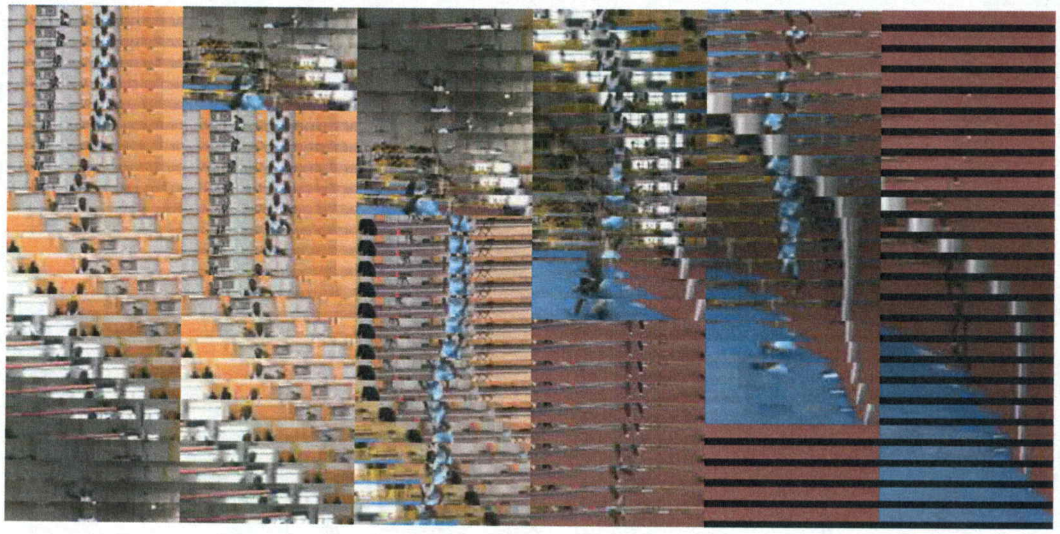

Figure 4.20: "High Jump" Clip Using Row Slice Width of 24 pixels

4.3 CNN Architecture

DeepSlice consists of two convolutional layers and their respective max pooling layers, followed by two consecutive convolutional layers, a pooling layer, and two more convolutional layers. Finally, a max pooling layer is connected to a single fully connected feature vector, which is connected to a Softmax output layer. Because the width of the image strip is usually several thousand pixels wide, the convolution layers contain a kernel of size (3x3), but the pooling windows are of size (2x2), or (1x2).

The first four convolutional layers contain 60 filters, while the last two convolutional layers contain 60 and 50 filters, respectively. Conv1, 3a, 3b, 4a and 4b use a stride of 3, with Conv2 using a stride of 1. Using a stride of 3 for the first convolutional layer, in addition to the slice widths of 3, helps to preserve the temporal information [3]. The last pooling layer contains a kernel of size 1x2 to account for the large width:height ratio of the image strip. Finally, the feature vector contains 1000 nodes, with the Softmax layer containing 101 output nodes.

Each layer contains a Rectified Linear Unit (ReLU) activation function, and the weights are initialized using a Gaussian distribution with a standard deviation of 0.01. The feature vector contains a dropout ratio of 0.5 [29]. Finally, the network is optimized using Stochastic Gradient Descent, with a momentum coefficient of 0.9, and a base learning rate of 0.003. The learning rate was decreased by a factor of 10 every 10000 iterations. DeepSlice was trained on an Nvidia Tesla K20c GPU and we used the *caffe* framework [30] to implement our neural networks for both the training and testing phases. The temporal slicing and preprocessing of the dataset was performed using the Python language, as well as the NumPy library, OpenCV, and FFMpeg. Tables 4.1 and 4.2 contains the network architecture and hyperparameters of the DeepSlice network.

Table 4.1: DeepSlice Architecture

Layer	Shape	Parameters	Initialization
data	(3, 128, 5040)		
conv1	(60, 3, 3)	stride=3	Gaussian(std=0.01)
pool1	(2, 2)	stride=2	
conv2	(60, 3, 3)	stride=1, bias=1	Gaussian(std=0.01)
pool2	(2, 2)	stride=1	
conv3a	(60, 3, 3)	stride=3	Gaussian(std=0.01)
conv3b	(60, 3, 3)	stride=3	Gaussian(std=0.01)
pool3	(1, 2)	stride=1	
conv4a	(60, 2, 2)	stride=2	Gaussian(std=0.01)
conv4b	(50, 3, 3)	stride=3	Gaussian(std=0.01)
pool4	(1, 2)	stride=1	
fc5	1000	dropout=0.5=1, bias=1	Gaussian(std=0.005)
fc6	101		Gaussian(std=0.01)

Table 4.2: DeepSlice Hyperparameters

Hyperparameter	Value
Base learning rate	0.003
Gamma	0.1
Step Size	10000
Momentum coefficient	0.9
Weight decay	0.0005

4.4 Summary

The DeepSlice network experimentation was carried out solely on the UCF-101 dataset, primarily because the dataset provided enough training data to train the network to be able to adequately distinguish between various human actions, even several that are very similar, such as "Apply Eye Makeup" and "Apply Lipstick". The UCF-101 dataset is well grouped, as it distinguishes between not only various clips, but groups of clips that share a similar context or background. The network architecture that was listed in this chapter, as well as the temporal slicing technique that was described, is the foundation for the experimental work performed in the following chapter.

CHAPTER 5

EXPERIMENTS AND RESULTS

This chapter showcases the results produced by the DeepSlice network. This includes the experimentation that determined the preprocessing techniques that rendered the best accuracy of the network, as well as the network parameters that also produced optimal results. This chapter also includes our observations on specific classes of the UCF-101 dataset that produced either noticeably positive or negative results. Finally, the computational performance of DeepSlice is observed and compared to the C3D network, and the training time is shown to be improved, while still achieving sufficient training to produce higher classification results as well.

The network was trained and validated using the UCF-101 train and test split-1 [6]. This step ensures that video clips taken from the same long video sample or similar environments are not duplicated in the training phase. These are the same training and test datasets used in experiments such as [3]. The training set includes a copy of each clip that is flipped horizontally to expand the training set.

5.1 Slice Width and Orientation

To determine a reasonable segmentation width for the columns extracted from each frame, we tested DeepSlice on images consisting of slices of 3, 6, 12, and 24 pixels in width, using a convolutional filter size of 3x6. Each dataset was tested using the same dataset split for 20,000 iterations with a minibatch size of 20. The results are shown in Table 5.1.

Table 5.1: Effect of Column Slice Width on DeepSlice Performance

Slice Width	Top-1 Accuracy	Top-5 Accuracy
3	**53.75%**	**75.76%**
6	19.45%	41.01%
12	21.00%	41.49%
24	52.99%	75.46%

An interesting observation from Table 5.1 is that slice widths of 3 and 24 both produce significantly impressive results. The slices of widths 6 and 12 provide a more equal balance of spatial and temporal data for each section of the video frames, while the width of 3 provides significant temporal information, and 24 provides significant spatial information to the network. This allows the network to learn to recognize the features and motion more easily, thus leading to higher classification accuracy.

The same process was repeated using row slicing instead of column slicing. The only difference in the network architecture itself was that any asymmetric kernels were rotated by 90 degrees. This is due to the fact that rowwise segmentation results in a 3600x171 pixel image, instead of a 128x5130 pixel image with the column segmentation. Thus, the 3x6 kernels using in the convolutional layers were 6x3 kernels for the row sliced dataset. The results of this experiment are shown in Table 5.2. A side-by-side comparison of the two preprocessing techniques is shown in Table 5.3.

Table 5.2: Effect of Row Slice Width on DeepSlice Performance

Slice Width	Top-1 Accuracy	Top-5 Accuracy
3	21.25%	42.37%
6	21.49%	40.49%
12	22.47%	43.23%
24	22.53%	**45.00%**

Table 5.3: Results of Column Slicing vs Row Slicing

Slice Width	Column Accuracy	Row Accuracy
3	**53.75%**	21.25%
6	19.45%	21.49%
12	21.00%	22.47%
24	52.99%	22.64%

Although the row accuracy is slightly higher than column accuracy for 6 and 12 pixel slices, the 3 and 24 pixel column slices perform drastically better than any of the other preprocessing methods. A possible explanation for this is that many instances of the actions contained in this dataset occur in the vertical direction, which would be more prominently displayed in columns instead of rows. In addition, the human targets in the video clips are often in vertical, erect positions for many of the classes. This would indicate that the action and motion is more easily recognized and extracted when the spatial information is maintained in the vertical dimension, versus the horizontal dimension. Therefore, for the majority of experiments in the remainder of this paper, the datasets used are the column-wise slicing datasets.

5.2 Kernel Size

We tested DeepSlice using three different kernel sizes within the convolutional layers. The kernel sizes tested were 3x3, 3x6, and 3x9. Each network was trained for 20,000 iterations, and the results are shown in Table 5.4.

Table 5.4: Effect of Kernel Size on DeepSlice Performance

Slice Width	**3x3**	3x6	3x9
3	**59.48%**	53.75%	54.21%
6	15.54%	19.45%	15.82%
12	16.52%	21.00%	18.75%
24	57.77%	52.99%	38.20%

Based on these results, the temporal slicing of width 3 appears to produce the best results with a filter size of 3x3. This suggests that the network performs better when it focuses on spatial context in earlier layers and delay integration of temporal information.

5.3 Classification Accuracy

Additional experimentation was conducted on the 3 and 24 width datasets, with the 3 width dataset producing the highest results. The loss for the training of DeepSlice for the 3 width dataset is shown in Figure 5.1.

After training for approximately 45 epochs (40,000 iterations), DeepSlice reached its optimal classification accuracy of 61.42%. Further details are shown in Table 5.5.

The frame strips that were produced for each video clip were created by taking slices of 3 pixels from each frame in a video clip. This ensured that enough spatial

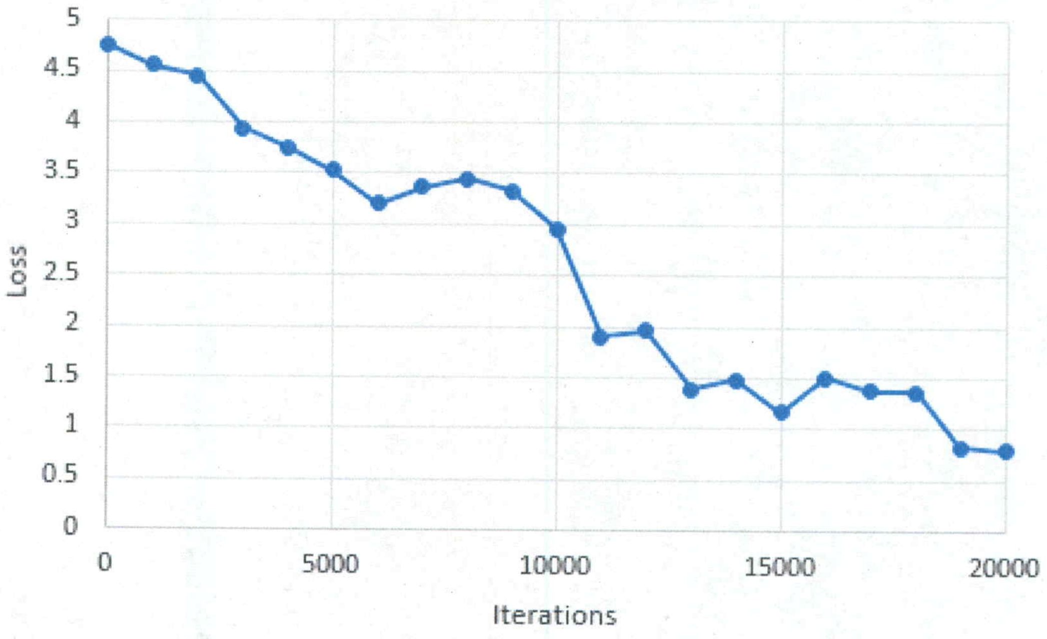

Figure 5.1: Loss during DeepSlice Network Training

Table 5.5: DeepSlice Classification Results for Slice Width of 3

K	Top-K Results
1	61.42%
5	81.12%
10	88.57%
50	98.75%

context was preserved so that the network was able to learn certain features about the subject of each video and follow the subject's motion accordingly, while also providing enough encoded temporal information for the network to make a fairly accurate prediction based on the entire video as a whole. The confusion matrix is shown in Figure 5.2 as a normalized gray-scale heat map.

We compare our results to other experiments that trained a network from scratch using the same UCF-101 dataset and the same training and test split. These results are shown in Table 5.6, and do not include any results that were obtained by using any external or post-training techniques such as Support Vector Machines (SVMs), feature extractors, fine-tuning, or pre-trained networks. All accuracies in Table 5.6 are also assumed to be top-1 accuracies.

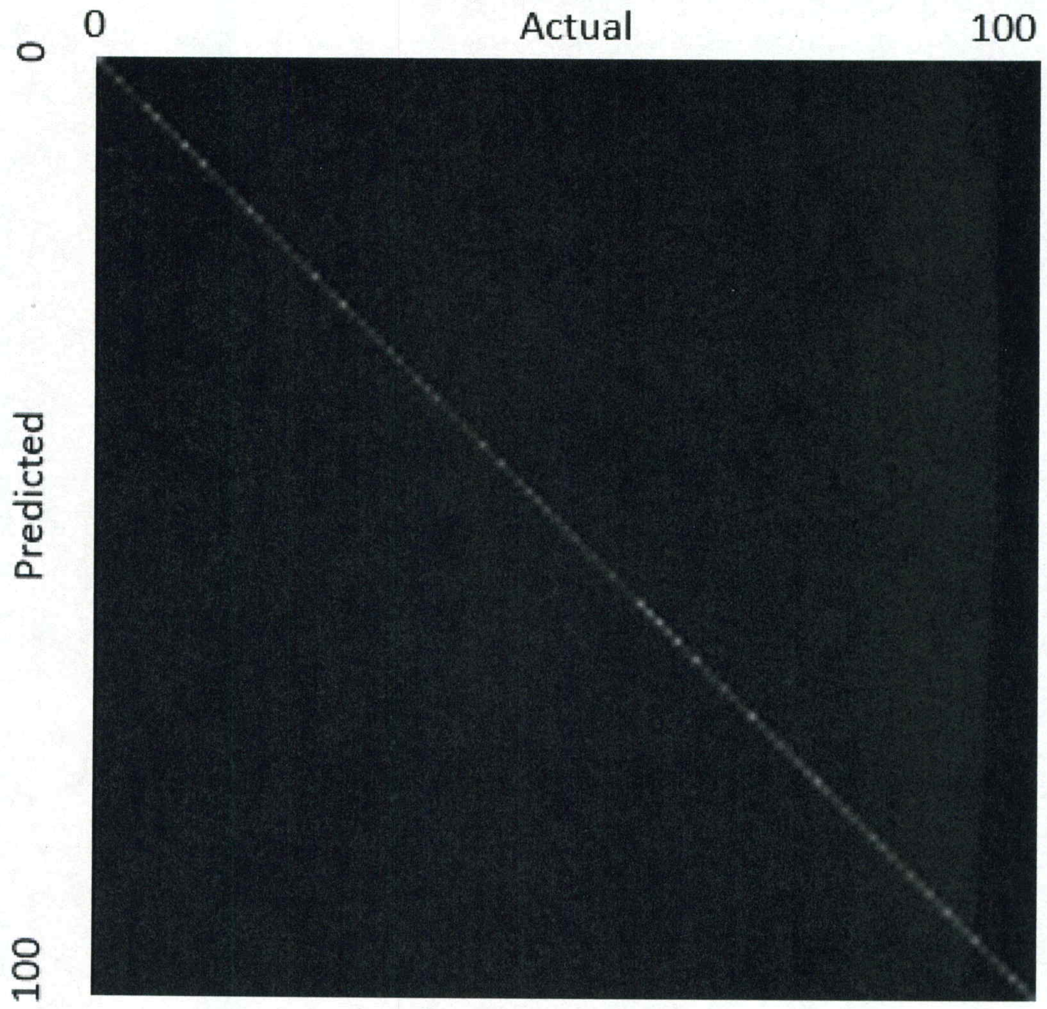

Figure 5.2: Confusion Matrix

Table 5.6: Network Accuracy Comparison

Network	Accuracy
DeepSlice	**61.42%**
C3D [3] (based on 3-fold split of UCF-101 dataset)	44%
Slow Fusion [2]	41.3%
Three Order [31]	38.6%
Two-Stream ConvNet [9]	48.6%

Tables 5.7 and 5.8 contain information that is not easily derived from the gray-scale confusion matrix. The precision and recall of each individual class was computed, and these results were then used to calculate the F1-Score of each class. The ten classes with the highest F1-Score are shown in Table 5.7 and the ten classes with the worst F1-Score are shown in Table 5.8. Each class contained an average of 32 individual video clips that were used to test the trained network, and each test video clip is taken from a group that was not previously shown to the network during the training phase. Finally, Table 5.9 shows the most common misclassifications that occurred during the testing phase.

Table 5.7: Highest Classification Rate by F1-Score

Class	Precision	Recall	**F1-Score**
Breast Stroke	1.0000	0.9583	**0.9787**
Typing	0.8889	0.9412	**0.9143**
Playing Guitar	0.9211	0.8974	**0.9091**
Jump Rope	0.9032	0.9032	**0.9032**
Billiards	0.8718	0.9189	**0.8947**
Playing Dhol	0.8444	0.9268	**0.8837**
Table Tennis Shot	0.9091	0.8571	**0.8824**
Playing Daf	0.8140	0.9459	**0.8750**
Wall Pushups	0.8529	0.8788	**0.8657**
Tennis Swing	0.8500	0.8500	**0.8500**

Table 5.8: Lowest Classification Rate by F1-Score

Class	Precision	Recall	**F1-Score**
Balance Beam	0.1818	0.1852	**0.1835**
Pizza Tossing	0.2778	0.1724	**0.2128**
Blowing Candles	0.2500	0.2222	**0.2353**
Mopping Floor	0.2368	0.3214	**0.2727**
Baby Crawling	0.2813	0.2727	**0.2769**
Haircut	0.3333	0.2727	**0.3000**
Rock Climbing Indoor	0.2727	0.3429	**0.3038**
High Jump	0.3103	0.3000	**0.3051**
Walking With Dog	0.3056	0.3548	**0.3284**
Floor Gymnastics	0.3143	0.3548	**0.3333**

Upon looking at Table 5.9, it is worth noting that several of the misclassifications made were due to major similarities between a majority of the videos between the predicted and the actual class. The most common case, "Balance Beam" and "Uneven Bars", occurs due to the striking similarities in the background and angle of the videos. Both videos are commonly shot in a gymnasium, and the human ac-

Table 5.9: Most Common Misclassifications

Predicted Class	Actual Class	Occurrences
Balance Beam	Uneven Bars	6
Baseball Pitch	Golf Swing	5
Walking With Dog	Soccer Juggling	5
Shaving Beard	Applying Eye Makeup	4
Punch	Floor Gymnastics	4
Pole Vault	High Jump	4
Basketball Dunk	Sumo Wrestling	4

tion contains very similar movement, such as flips and other gymnastics maneuvers. Classes such as "Shaving Beard" and "Applying Eye Makeup" contain videos that are taken from a similar viewing angle, and the human subject is commonly looking into a mirror during the video, which also may lead to the common misclassification. Actually, this is an interesting misclassification and indicates that our network is able to determine a person is doing something with his or face. Likewise, "Baseball Pitch" and "Golf Swing" contain similar quick movements of humans contrasted by a similar background, and the same theory can be applied to "Pole Vault" and "High Jump". It is not as apparent why such classes such as "Walking With Dog" and "Basketball Dunk" are misclassified as they are. One possible explanation is that during the course of each of these actions, the human subject remains upright during the duration of the video clip, in both the predicted class and its corresponding actual class. The column-wise slicing will preserve the length of the moving subject, and this can possibly cause such misclassifications to occur.

The classes containing the highest F1-Score usually contain an element or background that is consistent across all videos in the class, leading to such a high classification rate. For example, "Breast Stroke" almost always contains a background containing water, something learned by the network. Similarly, "Typing" and "Playing Guitar" contain keyboards and guitars, respectively. These objects are able to be learned as features by the network and this greatly contributes to the higher F1-Scores. Inversely, classes such as "Balance Beam" and "Pizza Tossing" do not contain such distinguishable features, or are extremely similar to other classes, as is the case with "Balance Beam." It is also possible that classes such as "Pizza Tossing" and "Blowing Candles" contain features that are not recognized by the network, possible due to the column-wise slicing. This would be a factor that would contribute to the low classification accuracy for such classes.

5.4 Performance

An additional benefit to the DeepSlice network is that it is able to achieve significantly improved classification accuracy without compromising the training time compared to other networks such as C3D. Because each training sample used in DeepSlice represents an entire video clip, the entire UCF101 dataset can be propagated through the network in less time. The fact that the DeepSlice network also handles each frame strip as a two-dimensional image instead of a three-dimensional video, means that the network is able to increase the batch size of the network based on the GPU specifications, and this also contributes to the decreased training time.

However, there are few direct comparisons to be made between the performance of C3D [3] and DeepSlice. The root cause of this is that fact that each video clip is processed in its entirety with C3D. However, only a fixed selection of frames are extracted from each video in the DeepSlice preprocessing techniques. This means that not all frames of a certain video clip will be seen by our network. The amount of video data that is skipped by the preprocessing method increases as the length of the video clip increases. However, with C3D, each video clip is split into fixed-duration segments, which are then treated as separate training samples. With this in mind, note that this is why DeepSlice is trained on 19,982 samples, one per video clip. However, C3D trains on 46,548 samples, after each of the original 19,982 clips are split into the fixed-length segments. This factor, along with the fact that 3D convolution is more computational expensive than 2D convolution, are the biggest reasons that the training time of C3D is drastically longer that the training time for

DeepSlice. However, it is also noteworthy that DeepSlice is able to achieve a higher classification accuracy, in spite of less data being propagated through the network. Further information is shown in Figure 5.10 below. The times shown in Figure 5.10 are for the training only, not the validation or testing times.

Table 5.10: Training Time Comparison

	C3D	DeepSlice
Batch Size	5	20
Train Samples	46,548	19,982
Iterations	148,953	40,000
Epochs	14	45
Time (hours)	**24.83**	**8.88**

A possible solution to high training times is to reduce the sampling rate of the videos during the preprocessing stage. However, after a certain threshold is reached, the network is not provided enough spatial or temporal information and the classification accuracy begins to decrease. Further experiments in [3] and [2] show that the number of frames sampled is an important factor in the classification accuracy of any network.

5.5 Summary

Through experimentation, we determined that the achieved higher classification rates with column-wise frame slicing, compared to row-wise slicing. Afterwards, it was concluded that slice widths of 3 and 24 pixels yield higher accuracy compared to 12 and 6 pixels, with 3 pixels edging out 24 pixels for the highest classification results. With the optimal kernel size of 3x3, the classification rate of the DeepSlice network peaked at 61.42% and also required less training time to achieve this accuracy, compared to 3D convolution as illustrated by C3D. Upon looking at the most common misclassifications, it can be reasonably concluded that many of the mistakes made by the network were fairly reasonable, which further illustrates the capability of the network to track and learn various human movements and motions.

CHAPTER 6

CONCLUSION

We demonstrate in this thesis an innovative and improved approach to training convolutional neural networks on spatio-temporal datasets. By mapping the temporal dimension of the video frames into spatial data during the temporal slicing stage, the computational overhead of the temporal convolution can be mitigated and decreases the overall training time of the network. Moreover, this approach also significantly improves the classification accuracy of the network as well, as the experiments conducted for this paper demonstrate around 20% improvement over current state-of-the-art networks trained from scratch on the same dataset.

The two-fold benefit of our approach to human action classification using deep convolutional neural networks reduces the workload of the training process. Through the compression of the temporal data, the network only encounters two dimensions. This aspect, which results in faster training time and improved accuracy, reduces the time needed to thoroughly train a network, allowing results to be obtained in a faster, more efficient manner. This improvement can be achieved without compromising the classification accuracy when compared to other networks that use 3D convolution or other computationally expensive approaches. The experimentation on the varying kernel sizes provided further evidence that decreasing the kernel size often has beneficial results, similar to the work displayed in [3]. This further strengthens our confidence in the current DeepSlice architecture, and serves as a formidable foundation for continued work in this field.

There is much additional work that can be performed concerning this approach. As with all artificial neural networks, there is an infinite amount of possible configurations for the network architecture and hyper-parameters that may lead to increased performance. Further adjustments to the convolution and pooling windows may also improve performance. Another advancement that can be made is by experimenting with DeepSlice on another dataset such as Sports1M [2]. One possible approach to this would be to train the network on Sports1M, but use UCF101 as the validation/testing set to demonstrate DeepSlice's robustness. Finally, there are further experiments that could be conducted with the frame preprocessing that might also lead to better results. These include changing the number of frames sampled from each video, and the number of sequential pixel columns sampled. Further experimentation can be conducted to determine a time-space tradeoff with the number of sampled frames, and the size of the frame strips in relation to the computational processing time of the network, and its relationship to the accuracy of the network. More potential experimentation can be performed to determine if spacing of the temporal slices is detrimental to the network accuracy. If this is not the case, it is possible to maintain the network's original accuracy while significantly reducing the computation and training time and further increasing the speedup of the network.

APPENDICES

APPENDIX A

UCF-101 CLASSES

1. ApplyEyeMakeup	13. BlowDryHair	25. CuttingInKitchen
2. ApplyLipstick	14. BlowingCandles	26. Diving
3. Archery	15. BodyWeightSquats	27. Drumming
4. BabyCrawling	16. Bowling	28. Fencing
5. BalanceBeam	17. BoxingPunchingBag	29. FieldHockeyPenalty
6. BandMarching	18. BoxingSpeedBag	30. FloorGymnastics
7. BaseballPitch	19. BreastStroke	31. FrisbeeCatch
8. Basketball	20. BrushingTeeth	32. FrontCrawl
9. BasketballDunk	21. CleanAndJerk	33. GolfSwing
10. BenchPress	22. CliffDiving	34. Haircut
11. Biking	23. CricketBowling	35. Hammering
12. Billiards	24. CricketShot	36. HammerThrow

37. HandstandPushups
38. HandstandWalking
39. HeadMassage
40. HighJump
41. HorseRace
42. HorseRiding
43. HulaHoop
44. IceDancing
45. JavelinThrow
46. JugglingBalls
47. JumpingJack
48. JumpRope
49. Kayaking
50. Knitting
51. LongJump
52. Lunges
53. MilitaryParade
54. Mixing
55. MoppingFloor
56. Nunchucks
57. ParallelBars
58. PizzaTossing
59. PlayingCello
60. PlayingDaf
61. PlayingDhol
62. PlayingFlute
63. PlayingGuitar
64. PlayingPiano
65. PlayingSitar
66. PlayingTabla
67. PlayingViolin
68. PoleVault
69. PommelHorse
70. PullUps
71. Punch
72. PushUps
73. Rafting
74. RockClimbingIndoor
75. RopeClimbing
76. Rowing
77. SalsaSpin
78. ShavingBeard
79. Shotput
80. SkateBoarding
81. Skiing
82. Skijet
83. SkyDiving
84. SoccerJuggling
85. SoccerPenalty
86. StillRings
87. SumoWrestling

88. Surfing
89. Swing
90. TableTennisShot
91. TaiChi
92. TennisSwing
93. ThrowDiscus
94. TrampolineJumping
95. Typing
96. UnevenBars
97. VolleyballSpiking
98. WalkingWithDog
99. WallPushups
100. WritingOnBoard
101. YoYo

REFERENCES

[1] Michael A. Nielsen. *Neural Networks and Deep Learning.* Determination Press, 2015.

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

[3] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[4] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[5] Graham W Taylor, Rob Fergus, Yann LeCun, and Christoph Bregler. Convolutional learning of spatio-temporal features. In *European conference on computer vision*, pages 140–153. Springer, 2010.

[6] 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.

[7] Dan C Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1237. Barcelona, Spain, 2011.

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

[9] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision*, pages 20–36. Springer, 2016.

[10] Andrea Tacchetti, Leyla Isik, and Tomaso Poggio. Spatio-temporal convolutional neural networks explain human neural representations of action recognition. *arXiv preprint arXiv:1606.04698*, 2016.

[11] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[12] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[13] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[14] D. Warde-Farley, I. J. Goodfellow, A. Courville, and Y. Bengio. An empirical analysis of dropout in piecewise linear networks. *ArXiv e-prints*, December 2013.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

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

[17] Andrew G Howard. Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402*, 2013.

[18] Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. Describing videos by exploiting temporal structure. In *Proceedings of the IEEE international conference on computer vision*, pages 4507–4515, 2015.

[19] Sergio Guadarrama, Niveda Krishnamoorthy, Girish Malkarnenkar, Subhashini Venugopalan, Raymond Mooney, Trevor Darrell, and Kate Saenko. Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 2712–2719, 2013.

[20] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013.

[21] Adhavan Jayabalan, Harish Karunakaran, Shravan Murlidharan, and Tesia Shizume. Dynamic action recognition: A convolutional neural network model for temporally organized joint location data. *arXiv preprint arXiv:1612.06703*, 2016.

[22] Agne Grinciunaite, Amogh Gudi, Emrah Tasli, and Marten den Uyl. Human pose estimation in space and time using 3d cnn. In *Computer Vision–ECCV 2016 Workshops*, pages 32–39. Springer, 2016.

[23] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *CoRR*, abs/1406.2199, 2014.

[24] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[25] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[26] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[27] Chong-Wah Ngo, Ting-Chuen Pong, and Hong-Jiang Zhang. On clustering and retrieval of video shots. In *Proceedings of the ninth ACM international conference on Multimedia*, pages 51–60. ACM, 2001.

[28] Chong-Wah Ngo, Ting-Chuen Pong, and Hong-Jiang Zhang. On clustering and retrieval of video shots through temporal slices analysis. *IEEE Transactions on Multimedia*, 4(4):446–458, 2002.

[29] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[30] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[31] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pages 527–544. Springer, 2016.