VIDEO MANIPULATION DETECTION VIA RECURRENT RESIDUAL FEATURE LEARNING NETWORKS

Matthew J. Howard
Computer Science and Engineering
University of California, Santa Cruz
matthoward@ucsc.edu

Alexander S. Williamson
Computer Science and Engineering
University of California, Santa Cruz
alswilli@ucsc.edu

June 3rd, 2019

ABSTRACT

Over the past decade, the increased adoption of Internet-connected technology has led to a substantial growth in the amount of videos produced and digested by people around the world. Subsequently, manipulated content within videos has become far more common and difficult to detect. Material of this nature poses a significant problem as it provides a way to falsely affect a viewer's beliefs, potentially with serious repercussions. In this thesis, we propose a Deep Learning (DL) architecture that leverages techniques from Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) to detect a variety of video manipulations on a frame-by-frame basis. More specifically, we develop a model which analyzes manipulated video segments represented as sequences of images via a joint Residual Network (ResNet) feature extractor and Long Short-Term Memory (LSTM) network to classify video frames that exhibit signs of manipulation and determine which type of manipulation was applied. We train our model on a modified subset of the UCF101 Action Recognition Video Data Set [1] which we alter with a set of four manipulation types: object insertion, image compression, frame black out, and frame blurring. We evaluate the classification accuracy of our model by creating manipulation "blocks", or sets of consecutive manipulated video frames, and varying the length of blocks, distance between blocks, and distributions of manipulations within blocks. Experimental results demonstrate that our model achieves high ($>$90%) classification accuracy in the presence of increased frame class transitions and when non-manipulated frames hold a majority within the sequence; however, despite outperforming a comparative RNN baseline, our model also demonstrates room for improvement when manipulated frames constitute a majority of the video sequence and when frame class transitions are sparse.

1 Introduction

As Internet-connected technology becomes more embedded into the lives of millions, it increases the amount of digital media that is digested and disseminated around the world. In particular, the enormous growth of video-sharing and streaming services suggests that an increasing number of individuals gather information from digital video content. Unfortunately, the immense growth and popularity of video media has created an environment well-situated for exploitation. As with the rise in "fake news", the prospect of "fake" or manipulated videos presents an enormous challenge to preventing misleading and dishonest information from spreading. Further, the implications of manipulated videos ranges from friendly alterations to major modifications that change the nature of the original video. In particular, manipulations may be used to hide information (e.g., obscuring objects or persons), while others may be used to alter the way information is presented (e.g., replacing objects or persons). Additionally, modern manipulation techniques allow for increasingly complex alterations that are far more difficult to detect with the human eye. These techniques can also be applied in less time than ever before, creating concern over the reputability of online videos covering important issues on a national or global scale. Considering these characteristics, methods for the detection of video manipulations have become quite desirable; however, work in this domain has thus far been limited. In this work, we seek to contribute to the advancement of these methods and present a statistical deep learning model for the detection of common video manipulations.

In the following sections, we first describe the goals of our thesis as well as related work in the space of manipulation detection. Then, we discuss our proposed approach, our models, and a modified video dataset that we construct to train our model for the task of detecting video manipulations. Finally, we demonstrate the performance of our model in classifying different types of manipulations within video segments across several experiments, highlighting advantages and disadvantages of our proposed model and suggesting several avenues to extend the current work with future research.

1.1 Background

A growing body of research investigating the detection of manipulations within images and videos serves as motivation for the work presented in this thesis. Research that has been focusing on detection of manipulations in images largely focuses on subtle changes to lighting, contrast, and objects in the background of images, while much of the related research for videos specifically focuses on detecting the forgery of human faces, or so called 'deep fakes'. Deep fakes are commonly associated with fake-news, and more work has recently been conducted in hope of being able to detect them in light of the looming 2020 U.S. elections. Several comedic deep fakes of former President Barack Obama have been created and posted online that are shockingly realistic, making people concerned that videos like this could be released rapidly during the upcoming elections that could sway votes in a desired direction maliciously [22]. This concern has even prompted the creation of startups like Deeptrace which aim to create methods to combat these types of manipulations [23]. Below, we highlight two separate related papers involving the detection of manipulations in images and facial manipulations in videos, respectively, which we believe most proficiently represent the body of research being conducted in the two areas.

The first related work is a paper produced by Adobe Research that focuses on detecting various manipulations in images [4]. The three types of manipulations the paper examines are 'splicing' (object insertion), 'copy-move' (object duplication), and 'removal' (object removal), all of which are very common types of image manipulations and are the most similar in complexity to the types of manipulations we cover in our thesis. In order to detect these manipulations, the authors utilize a Faster R-CNN [12] neural network (NN) that looks at both the standard RGB images and their corresponding noise stream images when generating the final class predictions. The first stream is an RGB stream, which uses ResNet [15] to learn image features that are more present in the RGB stream like edges and high contrast areas and feeds these features into a Region Proposal Network (RPN). The RPN uses these features to create Regions of Interest (RoI) which the model analyzes individually to create more accurate final bounding box and class predictions. The second stream is a Noise Stream, which is helpful at identifying splicing noise that would otherwise be unnoticed in RGB, uses steganalysis rich model (SRM) to produce the noise features from the RGB images. As with the RGB stream, RoIs are also created but only classification predictions are outputted. Bilinear pooling finally combines the detected features from both streams for each image for producing the final classification predictions. The purpose of using this type of two-stream network, region-based network is to further increase the potential of being able to detect visual inconsistencies such as tampered contrast and boundaries of object contained within images. The paper uses many different datasets to train and test the model's performance, which most notably include the NIST16 [13] and CASIA [14] datasets of manipulated images which have been masked with various post processing techniques to further test robustness. The NIST16 dataset contains several different tampering techniques applied to images that are post-processed further to hide transformations. The CASIA dataset contains for of the same kinds of tampering to images but with less post-processing and manipulation types. Ground truth masks are also obtained from each dataset to be used during evaluation. In total this creates about 6000 modified images. Due to this smaller size, the model is pre-trained on a modified COCO dataset [16] of 42,000 original and modified[1] image pairs. COCO is a very common and large dataset used for training model to classify object in images. The final model is compared against a couple baseline models which resemble simplified and slightly augmented versions of the final model. The authors also run additional tests with augmentations to account for resizing and compression of images. The final evaluation metrics being used are pixel-level "Area Under the receiver operating characteristic Curve" (AUC) and F_1 score. For the NIST dataset, the final model achieves the highest scores across all baseline models with and AUC of 0.937 and an F_1 score of 0.722. Results when augmentations are applied achieved similar values for both metrics.

The second related work is a paper from Purdue University that more closely mirrors the problem structure of our thesis, since it focuses on detecting manipulations in videos using a combination of a CNN feeding frame features into an RNN to develop temporal dependencies before producing final class predictions. More specifically, though, it focuses on detection of deep fakes rather than detection of simpler manipulations highlighted in the previous related work and in our thesis [2]. Using a dataset of 600 deep fake videos comprised of 300 selected videos from online video sources and another 300 from the HOHA dataset [11], the authors input a fixed length sequence of frames from each video into their model. The HOHA dataset is comprised of realistic actions of humans in films, news segments,

[1] Objects from some images were copied and pasted into other images

and other popular sources, and is used here to help the model recognize deep fake features better since the content from the HOHA dataset contains videos types that are often used for deep fakes (i.e., contain actors or important people giving monologues). The model is comprised of a special type of layer called an Convolutional LSTM, which is divided up into a CNN and a LSTM. The CNN used here is the state-of-the-art InceptionV3 [9] architecture with pre-trained ImageNet [10] weights. This architecture allows for high quality feature extraction from the video frames, with the final 2048-dimensional feature vector output being fed directly into the LSTM as input. The LSTM takes the concatenated features from the CNN and analyzes them for any temporal dependencies. A fully connected layer followed by a dropout regularization layer of 50% is also applied before being fed into a softmax layer for final classification prediction. Additional preprocessing also helps aid the feature detection of the layers, with subtraction of the mean channel being applied, each frame being resized to 299x299 pixels, and setting the Adam optimizer with a learning rate of 1e-5 and decay of 1e-6. Once these preprocessing steps are applied, the entire Convoultional LSTM model when evaluated with size 20, 40, and 80 frame sequences achieves a validation accuracies of 96.9%, 97.1%, and 97.2%, respectively. the validation data represents 15% of the initial training data being sampled.

The above related work represents characteristics of our work very well. With the Adobe work, we adopt the more simple yet still important choices for manipulations to be applied to frames in images, while also adding a few of our own. With the Purdue work, we adopt a similar approach in how we detect our manipulations via dividing video samples into short, fixed sequences of frames and then feeding this input through a CNN for feature extraction and an LSTM for more temporal feature detection and concatenation. These adoptions help create a final proposed model and framework pipeline that is unique, while still utilizing cutting edge techniques in the area of research to reach our goals and examine and important topic.

1.2 Existing Algorithms and Models

Much of the basic design behind the models used in the work related to our thesis stems from the use of CNNs and LSTMs. CNNs provide a way to analyze images in a way that highlights various unique aspects, such as edges, similar looking objects and lighting, using specific filter matrices applied to hidden layers of neurons. These filters allow for distinct characteristics of images to be identified and then classified. CNNs are often comprised of many convolutional, pooling and fully-connected layers, and the filters simplify the representation of the output at each layer the further into the network the input gets. In comparison to other image feature generalization techniques, such as standard neural networks, this allows for more efficient detection of crucial image features. LSTMs build off of RNNs to allow for better use of past information when deciding how to interpret and make connections to input passed in the network in the present. This effect is achieved by preserving the error that is backpropagated through an RNN's layers using several different types of memory storage cells. These cells, referred to often as "gates", are activated via the use of sigmoid functions and have their own set of weights that are updated with each time step when determining whether to release or store information. In comparison to Gated Recurrent Units (GRUs), LSTMs contain one more gate and have been shown to be computationally less efficient. However, the output of LSTMs may be better suited for different types of input, sometimes providing better results, so they are still utilized in most scenarios.

Long-term Recurrent Convolutional Networks (LRCN) [6] are a class of NN models that seek to unify image classification networks with sequence learning to enable tasks that map sequences of images to sequences of outputs. More specifically, LRCNs are an adapatable framework consisting of convolutional feature learning networks that supply RNNs (typically LSTMs) with convolutional feature sequences in which convolutional and recurrent parameters are learned jointly for the task. In particular, the flexible LRCN framework enables high-performing model learning across a variety of image sequence tasks when fitted with task-appropriate convolutional and recurrent layers. For example, Donahue et. al [6] experiment with various combinations of CNNs and RNNs, including CaffeNet [24] and VGGNet [25] for the convolutional layers, and LSTMs and "vanilla" RNNs for the recurrent layers, applying their models to tasks of image captioning, video description, and activity recognition. On UCF101, LRCN models achieve 82% classification accuracy for the action recognition task.

In addition to LRCN, many other models exist in the space of sequence learning from sequential image inputs. One such model, two-stream CNNs [19] address the problem of activity recognition within videos by decomposing a convolutional architecture into spatial and temporal networks. In the spatial network, action recognition is performed on still images, while the temporal network constructs optical flows between frames to highlight intensity of motion between pixels in consecutive frames. The two-stream composition allows for transfer learning of pre-trained image classification models for the spatial stream, but relies on raw video data to train the temporal optical flow network. Overall, the two-stream model performs exceptionally well utilizing pre-trained image classification networks, achieving 87% classification accuracy on the UCF101 action recognition dataset.

1.3 Thesis Goals and Approach

The main goal of our thesis is to successfully detect a small set of video frame manipulations (object insertion, image compression, frame black out, and frame blurring) for frames in a given video and determine where in the video they occur, if at all. A secondary goal of our thesis is to create a framework that allows for future work in the area of video manipulation detection. In addition, our framework should allow for any dataset of videos to be selected and augmented with frame-level manipulations, and which can be extended to incorporate more realistic and complex manipulation scenarios. Finally, irrespective of our manipulation framework, we aim to study how patterns of manipulations affect model learning ability, in particular studying how our deep learning models perform against a wide array of potential manipulation patterns.

The methods we propose in this thesis to detect video manipulations on a frame-to-frame basis share many characteristics with the final model(s) described in the related works above. We utilize the well-known UCF101 [1] video dataset for action recognition and create a modified dataset of approximately 1000 videos for training our model. Specifically, we apply several different manipulations to the individual video frames in various patterns for each of the videos for the purpose of model development and experimentation. Additionally, we construct our video manipulation classification model from a combination of convolutional and pooling layers via the state-of-the-art model ResNet50, similar to the methods utilized by related work. However, our work extends this design by adding a LSTM network which takes as input the output of the ResNet model to learn temporal relationships that further help classify video manipulations. Our results show that this extension proves to be very helpful with improving the classification results of our model when the sequences of frame manipulations in training input become increasingly more complex. Thus, our contribution is two-fold; we provide a framework that generates various manipulated sets of video data from an unmodified dataset of videos and we construct a prototype model that is able to detect the manipulations in these generated videos with high accuracy, showing that it is capable of being generalized to detect manipulations in real-world video data.

The following sections are structured as follows: Section 2 focuses on our dataset, going into details regarding the preprocessing and manipulations that are applied to it; Section 3 covers the context in which we build our model, as well as the details regarding our final model's architecture and the other architectures we explored before it; Section 4 dives into the Experiments we used to test the accuracy and robustness of our model, covering both how they were conducted as well as an analysis of the results we achieved. Section 5 describes the contributions from each member who worked on the thesis; and, finally, Section 6 outlines future work that could help improve upon the work conducted here and Section 7 concludes and summarizes our thesis as whole.

2 Dataset

In this thesis, we develop a preprocessing framework for automatically generating manipulated videos from any provided video dataset in parallel with our proposed manipulation detection model. For this work, we apply our manipulation framework on the UCF101 action recognition dataset for model development, training, and experimentation. In the following sections, we first describe the UCF101 dataset and then describe our preprocessing framework, including details on how we apply specific manipulations to the videos.

2.1 UCF101

UCF101 is a dataset comprised of 13,320 videos sourced from YouTube depicting 101 types of action categories. We utilize the UCF101 dataset for model development and experimentation for several reasons, specifically the videos offer a high degree of variety in camera movement, lighting conditions, and background context, while also capturing realistic and dynamic human scenarios as opposed to staged or static actions.

2.2 Preprocessing

Several preprocessing steps are taken to ensure that our data is consistent and manageable. During preprocessing, we assume that we are provided a fixed selection of videos to choose from, and apply the following steps:

1. *Configuration*. We set parameters to determine the number of video segments, video segment length, and manipulation patterns for blocks of manipulations including block length, block spacing, and manipulation distribution across blocks. Parameter configurations are designed to enable experimentation and testing across many manipulation patterns.

2. *Video Parsing*. Videos are parsed into sequences of images (frames) via OpenCV[2], with each video broken into segments according to the configuration of video segment length. Each frame is scaled to a fixed size of 224x224x3 pixels, and each pixel value is normalized from 0-255 to 0-1.
3. *Manipulation Assignment*. Each parsed frame within each video segment is assigned a manipulation class from a set of available manipulations according to the configuration of manipulation patterns. For example, frames 1-15 may be assigned a *normal*, or non-manipulated class, while frames 16-20 are assigned a *black* manipulation class.
4. *Manipulation*. Once each video segment has been assigned manipulation classes, manipulations are applied individually to each frame according to the assignment. Manipulations are handled by applying individual image transformations on frames according to their class assignment.
5. *Data Preparation*. Each manipulated video segment is stored individually as a sequence of pixel-arrays alongside corresponding class assignment labels which we store as one-hot encoded label vectors. Pairs of pixel-array sequences and label sequences are then split into training and validation sets using an 80%/20% separation. Each pair of pixel-array sequence and label sequence is then stored on disk and available during training and evaluation with flexibility for batching as necessary.

2.3 Manipulation Details

We construct four unique video manipulations to apply during data generation in Figure 1:

- The first manipulation, *black*, converts a video frame into all black pixels. This manipulation represents a simple, yet common scenario in which an entire frame is occluded from view. We implement this manipulation by zeroing out all pixel values within a frame.
- The second manipulation, *compressed*, applies a random amount of compression to a frame. This manipulation represents the scenario where a video loses quality, either intentionally, or by other means, such as a loss in network bandwidth. We implement this manipulation by applying lossy JPEG compression via PIL[3], with quality randomly selected between 1-10%. A range of low quality values are selected to simulate a variety of compression scenarios that noticeably affect the original video segment.
- The third manipulation, *insert*, inserts an image of an object with a transparent background over a given frame. This manipulation represents the scenario where an object that was not originally present is added into the scene depicted by the video. We implement this manipulation by pasting via PIL a single, transparent object PNG on top of a given frame, selecting randomly from a pool of 20 total PNGs[4]. In order to generalize the scenario further, each PNG is inserted at a random location at a random size (between 30% - 50% of the frame size).
- The final manipulation, *blurred*, applies a Gaussian blur to a random rectangular region within a given frame. This manipulation represents the scenario where a select region of a video segment is occluded from view. Additionally, we identify this manipulation to be different from *black* and *compressed* at a functional level due to the specificity of the region that is manipulated (as opposed to the whole frame) and the direct transformation of specific pixel values. We implement this manipulation by selecting a random rectangular region with 20-50% width and height to that of the full frame, then apply a Gaussian blur via PIL to all pixels within that region.

3 Models and Algorithms

In this section, we first describe the problem setting for the task of detecting manipulations within videos and then describe our model exploration and development process. Finally, we describe in detail the proposed NN architecture of our video manipulation detection model.

3.1 Problem Setting

Video manipulation is the process of altering the content of a video in any manner that changes the visual content displayed by the video from its original state. In digital videos, displayed content consists of a series (or sequence)

[2]OpenCV Library: https://opencv.org/
[3]Python Imaging Library (PIL): https://pillow.readthedocs.io/en/stable/
[4]PNGs collected from FreePNGs: https://www.freepngs.com/

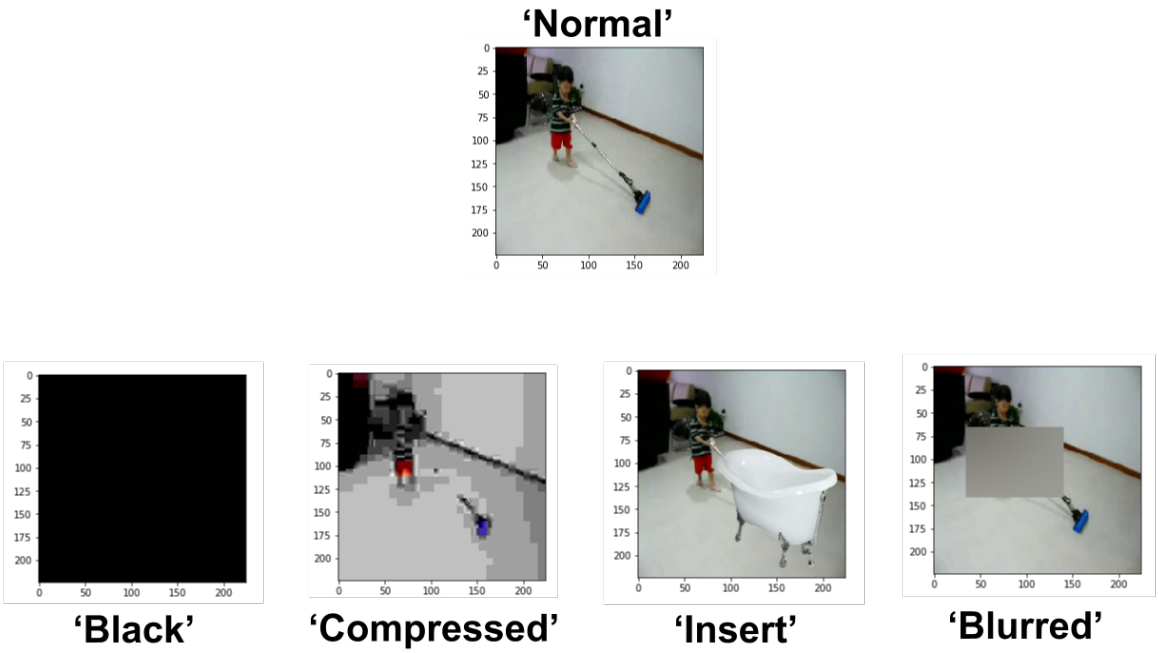

Figure 1: Manipulated frame examples for *black*, *compressed*, *insert*, and *blurred* manipulations.

of images (frames) displayed in rapid succession. Consequently, video manipulation involves any alteration to visual content within any frame of the video.

Several applications exist for classifying sequences of images, including methods for action recognition; however, detecting manipulations within videos presents several distinct challenges from these applications. First, manipulations are often applied for varying lengths of time from the full length of the video down to a single frame. For example, an object could be inserted into the background of a video for an extended period of time, or a person's face could be blurred out for a single frame that they appear in. Beyond the unbounded length of potential manipulations, an additional challenge presents when combinations of manipulations are applied to the same video. Manipulations are often applied non-uniformly across a video, as can be imagined in the case where a person's face is blurred out whenever they are on screen and another several frames are blacked out to obscure an activity occurring within the video. Lastly, in contrast to action recognition and similar tasks that typically involve human individuals accompanied by a set of objects that are utilized for specific actions (e.g., baseball mitts for the action of playing baseball), video manipulations may be present in all imaginable contexts. Considering these challenges, we seek to build a prototype model that is both robust to the unpredictability in which manipulations might be applied and able to decipher which type of manipulation is applied when one is detected.

3.2 Model Exploration

In developing a robust statistical learning model for detecting and classifying video manipulations, we first turn to successful state-of-the-art models in video activity recognition.

We consider LRCN [6] as a starting point for our proposed model implementation. The LRCN model is comprised of convolutional feature extraction layers that feed into an LSTM network and recent adaptations of this model have been useful at both extracting visual features from frames of a video and developing predictions based on these features in a sequence of frames. For LRCN networks, the convolutional feature extraction layers are typically in the architecture described by VGGNet [7] or AlexNet [17]. Both of these CNNs are highly-renowned for their performance on a variety of image recognition tasks; however, more recent advances such as ResNet demonstrate stronger classification power than the former architectures.

Importantly, the LRCN model allows for flexibility in the the feature extraction layers, with the ability to swap in any desired CNN without compromising the integrity of the overall architecture. This flexibility allows for alterations to and tuning of the overall architecture for the specific task of detecting video manipulations. In contrast, other

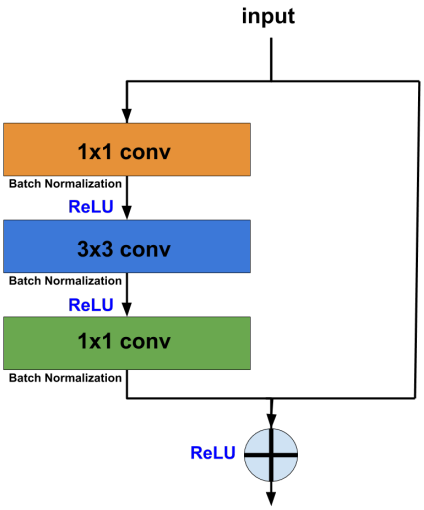

Figure 2: Residual network block for ResNet-50 [15]. Residual blocks are repeated in succession to extract features from the input.

Table 1: Proposed RNN architecture layers.

Layer	Size (Units/%)	Output Shape
Input (Frame Sequences)	$n_{samples}$	$n_{samples} \times 20 \times 224 \times 224 \times 3$
ResNet50	(see [15])	$n_{samples} \times 20 \times 7 \times 7 \times 2048$
GlobalMaxPooling	–	$n_{samples} \times 20 \times 2048$
Dropout	50%	$n_{samples} \times 20 \times 2048$
LSTM	30	$n_{samples} \times 20 \times 30$
Softmax Classifier	$n_{classes}$	$n_{samples} \times 20 \times n_{classes}$

sequential image models such as two-stream CNNs [19] are strongly designed for a specific task (action recognition), presenting difficulties in maintaining architectural integrity when transferring tasks.

3.3 Proposed Network Architecture

In designing our model, we consider the classification power of our feature extraction layers to play a pivotal role in the outcome of our predictions. Additionally, we also seek flexibility in our recurrent layers to adjust to the wide variety of video manipulation arrangements that we seek to capture.

With these considerations, we propose a modified version of the LRCN architecture that takes advantage of the state-of-the-art image classification model ResNet for feature extraction. More specifically, we first construct a feature extraction network that utilizes the architecture of ResNet-50. Then, we apply global max pooling and dropout regularization [18] on the extracted features to improve the generalizability of our model and reduce overfitting. Finally, we feed the regularized features into an LSTM network to decipher sequential feature dependencies before outputting a classification prediction. Figure 2 provides an overview of the repeating residual block architecture of ResNet-50 while Figure 3 and Table 1 summarize the components of our full proposed architecture. With respect to the sequential layer profiles of our model shown in Figure 3, the output shape of the previous layer is the input shape for the next. Additionally, the value of 20 in the shapes refers to the number of frames in the sequences being processed, while the starting 244x224x3 shape value in the input layer refers to the number of pixels. For all non-recurrent layers, we utilize a time-distributed approach that shares one set of non-recurrent network weights across all frames within a provided sequence, such that the model is updated with respect to the full sequence of frames during training.

The addition of regularization into our model is crucial for the setting of video manipulations since video manipulations can accompany any imaginable video content, and we desire a model that is context-agnostic. For example, a

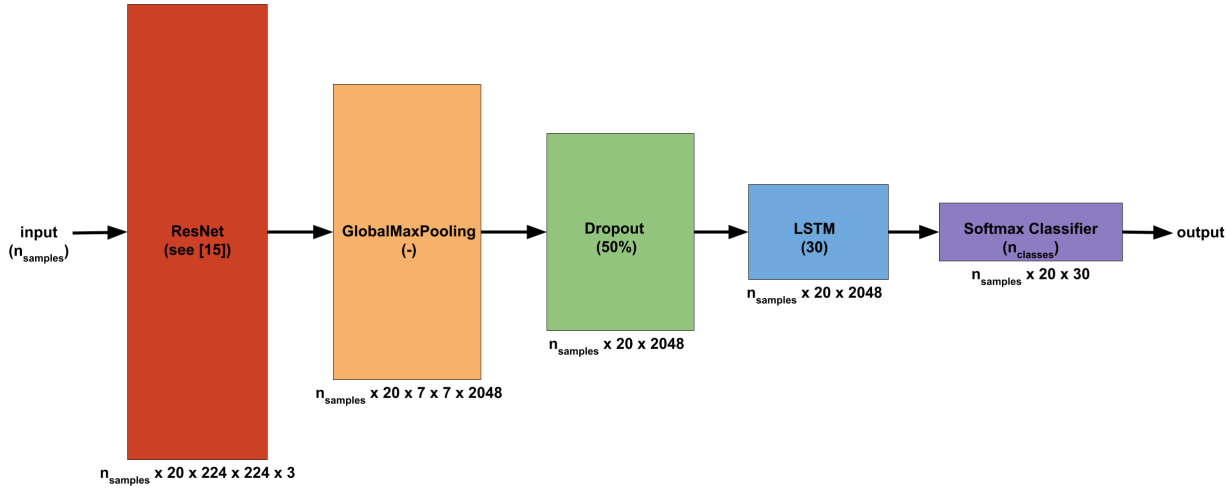

Figure 3: Proposed network architecture with corresponding layer input shapes listed below each layer block and sizes listed below each layer label.

video manipulation model that learns to detect manipulations exclusively on videos of animals may experience difficulty detecting manipulations on videos containing humans if it was overfit to the animal videos. Considering the combination of regularization with the powerful residual feature extraction networks that feeds into an LSTM layer, our proposed model provides a promising architecture for the task of video manipulation [21].

4 Experimental Results

In this section, we first outline several experiments conducted to test the predictive performance of our video manipulation detection model, and then discuss our findings.

4.1 Experiment Design

In considering the challenges for detecting manipulations within videos, we conduct five separate experiments that aim to study the impacts of manipulation uniformity, manipulation length, and quantity of manipulations on the temporal predictive classification accuracy of our proposed model.

We conduct five separate experiments to highlight the different strengths and weaknesses of our proposed model in terms of these characteristics. In each experiment, our goal is to measure the predictive classification accuracy of our model on all manipulated frames within a video. Further, in each experiment, we construct a unique arrangement of manipulation characteristics, as described in Figure 4 to examine how manipulation arrangement characteristics affect performance. In comparison with existing models, we analyze performance of our proposed model against an LRCN baseline for Experiments 1 and 2. The tested LRCN baseline consists of a VGG-16 convolutional feature network followed by 50% dropout regularization and a 256-unit LSTM network.

All experiments are executed on a single machine with an Nvidia 980Ti GPU with 6GB of VRAM, a 7th Gen Intel i7 CPU, and 16GB of system RAM. We build our NN models utilizing the Keras API [20]. For each experiment we randomly select 1000 20-frame video segments[5] to train our model from the UCF101 dataset and apply manipulation modifications as described in Section 2. From this modified set, we select 80% of the video segments for training our

[5]We limit ourselves to only 1000 samples per experiment due to time and computing resource constraints

model, hold out 20% for validation, and select another 200 unseen modified segments for the experimental testing of our model. In each experiment, we train our model for 50 epochs and evaluate on the model with weights associated with the highest validation accuracy obtained throughout the training epochs.

In each experiment, we employ k-folds cross validation with $k = 5$ to compute average predictive values for all evaluations to more adequately measure the generalized performance of our model.

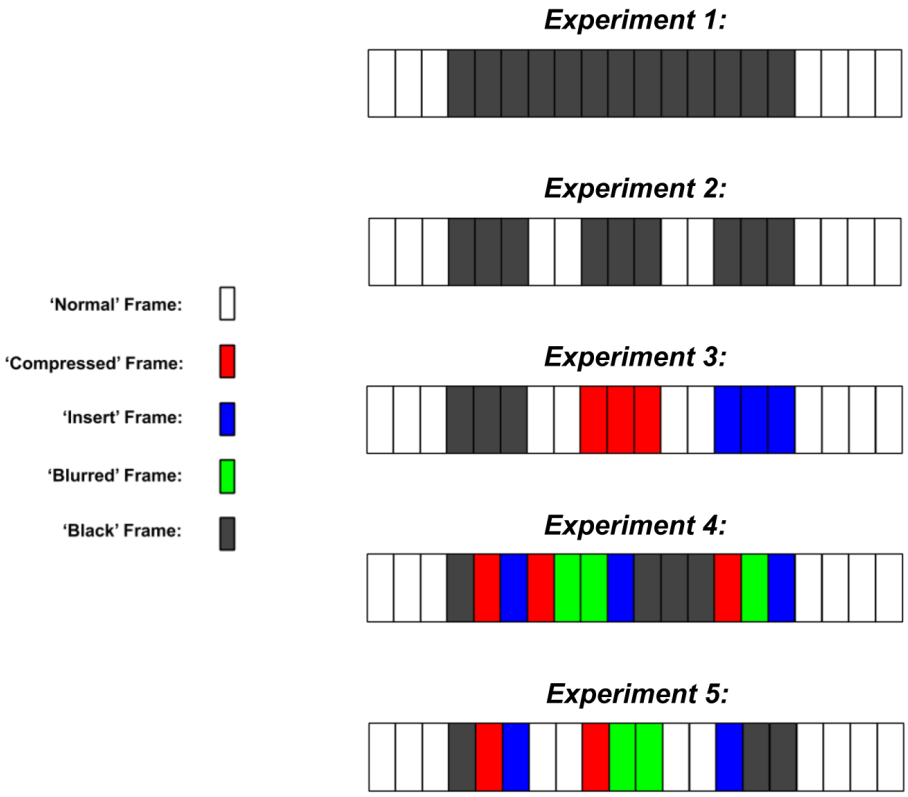

Figure 4: Example manipulation patterns for experimental configurations.

4.2 Experiment 1: Uniform Single-Block Single-Manipulation

The first experiment, *uniform single-block single-manipulation*, applies a single manipulation (selected at random) to a single block of consecutive frames within each provided sample segment. The start position of the manipulated block is selected at random for each segment. In conducting this experiment, we aim to study how our proposed model reacts to the length of a sequence of manipulated frames. We vary the length of our manipulation blocks from 5% (1 frame of 20) up to 65% (13 frames of 20), and measure both the individual accuracy for predicting each type of manipulation as well as the total accuracy across all manipulations. Results for experiment 1 when conducted using our proposed model are tabulated in Table 2, while the results for the same experiment using the LRCN baseline model are tabulated in Table 3.

From Table 2, we see a clearly observed decrease in video manipulation classification accuracy as the size of the manipulated block is increased, except in the case of the *compress* manipulation. The results for the *black* manipulation class emphatically demonstrate this trend, starting at 91.7% classification accuracy with a single frame block, dropping to 56.9% accuracy at 40% of the video length, and finally to a meager 19.4% accuracy at 65% of the video length. A potential explanation for this lies in the amount of non-manipulated context that is available in a video segment. That is, as a larger fraction of the video segment is manipulated, less non-manipulated content is available for the model to decipher between, leading to the model interpreting that the majority of what it sees is in fact the non-manipulated class. This is true especially in the case of the *black* manipulation, because the entire content of the original frame is modified, leaving behind no contextual artifacts from the original video segment. Further, we believe that the *compress* manipulation demonstrates a reverse trend to that of *black* for several reasons. First, compression does not

Table 2: Classification accuracies for Experiment 1 (*Uniform Single-Block Single-Manipulation*) – Proposed Model.

	Block Size			
Manipulation	1 (5%)	4 (20%)	8 (40%)	13 (65%)
Compressed	0.749	0.960	0.997	0.958
Insert	0.498	0.867	0.667	0.375
Blurred	0.333	0.845	0.517	0.517
Black	0.917	0.600	0.569	0.194
All Manipulated	0.651	0.852	0.731	0.504

Table 3: Classification accuracies for Experiment 1 (*Uniform Single-Block Single-Manipulation*) – LRCN.

	Block Size			
Manipulation	1 (5%)	4 (20%)	8 (40%)	13 (65%)
Compressed	0.000	0.000	0.000	0.159
Insert	0.111	0.000	0.000	0.279
Blurred	0.000	0.000	0.000	0.502
Black	1.000	1.000	1.000	0.942
All Manipulated	0.281	0.252	0.268	0.417

suppress original video content nearly as strongly. Second, many of the videos we consider possess "quality artifacts" (pixelation, etc.), and thus our model is able to more adequately determine a *compress* manipulation from a quality artifact when the length of the *compress* manipulation is increased. In comparison, the tested LRCN baseline (Table 3) exhibits an entirely opposite performance pattern to our proposed model. Specifically, LRCN is able to capture the *black* manipulation with high degree of accuracy (94%) across all block sizes; however, it fails to correctly classify all other manipulations with any degree of success at any block size. In addition, LRCN tends to classify manipulations more accurately as block size increases to the size of the entire video segment, exhibiting a reverse trend to that of our proposed model. Overall, our results exhibit a significant performance increase over the LRCN baseline for a majority of the manipulations, but struggles to maintain that performance with an increase to the block size.

4.3 Experiment 2: Uniform Multi-Block Single-Manipulation

The second experiment, *uniform multi-block single-manipulation*, applies a single manipulation (selected at random) to multiple fixed-length blocks within each provided video segment. The start position of the first manipulated block is selected at random for each segment and each block is assumed to be separated from another block by a fixed amount of spacing. In this experiment, we seek to study how the fragmentation of a single manipulation applied across a video segment affects our model differently than in the single-block setting. We consider configurations on the number of blocks, the distance between blocks, and the size of blocks. We vary the number of blocks between 2 and 5, the size of blocks from 5% (1 frame) to 35% (7 frames), and the spacing between blocks from 5% (1 frame) to 75% (15 frames). Experimental results for our proposed model are displayed in Table 4, while results for the LRCN baseline model are displayed in Table 5.

From Table 4, we again note a decrease in video manipulation classification accuracy as the size of blocks is increased, especially when block spacing is smaller. Since all blocks are applied the same manipulation, these trends align with our observations for Experiment 1. Interestingly, the results demonstrate that classification accuracy tends to increase as block spacing is increased, though this trend becomes weaker as the block sizes increase, especially when the total number of manipulated frames (block size $\times$ n_{blocks}) exceeds 50%. In alignment with our discussion for Experiment 1, increasing block spacing introduces more original video context and provides a larger transition buffer between non-manipulated and manipulated frames. In the best-performing case, our model achieves 90.6% classification accuracy for 3 blocks of size 1 (15% total manipulated frames) and 30% block spacing. In the worst-performing case, our model achieves 32.3% classification accuracy for 5 blocks of size 3 (75% total manipulated frames) and 5% block spacing. In general, as the video segment becomes less uniform (i.e., more diversity in classes between frames), the accuracy of our model improves. For the LRCN baseline (Table 5), highly different trends are observed, specifically with respect to the number of blocks where LRCN performs better as more blocks are introduced. Further, LRCN also demonstrates

Table 4: Total manipulation classification accuracies for Experiment 2 (*Uniform Multi-Block Single-Manipulation*) – Proposed Model.

	Block Spacing				
Block Size	1 (5%)	3 (15%)	6 (30%)	10 (50%)	15 (75%)
1 (5%)					
$n_{blocks} = 2$	0.823	0.800	0.768	0.842	0.810
$n_{blocks} = 3$	0.793	0.779	0.906	–	–
$n_{blocks} = 4$	0.836	0.877	–	–	–
$n_{blocks} = 5$	0.677	0.771	–	–	–
3 (15%)					
$n_{blocks} = 2$	0.815	0.789	0.824	0.709	–
$n_{blocks} = 3$	0.626	0.747	–	–	–
$n_{blocks} = 4$	0.598	–	–	–	–
$n_{blocks} = 5$	0.323	–	–	–	–
5 (25%)					
$n_{blocks} = 2$	0.681	0.645	0.633	0.653	–
$n_{blocks} = 3$	0.495	–	–	–	–
7 (35%)					
$n_{blocks} = 2$	0.525	0.577	0.562	–	–

Table 5: Total manipulation classification accuracies for Experiment 2 (*Uniform Multi-Block Single-Manipulation*) – LRCN.

	Block Spacing				
Block Size	1 (5%)	3 (15%)	6 (30%)	10 (50%)	15 (75%)
1 (5%)					
$n_{blocks} = 2$	0.211	0.158	0.193	0.175	0.281
$n_{blocks} = 3$	0.214	0.176	0.228	–	–
$n_{blocks} = 4$	0.214	0.229	–	–	–
$n_{blocks} = 5$	0.250	0.232	–	–	–
3 (15%)					
$n_{blocks} = 2$	0.449	0.231	0.278	0.286	–
$n_{blocks} = 3$	0.325	0.420	–	–	–
$n_{blocks} = 4$	0.503	–	–	–	–
$n_{blocks} = 5$	0.658	–	–	–	–
5 (25%)					
$n_{blocks} = 2$	0.468	0.555	0.302	0.277	–
$n_{blocks} = 3$	0.568	–	–	–	–
7 (35%)					
$n_{blocks} = 2$	0.584	0.599	0.508	–	–

improved classification accuracy with increased block spacing when block size is 5% and 15%, but this trend reverses for block sizes 25% and 35%. In total, the highest achieved accuracy for the LRCN baseline is 65.8% with 5 blocks of size 15% and block spacing of 5%. As in Experiment 1, this result indicates that LRCN starts to perform well when a large majority of the video segment is associated with a specific manipulation class. Overall, as seen in Experiment 1, our proposed model performs superiorly for most situations seen in this experiment in comparison to the LRCN baseline model.

4.4 Experiment 3: Uniform Multi-Block Multi-Manipulation

The third experiment, *uniform multi-block multi-manipulation*, applies a single random manipulation to each block. In contrast with experiment 2, we aim to study how several different manipulations in sequence affects the performance of our model compared to a single repeated manipulation. We vary the number of blocks from 2 to 4, the size of blocks from 5% to 35%, and the number of manipulations per video from 2 to 4. Again, the start position of the first manipulated block is selected at random (provided it meets the fixed experimental criteria). Results for experiment 3 are shown in Table 6.

Table 6: Total manipulation classification accuracies for Experiment 3 (*Uniform Multi-Block Multi-Manipulation*).

	Number of Manipulations/Blocks		
Block Size	2	3	4
1 (5%)	0.947	0.912	0.952
3 (15%)	0.944	0.969	0.962
5 (25%)	0.956	0.960	–
7 (35%)	0.946	–	–

From Table 6, we report general success in correctly classifying manipulated video frames, achieving above 91% classification accuracy in all experimental configurations. In addition, we notice a mild trend in improving classification accuracy as the number of unique manipulations is increased. Again, this trend can be attributed to the uniformity of the manipulated video segment, as more unique blocks allow our model to discern that the true, non-manipulated video content lies between segments of manipulations. For example, with block size 15%, we achieve 94.4% classification accuracy for 2 blocks of unique manipulations, while achieving 96.9% and 96.2% accuracy for 3 and 4 blocks of unique manipulations.

4.5 Experiment 4: Random Single-Block Multi-Manipulation

The fourth experiment, *random single-block multi-manipulation*, applies random manipulations to all frames within a single fixed-length block for all video segments. In this experiment, we seek to understand how manipulation variety and non-uniformity impacts our model performance. We vary the size of manipulated blocks from 20% (4 frames) to 65% (13 frames) and also vary the number of manipulations that are selected from from 2 to 4. Results for this experiment are provided in Table 7. As a reminder, these accuracies represent the total accuracy for detection of manipulated frames for each scenario, meaning the accuracy of non-manipulated frames is not included for the calculations.

Table 7: Total manipulation classification accuracies for Experiment 4 (*Random Single-Block Multi-Manipulation*).

	Number of Manipulations		
Block Size	2	3	4
4 (20%)	0.991	0.982	0.978
8 (40%)	0.941	0.965	0.978
13 (65%)	0.963	0.970	0.973

From Table 7, we notice a stark improvement in the single-block setting over the results reported for Experiment 1. Specifically, we achieve greater than 94% classification accuracy in detecting manipulated video frames in all tested scenarios. Although this scenario, where a single-block of variable length is applied random manipulations, is unrealistic in practical settings, it again highlights the strength of our model in detecting video manipulations when there is high variability between frames. In addition, the model excels at classifying manipulated frames even when a large number of frames are manipulated, such as in the case of 95% block size. In this case, the uniformity argument again holds, as the accuracy increases from 96.6% when 2 manipulations are assigned to the block to 98.7% and 98.9% when 3 and 4 manipulations are assigned, respectively.

4.6 Experiment 5: Random Multi-Block Multi-Manipulation

Finally, the fifth experiment, *random multi-block multi-manipulation*, applies a random manipulation to each frame within a sequence of fixed-length repeating blocks that are separated by a fixed distance. In this experiment, we seek to again study how variations in manipulation patterns differ depending on the length of manipulated sequences and the distance between manipulations. We again vary the quantity of manipulated blocks from 2 to 5, block size from 15% (3 frames) to 35% (7 frames), and the number of unique manipulations from 2 to 4. Results from experiment 5 are shown in Table 8.

Table 8: Total manipulation classification accuracies for Experiment 5 (*Random Multi-Block Multi-Manipulation*).

	Number of Manipulations		
Block Size	2	3	4
3 (15%)			
$n_{blocks} = 2$	0.822	0.991	0.982
$n_{blocks} = 3$	0.977	0.963	0.972
$n_{blocks} = 4$	0.990	0.978	0.992
$n_{blocks} = 5$	0.952	0.952	–
5 (25%)			
$n_{blocks} = 2$	0.996	0.975	0.967
$n_{blocks} = 3$	0.945	0.982	0.986
7 (35%)			
$n_{blocks} = 2$	0.969	–	–

From Table 8, we observe high-performing results similar in nature to those observed in Experiment 4, achieving accuracies above 94.5% in all tested scenarios, except for an 82.2% accuracy result for 2 blocks of size 15% and 2 manipulations. Similar to the other experiments, the lowest accuracy is likely a result of uniform blocks that are generated by the randomization process leading to a increase in uniformity, as we see highly accurate predictions when 2 manipulations are used at higher block sizes that are less likely to result in uniform manipulation segments. Once more, the negative trend in accuracy seen in experiment 2 for our proposed model becomes non-existent as manipulations are varied within blocks themselves.

4.7 Summary of Results

In summary, our proposed model demonstrates a strong ability to detect and classify video manipulations in a wide variety of manipulation scenarios; however, the model also highlights a strong weakness in dealing with long consecutive sequences of manipulations. In particular, our model achieves poor accuracy when a manipulation is applied to a majority (50%) of frames in a video sequence, specifically when the segments of manipulation are contiguous or separated only by a small amount of frames. In contrast, our model performs extremely well (roughly 90% accuracy) when there is large spacing between manipulations and when manipulation blocks are not uniform. This result indicates that our LSTM network is able to effectively utilize the regularized ResNet features to recognize class changes between frames, suggesting that our pipeline and model are extendable to other types of manipulations in videos. In comparison with a tested LRCN baseline, our model demonstrates remarkable performance in overall classification accuracy, achieving vastly higher accuracies in Experiments 1 and 2 and demonstrating a stronger ability to classify manipulations when they appear in smaller fractions of the video segment or with increased frame class transitions. Overall, the experimental results demonstrate a partial success in pursuit of detecting video manipulations and present a clear avenue for future work in the area of majority-video manipulations, which we highlight alongside other ideas in Section 6.

5 Thesis Contributions

5.1 Matt's Contribution

Matt contributed to several areas of the thesis, including a large effort in designing the infrastructure and overall pipeline from data generation to model experimentation. In particular, Matt created the data generation framework to

enable modular creation of dataset variations (relating to the different experimentation setups) that were used in the development and evaluation of our models. Specifically, the data generation framework enables automatic calculation of frame manipulations for selected input videos that adhere to configuration parameters and ensure the properties of manipulation distributions. Further, he was heavily involved in the creation of code for manipulating data into a digestible format for the computational models, as well as experimentation with sequential filtering and additional preprocessing components, some of which are not reported in this work but provide opportunity for future research in the handling of video data for the task of manipulation detection. The developed code allows for modular extension of the presented framework for future work and comprises a testing sandbox that is data-independent allowing the application of various sequential models with limited requirements and barriers to execution. In addition to model development, Matt was also involved in idea generation, background research, and discussion around designing and building the network architecture that is proposed in this work, specifically in regard to work surrounding the design and implementation of RNNs. Finally, Matt contributed an equal portion of the writing, formatting, and editing process that went into the final thesis report.

5.2 Alex's Contribution

Alex contributed in a variety of different ways to the thesis. A large part of his efforts went into the implementation and research of potential improvements to some of the preliminary models that eventually evolved into the final layers present in the proposed model. Some of the improvements he examined involved trying out several hyper parameter combinations and sizes that other successful models have used in the past when dealing with extracting features from images to see whether they would affect the results of experiments positively. Using intuition from these trials, he eventually created models that performed much better at detecting certain manipulations. He also created code in this area to test multiple of these different parameters across different test models consecutively, which allowed for faster analysis of results. Furthermore, in order to improve the accuracy of results, he experimented more with the preliminary models using different video datasets to determine whether better feature generalization could be achieved in more controlled video scenarios. Additionally, he played a role in creating the code for adding manipulations to the video sequences and implementing the various final experiments. This involved further research into python imaging libraries and techniques that could successfully replicate the manipulations seen in real-world manipulations most successfully. Along with this code, he helped form the Jupyter notebook pipeline for running these experiments, thus defining where training and validation data is created, fitted to the model, and analyzed. All the final experiments were conducted by him on his local machine, as well. Moreover, Alex was involved in discussions related to the ideas of the high level design of the entire thesis framework and helped code some of the functions that went into it. For example, one of the functions he helped write the code for equally distributes the files from the UCF101 dataset to their respective class folders for the training and validation sets. On top of all the code he wrote, he also took part in writing and formatting an equal portion of the final report, forming some of the diagrams seen throughout in the process. Finally, much of the work prior to working on the thesis involving background research for potential thesis ideas was conducted by Alex.

6 Future Work

In this thesis, we developed a preprocessing framework for automatically manipulating video segments as well as a deep learning-based video manipulation detection model. Before discussing avenues for future work, we acknowledge that the manipulations we chose to detect in this thesis are rather simplistic in comparison to more common forms of video manipulations today such as deep fakes (see [2]). In practice, deep fakes represent modifications to videos that are typically present throughout the entire video and change with respect to the faces that are being used to create the deep fake, where as our manipulations may or may not be present throughout an entire video and aim more to obscure content rather than alter it, except in the case of *insert*. For political video manipulation, deep fakes can be used to alter the facial expression and words that an important political figure is saying, which is very powerful and hard to detect without close inspection. Importantly, our manipulations, though not as powerful or convincing, share qualities around the implications of manipulated videos, specifically the insertion of malicious objects into videos or occlusion of important areas of scenes. Further, our manipulation framework provides a pipeline to investigate how manipulation patterns affect model learning and performance, which remains an important component for detecting manipulated videos, regardless of manipulation quality.

In the following sections, we highlight several avenues for future research within our current framework, including the development of more sophisticated automated video manipulation techniques and preprocessing techniques that could enhance detection of complex manipulated content.

6.1 Manipulation Enhancements

One improvement we propose to the work presented in this thesis is the enhancement of the various manipulations we choose to evaluate our model with. To start, we note that our *insert* manipulation does not cover the most realistic cases of the manipulation, making our model less robust to potential real-world use cases. With more research into appropriate methods, we could edit the inserts within our framework and add more realistic features such as background object tracking, image style blending, and realistic object placement based on visual context. This may increase processing time, but the overall results would be a better representation of our model's capability. Methods for generating videos with these types of object insertions include crowdsourcing, employment of graphic video professionals, or advanced computational manipulation techniques that automate the tracking of objects. Additionally, we could add more complex manipulations that we chose not to include in our current work. One such manipulation is a *delete* manipulation, which selects a random amount of frames to be deleted/dropped at a random point within a given video. This manipulation does not provide clear visual information that stays in frame for multiple video frames, making it much more difficult for our model to generalize features for in its current state. Additionally, our framework is more suited for the current types of manipulations we consider that alter only viewable content in a way that preserves the number of frames, so additional test cases would have to be created. A few other types of more complex manipulations could be cloning and removal of objects within videos. These manipulations present clear visual differences within each modified frame, which suits our framework well, while also being very relevant to the real-world manipulations in videos that are having the largest impact on people's beliefs. Furthermore, we can provide an wider range of variability with manipulation parameters we have chosen in our experiments that control how active the manipulations are on the frames. We've opted for testing on a small range of values for compression, blurring ratio, and image insertion size to keep our experiments manageable given the constraints of the thesis, but we could expand upon our results by looking at more extreme values that might be seen in practical settings. This may also warrant additional experiments that show how our model handles scaling these various parameters, further exposing how robust it can be. Finally, future research should also investigate the presence of multiple manipulations in a given frame, as well as localizing manipulations within a frame.

6.2 Improvements to Proposed Model

A straightforward improvement we propose is improving the manipulation classification accuracy for our proposed model across all potential patterns of manipulations, such as in the case of long sequences of uniform manipulation. One way we could achieve this goal is by experimenting with different types of NN architectures and hyper parameters. There are a variety of other convolutional and recurrent layers, for example, that are listed in the Keras documentation that can be used to achieve the outcome we desire with our ideal model. These layers include such layers as the Conv3D, ConvLSTM2D, and GRU layers. The Conv3D and ConvLSTM2D layers act as an alternative to the TimeDistributed convolutional layers utilized in our TimeDistributed Resnet network since they equivalently add an additional time dimension to their input parameters for sequence handling, while the GRU layer is an alternative RNN to LSTM layers which works better with certain types of input. We experimented minimally with a few of these special layers types in prior test model architectures, but the work presented in this thesis showcases our best-performing model. With more experimentation, more advanced architectures that address the shortcomings of our proposed model could significantly improve classification accuracy.

6.3 Improved Frame Preprocessing

A third improvement we propose is more advanced preprocessing steps to maximize the feature generalization of our proposed model. Often with similar machine learning-based works, the frames of the images become augmented in a way that can accentuate certain desired characteristics for the model to detect and categorize. For example, images are often rotated for image classification models to assist models in detecting objects in an orientation-agnostic manner.

One potential preprocessing method concerns subtracting out the mean visual features for each given video from every frame in each video, respectively. Ideally, once applied, the visual changes (manipulations) from frame to frame would be more pronounced, thus allowing the model to identify objects that are not present throughout the length of the video, or that change coordinates. An alternative method in the same vein of this preprocessing step would be to subtract from each frame the visual features of the preceding frame for each given video. In doing this, only visual changes that have manifested from frame to frame will be visible to the model, making manipulations more obvious to our classification network. We briefly experimented with these types of preprocessing steps within our data generator, and occasionally achieved better results; however, results varied unpredictably, so we excluded them from our final preprocessing framework, though if given more attention we are confident that these additions may help our model succeed, especially with addition of more complicated manipulations.

6.4 Prediction Post-Processing

Another potential improvement involves analyzing our predictions *ex post facto* in a manner that bolsters the overall predictive power. For example, we could apply our manipulation classifier on a sliding window of frames across the video, averaging predictive values for each window to determine the final classification. This approach exposes our classification network to more video context and may help dampen effects where a manipulation extends for a long period of time. Another benefit arises in the case where our prediction model performs poorly when a manipulation falls at the end of a window. By sliding a window along the frames of a video, the manipulation is forced to different sections of the window, and the averaging operation could help limit the impact of poor-performing manipulation patterns.

7 Conclusion

In conclusion, we have successfully created a framework that, given any dataset of videos, produces a new dataset with select manipulations applied to the original dataset of videos for use in training DL-based models to detect video manipulations. Additionally, we have produced a prototype model that performs extremely well at detecting and classifying manipulations on a frame-to-frame basis in most situations involving detection of manipulated video frames. Our proposed model utilizes a modified version of the LRCN architecture which incorporates the state-of-the-art ResNet classification network for powerful feature extraction as well as several forms of regularization to promote generalizability, and finally a recurrent LSTM network to capture temporal dependencies between features in classifying manipulations. In comparison with a VGGNet-based LRCN baseline model, our model vastly outperforms the baseline in correctly classifying manipulated frames, including achieving 65.1%, 85.2%, 73.1%, and 50.4% accuracy for block sizes of 5, 20, 40, and 65%, respectively, in Experiment 1, as opposed to 28.1%, 25.2%, 26.8%, and 41.7% accuracy for the baseline. Furthermore, for Experiments 3, 4, and 5, our proposed model achieves an average prediction accuracy 96.3%, with the highest overall accuracy attained at 99.6%. While our model lacks high-performing results on manipulation patterns in which a large majority of a video segment is manipulated, it demonstrates a stark improvement over a standard LRCN baseline and our work presents a foundation for improvement to account for more common, complex video manipulations in future iterations.

References

[1] Soomro, Zamir, and Shah. "UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild." https://arxiv.org/abs/1212.0402. https://www.crcv.ucf.edu/data/UCF101.php. University of Central Florida. Accessed March 2019.

[2] Guera and Delp. "Deepfake Video Detection Using Recurrent Neural Networks." https://engineering.purdue.edu/ dgueraco/content/deepfake.pdf. Purdue University. Accessed June 2019.

[3] "YouTube M8 DataSet." https://research.google.com/youtube8m/. Video Understanding hroup, Google Research. Accessed March 2019.

[4] Zhou, Han, Morariu, and Davis. "Learning Rich Features for Image Manipulation Detection." https://theblog.adobe.com/spotting-image-manipulation-ai/. University of Maryland, College Park Adobe Research. Accessed March 2019.

[5] https://www.freepngs.com/. freePNGs. Accessed March 2019.

[6] Donahue, Hendricks, Rohrbach, Venugopalan, Guadarrama, Saenko, and Darrell. "Long-term Recurrent Convolutional Networks for Visual Recognition and Description." https://arxiv.org/pdf/1411.4389.pdf. UC Berkeley UT Austin UMass Lowell. Accessed March 2019.

[7] Simonyan and Zisserman. "Very Deep Convolutional Networks For Large-Scale Image Recognition." https://arxiv.org/pdf/1409.1556.pdf. University of Oxford. Accessed May 2019.

[8] Refaeilzadeh, Tang, and Liu. "Cross Validation." http://leitang.net/papers/ency-cross-validation.pdf. Arizona State University. Accessed May 2019.

[9] Szegedy, Vanhoucke, Ioffe, and Shlens. "https://arxiv.org/pdf/1512.00567v3.pdf." https://arxiv.org/pdf/1512.00567v3.pdf. Google Inc. University College London. Accessed June 2019.

[10] http://www.image-net.org/. Stanford University Princeton University. Accessed June 2019.

[11] Laptev, Marszalek, Schmid, and Rozenfeld. "Learning Realistic Human Actions from Movies." https://www.di.ens.fr/ laptev/actions/. INRIA Rennes INRIA Grenoble Bar-Ilan University. Accessed June 2019.

[12] Ren, He, Girshick, and Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." https://arxiv.org/pdf/1506.01497.pdf. University of Science and Technology of China Microsoft Research Facebook AI. Accessed May 2019.

[13] "Nimble Challenge 2017 Evaluation - Data Resources." https://www.nist.gov/itl/iad/mig/nimble-challenge-2017-evalation. NIST. Accessed May 2019.

[14] Dong, Wang, and Tan. "CASIA Image Tampering Detection Evaluation Database." https://ieeexplore.ieee.org/document/6625374/authorsauthors. Chinese Academy of Sciences. Accessed May 2019.

[15] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[16] Lin, Patterson, Ronchi, Cui, Maire, Belongie, Bourdev, Girshick, Hays, Perona, Ramanan, Zitnick, and Dollar. "COCO - Common Objects in Context." http://cocodataset.org/home. CVDF Microsoft Facebook Mighty AI. Accessed May 2019.

[17] Krizhevsky, Sutsukever, and Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf. University of Toronto. Accessed May 2019.

[18] Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." https://www.cs.toronto.edu/ hinton/absps/JMLRdropout.pdf. University of Toronto. Acessed May 2019.

[19] Simonyan, Karen, and Andrew Zisserman. "Two-stream convolutional networks for action recognition in videos." Advances in neural information processing systems. 2014.

[20] Chollet, François and others. Keras. https://keras.io. 2015.

[21] Howard and Williamson. "Video-Detection." https://github.com/matthewjhoward/video-detection. GitHub. Accessed May 2019.

[22] Vincent. "Watch Jordan Peele use AI to make Barack Obama deliver a PSA about fake news." https://www.theverge.com/tldr/2018/4/17/17247334/ai-fake-news-video-barack-obama-jordan-peele-buzzfeed. Vox Media. Accessed June 2019.

[23] Patrini, Cavalli, Ajder, and Cullen. https://www.deeptracelabs.com/. Accessed June 2019.

[24] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in ACM MM, 2014.

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in ICLR, 2015.