Political-advertisement video classification using deep learning methods

by

Aashish Mani Dhakal

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Adisak Sukul, Co-major Professor
Wallapak Tavanapong, Co-major Professor
David A. M. Peterson

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this thesis. The Graduate College will ensure this thesis is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2019

DEDICATION

I would like to dedicate my thesis to my parents Mr. Ananta Ram Dhakal and Ms. Gita Luintel, and my sister Aakankshya for their unlimited support. I would also like to thank my friends and colleagues who have helped me throughout my graduate studies. Last but not the least, my advisors for their constant motivation and guidance.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

ABSTRACT

Today's digital world consists of vast multimedia contents: images, audios and videos. Thus, the availability of huge video datasets have encouraged researchers to design video classification techniques to group videos into categories of interest. One of the topics of interest to political scientists is automated classification of a video advertisement into a political campaign ad category or others. Recent years have seen a plethora of deep learning-based methods for image and video classification. These methods learn feature representation from the training data along with the classification model. We investigate the effectiveness of three recent deep-learning based video classification techniques for the political video advertisement classification. The best technique among the three yields an accuracy of 80%. In this thesis, we further improve the classification accuracy by combining the results of classification of text features with that of the best deep learning methods we studied. Our method achieves the classification accuracy of 91%.

CHAPTER 1. INTRODUCTION

Online video advertising has been a major source of marketing in recent years. Political parties have also changed the way they conduct political campaigns. Rather than using the old technology of printed media and television broadcast, political parties have increasingly used various web-based techniques to advertise their party and candidates online. YouTube [1] has been one of the prominent sources for election campaign advertising.

Out of four million advertisements on YouTube we collected during 2014 to 2019, only 3-4% of the advertisements are about election campaigns to promote, attack a party or a candidate, or contrast candidates. This type of video typically has information about the sponsor who endorses the content of the advertisement; it does not include news or any kind of shows that mention campaigners or political parties. We refer to this type of video as political ads hereafter. The goal of this thesis is to find the best algorithm that automatically identifies political ads with over 90% accuracy in order for political scientists to study micro-targeting of political ads.

Previously, feature based techniques have been investigated for political ad classification, P-Ads Video Classifier [2]. These techniques use Term-Frequency (TF) and Term-Frequency Inverse Document Frequency (TF-TDF) features of text obtained from Optical Character Recognition (OCR) of image frames and speech-to-text of audio to classify political video ads. The best feature based technique gave the accuracy of around 79% on our CyAds dataset, which has 2479 ads: 1233 political and 1246 non-political ads. We aim to improve the accuracy further by investigating three recent deep learning techniques [3-5] that were successful for other video classification tasks.

We implemented CNN Fusion method in [3], which applied 3D convolutional filters on a sequence of frames. The method was evaluated on the Sports-1M dataset [3] which has 1 million videos in 487 categories. Furthermore, four different fusion methods, single frame, late fusion, early fusion, and slow fusion, were studied. Two different spatial resolutions: a low-level context stream and a high-resolution fovea stream, were used as input. On the UCF-101 [4] dataset, which has 13220 videos in 101 action categories, 3D CNN with slow fusion [3] significantly improved the classification accuracy to 65.4%, up from 41.3% by the compared feature based technique.

We implemented the first of the two models studied in [5]. The first model explored five different pooling architectures: Convolutional Pooling, Late Pooling, Slow Pooling, Local Pooling, and Time-domain Convolution. Among them, Convolutional Pooling offered the best accuracy. The second model connected Long Short-term Memory (LSTM) to the final CNN layer and used a softmax layer to predict the class for each frame. The confidence score of each frame for a video was averaged and the video was classified into the category with highest average confidence score. The first model with Convolutional Pooling gave an accuracy of 90.8% on the Sports-1M dataset, but only 88.6% on the UCF-101 dataset.

We implemented Long-term Recurrent Convolutional Networks (LRCN) introduced in [6]. LRCN processed a variable length visual input with a CNN that output to a stack of recurrent sequence models. The recurrent models produced a final variable-length prediction. Both the CNN and LSTM weights were shared across time, resulting in a representation that scaled to arbitrarily long sequences. LR-CN offered 82% accuracy on activity classification on the UCF-101 dataset [6].

We ran all of the three aforementioned deep learning models on the UCF-101 [4] dataset to verify our implementation before running them on our CyAds dataset. When weighing the prediction results from the best of the three deep-learning models and the best P-Ads Classifier [2] from our previous work, we achieved the highest accuracy of 91%.

The paper is organized as follows. Chapter 2 provides a brief summary of related works. Chapter 3 describes the methodology. We describe the experimental design, results and analysis in Chapter 4. The conclusion and the description of the future work are described in Chapter 5.

CHAPTER 2. RELATED WORKS

There have been plenty of techniques proposed in the realm of video classification. The older video classification techniques are feature based techniques and the more recent techniques use deep learning.

2.1 Feature Based Techniques

The general idea was to use text-based features, audio-based features and video-based features to perform the task of video classification. The methods in [7-9] used closed captioning, a method of letting hearing-impaired people know what is being said in a video by displaying text of the speech on the screen, as text feature. Wei *et. al* [10] used audio and video features to first to detect the video shots and then these shots were grouped into scenes. They then used the OCR from detected scenes and close captioning to generate text to classify news video into types of news stories. Wang *et. al* [8] used transcripts of dialogs extracted from speech as audio based features along with closed captioning. Brezeale *et. al* [9] used the color histograms in the RGB space as visual features for classification in addition to closed captioning and subtitles (if present) as text based features. Jasinschi *et. al* [11] used audio from the video to produce probability values for six categories: noise, speech, music, speech + noise, speech + speech, and speech + music and used these as audio based features. They also used visual features to detect commercials and closed captioning as text based features to classify videos. Thus, various feature based techniques have been implemented, which use either text bases, audio based, video based features, or combination of multiple of these features for video classification.

Table 1: *Recent feature based techniques for video classification*

Publications	Closed Captioning	Audio	Color	OCR
Zhu *et. al*, 2001 [11]	✓			✓
Wang *et. al*, 2003 [8]	✓			
Brezeale *et. al*, 2008 [9]	✓		✓	
Wei *et. al*, 2000 [7]		✓		
Jasinschi *et. al*, 2001 [10]	✓	✓		

2.2 Deep Learning Methods

Hinton *et. al.* [12] introduced Deep Belief Networks (DBNs) to greedily train each layer of the network to train deep networks in a better way. This sparked research in deep networks and thus many ways to train deep nets are available. As a result, many deep learning based networks are available to classify the videos. Frame classification methods were one of the popular video classification methods using deep learning techniques. A video clip was treated as a collection of frames. Features of each frame were extracted using one of the CNN (Convolutional Neural Network) models pre-trained on ImageNet [13], AlexNet [14], VGGNet [15], GoogleNet [16], and ResNet [17]. Finally, frame-level features were averaged into a video-level representation as input to a classifier, e.g., Support Vector Machine (SVM). Zha *et. al.* [18] systematically studied the performance of image-based video recognition using features from different layers of deep models together with multiple convolutional kernels for classification. They demonstrated that off-the-shelf CNN features coupled with kernel SVMs can obtain decent recognition performance. Frame classification methods do not account for

the temporal aspect, i.e. they do not consider the relationship between frames with time, thus losing motion information.

The effectiveness of CNNs on a variety of tasks lies in their capability to learn features from raw data as an end-to-end pipeline targeting a particular task. Therefore, in contrast to the image-based classification methods, there have been many works focusing on applying CNN models to learn hidden spatio-temporal feature patterns in a video. End-to-end CNN architectures made use of the temporal features in a video. Ji *et. al.* [19] introduced the 3D CNN model that operates on stacked video frames, extending the traditional 2D CNN designed for images to the spatio-temporal space. The 3D CNN utilized 3D kernels for convolution to learn motion information between adjacent frames in volumes segmented by human detectors. Training of CNNs with inputs of 3D volumes is usually time-consuming. To effectively handle 3D signals, Ji *et. al.* [20] introduced factorized spatio-temporal convolutional networks that factorize the original 3D convolution kernel learning as a sequential process of learning 2D spatial kernels in the lower layer.

Motivated by the fact that videos can naturally be decomposed into spatial and temporal components, Yi *et. al.* [15] proposed a two-stream approach that breaks down the learning of video representation into separate feature learning of spatial and temporal clues. The authors first adopted a typical spatial CNN to model appearance information with raw RGB frames as inputs. For temporal clues among adjacent frames, they explicitly generated multiple-frame dense optical flows, upon which a temporal CNN is trained. The authors reported promising results on two action recognition benchmarks. As the two-stream approach contains many implementation choices that may affect the performance, Ye *et. al.* [21] evaluated different options, including dropout ratios and network architectures.

The temporal CNN in the two-stream approach by Yi *et. al.* [15] explicitly captured the motion information among adjacent frames, which, only depicts movements within a short time window. This is not sufficient for video analysis, since complicated events in videos usually consist of multiple actions over a long time period. Therefore, researchers have recently attempted to leverage RNN models to account for the temporal dynamics in videos. Among these RNN models, LSTM is a good fit without suffering from the "vanishing gradient" effect, and has demonstrated its effectiveness in several tasks such as image/video captioning. Wu *et. al.* [22] fused the outputs of LSTM models with CNN models to jointly model spatio-temporal clues for video classification and observed that CNNs and LSTMs are highly complementary. Feichtenhofer *et. al.* [23] improved the two-stream approach by exploring a better fusion approach to combine spatial and temporal streams. They found that two stream approach is better at modeling correlations of spatial and temporal streams.

It was observed that CNNs and LSTMs are highly complementary. Using all of the frames in a video was computationally expensive and may degrade the performance of recognizing a class of interest as not all the frames are relevant. This issue has motivated researchers to leverage the attention mechanism to identify the most discriminative spatio-temporal volumes that are directly related to the targeted semantic class. Sharma *et. al.* [24] proposed the first attention LSTM for action recognition with a soft-attention mechanism to attach higher importance to the learned relevant parts in video frames. Gavrilyuk *et. al.* [25] introduced VideoLSTM, which applied attention to convolutional LSTM models to discover relevant spatio-temporal volumes.

Table 2: *Recent deep learning techniques for video classification*

Deep Learning Techniques	**Publications**
End-to-End CNN (3D-CNNs)	Karpathy *et. al*, 2014 [3] Ji *et. al*, 2013 [19] Tran *et. al*, 2015 [26]
Two Stream Approach	Simonyan *et. al*, 2014 [15] Ge *et. al*, 2013 [20] Ye *et. al*, 2015 [21]
Modeling Long-Term Temporal Dynamics (Mostly RNN + CNN)	Donahue *et. al, 2017.* [6] Ng *et. al*, 2015 [5] Wu *et. al*, 2015[22]
Incorporating Visual Attention	Sharma *et. al*, 2015 [24] Li *et. al*, 2018[24]

2.3 Implemented Deep-learning methods

We describe the three method**s** we implemented in this study in detail.

2.3.1 CNN Fusion

Karpathy *et. al* [3] proposed CNN Fusion and designed a Sports-1M dataset for evaluating the classification methods they studied. CNN Fusion basically focuses on answering certain questions like: What kind of temporal connectivity pattern in CNN architecture is able to take advantage of local motion in a video? How does this additional information improve the performance?

Images are easily cropped and scalable to a certain, fixed size but videos are not, thus videos cannot be processed with the same architecture. Thus, this model converted each video into smaller clips to learn spatio-temporal features. They proposed four different architectures, Single Frame, Late Fusion, Early Fusion and Slow Fusion, to explore the fusion of information over a temporal dimension through the network.

The general architecture used in all the CNN Fusion models is as follows: *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-FC(4096)-FC(4096),* where *C(d, f, s)* indicates a convolutional layer with *d* filters of spatial size *(f *f),* applied to input with stride *s. FC(n)* represents a fully connected network with *n* nodes. *P* are pooling layers that pool spatially in non-overlapping regions. *N* denotes normalizing layers and has the same parameters as those of [14].

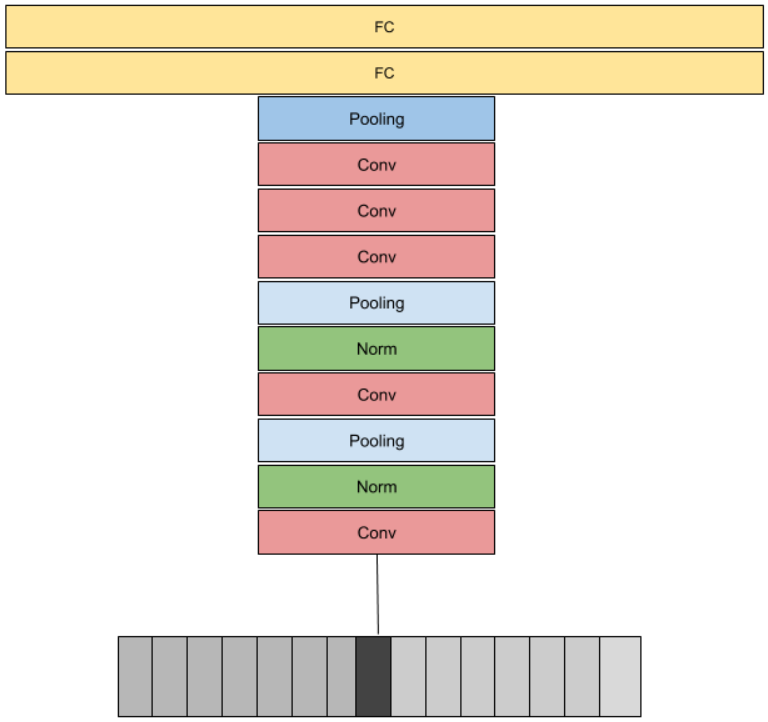

Figure 2-1: Single Frame Architecture

The above architecture is a single frame architecture for fusing information over temporal dimension through the network. Convolutional, Normalization, Pooling and Fully Connected layers are represented as Conv, Norm, Pooling and FC respectively. Single frame method only incorporates the static appearance in the classification category. The model takes one frame at a time. Each frame is classified separately and the confidence score for each frame per class is averaged. The video is classified to the class with the highest confidence score.

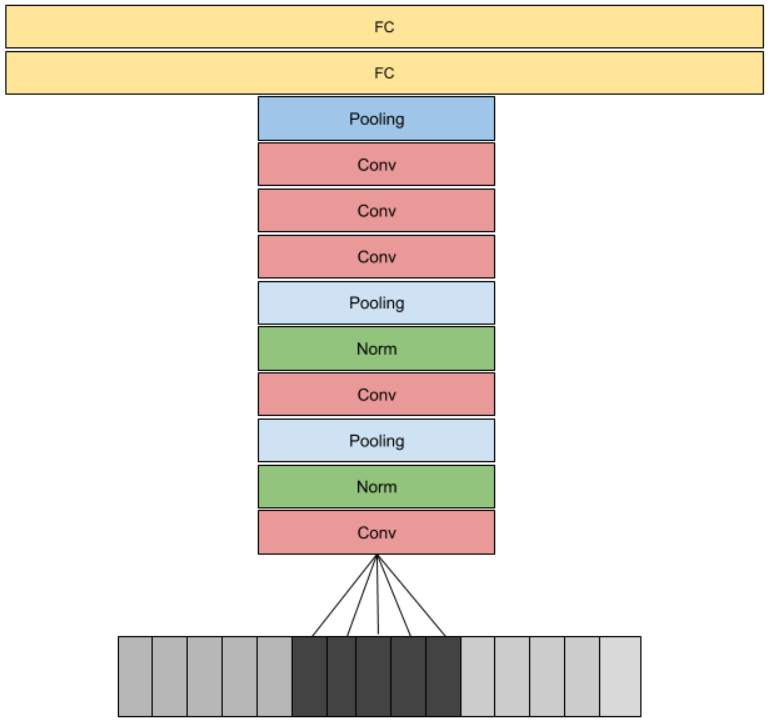

Figure 2-2: Early Fusion Architecture

The above architecture is an early fusion architecture for fusing information over temporal dimension through the network. Convolutional, Normalization, Pooling and Fully Connected layers are represented as Conv, Norm, Pooling and FC respectively. Early fusion method combines information across an entire time window immediately on pixel level. A clip is divided into 15 frames and the information of the middle 5 frames are combined on pixel level. A single clip of 15 frames is classified at a time. The confidence score of all the clips are averaged and the video is classified to the class with the highest confidence score.

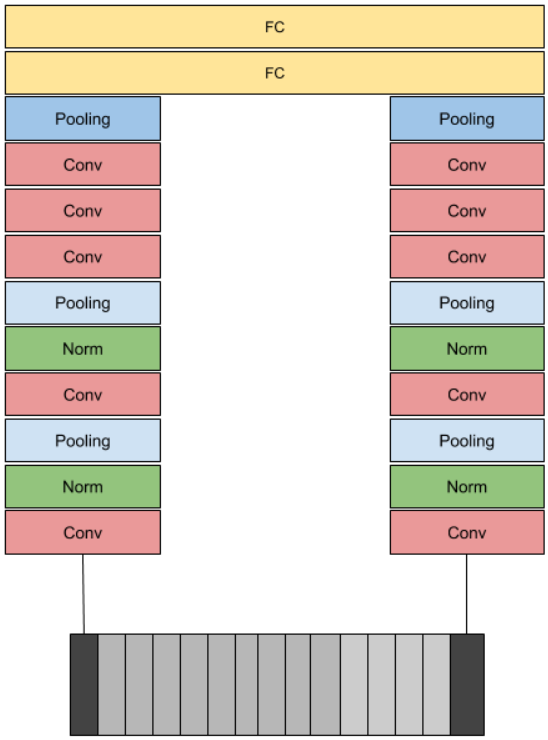

Figure 2-3: Late Fusion Architecture

The above architecture is a late fusion architecture for fusing information over temporal dimension through the network. Convolutional, Normalization, Pooling and Fully Connected layers are represented as Conv, Norm, Pooling and FC respectively. Late Fusion model combined two separate single-frame networks; they run in parallel until the last Convolutional Layer *C(256, 3, 1)* with shared parameters at a distance of 15 frames. The two networks are merged in the first fully connected layer. Thus, it cannot actually detect any motion but the first fully connected layer detects global motion characteristics by comparing outputs of both

networks. A single clip of 15 frames is classified at a time. The confidence score of all the clips are averaged and the video is classified to the class with the highest confidence score.

2.3.2 CNN-LSTM

Ng *et al.* [5] proposed two models capable of handling full length videos. The first method explored various temporal network architectures to adapt a normal CNN. The second method explicitly modeled the video as an ordered sequence of frames. The model employed recurrent neural networks that use Long Short-Term Memory *(LSTM)* cells that are connected to the output of the underlying CNN.

For both of the above methods, the authors investigated two different classes of CNN architectures which are capable of aggregating information in video-level. The first class of techniques were investigated on different pooling architectures to see which of the pooling techniques perform the best. Temporal feature pooling is one of the extensively used techniques for video classification [16][21][27].

The pooling operation is directly incorporated as a layer. The authors carried out various experiments to see where to put the temporary pooling layer in the network architecture. The authors conducted various experiments on different pooling methods and mainly the particular layer whose features were aggregated. They experimented with various pooling networks rather than just using max pooling. They also experimented with implementing a fully connected layer before a pooling layer. They experimented with various types of pooling architectures.

The pooling architectures are: (a) Conv Pooling, (b) Late Pooling, (c) Slow Pooling, (d) Local Pooling and (e) Time-Domain Convolution. Conv Pooling is described further in Chapter 3.

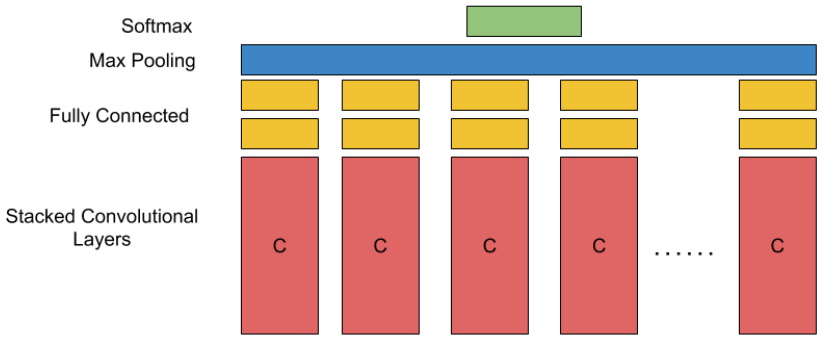

Figure 2-4: Late Pooling

Late Pooling first passes convolutional features through two fully connected layers before applying the max-pooling layer. The weights of all convolutional layers and fully connected layers are shared. Softmax layer is used for video classification at the end.

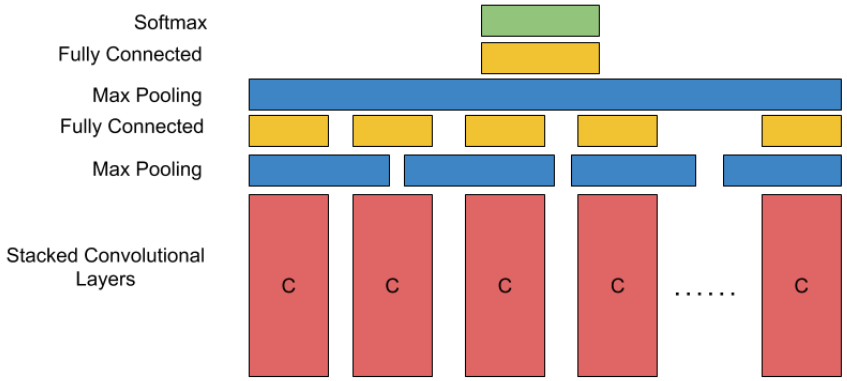

Figure 2-5: Slow Pooling

Slow Pooling hierarchically combines frame level information from smaller temporal windows using two max pooling layers. First, max-pooling is applied over 10-frames of convolutional features with the stride of 5. Each max-pooling is then followed by a fully-connected layer that combines the outputs of all fully-connected layers. Softmax layer is used to classify the video at the end.

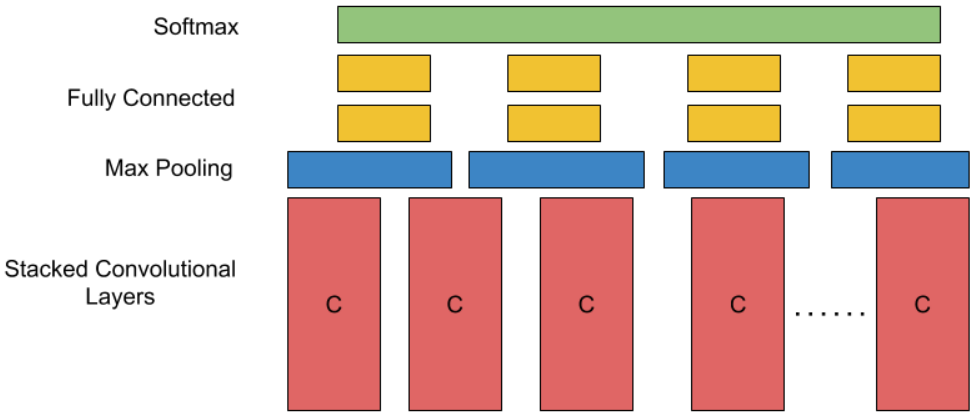

Figure 2-6: Local Pooling

Similar to Slow Pooling, Local Pooling combined frame level features locally after the last convolutional layer but contains only a single max-pooling after the convolutional layers. This is then followed by two fully connected layers, with shared parameters. Finally, all layers are connected using a larger softmax layer which is used to classify the video at the end.

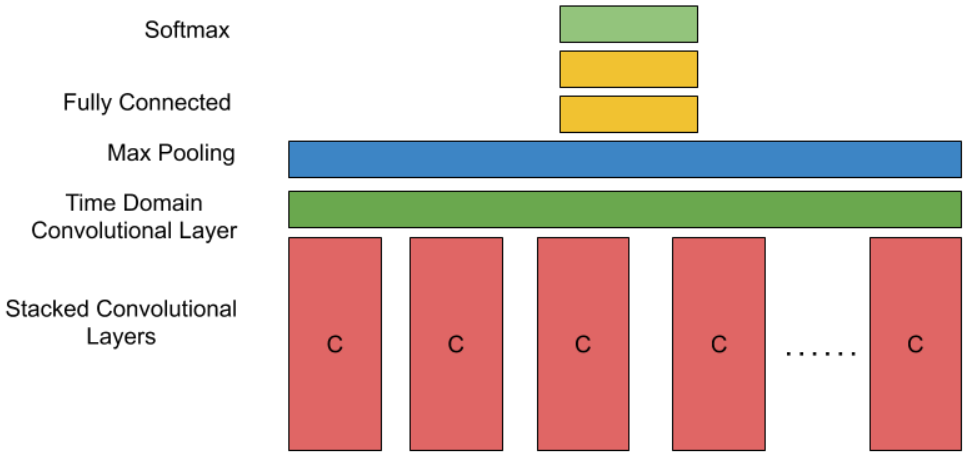

Figure 2-7: Time Domain Convolutional Pooling

Time-Domain Convolutional Pooling contains an additional time-domain convolutional layer before applying max pooling across all the frames. Max-pooling operation is performed on the temporal domain after the time-domain convolutional layer. The main idea of this model is to be able to capture similarities between frames within a small temporal window.

The authors also experimented with an architecture based on GoogLeNet [15]. This architecture uses the max-pooling operation after the dimensionality reduction (average pooling) layer in GoogLeNet. This max pooling layer is connected to two fully connected layers and finally the softmax layer. The architecture was further enhanced by adding two fully connected layers of size 4096 with Rectified Linear Unit (ReLU) activations on top of the 1000D (Dense Net) output but before the softmax layer.

2.3.3 LR-CN

Donahue *et. al* [6] proposed Long-term Recurrent Networks (LR-CN). It is a class of architecture for visual recognition and description which combined convolutional layers and long-range temporal recursion and is end-to-end trainable. This paper focused on three different activities related to video: video activity classification, image caption generation and video caption generation.

LR-CN model combined a deep hierarchical visual feature extractor (such as CNN) with a model that can learn to recognize and synthesize temporal dynamics for tasks involving sequential data like audios, videos, etc. In LR-CN, each visual input x_t, at time t, was passed through a feature transformation $\phi_v(.)$ with parameters *V*, usually a CNN, to produce a fix length vector representation $\phi_v(x_t)$. The outputs of ϕ_v were then passed into a recurrent sequence learning module. This paper focused on solving three different problems: Activity Recognition, Image Captioning and Video Captioning.

Image Captioning takes fixed input and gives sequential outputs. The input is a single image and is represented by x and the output is the caption, which is sequence of words represented by $<y_1, y_2, y_3, \ldots, y_t>$, where *t* is the number of words in the caption. Thus, it can be represented by $x \longmapsto <y_1, y_2, y_3, \ldots, y_t>$.

Video Captioning takes sequential number of inputs and gives sequential outputs. The inputs are sequences of images $x_1, x_2, x_3, \ldots, x_t$, where *t* is the total number of input frames and the output is $y_1, y_2, y_3, \ldots, y_{t'}$ where *t'* is the number of words in the caption. This can be represented by $< x_1, x_2, x_3, \ldots, x_t > \longmapsto < y_1, y_2, y_3, \ldots, y_{t'}>$

CHAPTER 3. METHODOLOGY AND PROPOSED WORK

We implemented existing feature based technique, P-Ads Video Classifier to classify the videos. We extracted texts, OCR and speech to text, from the videos. We then used the texts to calculate Term frequency (TF) and Term frequency-Inverse Document Frequency (TF-IDF). We first used only TF as a feature to classify the videos and then use TF-IDF separately as a feature to classify political videos from non-political ones. As deep learning methods for video classification showed promising results, we investigated the effectiveness of existing deep learning techniques for political video ad classification.

3.1 Images and Feature Extraction

For the pre-processing step required for both the feature based techniques and deep learning models, we extract the images (at the playback rate of the videos) from all the videos using FFMPEG [28]. For the feature based technique, we do not make any further modifications. For the implementation of first two deep learning techniques [3][4], we resize the images of 224x224 to 170x170 pixels. For the implementation of [5], we extract features from the video using Extractor [29], which extracts the metadata from all the images in the video. The metadata includes properties of the images, for e.g. the size of the image, ISO, aperture size, quality, etc. We then make a Numpy [30] sequence to store all the information.

3.2 P-Ads Video Classifier

3.2.1 Text based features:

The first feature taken into account is text from OCR from the last three frames of an input video. We consider only the last three frames because political videos have information

about the sponsor of the advertisement at the end of the video. The second set of feature taken into account are speech-to-text transcript. Texts from both the OCR from frames and the transcript from speech-to-text are extracted using Google API [31]. Both of these features are taken into account to train the model. The next step involves removing all the stop words and most of the punctuations from the text documents using Natural Language Processing library NLTK [32].

3.2.1.1 OCR text extraction

The main task is to extract the important frames from the video. Since we can extract frames at various frame rates, choosing the right rate is difficult task. The frames are extracted from all the videos at 1 frame per second (fps) using FFMPEG [28]. Our main aim is to get the end frames which generally have the information about the sponsors and if the advertisement is being sponsored. We clean up the texts from OCR by removing all the stop words and then concatenate all the texts into a single file.

3.2.1.2 Speech to Text generation

We first of all extract the complete audio from the video. We then transcribe the extracted audio files using Google's Speech API [31]. This API only generates valid English dictionary words as outputs. Therefore, we don't need to clean the output texts. After the completion of the process, the transcription text for each individual advertisement video is stored in a separate file.

All the texts from the OCR and audio are then transcript into a single file. The texts are then used to train multiple classifiers and then tested. We use multiple classifiers like Naïve Bayes, Support Vector Machines and Logistic Regression and some boosting algorithms like AdaBoost and Gradient Boost to test the best classifier.

Then, the training text dataset is vectorized. Vectorization is a process of converting a text document dataset into feature vectors. Tf or term-frequency of a token in a document, is the number of times the token is present in a text document. IDF or inverse document frequency is a logarithmic function of ratio of number of text documents in the corpus and number of documents where the token appears.

TF or term frequency score of each token is calculated as [33]:

$$TF\ (t, d) = 1 + \log f_{t,d} \text{ or zero if } f_{t,d} = \text{zero}$$

where t = term, d = document, $f_{t,d}$ = count of term t in document d

IDF or Inverse document frequency is calculated as [33]:

$$IDF\ (t, D) = \log (N / |\{d \in D : t \in d\}|$$

where N = total number of documents, D = set of all documents

TF-IDF = TF * IDF.

Vectorization process is completed when all the n-gram in the training corpus dataset are assigned a tf-idf weight. TF-IDF score of each individual token in the n-gram model was considered as a feature in the classification model.

3.2.1.3 Keyword based features:

According to Federal Election Commissions guidelines, any political campaign ads must contain disclaimers about the sponsors and who is it for. If the keywords are present in the advertisement, the video can be classified as a political one. The disclaimer keywords used are: Paid for, Approved by, For president, Authorized by, Responsible for, Approve this message, candidate, president.

3.3 Recent Deep Learning Models

P-Ads Video Classifier, in our case, uses only text based features from OCR and speech to text. This technique does not take visual features and other features like, the motion of the video into account. We implemented three state of the art deep learning techniques, CNN Fusion, CNN-LSTM and LR-CN techniques which takes all these features to classify political videos. These video classification techniques were shown to perform better than the feature based techniques and give good results in UCF-101 dataset [6].

3.3.1 CNN Fusion

Slow Fusion method performs the best from all the methods suggested in [3], which was implemented. This model is a balanced approach where two networks run parallel to fuse temporal information throughout the network. The higher layer has access to more temporal and spatial information. Out of 15 frames, the middle 10 frames are selected and then the first convolution layer is extended to apply to every filter of Temporal length T=4 with stride 2. It produces 4 responses and we have 4 parallel networks running. Second and third layers carry the information across all the 10 frames, which enables the third convolutional layer to have access to all the information across the 10 frames.

The architecture shown in Fig 3-1 is a slow fusion architecture for fusing information over temporal dimension through network. Convolutional, Normalization, Pooling and Fully Connected layers are represented as Conv, Norm, Pooling and FC respectively. Slow Fusion method makes sure that the temporal information is slowly fused throughout the network such that the higher layers get access to more global information in both spatial and temporal dimensions.

This method performed the best out of all of these four architectures proposed in the model as it keeps the temporal information for 2 layers rather than other models where the temporal information was not considered at all. Thus, we implemented the Slow Fusion method in in our work.

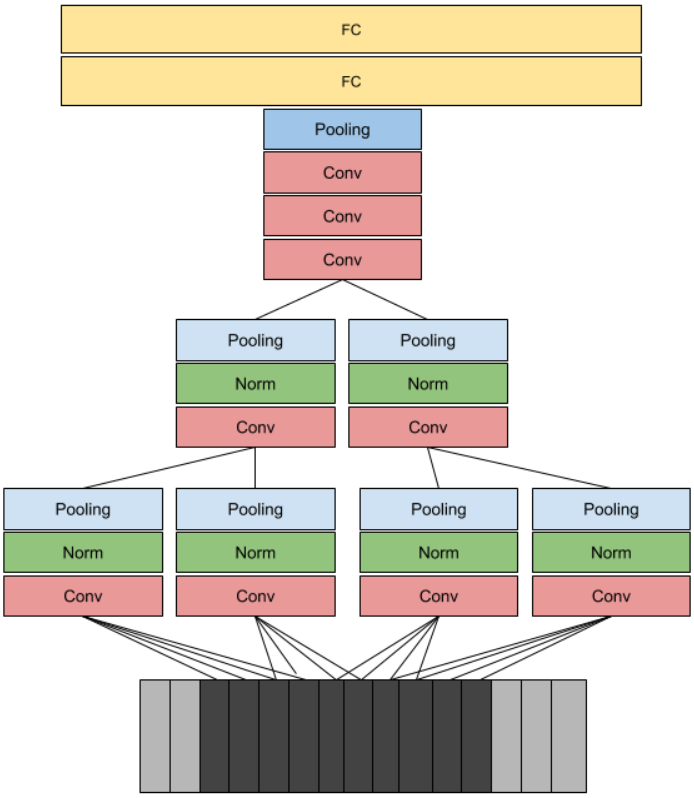

Figure 3-1: Slow Fusion Architecture

3.3.2 CNN-LSTM

Ng. *et.* al. [5] explored various temporal network architectures to adapt a normal CNN to perform better. This method explicitly models the video as an ordered sequence of frames. The model employed recurrent neural networks that uses Long Short-Term Memory *(LSTM)* cells that are connected to the output of the underlying CNN.

This method uses GoogLeNet [16] as the CNN architecture. GoogLeNet uses a network-to-network approach, stacking Inception modules to form a network. It takes a single image as an input. The images are then passed through multiple inception modules, each of which is applied, in parallel, 1*1, 3*3, 5*5 convolution and max-pooling operations. The results of these operations are concatenated.

The pooling operation is directly incorporated as a layer. This enables to experiment with different locations of the temporal pooling layer with respect to the network architecture. Conv Pooling model performs max-pooling over the final convolutional layer across the video's frames. The spatial information in the output of the convolutional layer is preserved through the max pooling operation over the time domain

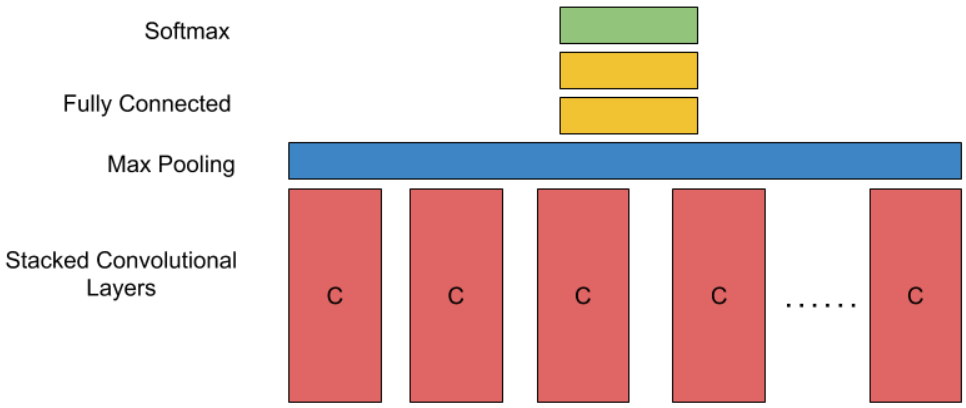

Figure 3-2: Convolutional Pooling

We implemented recurrent neural networks to explicitly consider sequences of CNN activations. The variations between frames may encode additional information as videos

contain dynamic content. We used a deep LSTM architecture [34] where output of one layer of LSTM was used as an input for the next layer. There were various experiments done and best results were given by using five stacked LSTM layers, each with 512 memory cells. The LSTM layers were followed by a softmax classifier, which made prediction at every frame.

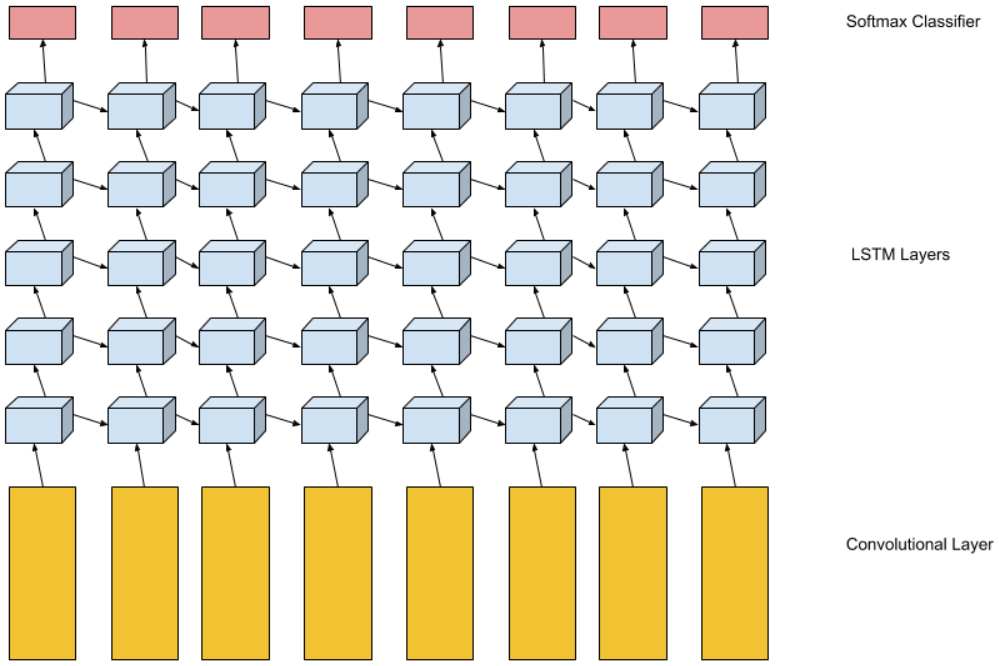

Figure 3-3: *Deep Video LSTM Architecture*

This model takes 120-frames from a video for the classification. We randomly choose 120 frames from the total number of frames extracted from the video. Deep Video LSTM Architecture takes input from the final CNN layer at each consecutive video frame. The outputs

of CNN are processed forward through time and upwards through five layers of stacked LSTMs. A softmax layer predicts the class at each time step. By expanding small networks to larger ones and fine tuning, significant speedup was achieved compared to training a large network from scratch.

In order to combine LSTM frame-level predictions into a single video-level prediction they tried several approaches but the best results were given by linearly weighing the predictions and averaging the result. The predictions were done on one frame at a time. In order to make the video level prediction, the confidence scores of the classification results of all the frames are averaged. The predicted class is the one with the highest confidence score.

3.3.3 LR-CN

In LR-CN, each visual input x_t is passed through a feature transformation $\phi_v(.)$ with parameters *V*, usually a CNN, to produce a fix length vector representation $\phi_v(x_t)$. The outputs of ϕ_v are then passed into a recurrent sequence learning module. Activity recognition model takes sequential images $x_1, x_2, x_3, \ldots, x_t$ where *t* is the number of frames, as inputs and gave a static output. It can be represented as $< x_1, x_2, x_3, \ldots, x_t > \mapsto y$. It takes videos of arbitrary length *T* as input, but with goal of predicting a single label. Each frame in a length *T* sequence was the input to a single convolutional network. LR-CN is trained to predict the video's activity class at each time step. To produce a single label prediction for an entire video clip, we average the label probabilities, across all frames and chose the most probable model.

The CNN component of LR-CN in the activity recognition problem is a hybrid of *CaffeNet* [35] which is a minor variant of *AlexNet* [14] and the network. In this model, classification of the whole video is done by averaging scores all the video frames. The most

influential hyperparameters were found to be the number of hidden units in the LSTM. Various experiments were performed and hidden units of size 1024 yielded the best results.

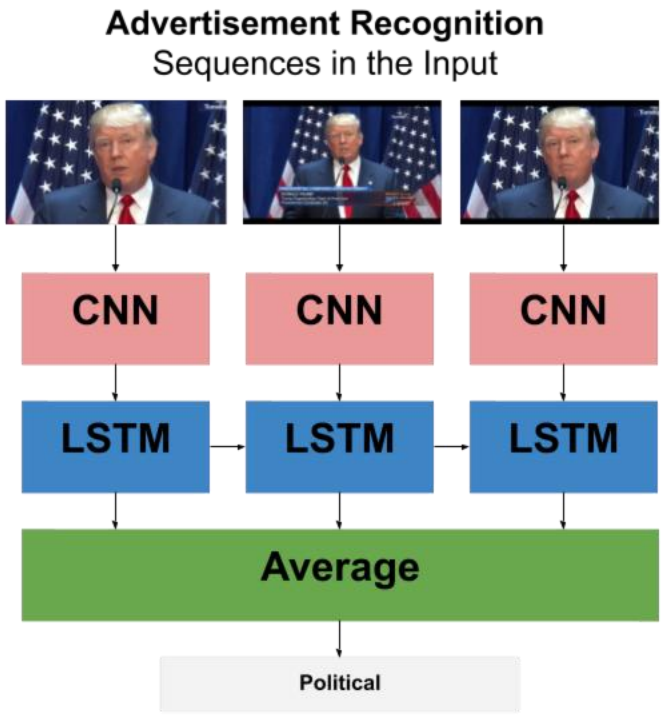

Figure 3-4: Activity Recognition using LR-CN

3.4 Combined Model

We get results from both the P-Ads Video Classifier and all the implemented deep learning models. CNN Fusion give us the comparable accuracy on both the CyAds and UCF-101 datasets. Thus, we take CNN Fusion and results from P-Ads Video Classifier (both OCR and speech-to-text) to perform the classification.

We take the confidence score from both the CNN Fusion and the P-Ads Video Classifier for each video. We first average the confidence scores from both the models for a video. The video is classified into the category, either political or non-political with the highest confidence score. We can choose either of CNN-LSTM or LR-CN to combine it with the results from the P-Ads Video Classifier, but these models give vast difference in accuracy while implementing on the UCF-101 dataset and on the CyAds dataset.

CHAPTER 4. EXPERIMENTS AND ANALYSIS

4.1 Datasets and Performance Metrics

4.1.1 Datasets

All of the deep learning experiments mentioned in Chapter 3 were conducted on the UCF-101 dataset [4]. The dataset consists of 13,220 videos of 101 classes. Each class consists of videos that shows only one activity. Example classes are applying make-up class, horse riding class, and playing soccer class. Most of these videos are short, 10 seconds or less. They are used for activity recognition.

We conducted all the experiments, P-Ads Video Classifier and the deep learning methods mentioned in Chapter 3, on our own dataset, CyAds dataset along with the UCF-101 dataset. We collected the advertisements from sources like PCL Stanford [36] and YouTube [1], and labelled the data as either political or non-political video. CyAds dataset consists of 2,479 video ads: 1,233 political and 1,246 non-political video ads. Political advertisements show a candidate or a party promoting or attacking candidates in the same or another party. Non-political advertisements consist of advertisements for cars, insurance, restaurants, etc. Table 4 shows that we selected roughly the same number of videos from each sub-category of non-political videos from YouTube [1].

Table 3: *Datasets used for the experiments*

Dataset	**Number of Videos**	**Number of Classes**
UCF-101	13,220	101
CyAds	2,479	2

Table 4: *Number of videos in each category of non-political video ads*

Class	**No. of videos**
Insurance	257
Cars	234
TV Shows	223
Games	198
Restaurants	187
Miscellaneous	147

4.1.2 Performance Metrics

To measure the performance of all the classification models, we used standard performance metrics: Accuracy (A), Precision (P), recall (R) [37], and F1 score [38]. These metrics are defined on True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN).

$$\boldsymbol{P} = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FP}} ; \boldsymbol{R} = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FN}} ; \boldsymbol{F1} = \boldsymbol{2} * \frac{\boldsymbol{P * R}}{\boldsymbol{P + R}} ; \boldsymbol{A} = \frac{\boldsymbol{TP + TN}}{\boldsymbol{TP + FP + TN + FN}}$$

4.2 Feature based Classifier Training

All the deep learning classifiers were trained with early-stopping, which enabled us to specify an arbitrary large number of training epochs and stop training once the model performance stops improving on a holdout validation dataset.

We experimented with the P-Ads Video Classifier on the CyAds dataset. We then proceeded with application of the three deep learning classifiers on the UCF-101 dataset and finally continue training and testing them on the CyAds dataset.

4.2.1 P-Ads Video Classifier

The dataset was split randomly into the training dataset and test dataset with 70% and 30% split respectively. We trained the P-Ads Video Classifier using 5-fold cross-validation on the CyAds dataset summarized in Table 3. We used Scikit-learn [39] for data mining and data analysis. We experimented with different linear classifier algorithms: Bernoulli Naïve Bayes, Stochastic Gradient Descent, Logistic Regression, and Support Vector Machine (SVC) with linear kernel. We also experimented with AdaBoost and Gradient Boost algorithms to see their performance.

In order to find suitable values of the hyper parameters, we used the grid search algorithm [40]. We experimented with different word n-gram models and the best result was with n-grams in the range (1, 3).

We first trained our classifier on text-features. We investigated Term Frequency (TF) and Term Frequency-Inverse Document Frequency (TF-IDF) feature representation. TF gave higher accuracy than TF-IDF while just using text extracted from speech thus the results were based upon it. Table 5 and Table 6 show that Random Forest classifier performed the best out of all the implemented algorithms. Table 5 shows that it gave an accuracy of 0.80, precision of 0.79, recall of 0.68, and F1-score of 0.72 on the training dataset of the CyAds dataset. Table 6 shows that it gave an accuracy of 0.78, precision of 0.77, recall of 0.64, and F1-score of 0.70 on the test dataset of the CyAds dataset. The best parameters for each algorithm are listed in Appendix A.

Table 5: *Five-fold cross validation performance of P-Ads Video Classifier using TF extracted from speech to text on the training dataset of the CyAds dataset*

Algorithm	Accuracy	Precision	Recall	F1-Score
Stochastic Gradient Descent	0.60	0.61	0.72	0.69
SVC	0.67	0.69	0.54	0.59
Bernoulli Naïve Bayes	0.64	0.78	0.48	0.60
Random Forest	0.80	0.79	0.68	0.72
Ada Boost	0.62	0.68	0.62	0.58
Gradient Boost	0.71	0.70	0.70	0.71

Table 6: *Performance of P-Ads Video Classifier using TF extracted from speech to text on the test dataset of the CyAds dataset*

Algorithm	Accuracy	Precision	Recall	F1-Score
Stochastic Gradient Descent	0.56	0.54	0.84	0.65
SVC	0.63	0.66	0.54	0.59
Bernoulli Naïve Bayes	0.67	0.78	0.48	0.60
Random Forest	0.78	0.77	0.64	0.70
Ada Boost	0.58	0.64	0.58	0.54
Gradient Boost	0.67	0.67	0.67	0.67

We then took texts from both the OCR and speech which was transcript into a single file. We evaluated Term Frequency (TF) and Term Frequency-Inverse Document Frequency (TF-IDF). TF-IDF gave higher accuracy than TF when we take all the texts from the video. The reported results shown in Table 7 and Table 8 are based on TF-IDF. Bernoulli Naïve Bayes performed the best. Table 7 shows that it gave an accuracy of 0.892, precision of 0.886, recall of 0.895, and F1-Score of 0.889 on the training dataset of the CyAds dataset. Table 8 shows that it gave an accuracy of 0.863, precision of 0.858, recall of 0.862, and F1-Score of 0.862 on the test dataset of the CyAds dataset. The best parameter value for each algorithm is presented in Appendix A.

Table 7: *Five-fold cross validation performance of P-Ads Video Classifier using TF-IDF extracted from OCR and speech to text on the training dataset of the CyAds dataset*

Algorithm	Accuracy	Precision	Recall	F1-Score
Stochastic Gradient Descent	0.786	0.794	0.792	0.793
SVC	0.823	0.815	0.823	0.823
Bernoulli Naïve Bayes	0.892	0.886	0.895	0.889
Random Forest	0.838	0.842	0.836	0.836
Ada Boost	0.798	0.796	0.790	0.790
Gradient Boost	0.739	0.666	0.643	0.643

Table 8: *Performance of P-Ads Video Classifier using TF-IDF extracted from OCR and speech to text on the test dataset of the CyAds dataset*

Algorithm	Accuracy	Precision	Recall	F1-Score
Stochastic Gradient Descent	0.748	0.763	0.758	0.761
SVC	0.782	0.776	0.784	0.781
Bernoulli Naïve Bayes	0.863	0.858	0.862	0.862
Random Forest	0.798	0.802	0.796	0.796
Ada Boost	0.754	0.752	0.750	0.750
Gradient Boost	0.698	0.635	0.603	0.603

4.3 Deep Learning Video Classification Techniques

We trained the three deep learning models on both the CyAds Dataset and the UCF-101 Dataset. The datasets were randomly split into training dataset and testing dataset with 70% and 30% split respectively. We trained them with all the hyper-parameters mentioned in the Appendix A below.

4.3.1 CNN Fusion

We used the pre-trained InceptionNet Version 3, which was followed by 2D global average pooling. We then used DenseNet with 1024 hidden units and then finally Rectified Linear Unit (ReLU) as an activation function. Finally, a SoftMax layer was added at the end to make the binary prediction.

Figure 4-1 shows the shows the training accuracy of CNN Fusion on the CyAds Dataset. The accuracies of CNN Fusion model on the UCF-101 test dataset and CyAds test dataset are 0.802 and 0.787 respectively.

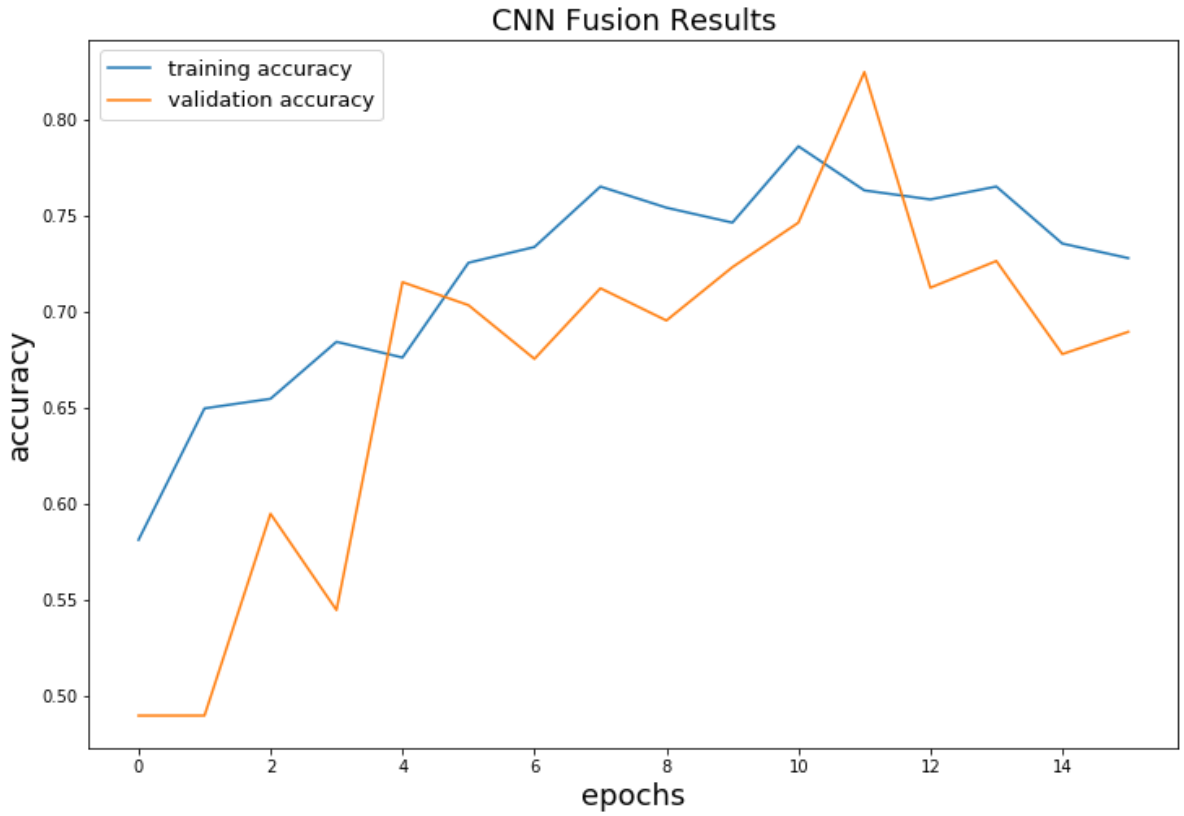

Figure 4-1: Performance of CNN Fusion on the CyAds train Dataset

4.3.2 CNN-LSTM

The pooling model was first optimized on a cluster using Downpour Stochastic Gradient Descent [41] starting with a learning rate of 10^{-5}. For LSTM, we used the same optimization method with a learning rate of $N * 10^{-5}$ where N is the number of frames. The learning rate was exponentially decayed over time. To reduce CNN training time, the parameters were initialized from a pre-trained ImageNet model and then fine-tuned on Sports-1M videos. The video labels were back propagated on each frame rather than once per clip.

For the final results on LSTM, during training, the gradients were back-propagated through convolutional layers for fine tuning.

Figure 4-2 shows the shows the training accuracy of CNN-LSTM on the CyAds Dataset. The accuracies of CNN-LSTM model on the UCF-101 test dataset and CyAds test dataset are 0.886 and 0.807 respectively.

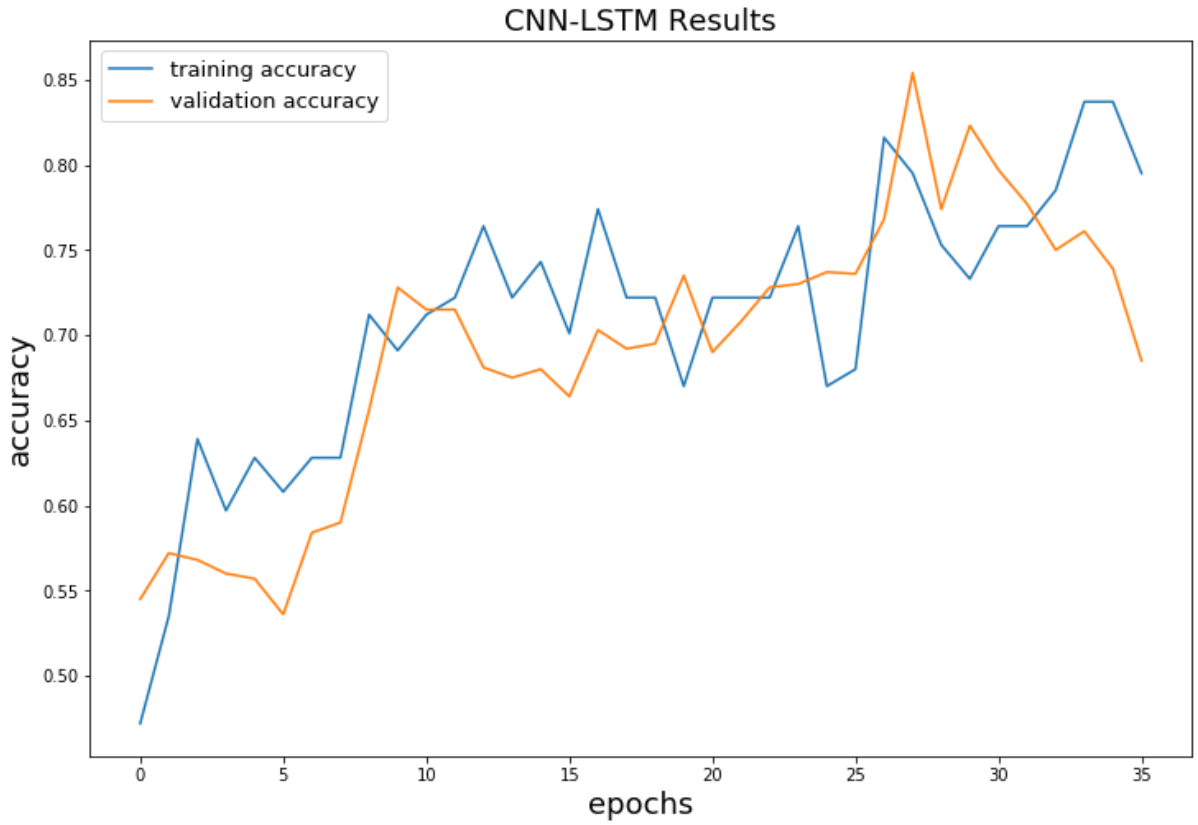

Figure 4-2: Performance of CNN-LSTM on the CyAds train Dataset

4.3.3 LR-CN

In this model, we resized the images to the resolution of 240 x 320 pixels and augmented them to the resolution of 227 x 227 pixels. We trained all the LR-CN networks with

video clips of 16 frames. The frames are sampled randomly out of all the frames extracted from the video. LR-CN was trained to predict the video's class at each time step. We averaged the label probabilities i.e. the outputs of the network's SoftMax layer, across all frames and chose the most probable label. The existing network was pre-trained on the 1.2M image ILSVRC-2012 [13] classification training subset of ImageNet dataset.

Figure 4-3 shows the training accuracy of LR-CN on the CyAds Dataset. The accuracy of the LR-CN model on the UCF-101 test dataset and CyAds test dataset are 0.829 and 0.698 respectively. The videos on the CyAds dataset are longer in length than the ones in UCF-101 dataset. Since, this model takes only 16 frames we see a huge difference in the accuracies in the two datasets.

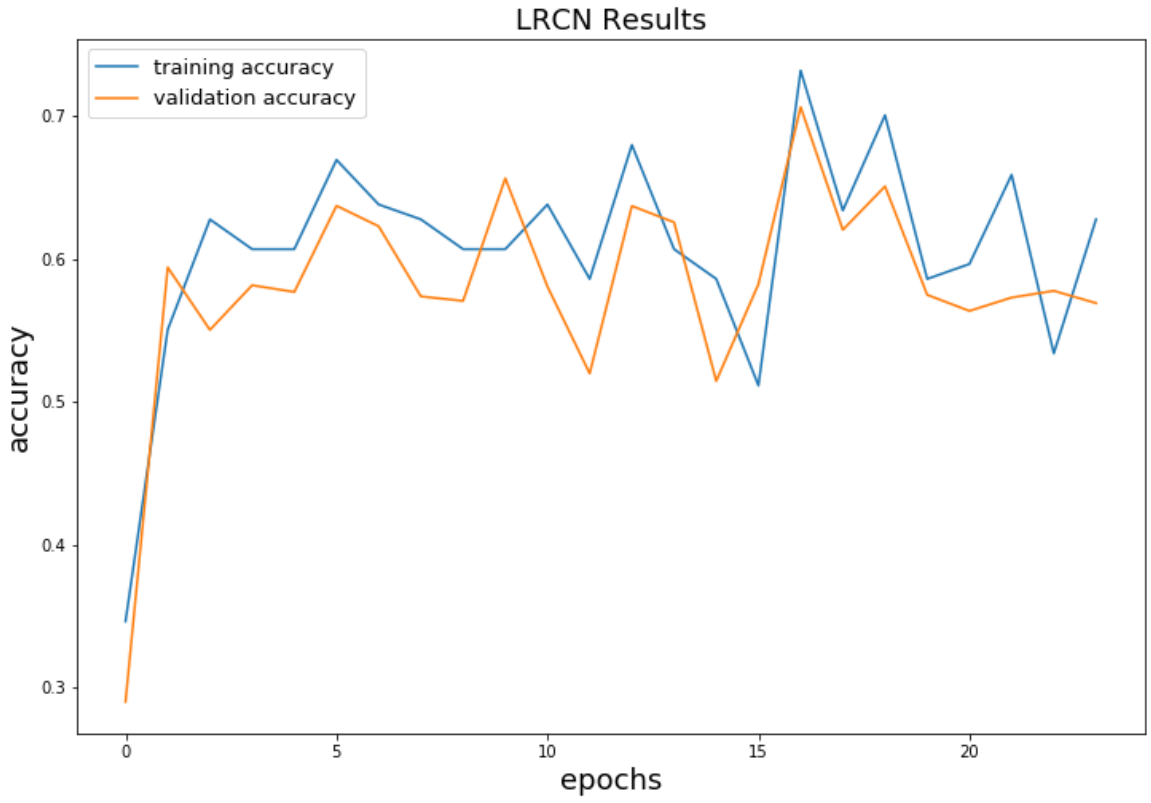

Figure 4-3: Performance of LR-CN on the CyAds train Dataset

4.4 Combined Model

When using Random Forest Model on TF features from speech-to-text transcript using P-Ads Video Classifier on CyAds test dataset, we achieved the accuracy of 0.78, precision of 0.77, recall of 0.64 and F1-score of 0.70 which was the best result from all the implementations. When we further added TF-IDF features on text from OCR and speech-to-text transcript using P-Ads Video Classifier on CyAds test dataset we got an accuracy of 0.863, precision of 0.858, recall of 0.862 and F1-score of 0.860 with Bernoulli Naïve Bayes, which was the best result from all the implementations.

CNN Fusion gave an accuracy of 0.787 on the CyAds test dataset whereas CNN-LSTM model gave an improved accuracy of 0.807. LR-CN model gave a lower accuracy of 0.698 compared to the previous ones.

CNN Fusion method takes all the frames during classification while CNN-LSTM and LR-CN do not take all the frames generated from the video. Thus, the accuracy in the UCF-101 dataset and the CyAds was the closest. Hence, we combined the results from the CNN Fusion and the P-Ads Video Classifier.

We get confidence scores for each video from the CNN Fusion and also from the P-Ads Video Classifier for each category, political and non-political. We first experimented by assigning equal weights to the confidence scores from both the CNN Fusion and the P-Ads Video Classifier. Then, we experimented with assigning different weights for different models. We tried with the weights of 30%, 40%, 60% and 70%, respectively, for the P-Ads Video Classifier with the remaining weight assigned to CNN Fusion.

We achieved an accuracy of 0.87 when equal weights were assigned to both the P-Ads Video Classifier and the CNN Fusion. We then experimented with assigning different weights to the P-Ads Video Classifier. The best results were when we assigned 40% weight to the

confidence score from P-Ads Video Classifier and 60% weight to the confidence score from CNN Fusion. We achieved an accuracy of 0.91, precision of 0.90, and recall of 0.93 and F1-score of 0.91.

Table 9: *Results of the combination of the best P-Ads Video Classifier using TF-IDF and CNN Fusion on the test dataset of the CyAds dataset*

Algorithm	Accuracy	Precision	Recall	F1-Score
P-Ads Video Classifier + CNN Fusion (30%/70%)	0.90	0.86	0.95	0.91
P-Ads Video Classifier + CNN Fusion (40%/60%)	0.91	0.90	0.93	0.91
P-Ads Video Classifier + CNN Fusion (50%/50%)	0.87	0.89	0.84	0.87
P-Ads Video Classifier + CNN Fusion (60%/40%)	0.86	0.89	0.85	0.86
P-Ads Video Classifier + CNN Fusion (70%/30%)	0.86	0.88	0.83	0.85

Out of the 743 videos in the CyAds test Dataset, 676 videos were classified correctly and 67 videos were misclassified using the best combined method. Out of the 371 political videos in the CyAds test Dataset, 37 were misclassified as non-political. Similarly, out of the 372 non-political videos, 35 were misclassified as political. Out of the correctly classified videos, 85% of the videos were less than 30 seconds long and out of the incorrectly classified videos, 79% of the videos were longer than 30 seconds.

The political videos that were classified as non-political generally had similar properties to a non-political video. For e.g. political videos with huge mass of people, where people interacting with each other, where vehicles were present, where there were many texts in the video were classified as a non-political one.

Similarly, non-political videos where people were addressing an audience, where people were talking to each other, where insurance advertisers were present and talking were classified as a political video.

CHAPTER 5. CONCLUSION

We investigated various approaches for classification of political and non-political videos. We started with implementation of feature based approach, P-Ads Video Classifier to see the results. This approach does not take the video information into account. We implemented various deep learning video classification methods to use the features of a video into account. We found that the deep learning techniques yielded better results than feature based techniques.

We implemented P-Ads Video Classifier and three deep learning models. The results are on the CyAds test dataset. We found that just using OCR from the last 3 frames of a video in P-Ads Video Classifier gave us an accuracy of 78.7% while using both OCR from frames and speech-to-text transcript yielded an accuracy of 86.2%. Thus, we choose P-Ads Video classifier which uses both OCR from frames and speech-to-text transcript.

CNN Fusion gave us an accuracy of 78.7%. CNN-LSTM gave an accuracy of 80.7% and LR-CN gave an accuracy of 69.8%. The combination model combined results from P-Ads Video Classifier and CNN-Fusion, which gave an accuracy of 90.1%.

REFERENCES

[1] YouTube, “YouTube.” [Online]. Available: https://www.youtube.com/.

[2] B. Banerjee, “Machine Learning Models for Political Video Advertisement Classification,” (2017). Creative Component. 1443. [Online]. Available: https://lib.dr.iastate.edu/creativecomponents/1443

[3] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, “Large-scale Video Classification with Convolutional Neural Networks,” In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1725–1732, 2014.

[4] J. Donahue, L. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, T. Darrell, “Long-Term Recurrent Convolutional Networks for Visual Recognition and Description,” *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2625-2634, 2015

[5] J. Y. H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, “Beyond Short Snippets: Deep Networks for Video Classification,” In *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 4694–4702, 2015.

[6] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset for 101 human actions classes from videos in the wild". C*oRR,* abs/1212.0402, 2012.

[7] W. Zhu, C. Toklu, and S. P. Liou, “Automatic news video segmentation and categorization based on closed-captioned text,” In *Proc. of IEEE Int. Conf. Multimed. Expo*, pp. 829–832, 2001.

[8] P. Wang, R. Cai, and S. Q. Yang, “A hybrid approach to news video classification with multi-modal features,” In *Proc. of ICICS-PCM 2003 Jt. Conf. 4th Int. Conf. Information, Commun. Signal Process. 4th Pacific-Rim Conf. Multimed.*, vol. 2, no. December, pp. 787–791, 2003.

[9] D. Brezeale and D. J. Cook, “Automatic Video Classification: A Survey of the Literature,” *IEEE Trans. Syst. Man, Cybern. Part C (Applications Rev.*, vol. 38, no. 3, pp. 416–430, May 2008.

[10] Q. Wei, L. Gu, H. Jiang, X.-R. Chen, and Z. Hong-Jiang, “Integrating Visual , Audio and Text Analysis for News Video,” In *Proc. of* International Conference on Image Processing, pp. 520-523, 2000.

[11] R. S. Jasinschi and J. Louie, “Automatic TV program genre classification based on audio patterns,” *EUROMICRO,* pp. 370–375, 2001.

[12] G. Hinton, S. Osindero, and Y.-W. Teh, "Communicated by Yann Le Cun A Fast Learning Algorithm for Deep Belief Nets 500 units 500 units," *Neural Comput.*, vol. 18, pp. 1527–1554, 2006.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," *IEEE Conference Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks." *NIPS, pp. 1097-1105,* 2012.

[15] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," *NIPS*, pp. 568–576, Sep. 2014.

[16] C. Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke and Andrew Rabinovich, "Going Deeper with Convolutions," *IEEE Conference Computer Vision and Pattern Recognition,* pp 1-9, 2015.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *IEEE Conference Computer Vision and Pattern Recognition (CVPR),* pp. 770–778, 2016.

[18] S. Zha, F. Luisier, and W. Andrews, "Exploiting Image-trained CNN Architectures for Unconstrained Video Classification." *CORR,* pp 56-65, 2015.

[19] S. Ji, W. Xu, M. Yang, and K. Yu, "3D Convolutional neural networks for human action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–231, 2013.

[20] T. Ge, K. He, Q. Ke, and J. Sun, "Optimized Product Quantization for Approximate Nearest Neighbor Search," *IEEE Conference Computer Vision and Pattern Recognition (CVPR)* pp. 2329–2336, 2013.

[21] H. Ye, Z. Wu, R.-W. Zhao, X. Wang, Y.-G. Jiang, and X. Xue, "Evaluating Two-Stream CNN for Video Classification," In *Proc. of the 5th ACM International Conference on Multimedia Retrieval,* pp. 435-442, 2015.

[22] Z. Wu, X. Wang, Y.-G. Jiang, H. Ye, and X. Xue, "Modeling Spatial-Temporal Clues in a Hybrid Deep Learning Framework for Video Classification," In *Proc. of the 23rd ACM International conference on Multimedia,* pp. 461-470, 2015.

[23] C. Feichtenhofer, A. Pinz and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 1933–1941, 2016.

[24] S. Sharma, R. Kiros, and R. Salakhutdinov, “Action Recognition using Visual Attention,” *CoRR,* pp. 1–11, 2015.

[25] Z. Li, K. Gavrilyuk, E. Gavves, M. Jain, and C. G. M. Snoek, “VideoLSTM convolves, attends and flows for action recognition,” *Computer Vision and Image Understanding*, vol. 166, no. October 2017, pp. 41–50, 2018.

[26] D. Tran, L. D. Bourdev, R. Fergus and M. Paluri, “C3D: Generic features for video analysis.,” *CoRR*, pp. 36-48, 2015.

[27] S. Sabour, N. Frosst, and G. E. Hinton, “Dynamic Routing Between Capsules,”. In *the Annual Conference on Neural Information Processing Systems (Nips)*, 2017.

[28] “FFMPEG Project.” [Online]. Available: https://ffmpeg.org.

[29] “Extractor.” [Online]. Available: https://docs.scipy.org/doc/numpy/reference/generated/numpy.extract.html

[30] NumPy, “NumPy.” [Online]. Available: https://numpy.org/.

[31] Google, “Google API.” [Online]. Available: https://developers.google.com/apis-explorer.

[32] NLTK, “NLTK, Natural Language Processing Toolkit.” [Online]. Available: https://nltk.org.

[33] G. Salton and C. Buckley, “Term-weighting Approaches in Automatic Text Retrieval", In *Information Processing & Management, 24(5): 513-523,* 1988.

[34] A. Graves, A. R. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” In *Proc. of (ICASSP), IEEE*, no. 6, pp. 6645–6649, 2013.

[35] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Grishick, S. Guaderrama, T. Darrell, “Caffe: Convolutional Architecture for Fast Feature Embedding,” In *Proc. of the 22nd ACM International Conference on Multimedia, ACM*, pp. 675–678, 2014.

[36] Stanford, “PCL [Online].” [Online]. Available: http://pcl.stanford.edu/.

[37] C. D. Manning, P. Raghavan and H. Schutze, “Introduction to Information Retrieval.” Cambridge University, New York, NY, USA, pp. 155, 2008 [Online]. Available: https://nlp.stanford.com/IR-book/irbookonlinereading.pdf.

[38] C. D. Manning, P. Raghavan and H. Schutze, “Introduction to Information Retrieval.” Cambridge University, New York, NY, USA, pp. 155, 2008 [Online]. Available: https://nlp.stanford.com/IR-book/irbookonlinereading.pdf.

[39] Scikit-Learn, “Scikit-Learn Python Library for Machine Learning.” [Online]. Available: https://scikit-learn.org/.

[40] Scikit-Learn, “GridSearch,” *scikit-learn v0.21.3*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

[41] S. Shaley-Shwartz, S. Ben-David, "Stochastic Gradient Descent" in "*Understanding Machine Learning: From Theory to Algorithms."* Cambridge: Cambridge University Press, pp. 150-166, 2014.”

[42] M. D. Zeiler and R. Fergus, “Visualizing and Understanding Convolutional Networks,” In *ECCV,* pp. 818–833, 2014.

APPENDIX HYPERPARAMETERS

P-Ads Video Classifier using TF extracted from text from speech

Stochastic Gradient Descent: loss='hinge', penalty='l2', alpha=0.001

Support Vector Classification: kernel='linear', C=1, probability=True

Bernoulli Naïve Bayes: alpha=1.0, binarize=0.0

Random Forest: n_estimators=100, random_state=0, max_depth =2

Ada Boost: n_features=4, n_samples=100, learning_rate=1.0, random_state=0

Gradient Boost: loss='deviance', learning_rate=0.1, n_estimators=100

P-Ads Video Classifier using TF-IDF extracted from text from speech and OCR

Stochastic Gradient Descent: loss='hinge', penalty='l2', alpha=0.01

Support Vector Classification: kernel='linear', C=1, probability=True

Bernoulli Naïve Bayes: alpha=1.0, binarize=0.0

Random Forest: n_estimators=10, random_state=0, max_depth =2

Ada Boost: n_features=4, n_samples=100, learning_rate=0.1, random_state=0

Gradient Boost: loss='deviance', learning_rate=0.01, n_estimators=100

CNN Fusion

Number of Epochs: 100 (Early Stopping, patience = 10)

Dropout Ratio: 0.5

Losses: binary cross entropy

Kernel regularizer: 0.001

Batch size: 32

CNN-LSTM

Number of Epochs: 100 (Early Stopping Enabled)

Learning Rate: 10^{-5}

Decay Rate: 10^{-6}

Dropout Ratio: 0.4

Losses: binary cross entropy

Kernel regularizer: 0.0001

Batch size: 32

LR-CN

Number of Epochs: 100 (Early Stopping Enabled)

Learning Rate: 10^{-5}

Decay Rate: 10^{-5}

Dropout Ratio: 0.5

Losses: binary cross entropy

Kernel regularizer: 0.001

Batch size: 32