Deepfake Detection: Building the Optimal Model

Submitted for the Master of Science,
London School of Economics, University of London

Department of Statistics 2020

Commissioned by Samsung

Supervisors

Dr. K. Kalogeropoulos

Dr. G. Tzimiropoulos

Candidate numbers

42363, 36878, 47602

August 26, 2020

THE LONDON SCHOOL OF ECONOMICS AND POLITICAL SCIENCE

Acknowledgements

Throughout the writing of this dissertation we have received a great deal of support and assistance.

We would first like to thank our internal supervisor, Dr. Kostas Kalogeropoulos, whose guidance was central to the formulation and focus of our research.

Additionally we would like to thank our external supervisor, Dr. Georgios Tzimiropoulos, whose expertise was central to gaining an understanding of the latest developments in computer vision and avenues for potential improvement, and Samsung, who commissioned this project.

We would also like to thank Prof. Milan Vojnovic whose guidance throughout our Masters programme was invaluable. In particular, Milan's course on Artificial Intelligence and Deep Learning sparked our passion for computer vision and laid the groundwork for further exploration into deepfake detection.

Contents

List of Figures

List of Tables

Executive Summary

Context

Deepfakes, videos or images containing a modified or swapped face produced by a deep neural network, are an increasing and unprecedented threat. Online deepfake content is doubling every six months. As of June 2020, approximately 50,000 deepfake videos were identified by Ajder et al. (2020). An overwhelming 63% of all deepfake videos online on June 2020 contained pornographic content. The victims suffer loss of professional and educational opportunities, stalking, psychological damage and defamation (Citron and Franks, 2014). An additional threat of deepfakes presents itself as undetected political deepfake material, which is capable of severe damage in election campaigns as well as in financial markets (Nguyen et al., 2019). The increasing prevalence of deepfakes results in a society suspicious of online visual content. If deepfake creation becomes more accessible and harder to detect, all online images and videos could lose credibility.

Unfortunately, it is clear that deepfakes are here to stay. Their increasing abundance, accuracy and detrimental societal impact mean that the development of high quality detection instruments is of central importance, a non-trivial and increasingly difficult task as deepfake generators become more sophisticated. The purpose of this paper is to develop an optimal deepfake detection model, maximising its accuracy. To achieve this, we set three sub-objectives. The first is to reproduce the results of "FaceForensics++: Learning to Detect Manipulated Facial Images" by Rossler et al. (2019) providing a platform for further improvement. Second, we analyse whether variations of data augmentation and hyperparameter selection can improve accuracy. Lastly, we explore temporal models in comparison to frame-based architectures.

We run all analyses on FaceForensics++ data, which was the most sophisticated dataset available at the beginning of our research. We use Google Cloud Platform's AI Platform to build and train our models. We extract every fifteenth frame from each video, and then obtain cropped faces from these frames using RetinaFace (Deng et al., 2019). After the above data pre-processing, we proceed to tackle our sub-objectives.

Main Findings

This paper's main findings are subdivided according to their respective research sub-objectives. The first one regards reproducing FaceForensics++'s results. We replicate the test accuracy of 99.26% in Rossler et al. (2019) by achieving 99.29% using an Xception base. Moreover, we experiment with a variety of alternative convolutional bases and find that EfficientNet provides the joint highest validation accuracy on FaceForensics++ data. EfficientNet also achieves the highest accuracy on *ImageNet* which is strongly correlated with transfer accuracy. We therefore adopt EfficientNet as our convolutional base, other than for the augmentation experiment, which uses MobileNet due to its lightweight nature.

Our next finding regards the second sub-objective. Train-time augmentation is central to improving deepfake detection. A recent class of information-dropping augmentations aims to prevent overfitting and encourages better generalisation. GridMask is one example that has shown promise and outperforms others (Chen, 2020). Hence, we include GridMask among our custom developed methods inspired by Seferbekov (2020): GridMask-Alternative, FacialArtifacts, FacialArtifacts-Alternative and FaceMask. We perform experiments training six MobileNets changing only the augmentation method. We find that the resulting video validation accuracies, which are already very high for our dataset, do not show any significant advantage in favour of using information-dropping approaches in this particular setting. Nevertheless, further work should explore these techniques' ability to avoid overfitting and increase generalisation. Next, we turn to optimising augmentation for our dataset with DeepAugment, an automatic augmentation selector. The obtained optimal policy includes promising augmentations. However, their magnitudes require further calibration, hence, we incorporate elements of the optimal policy with some magnitude adjustments. Simultaneously, we perform a classical frequency analysis inspired by Durall et al. (2019), which is essentially a heavy augmentation on the data. This algorithm is limited to medium- to high-resolution images, and the optimised model achieves a 87.90% validation accuracy. This accuracy is far below the baseline and so we do not explore this concept further.

Our third set of findings regards training techniques. We analyse hyperparameters which are central to the model's success aiming to find the optimal combination for our model architecture and dataset. The primary goal is to find a suitable range of hyperparameters for our random search. We begin with the learning rate, finding that values of 0.00001, 0.0001 and 0.001 lead to the highest accuracies. We also learn that while a constant learning rate can perform well if tuned correctly, warming up the learning rate for a few epochs and then decaying the rate using a cosine annealing function often leads to more stable, promising results. We progress to experiment with batch size where results indicate that smaller batch sizes (less than 32) allow for a greater range of successful learning rates, stronger regularisation and increased likelihood of escaping sub-optimal local minima. Aside from these pivotal hyperparameters, we investigate label smoothing, weight decay, dropout and class weights, all three of which can positively affect accuracy. With the above findings in mind, we create a search space consisting of all promising hyperparameters and train fifty models to convergence. Nine models perform with impressive validation accuracies. These are included in our final model ensemble, discussed towards the end of this summary.

Our final finding concerns the third sub-objective. We explore whether temporal models can improve accuracy. During deepfake generation, an autoencoder is used on a frame-by-frame basis, and as a consequence it is unaware of any previous or future frames that are generated,

causing anomalies on a temporal scale. A convolutional neural network (CNN) extracts features for each frame, concatenates and passes them to a Long Short-Term Memory unit (LSTM) or Transformer. The best-performing convolutional LSTM achieves a 98.14% validation accuracy. The convolutional Transformer outperforms all LSTM models achieving 99.14%. There is no implementation of a convolutional Transformer in earlier literature, and its results are promising and should be investigated further.

Lastly, we amalgamate all our above findings into a final proposed model, evaluating it on the FaceForensics++ raw and low quality test set. The final model is an ensemble, consisting of nine different EfficientNets trained with optimised augmentations and a JPEG compression. A trained stacked approach meta-learner acts as a classifier to find the optimal way to combine each model in the ensemble. The final test accuracy on the raw data is 99.00%, just below that of Xception. The final test accuracy on the low quality data is 85.71%, outperforming Xception by 4.71% and all other papers reporting results on this test set.

Guidelines

Building on our findings, our first recommendation concerns our pre-processing pipeline. The face-extraction algorithm we developed uses the latest face detection software and is ready to be employed. Secondly, we highly recommend an automatic augmentation selector such as our tailored DeepAugment. Thirdly, we find random search essential for successful hyperparameter selection. The above provide guidelines for a deepfake detector approach should the end-user need tailoring to a specific dataset. However, if FaceForensics++ data is sufficient, we provide guidelines on using our final model below.

Our final product packages the pre-processing pipeline and trained ensemble. The product can be employed as follows: first pre-process new data with *extract_compressed_videos.py* and *retinaface-cropper.py* scripts. Then, run the output through our model, which can be loaded directly into Keras. This product is designed to be used on social media platforms or other websites containing frequently uploaded visual material. We suggest incorporating our model as an online tool that flags manipulated content to platform users, or automatically removes this content if a significant threat is detected. We also propose running the model at the content-uploading stage. Our final recommendation involves monetising the product. We suggest providing an API service to other companies where our technology can be used to detect manipulated content on their platforms. Our product can not only help to reduce the threat that deepfakes pose, but also contribute to preserving visual material's online credibility. We look forward to seeing its value in practice.

1 Introduction

1.1 Context

Attempts to falsify the identity of photographic subjects date back to the mid-19th century. It is believed that the technique came about as a result of the increased demand for lithographies, typically accompanied by many forged duplicates. Evidence of this can be found as far back as 1865, where Abraham Lincoln's face replaced that of Vice President John Calhoun on a portrait by A.H. Ritchie (Farid, 2011). Early face swaps made use of rudimentary methods to replace the subject's face – careful inspection reveals uncharacteristic features allowing most people to identify the lithography as fake. However, recent advances in deepfake technology have made the distinction almost invisible to the human eye.

The term deepfake is loosely defined, but for the purpose of this paper we define a deepfake as a video or image containing a modified or swapped face produced by a deep neural network. Contrastingly, a cheapfake is a fake video or image produced without a learning component (Dolhansky et al., 2019). The two most common deepfake creation techniques are Deepfake Autoencoders (DFAE) and variations of Generative Adversarial Networks (GANs).

DFAE methods use a shared encoder but different decoders. The idea behind this is that the shared encoders learn common features across various identities whereas the decoder extracts identity-specific features. The last stage involves running the source identity through the target decoder in order to obtain the face swap. The process is illustrated in Figure 1.

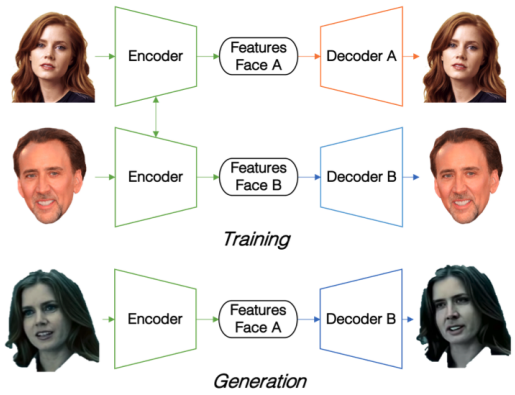

Figure 1: DFAE Procedure: The first two networks are trained using a shared encoder. The final network (shown in the last row) generates the deepfake using the shared encoder coupled with the decoder from the target face (Güera and Delp, 2018).

Contrastingly, GANs consist of two parts: a generator, which learns to create realistic deepfakes, and a discriminator, which attempts to distinguish fake and real data (Mo et al., 2018). During training, these two parts work together to produce deepfakes of such high quality that it is nearly

impossible for humans to distinguish real from fake, more so than DFAE.

Facial manipulations are typically split into two categories: identity and expression manipulations. Identity manipulations such as DFAEs and FaceSwap replace an individual's face in a target image with another person's face from a source image. The second category instead focuses on replacing certain regions of the face, preserving an individual's identity while modifying the facial expression. Widely used methods capable of carrying this out include Face2Face (Thies et al., 2016) and NeuralTextures (Thies et al., 2019). Humans perform poorly in recognising deepfake material. Rossler et al. (2019) studied human performance at deepfake detection, replicating a social media environment. The detection rate was reduced as image quality decreased, and it was lower than randomly guessing for two types of manipulations. Their results clearly indicate that manipulation technology is capable of generating deepfakes indistinguishable from real for a human eye.

Technological advances in facial manipulation have already had undeniable adverse effects on society. Online deepfake content is doubling every six months (Ajder et al., 2020). Pornography and politics are the two areas most affected. As of June 2020, approximately 50,000 deepfake videos were identified by Ajder et al. (2020). They discovered that an overwhelming 63% of all deepfake videos online in June 2020 contained pornographic content with an unequal distribution over gender, nationality and professions. All current pornographic deepfakes target females. Contrastingly, non-pornographic content consisted of 61% males. In both categories, most manipulated identities were high profile individuals from the entertainment industry. A small proportion of these identities consisted of politically and socially influential characters including news presenters, business owners and politicians.

The above mentioned adverse effects are manifold and the harmful effects of fake pornographic content are far-reaching. Victims suffer loss of professional and educational opportunities, stalking, psychological damage and defamation of reputation (Citron and Franks, 2014). A disturbing example of this occurred in the Phillippines in 2016 where the legal counsel to President Duterte used a faked sex video to justify the imprisonment of Senator Leila De Lima (Paris and Donovan, 2019).

Additionally, deepfakes are becoming increasingly destructive in the world of politics and cybersecurity. The prevalence of deepfakes results in a society suspicious of all online visual content. If deepfake creation becomes more accessible and harder to detect, all online image and video content could lose credibility. Thus, the impact of truthful visual material is undermined. Moreover, undetected manipulated material is capable of causing severe damage in the financial markets as well as to election campaigns (Nguyen et al., 2019). Deepfakes can also be used to fraudulently gain access to sensitive individual information.

It is clear that deepfakes are here to stay. Their increasing abundance and detrimental societal impact mean that the development of high quality detection measures is of central importance. The urgency of combatting deepfakes has been recognised by multiple academic groups and large companies, with a recent surge in deepfake detection research and public challenges focused on the construction of networks capable of precise deepfake detection. The central theme throughout this paper is to explore a variety of methods aiming to build the optimal model for deepfake detection, achieving the highest possible accuracy.

There are several convolutional architectures which perform well in deepfake classification. However, they suffer from poor performance when training and testing data are intrinsically mismatched. There are two potential causes of this misalignment: firstly, manipulated content, generated using techniques absent from the training data, tends to evade detectors. Secondly, manipulated content often undergoes a long sequence of transformations, and including all possible deepfake processing histories in the training dataset is not feasible. For these reasons the training dataset is an integral component in building a high performing model (Verdoliva, 2020). For our research, we use data from "FaceForensics++: Learning to Detect Manipulated Facial Images" by Rossler et al. (2019). We explain the choice of dataset and provide further information in Section 4.2.

1.2 Research Objectives

This paper's overarching purpose is to develop an optimal model for deepfake detection, maximising its classification accuracy. We achieve this through three sub-objectives:

1. Reproduce the results of FaceForensics++.
2. Determine whether data augmentation and its optimisation as well as the correct hyperparameter selection can improve model performance and generalisation.
3. Determine whether temporal model architectures produce superior results when compared to frame-based architectures and their corresponding aggregation techniques.

1.3 Summary of Main Results

This report initially sets out to achieve its first objective of reproducing FaceForensics++ results. After replicating their test accuracy with Xception, we find that alternative network structures are capable of better results. EfficientNet is considered the most promising architecture and is adopted as the convolutional base for our deepfake detector.

Our second set of findings involves the role of augmentations in model accuracy. We discover that information-dropping augmentations have shown great promise and an automated augmentation selector is capable of identifying suitable augmentations tailored to specific training data. We use DeepAugment for its combination of computational efficiency and reliable performance. Concurrently, we discover that a heavy augmentation grounded in signal processing

theory based on Durall et al. (2019) referred to as 'Simple Features' helps train simple classifiers within seconds but leads to sub-par accuracy.

Our third set of findings involves the role of hyperparameters in model accuracy. We find that specific learning rates (0.001, 0.0001, 0.00001), decaying rate schedules, small batches (less than 32) and label smoothing all have a positive effect on model performance. Moreover, techniques like weight decay, dropout and class weights further improve detection accuracy. Using this insight, we run a random search to find optimal combinations of hyperparameters. We obtain nine high performing models which are later included in our final ensemble.

Our fourth set of results covers temporal models that aim to identify inter-frame anomalies characteristic of deepfakes. We find that a convolutional Transformer model performs better than the Long Short-Term Memory (LSTM) architecture in validation accuracy and loss. With the above in mind, both models still perform with lower accuracies than frame-based methods. We conclude that temporal models should be investigated further on alternative datasets, but cannot be included in the final model if FaceForensics++ data is used. The convolutional Transformer is a novel type of model and does not exist in literature, to the best of our knowledge.

Lastly, we propose a final model comprising an ensemble of nine networks, built using different versions of EfficientNet bases. We test this model on both raw and low quality data. The final test accuracy on the raw data is 99.00%, just below that of Xception from FaceForensics++. The final test accuracy on the low quality data is 85.71%, outperforming Xception by 4.71% and all other papers reporting results on this test set. Based on our findings, we formulate guidelines in the conclusion to use and optimise our models.

1.4 Outline of the report

This report begins with a review of related work most relevant to the construction of high performing networks. Then, the research objectives are laid out to substantiate the structure of the next sections, after which the technical setup for the research approach is presented in detail. Section 5 reproduces the model from FaceForensics++ and presents baseline results to obtain the best performing convolutional base. Section 6 describes the data augmentation experiments and their results, and Section 7 further looks at heavy augmentations resorting to signal processing theory. To further improve the models, Section 8 presents the hyperparameter selection experiments. Section 9 explores temporal video detection techniques, using Long Short-Term Memory units and self-attention to maximise performance. Section 10 presents the final model and its performance on the FaceForensics++ test dataset. Finally we present our conclusion in Section 11, which summarises and links our results, discusses paths for future research and provides guidelines for the end user.

2 Related Work

This section presents earlier deepfake research. Firstly, we give a brief overview of deepfake generation techniques. We then discuss current deepfake detection approaches for images and videos. Finally, the section concludes with an overview of the role of data augmentation and hyperparameter selection in the highest performing detection approaches.

2.1 Deepfake Generation

Since 2017, the technology behind synthetic image generation and manipulation has progressed rapidly. Initially, most manipulation methods were relatively easy to detect due to the content's low resolution. However, this has been overcome with the aid of progressive growing – a new training approach proposed by Karras et al. (2017) applied to Generative Adversarial Networks (GANs). The resulting GANs synthesise high quality faces. Additional processes like correction of the shape, boundary, lighting and scaling are applied to the manipulated image or video to make the task of detecting fake visual material even trickier for the public (Dang et al., 2019). Furthermore, manipulation methods have evolved to train a unique model for every image or video manipulation rather than applying the same transformation to all data. Rossler et al. (2019) confirm that these approaches significantly complicate the detection. As this paper only uses data from that dataset, the next two paragraphs describe the deepfake generation methods employed in FaceForensics++.

The FaceForensics++ data comprise the two different manipulation methods, expression swap and identity swap. Expression swap consists of modifying the facial expression of the person whereas identity swap consists of replacing the face of one person in a video with the face of another person. The expression swap methods employed in their data are NeuralTextures and Face2Face. In both these methods, the first frames of each authentic video are used to obtain a 3D temporary facial identity, after which the expression over the remaining frames is tracked. Fake videos are then created by transferring the source expression parameters to the target video. NeuralTextures (Thies et al., 2019) uses original video data to learn a neural texture and rendering network of the target person, where a patch-based GAN-loss is used. Essentially, only the facial expression of the mouth region is modified. Face2Face is a computer graphics based approach, and can be considered a real-time facial re-enactment system.

The identity swap methods employed in FaceForensics++ are FaceSwap and DeepFake. FaceSwap consists of face alignment, image blending and Gauss-Newton optimisation to swap the source face to its target. The DeepFake method is based on two autoencoders with a shared encoder, trained to reconstruct training images of the source and the target face, respectively. For the generation of a fake image, the trained encoder and decoder of the source face are applied to the target face. The output of the autoencoder is blended with the rest of the image by Poisson image editing.

There are two approaches to generating deepfakes: learning-based and computer graphics-based approaches. The former is more modern and realistic than than the latter.

2.2 Fake Image Detection

Before the emergence of deep learning, detection techniques mainly comprised of non-learned forensic analysis. The top performing approaches looked for artifacts related to the in-camera processing chain (camera-based clues) or the out-camera processing history (editing-based clues) (Verdoliva, 2020). Other approaches focused on the audio-visual artifacts that existed mostly in the earliest generation of deepfake videos. In this line of research, a recent study by Matern et al. (2019) proposed fake detection systems based on relatively simple visual aspects such as missing details, strange eye colour and reflection in the teeth and eye area. In the end, 18 different facial action units related to movements of facial muscles were considered, and moreover, four features related to head movements were considered. Short video clips were then reduced to a feature vector of dimension 190 which the model used to classify videos. Such deepfake artifacts, however, generally cannot be traced in this way anymore.

While simpler methods showed promise, recent focus has shifted to deep learning based approaches achieving higher accuracies. CNNs are used primarily to detect the presence of artifacts from the face regions and surrounding areas of deepfakes. The most successful methods use a convolutional framework with minor adjustments targeting characteristic features found in deepfakes. The next few paragraphs report papers with results on the FaceForensics++ dataset, so that comparison is possible.

The best-performing CNN in the FaceForensics++ paper was based on Xception (Chollet, 2017b), a traditional CNN trained on *ImageNet* based on separable convolutions with residual connections. This architecture was transferred to a deepfake task by replacing the final fully connected layer with two outputs. It achieves an accuracy of 99% on raw data, and of 81% on low quality (LQ) data.

Instead of repurposing the final dense layers of a pre-trained CNN, many papers propose more novel techniques. Afchar et al. (2018) propose two compact detection networks focused on the mesoscopic properties of images which, when evaluated on the FaceForensics++ dataset, achieve an accuracy of 95% on raw data, and of 70% on LQ data. Xception outperforms these networks. Nguyen et al. (2019) incorporate multi-task learning for deepfake detection. They achieve accuracies above 92% on an older version of FaceForensics++, but achieve accuracies below 65% on the current FaceForensics++ dataset. Wang and Dantcheva (2020) study approaches based on 3D CNN, incorporating both motion and spatial information. Detectors in this paper are based on I3D and 3D ResNet approaches, and outperform Xception on LQ data from FaceForensics++. This is relevant as LQ images are a more realistic proxy for deepfakes in the wild. Capsule networks, as proposed by Nguyen et al. (2019), are introduced to accompany CNNs. These rely

on a dynamic routing algorithm, and are capable of describing the hierarchical pose relationships between object parts. These networks achieve a 93% accuracy on the FaceForensics++ data.

Furthermore, Dang et al. (2020) recently proposed attention mechanisms to further improve the training process. The proposed approaches do not report accuracy but achieve an Area Under Curve (AUC) of 99.4% on the FaceForensics++, higher than any of the papers above. The proposed attention map can be implemented easily and inserted into existing networks, through including a new convolutional layer and its associated loss function. In spite of the fact that a fair comparison of different deepfake detectors is hard, the attention mechanism approach can be considered state-of-the-art. A recent meta-review of deepfake literature concludes that many different approaches have been proposed in the literature, and they all seem to perform extremely well. However, despite these results they all show poor generalisation results to unseen databases (Tolosana et al., 2020).

2.3 Fake Video Detection

There are two approaches to detecting fake videos: frame-based approaches focusing on visual artifacts within each frame, and temporal approaches, instead investigating temporal features across frames in addition to their visual artifacts (Nguyen et al., 2019).

Frame-based approach Nguyen et al. (2019) and Afchar et al. (2018) use frame-based approaches where each video is treated as a collection of frames: selected frames are extracted from a video, fed into a network and visual artifacts are learnt separately on each image. This approach allows for the use of the same network structures present in the fake image detection scenario. Once all individual frames have been classified, their predictions are aggregated per video to obtain a video-level prediction. Afchar et al. (2018) provide evidence in favour of this method by arguing that "the effect of punctual movement blur, face occlusion and random misprediction can thus be outweighed by a majority of good predictions on a sample of frames taken from the video."

While aggregating frames using a simple mean has had recent success, there is room for a more refined approach depending on the dataset. Li and Lyu (2018) argue that due to the frequent illumination changes, head motions and face occlusions present in the majority of the videos, it is optimal to predict on all frames of a given video. The top third of these are then averaged for each video to produce aggregated video predictions.

Temporal approach Four approaches are highlighted here. Both Sabir et al. (2019) and Güera and Delp (2018) incorporate recurrent neural networks (RNNs) on top of CNNs. The idea behind such models is to exploit possible temporal discrepancies across frames. These discrepancies are not always clear to the human eye, but a prominent example is flickering of light in different

frames (Güera and Delp, 2018). To incorporate information from such discrepancies, these papers consider a recurrent convolutional neural network trained in an end-to-end fashion. Sabir et al. (2019) achieve an AUC of 96.3%, 96.9% and 94.3% for the FaceSwap, DeepFake, and Face2Face methods respectively in the FaceForensics++ dataset. Only LQ videos were selected in the analyses. Güera and Delp (2018) use a proprietary dataset and achieve a test accuracy of at least 96%.

Amerini et al. (2019) propose the adaptation of optical flow fields to identify potential inter-frame dissimilarities. An optical flow is a vector field computed from consecutive frames that can be used to compare motion across consecutive frames. The motivation behind this approach is that fake videos are more likely to have an unnatural optical flow due to the unusual movement of eyes, lips, or facial muscles. The results are sub-par compared to other papers, only achieving an accuracy of 80% on FaceForensics++.

Lastly, Song et al. (2018) propose a novel conditional recurrent generation network for deepfake detection. This network uses both audio and image features in the recurrent unit for temporal dependency, and a pair of spatial-temporal discriminators. They offer no results on the FaceForensics++ data but the concept is promising.

In this paper we explore the Transformer (Vaswani et al., 2017) to incorporate temporal information. As far as we are aware, there is no current implementation combining both CNNs and a Transformer.

2.4 Augmentation

Train-time augmentation plays a central role in improving deepfake detection models. The first benefit arising from image augmentations is that models observe seemingly new data thereby expanding the training observations. The second benefit is the avoidance of overfitting on training data, since modifications are different for each iteration leading to data often not being seen again in the exact original state. Data augmentation is essential for successful generalisation and high performance on previously unseen manipulations. Wang et al. (2020) explain that even simple and common image augmentations like geometric and colour alterations are robust to JPEG compression, blurring, and resizing. The importance of augmentation is also highlighted by Dolhansky et al. (2020), who mentioned "clever augmentations" as one of the main three themes in the top performing solutions of the Deepfake Detection Challenge (DFDC), hosted earlier this year. The next three paragraphs contain a brief description of more complex augmentation approaches and tools for data augmentation optimisation.

In 2017, Zhang et al. proposed an augmentation titled 'Mixup', which consists of convex combinations of image pairs and corresponding labels. As shown by tests on multiple datasets including *ImageNet*, their augmentation approach improves generalisation and robustness of

models by encouraging neural networks "to favor simple linear behavior in-between training examples." (Zhang et al., 2017, p.1)

Recently, there has been much progress in a different area of train-time augmentation, namely information-dropping approaches. These include methods such as Cutout (DeVries and Taylor, 2017), Random erasing (Zhong et al., 2017), Hide-and-Seek (Singh et al., 2018) and GridMask (Chen, 2020). GridMask is the latest of the four methods listed above, and is reported to outperform the others on multiple datasets. GridMask is statistically more likely to remove some but not all information from an image object due to its grid-like mask. Chen (2020) also mentions that while other information-dropping methods perform well on smaller datasets, they do not achieve the same success on larger ones.

Lastly, since not all augmentations are equally effective for different detection tasks, augmentation optimisation for a given dataset has attracted much attention. cubuk2018autoaugment proposed the first optimisation tool, AutoAugment. It aims to find optimal augmentation policies from a specified space of image transformations with the aid of reinforcement learning. Integration of augmentation policies suggested by AutoAugment has led to significant improvements in generalisation.

2.5 Hyperparameter Selection

Improved network architectures and augmentations are not solely responsible for improvements in detection accuracy. Optimisation methods, loss function tweaks and other training procedure refinements also play a major role (He et al., 2019). It is clear that the success of any deep learning model critically depends on the choice of hyperparameters. Model hyperparameters, unlike parameters which are learned during training, are values set before training. Hyperparameters can be thought of as a model's settings – these play a major role in obtaining precise models. Because optimal hyperparameters depend on the nature of data, they are not known ahead of time and therefore need to be tuned to achieve the highest possible accuracy on validation data. This section focuses on the most important hyperparameters and presents suggestions to calibrate their values found in past research.

The first, and most important hyperparameter, is the learning rate (Bengio, 2012). Goodfellow et al. (2016) suggest testing learning rates in the range 10^{-5} to 0.01. Bishop et al. (1995) explain that low learning rates can take too long to train and fail to reach optimal points whereas high learning rates can lead to divergent oscillations. Obtaining the optimal mix is critical. Learning rate schedules are capable of extracting further improvements from the initial learning rate selected above. Goyal et al. (2017) propose a warm-up strategy aimed at avoiding potential divergence in the early stages of training by linearly increasing the learning rate to the initial value during the first few epochs. This can be paired with a cosine annealing strategy, a learning

rate decay technique that has led to impressive results in the past few years (Loshchilov and Hutter, 2016).

The choice of batch size can also play a role in finding optimal weights. Some research suggests that large batch sizes result in a small range of learning rates which provide optimal solutions (Masters and Luschi, 2018). However, most research on batch sizes points towards small batches leading to superior results. Wilson and Martinez (2003), LeCun et al. (2012) and Keskar et al. (2016) show that large batches display drops in generalisation ability and are more prone to getting caught in sub-optimal local minima.

Taking further steps to ensure generalisation to all data, it is recommended that several forms of regularisation are included in the training process. Weight decay has been shown to suppress irrelevant components in weight vectors as well as reduce the effect of noise on predictions (Krogh and Hertz, 1992). Jia et al. (2018) propose applying regularisation to the weights whilst leaving the biases unregularised in order to maximise generalisation. Label smoothing, proposed by Szegedy et al. (2016), can supplement weight decay by converting hard allocations to soft allocations, thereby introducing some uncertainty. For example, a label of 1 might be converted to 0.99. Label smoothing introduces additional regularisation, improved generalisation and avoids over-confidence in the network by calibrating prediction confidence to prediction accuracy (Müller et al., 2019). Lastly, dropout complements the above two regularisation techniques by thinning networks during train time. It achieves this by randomly dropping units and their connections from networks during training, thereby reducing the possibility of overfitting (Srivastava et al., 2014).

While the above literature provides a valid general framework for hyperparameter selection, there is no universally accepted combination of hyperparameters. The choice is highly dependent on a wide range of factors including the dataset characteristics. For this reason, Bergstra and Bengio (2012) recommend conducting a random search on hyperparameters, increasing the likelihood of obtaining optimal hyperparameters without the excessive resources required by a grid search.

3 Research Objectives

The overarching purpose of this paper is to build the optimal model for deepfake detection, achieving the highest possible accuracy. We use performance on the FaceForensics++ raw and low quality test data as the standard for model success.

The above overarching aim can be split into three sub-objectives. Firstly, we plan to reproduce the work of FaceForensics++. That is, we intend to train CNNs on this publicly available video dataset and reproduce the test accuracy they achieved on raw data using Xception as a convolutional base. This provides a route to more complicated architectures and potentially higher accuracies.

Our second sub-objective is to determine whether augmentation and hyperparameter tuning can improve model performance and if so, which augmentations and hyperparameter values lead to the best results. Here, we pay particular attention to information-dropping augmentations, automatic augmentation optimisers, a simple features approach, hyperparameter combination trials and their contribution to model accuracy.

Our third sub-objective is to determine whether temporal models can lead to accuracy gains. Unlike standard convolutional architectures that make predictions on isolated frames, temporal models are capable of picking up nuances between frames. We explore whether these nuances are characteristic of deepfakes. We achieve this through the implementation of both a convolutional RNN and convolutional Transformer.

The three sub-objectives above investigate earlier detection techniques as well as novel architectures in pursuit of the optimal deepfake detector. These three sub-objectives encompass the overarching purpose, as the first is a solid basis to work further upon. The second then builds upon this basis by optimising the key elements of deep learning models, while the third investigates and ensures that all potential relevant information of deepfake detection is captured. After realising these research objectives, we produce recommendations and guidelines for end-users of our final product.

4 Setup

This section describes the setup required for our experiments, focusing on the computational resources, dataset and pre-processing steps which transform data into a usable format.

4.1 Methodology, Resources and Efficiency

All preliminary analyses and sample testing are run using the free Graphics Processing Unit (GPU) provided by Google Colab. However, very deep models tend to perform well on image recognition tasks and so the outdated GPU provided by Colab is insufficient for larger, more complex architectures. Moreover, the size of the datasets described in Section 4.2 requires all data to be stored on a Solid-State Drive (SSD) persistent disk that can be attached to instances during training. This is not possible with Google Colab. For these reasons, we applied for and obtained a Google Cloud Academic Research Grant of $5,000. Hence, we leverage the parallel use of multiple GPUs through Google Cloud Platform's AI Platform to train end-to-end models. All networks are trained using AI Platform notebook instances. Detailed instructions on recreating this procedure can be found in the Appendix. The notebooks required to train networks can be found in the GitHub repository.

We make use of standard buckets to store trained models, weights, accuracies and other supplementary information. However, data used to train networks cannot be stored in buckets as latency between the buckets and instances to which they are mounted is too high. The best alternative is to store data on an SSD thus minimising the I/O bottleneck. For this reason, all training, validation and test data are stored on a zonal SSD persistent disk. Because data is too large to fit in memory, we use a generator to read batches from disk directories during the training process. An important consideration is the time spent offloading data from the disk to the GPU memory, which is carried out by the CPU. Essentially, the CPU reads data from the disk into RAM and offloads the data to the GPU for training. To fully utilise the GPU, it is essential that this offloading time is as fast as possible. This is achieved with the aid of multi-threading. Instead of sequentially reading data using a single core, we use an 8 core machine which simultaneously reads batches of data to ensure there is always a batch waiting to be fed to the GPU. This minimises training time by ensuring the GPU is never idle. We typically use an 8 core machine with 30GB RAM and T4 or P100 GPUs depending on the hyperparameter complexity and number of trainable parameters.

4.2 Data

We conduct experiments using FaceForensics++ data, since it was the most sophisticated dataset available at the beginning of our research. For this dataset, Rossler et al. (2019) extract 1,000 distinct video clips from 977 collected YouTube videos. Then, these data are manipulated with four different methods to obtain five sets of video clips (including an original collection).

For training, we use the five collections of 1,000 video clips from FaceForensics++. However,

we perform the extraction of images and pre-processing steps on our own with some deviation from the methods employed in FaceForensics++. A detailed description is provided in Section 4.3. Still, we preserve the FaceForensics++ mutually exclusive split between training, validation and test videos.

4.3 Pre-Processing Data

The pre-processing pipeline consists of three stages: downloading videos, extracting every fifteenth frame in each video and extracting one face per frame. First, we run a script to download all videos manipulated using FaceSwap, Face2Face, DeepFake and NeuralTextures from the public dataset.

The second step consists of extracting frames. As the videos are around half a minute in length, each video consists of 900 frames on average. Since not all frames are relevant, we extract every fifteenth frame using the *extract_compressed_videos.py* script. This results in over 158,000 frames in total: 34,408 frames each for original, DeepFakes and Face2Face methods and 27,546 frames each for FaceSwap and NeuralTextures. These frames are stored in a .png format in their respective video folders.

The final step of pre-processing involves extracting square regions containing faces using the *retinaface-cropper.py* script. Our algorithm relies on the pre-trained model RetinaFace-R50 (Deng et al., 2019), a single-stage face detector with a ResNet50 backbone. By performing face localisation on a pixel-wise basis and incorporating joint extra-supervised and self-supervised multi-task learning, this model produces coordinates of two vertices describing the bounding box, and a corresponding probability of a detected element being a face. RetinaFace's accuracy outperforms the best performing model on the WIDER FACE detection challenge (Yang et al., 2016) by 1.1%, achieving an accuracy of 91.4% on face detection. Hence, we choose this model for its compatibility with Keras and superior detection ability. Lastly, our cropping algorithm loops over all frames, extracting a single face from each frame.

Expanding on this final step, the face detector begins by identifying all faces. Then the algorithm disregards elements which are less than 75% likely to be faces. The threshold probability is set to 0.75 because RetinaFace is exceptionally good at identifying faces, including sideways facial profiles and very small faces in the background (often labelling these with a probability of over 0.95). Given a scenario with multiple faces still remaining for a frame, the algorithm picks the face with the larger bounding box. We implement this rule after a manual inspection of the data, which verifies that in some cases the larger face has a lower probability and so the smaller face is selected. We find that in these scenarios, both faces have probability values of over 0.95 and so selecting the larger bounding box ensures the correct face is selected.

Since input images are traditionally squares, we reshape our bounding boxes; the shorter side

of each image is extended equally in both directions to match the size of the adjacent sides. This reshaping allows us to preserve face proportionality and avoid a potential negative impact on training and classification that a deformation could introduce.

Then, the bounding box is enlarged by 1.25. This value is chosen based on the trade-off between the benefit of including regions around the face and drawbacks of the larger storage size required as well as the inclusion of irrelevant information. A justification for a larger bounding box is that the near-facial regions are usually very important for classification, especially for identity manipulation methods, since their contrast with the facial region can uncover face warping artifacts characteristic of these identity manipulation methods. On the other hand, saving larger cropped images would require more storage space and increase computational complexity. Furthermore, including too large a portion of these near-facial regions could negatively impact detection later on since larger images require a greater downsizing factor. This would result in more information loss on the face itself, which is vital for detection, especially for fakes produced by facial expression manipulation methods. We find the scaling factor of 1.25 to be a good balance between inclusion of near-facial regions and keeping the image as small as possible. Finally, we inspect the DFDC's winning solutions to finalise the scaling factor value. Considering that the second-place solution reports the optimal value of 1.5-1.7 on the horizontal (shorter) side, we set our scaling factor to be 1.25 (a corresponding value for scaling up on the longer side).

In some cases, this enlarged bounding box may lie outside the original frame. In this case, a combination of initially shifting the box and then scaling down are applied to crop to the required region. This design ensures that the information preservation on an image level extends to a video level – since the number of frames from which a face is not extracted is minimised.

Overall, our pipeline extracts faces, mostly centrally positioned with differing levels of zoom. Our cropping script is designed to minimize the loss of important information, achieving an extraction rate of a face from a frame of over 99.9%. From this point onwards, we refer to the fully pre-processed frames of the videos (square images of extracted faces) as "frames."

5 Reproducing FaceForensics++

The first step in training a deepfake detector is to decide on a suitable neural architecture. Since 2012, significant advances in network architectures have lead to deep learning dominating the computer vision space. We begin our experimentation on deepfake detection by leveraging high performing pre-trained architectures. This section tackles the first sub-objective of reproducing the FaceForensics++ results, where the Xception architecture is the top performing model. We also test a variety of other networks with the aim of finding a suitable architecture for the rest of this paper.

5.1 Baseline Design

We adopt a rudimentary approach to training at this stage by using a standard set of hyperparameters across all architectures, included in the Appendix. Other than standardising data using samplewise centering, normalisation and resizing images to 224×224, no further image transformations are applied.

Using the baseline hyperparameters, we conduct experiments on six architectures with *ImageNet* initialised weights (Deng et al., 2009) for all networks except EfficientNet, where we use noisy-student initialised weights (Xie et al., 2020). These architectures include VGG16 (Simonyan and Zisserman, 2014), Xception (Chollet, 2017b), ResNet50 (He et al., 2016), MobileNet (Howard et al., 2017), EfficientNetB0 (Tan and Le, 2019) and DenseNet (Huang et al., 2017). We freeze the convolutional base of these networks, discard their top classification layer and replace it with three dense layers. We train these dense layers to convergence, unfreeze the base and fine-tune all weights until validation accuracy does not further increase for 7 successive epochs.

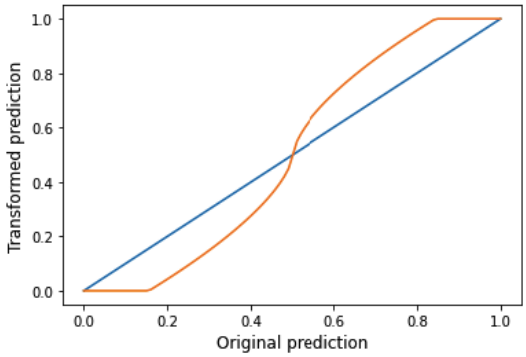

Figure 2: Transformation of individual image predictions for the transformed mean aggregation: Authentic labels are pushed closer to 0 and fake labels are pushed closer to 1.

For each architecture, we use four different methods to aggregate image predictions into video predictions. The first technique, simple mean, averages all image predictions per video. The second technique is fractional mean, inspired by Li and Lyu's work (2018). This aggregation method averages out the top fraction of all image predictions per video. In line with the authors,

we set our fraction to be a third. We speculate that this approach might be particularly suited to videos which have been only partially modified (for example only three seconds out of ten contain a deepfake). The third technique is a confident strategy mean, which takes the average of a subset of image predictions depending on the distribution of predictions per class: if one class overpowers the other significantly, then only the image predictions of that class are averaged to obtain a label for a given video (Seferbekov, 2020). The fourth technique uses a transformed mean (NTechLab, 2020), designed to make each image prediction more extreme. This concept is illustrated in Figure 2. These modified predictions are averaged to obtain an overall label for each video.

5.2 Results

We obtain a test accuracy of 99.29% (with fractional mean) using Xception as a convolutional base. This is in line with the reported test accuracy of 99.26% in the FaceForensics++ paper. The resulting validation accuracies for all architectures are shown in Table 1. It is clear that EfficientNet, Xception and DenseNet are the highest performing models. MobileNet, a much faster network, also performs well and is used in augmentation experiments in Section 6.1 for its lightweight nature.

Table 1: Validation accuracies using six different neural network architectures. Confident strategy and transformed means produce the same accuracies as the simple mean and, hence, are omitted.

Aggregation	**VGG16**	**Xception**	**ResNet50**	**MobileNet**	**EfficientNet**	**DenseNet**
Simple mean	0.7957	0.9929	0.8000	0.9914	0.9929	0.9929
Fractional mean	0.8000	0.9900	0.8000	0.9914	0.9914	0.9929

Other than for augmentation experiments, we adopt EfficientNet as our base from this point onwards for two reasons. Firstly, this baseline is achieved using EfficientNet-B0, a model with 5.3M parameters. We can upscale EfficientNet to B7, a model with 66M parameters outperforming B0 by 3.6% on *ImageNet*. Secondly, Kornblith et al. (2019) provide justification for the use of EfficientNets over Xception and DenseNet, all producing the same validation accuracies in our experiment. A strong correlation between *ImageNet* top-1 accuracy and transfer accuracy is suggested. EfficientNet is currently the highest performing network on *ImageNet*, and therefore we integrate its architecture into our models going forward.

While the results presented in this section are promising, they only serve as a guideline as to which architecture performs best. Higher accuracies can be expected once advanced augmentations, hyperparameter tuning and tweaked architectures are integrated. These are discussed in the Sections 6 and 8.

6 Augmentations

6.1 Information-Dropping Augmentation

6.1.1 Motivation

Convolutional neural networks have a strong tendency to overfit on training data due to their large scale and capacity to learn. Information-dropping augmentation can be leveraged to avoid overfitting, as evidenced by Seferbekov (2020) with his first placed DFDC model. We design an experiment to verify whether the inclusion of information-dropping augmentation can improve classification accuracy when used in combination with traditional augmentations such as horizontal flips, rotations and colour modifications.

6.1.2 Experiment Design

We test an established method, GridMask (Chen, 2020), and our custom implementations which include GridMask-Alternative, FacialArtifacts, FacialArtifacts-Alternative and FaceMask. While both versions of GridMask do not rely on any additional modules, FacialArtifacts uses *face_recognition* (Geitgey, 2020) to determine the position of facial landmarks. Similarly, FacialArtifacts-Alternative uses the *FaceParser* class (Shaoanlu, 2019) and FaceMask uses the *FaceParser* class as well as the *face_recognition* module to obtain required information on the location of regions of interest.

We conduct an experiment where we train six MobileNet models, all with *ImageNet* weights on the FaceForensics++ dataset (further detail on hyperparameters in Appendix). The only difference between each model is the type of information-dropping augmentation used, if any. Due to computational constraints, data is pre-processed with the information-dropping augmentation functions instead of performing real-time cutout augmentation. These functions augment images individually, allowing different parts of images, which belong to the same video, to be dropped each time.

6.1.3 Augmentation Methods

First we look at modules used to support the information-dropping augmentations and then we proceed to analyse each augmentation technique and its contribution to accuracy.

Supporting Modules When augmenting with the aid of *FaceParser*, the class is created externally and used later for each image to save time. In cases where an augmentation that requires additional module is chosen but no results from the module are obtained, the image is left unmodified; this occurs less than 0.5% of the time. The illustrations of facial landmarks obtained from *face_recognition* module and parsing map from FaceParser class are presented in Figure 3.

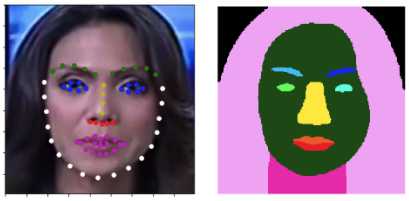

Figure 3: Supporting Modules for Information-Dropping Augmentations: facial landmarks from *face_recognition* module (left) and parsing map from *FaceParser* class (right)

GridMask For GridMask's implementation in TensorFlow, we rely on the work of Xie (2020). Unlike other information-dropping augmentations such as Cutout and Hide-and-seek, GridMask generates a grid-like mask, where preserved regions have a value of 1 and the regions that fall within the grid have a value of 0. The final step is a multiplication of the mask with an image, which produces an augmented image with a blacked out grid of squares as shown in Figure 4.

GridMask differs from Cutout, Random Erasing and Hide-and-Seek in the sense that the removal of image regions is not fully random. This is an advantage of GridMask as it is less likely to remove the entire face but more likely to remove some part of it, as required. Furthermore, unlike other methods mentioned above, GridMask cannot remove the whole object by design. The benefit of removing parts, but not all, of the object is achieving learning without memorisation. For these reasons we omit Cutout, Random Erasing and Hide-and-Seek in favour of GridMask.

Figure 4: GridMask

GridMask-Alternative GridMask-Alternative, shown in Figure 5, is our custom implementation and was inspired by Seferbekov's (2020) adaptation of original GridMask. Instead of cutting out a grid, half of the image is removed at random by cutting along the middle of one or both axes. Similarly to GridMask, GridMask-Alternative does not rely on any face-detection techniques. We hypothesise that due to our face-extraction algorithm, which produces square-shaped images with centrally positioned faces, GridMask-Alternative is extremely unlikely to cut out too much information from any given image.

Figure 5: GridMask-Alternative

FacialArtifacts FacialArtifacts, shown in Figure 6, cuts out a rectangular region over either the eyes, nose or mouth. Our only concern is that with this method we might remove too much information on regions around the eyes or mouth, which is disadvantageous when detecting facial expression manipulation methods.

Figure 6: FacialArtifacts

FacialArtifacts-Alternative This method is a refined version of FacialArtifacts; the difference is in the shape of removed region, shown in Figure 7. The hypothesised advantage of FacialArtifacts-Alternative's approach is that the removed regions provide a closer match to the facial features than the rectangles in the original version. This might be beneficial when detecting facial expression manipulations, since the important regions around the swapped facial features are still visible. Lastly, FacialArtifacts-Alternative relies on the *FaceParser* class. It is worth noting that the parsing map for each face is transposed twice and modified to make regions convex in both directions. While this is necessary for the eyes horizontally to fill in the gaps between the left and right eyes, it can be omitted to some extent if the approach becomes too computationally expensive and slow.

Figure 7: FacialArtifacts-Alternative

FaceMask FaceMask augmentation is implemented from scratch. The augmentation is performed by removing a left, right, top or bottom half of the face mask for a given image, demon-

strated in Figure 8. The cutting position is determined by the location of the nose: vertical cuts are performed along the average x value for nose bridge coordinates, while the horizontal cut is done along the y value for the lowest nose bridge coordinate. The outer perimeter of the face is obtained using the corresponding parsing map. We include this augmentation type as we feel it might prevent overfitting but keep sufficient relevant information by preserving parts of the face and all near-facial regions (narrow spaces around the face). We suppose that while this augmentation would be suited to facial expression manipulation detection, it would be particularly useful in detecting facial identity manipulations.

Figure 8: FaceMask

6.1.4 Results

Table 2: Validation accuracies using information-dropping augmentations.

Augmentation	Validation accuracy (on video dataset)			
	Simple mean	Fractional mean	Confident strategy	Transformed mean
GridMask	0.9914	0.9914	0.9914	0.9914
GridMask-Alt.	0.9929	0.9929	0.9929	0.9929
FacialArtifacts	0.9914	0.9900	0.9914	0.9914
FacialArtifacts-Alt.	0.9929	0.9900	0.9929	0.9929
FaceMask	0.9929	0.9900	0.9929	0.9929
No information-dropping aug.	0.9929	0.9914	0.9929	0.9929

As mentioned earlier, six models were trained with the same hyperparameters using all five augmentation techniques discussed above, as well as no augmentation. The validation accuracy values for videos, obtained with the four different aggregation methods discussed in Section 5, are presented in Table 2 for each of the six models.

Firstly, it is important to consider the image to video prediction aggregations and their relative performance to be able to draw conclusions on an augmentation methods' effectiveness for our validation video dataset. Given these data and model characteristics, a confident strategy and transformed mean yield the same results as the simple mean aggregation. Potentially, the confident strategy fraction parameter is set too high and instead a simple mean is calculated. Also, it is possible that the majority of image predictions are close to the extreme end values and hence the transformed mean aggregation has no effect. Additionally, it is important to note

that the highest validation accuracy on images is already very high for each trained model. This leaves less opportunity for the confident strategy and transformed mean to demonstrate an advantage over the other two aggregation methods. In conclusion, these two approaches should still be used among the other two for later experiments. However, we analyse the effectiveness of the five information-dropping augmentations in this section focusing on simple and fraction means only for the test accuracies.

The comparison of simple mean and fractional mean aggregations clearly shows that for our setting, the former is more, or at least as, accurate as the latter. Hence, we believe it is sufficient to use only the validation accuracies from simple mean aggregation when comparing information-dropping augmentation performance.

Proceeding to augmentations, the highest validation accuracy of 99.29% is achieved by models trained with GridMask-Alternative, FacialArtifacts-Alternative, FaceMask or without an information-dropping method. This accuracy is marginally better than the 99.14%, obtained by the other two methods. Since inclusion of either information-dropping augmentation does not show any improvement in validation accuracy over its exclusion, we believe that the results do not present significant evidence of potential test accuracy advancement which any of our five information-dropping augmentations could contribute to the later models. While we are aware of the significant advantage in terms of prevention of overfitting which information-dropping augmentation usually brings, we do not observe such a phenomenon on the FaceForensics++ validation dataset. Potentially, the similarity between training and validation sets causes the model, which is trained with no complex augmentation, to memorise the training data more closely, resulting in higher validation accuracy. Also, as mentioned earlier, the combination of our model architecture and hyperparameters with the FaceForensics++ dataset already produces very high validation accuracies, hence making it difficult for information-dropping augmentation to show significant improvement.

Overall, we choose not to experiment with these five information-dropping augmentations further. Firstly, inclusion of any of these methods does not show strong indication of potential test accuracy increase. Secondly, incorporating these augmentations further requires a lot of additional computational resources to fine-tune the proportion of images that get augmented. Hence, we turn to investigate augmentation optimisation to improve our models.

6.2 DeepAugment

While augmentations undoubtedly improve model accuracy and generalisation, finding optimal augmentations and accompanying hyperparameters is not trivial. Cubuk et al. (2018) provide a solution – AutoAugment, an automated augmentation selector. Although Chen (2020) found that GridMask outperforms AutoAugment's optimal policies, the FaceForensics++ dataset was

not included in Chen's 2020 study and the results in Table 2 suggest that there is room for accuracy improvement with an automated augmentation selector.

6.2.1 Theory

AutoAugment consists of two parts, a controller and a child model. The controller's task is to select the best performing augmentation policy and pass correspondingly augmented data to the child model for training. Reinforcement learning is leveraged to update the controller after each iteration using the child model's validation accuracy as a reward. The Proximal Policy Optimization algorithm is used as the gradient method to update the controller (Schulman et al., 2017). Finally after a given number of iterations, AutoAugment classifies the policy which yields the highest validation accuracy as optimal.

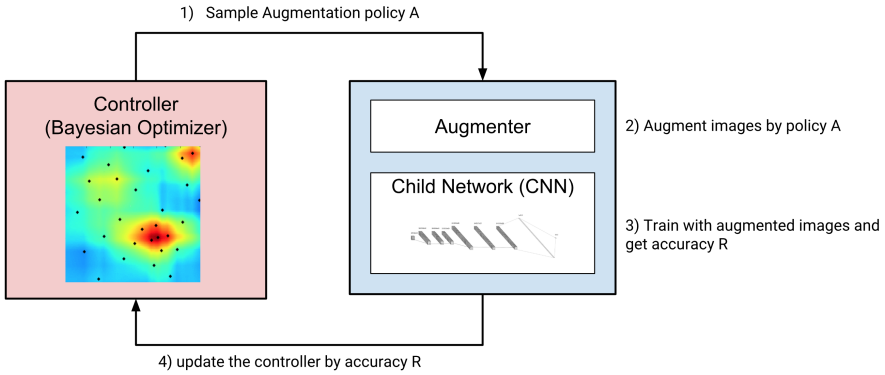

Figure 9: Tailored DeepAugment: A controller samples an augmentation policy, the augmenter transforms the images and the child model is trained using the augmented data. The average of the highest three validation accuracies is returned to the controller as reward and the policy is updated by a Bayesian optimiser which samples new policies using a Random Forest Estimator and Expected Improvement acquisition function. We run this procedure for 100 iterations.

While AutoAugment shows promising results, the reinforcement learning component requires 15,000 iterations and excessive computational power. We overcome this by tailoring a different version of AutoAugment, DeepAugment (Ozmen, 2019), to our dataset. Our tailored DeepAugment follows the same procedure as AutoAugment with two notable differences: firstly, reinforcement learning is replaced with Bayesian optimisation requiring significantly fewer iterations. Secondly, a minimised child model is used to reduce the number of trainable parameters, speeding up each iteration. Fewer and faster iterations lead to our tailored DeepAugment running up to 150 times faster than AutoAugment. The full process is illustrated in Figure 9.

6.2.2 Results

To identify augmentation strategies which suited our FaceForensics++ dataset best, we run our modified DeepAugment on a child EfficientNet-B0 and obtain an optimal policy shown in Table

3. Similar to AutoAugment, DeepAugment's optimal policy consists of multiple sub-policies, each containing two image transformations from the *imgaug* library (Jung et al., 2020) and corresponding magnitudes.

Table 3: DeepAugment's suggested optimal policy, which consists of five subpolicies (A, B, C, D and E). Each subpolicy has two augmentations applied consequently with specified magnitudes.

Subpolicy	First augmentation		Second augmentation	
	Type	Mag.	Type	Mag.
A	*salt and pepper*	0.902	*rotate*	0.97
B	*horizontal flip*	0.171	*additive gaussian noise*	0.751
C	*salt and pepper*	0.325	*crop*	0.634
D	*elastic transform*	0.653	*salt and pepper*	0.995
E	*hue and saturation*	0.414	*gamma contrast*	0.624

It is interesting to note that both colour-based augmentations present in Table 3, *hue and saturation* and *gamma contrast*, only occur once each and both in subpolicy E. No other sub-policy includes any colour-based augmentations. Instead, each of the other four sub-policies consists of one noise-based (*salt and pepper* or *additive gaussian blur*) and one geometry-based (*rotate*, *horizontal flip*, *crop* or *elastic transform*) augmentation.

Finally, visualisation of the optimal policy demonstrates that DeepAugment requires more computational resources to fully calibrate suggested magnitudes. Hence, we adopt these augmentations but choose to incorporate some intuition into building real-time augmentation for training data by tweaking the magnitudes and frequency of their application.

6.3 Final Augmentation Setup

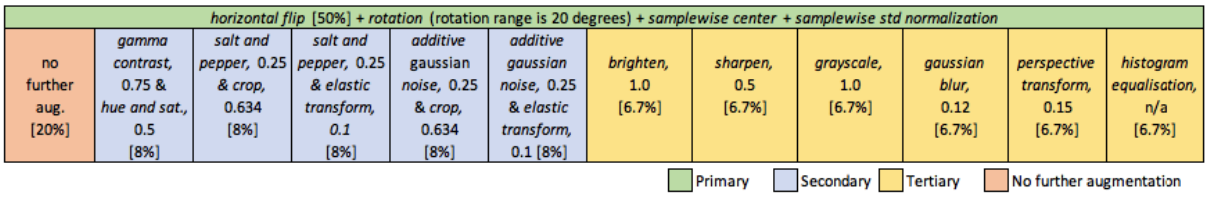

Figure 10: Final augmentation setup with three levels of augmentations. Primary augmentations (in green) are applied to all data, this is represented by the cell occupying all of the top row. Meanwhile secondary (in blue) and tertiary augmentations (in yellow) are performed mutually exclusively (each occupying a fraction of the bottom row). Each image transformation is specified in a format (*type*, magnitude) and probabilities (%), are recorded in brackets for each cell.

For the final augmentation setup (Figure 10), we create three levels of augmentation: primary, secondary and tertiary, classified by frequency of their application. While primary augmentations are applied by ImageDataGenerator on all data, secondary and tertiary augmentations are mutually exclusive and are performed by its *preprocessing_function*. Secondary and tertiary

groups each augment 40% of the data at random. The augmentations are displayed in Figures 11 (geometry-based), 12 (noise-based) and 13 (colour-based).

Our primary augmentations are *rotate* and *horizontal flip*, both geometry-based and available as part of ImageDataGenerator. The probability is set to be 50% for *horizontal flip* and rotation magnitudes are random with a maximum value of 20 degrees. Secondary augmentations are inspired by DeepAugment's suggested optimal policy in Table 3. We preserve the combination of colour-based augmentations of subpolicy E, while augmentations from the other four subpolicies are recombined to contain one augmentation from each group: *salt and pepper* or *additive gaussian blur* from noise-based, and *crop* or *elastic transform* from geometry-based. The resulting five subpolicies seen in blue in Figure 10 are executed mutually exclusively with equal probability of 8%. Finally, we include six additional augmentations, not chosen by DeepAugment, since models tend to generalise better if exposed to a wider variety of augmentations. These six image transformations become our tertiary augmentations: *gaussian blur*, *sharpen*, *brighten*, *all channel histogram equalization*, *perspective transform* and *grayscale*. These tertiary augmentations are applied mutually exclusively with each other and secondary augmentations, with 6.7% chance.

Figure 11: Geometry-Based Augmentations

Figure 12: Noise-Based Augmentations

Figure 13: Colour-Based Augmentations

7 Simple Features

Almost all simple methods of forgery detection have limited application to the domain of deepfakes, as these images are generated in such a sophisticated fashion that detecting these alterations requires increasingly sophisticated techniques. However, deepfakes generally still exhibit artifacts in their frequency domain that allow for differentiation between authentic and fake images using a simple method. This simple method is essentially a heavy augmentation and builds further upon the work on augmentations in Section 6. Durall et al. (2019) rely on a classical frequency analysis of images that reveals different behaviours at particular frequencies. Approaching deepfake detection without CNNs can be helpful since most CNN methods can be learnt by a GAN, that is, when a deep learning method is incorporated in the GAN's discriminator, the generator can be fine-tuned to learn a countermeasure for any differentiable forensic.

7.1 Theory

The theory of simple features is grounded in frequency domain analysis, which stems from signal processing. The image transformation consists of the spectral decomposition of the input data, which shows how an image's information is distributed over a range of frequencies (Durall et al., 2019). The transformation of an image has two parts. The first is a discrete Fourier transform. After this transformation is applied, the image is in a new domain but within the same dimensionality, that is 2D. The second part is an Azimuthal averaging applied such that a robust 1D representation of the Fourier Transform spectrum is created. Azimuthal averaging can be understood as compressing, gathering and averaging similar frequency components into a vector of features. The amount of features is reduced as such without losing relevant information.

Real and fake images display similar behaviour along the spatial frequencies, yet there is a clear offset between the real and fakes that allows the images to be classified. Generally, the spectral curve of fake images is just below that of real images. This algorithm is limited in the sense that it works well on medium to high-resolution images, but performs subpar in the low-resolution image domain. This limits its application in the real world, as many 'in-the-wild' deepfakes are compressed and downscaled to a lower resolution.

7.2 Experiment Design

The pipeline involves pre-processing the input image into its grayscale version, extracting the features using Discrete Fourier Transform and Azimuth averaging the 2D Amplitude spectrum, after which a 1D Amplitude spectrum is obtained, either of length 300 or 722, as inspired by Durall et al. (2019). The image has to be grayscaled first, as otherwise it contains 3 channels and it is not a 2D input. The classifier comprises 4 layers, with one 1024, one 512 and one 128-wide fully-connected layer, after which a dropout-layer follows. The network is optimised

using Adam with a learning rate of 0.005 and a decay of 10^{-6}. The pipeline is shown in Figure 14.

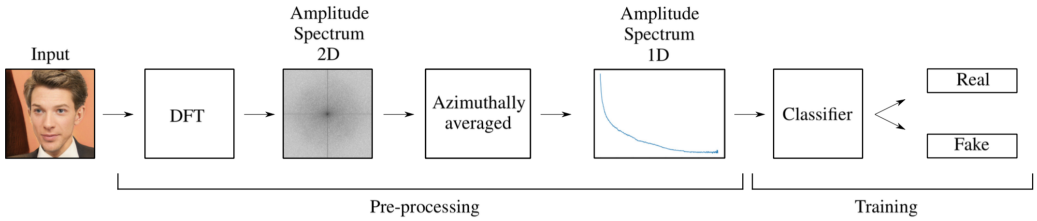

Figure 14: Simple Features Pipeline (Durall et al., 2019)

All 113,928 training images are converted to their 1D power spectrum of length 722 or 300, likewise for all 21,291 validation and 22,964 test images.

7.3 Results

The classifiers achieve an accuracy of at most 87.00% for the video classification task on the FaceForensics++ validation set. Detailed results are presented in Table 4.

Table 4: Video validation accuracies using simple features.

Manipulation method	300 features		722 features	
	Simple mean	Fractional mean	Simple mean	Fractional mean
Deepfakes	0.9714	1.0000	0.9500	0.9932
Face2Face	0.5143	0.8143	0.4576	0.7717
FaceSwap	1.0000	1.0000	0.9862	1.0000
NeuralTextures	0.8143	0.9429	0.6935	0.8644
Original	0.8786	0.5929	0.8798	0.7077
Overall	0.8357	0.8700	0.7923	0.8679

The 300-wide spectrum outperforms the 722-wide spectrum, which is unexpected as more features should encapsulate more information. Furthermore, FaceSwap and DeepFakes are picked up easily using simple features, while these features perform poorly on NeuralTextures and Face2Face. In fact, using a simple mean aggregation leads to a detection accuracy equivalent to guessing. The algorithm picks up Face2Face swaps only when using a fractional mean. Picking fractional instead of simple mean works well, except for authentic images, which, in the best model, are only detected accurately in 59.29% of the cases.

Durall et al. (2019) achieved a 91% video detection accuracy on a test set with a Support Vector Machine. These results are of limited value as their test set only contains 400 images, while ours contains 22,964. Therefore, their results are not very robust. With a maximum detection accuracy of 87.00% we drop this method and do not incorporate it in the final model.

8 Hyperparameter Selection

As mentioned in Section 2, hyperparameters are values set before training commences and play a major role in model accuracy. This section discusses the hyperparameter tuning techniques we employed, the results of these techniques and the impact of influential hyperparameters on model accuracy. This section also includes suggested values for the random search.

8.1 Hyperparameter Tuning

To obtain the optimal hyperparameter combination, we adopt the random search strategy in light of the discoveries described in the sections that follow. Our selection is motivated by past research, which shows that when compared with a pure grid search over the same domain, random search is capable of obtaining equivalent or superior models in a fraction of the computational time (Bergstra and Bengio, 2012). Bergstra and Bengio (2012) suggest the reason behind this is that only a few hyperparameters have a significant effect on results, and so a grid search that tests all combinations is a poor configuration algorithm – it spends too much time testing unpromising regions of the search space. Mathematically this can be explained by the highly dimensional grid search domain having a low effective dimensionality. While a grid search provides better coverage in higher dimensional spaces, the random search provides better coverage in the lower dimensional projections, some of which make up the important parameters. Figure 15 illustrates how a random search tests larger ranges of the important hyperparameter values. The sections that follow analyse individual hyperparameters to determine which values should be included in the grid search space. Results from the random search are presented at the end of this section in Table 5.

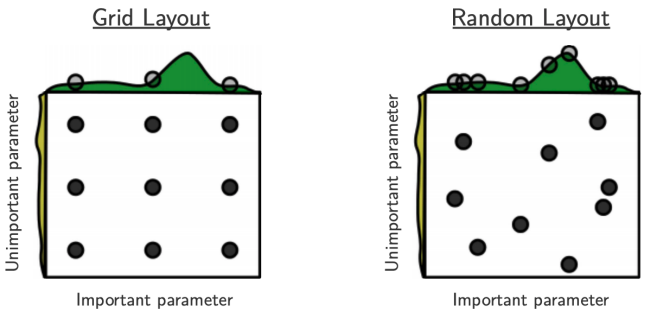

Figure 15: Nine trials for a grid and random search to optimise a function with low effective dimensionality or few important parameters. The random search explores nine distinct values of the important parameter whereas the grid search explores three (Bergstra and Bengio, 2012).

8.2 Optimisation Algorithm

The network optimiser, and specifically the learning rate, is widely accepted as the most important hyperparameter (Bengio, 2012; Goodfellow et al., 2016). Its importance arises from the notion that small values can take too long to train and might never reach optimal results whereas large learning rates can lead to divergent oscillations (Bishop et al., 1995). This section discusses

the experiments and methods we employed to attain the optimal learning rate, optimiser type, momentum and rate schedules in order to attain an optimiser yielding the highest results.

8.2.1 Learning Rate

Following the advice of Goodfellow et al. (2016) we test five learning rate values (0.1, 0.01, 0.001, 10^{-4}, 10^{-5}). We leverage transfer learning and drop the learning rate by a factor of 10 during the fine-tuning phase. We experiment with these five learning rates using an SGD optimiser, and two different standard values for Nesterov's momentum (0.9 and 0.99), training ten different models until convergence. We train these ten models for 30 epochs with a frozen base followed by 30 epochs of training with an unfrozen base. The results of this experiment are reported in Figure 16.

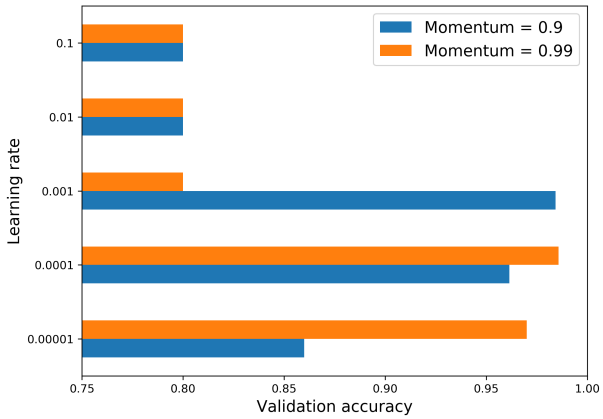

Figure 16: Video validation accuracy for five logarithmically spaced learning rates

Learning rates of 0.1 and 0.01 are too high for training an EfficientNet-B0, as these networks achieve precisely a 80.00% validation accuracy, meaning they predict that every image is fake, as 80% of the validation set consists of fake images. A learning rate of 0.001 achieves the highest accuracy combined with a momentum of 0.9, but achieves a low accuracy with a momentum of 0.99. Validation accuracies for 10^{-4} and 10^{-5} are also acceptable. For this reason, we include all learning rates in the random search, but place much larger probabilities on these three smallest learning rates as they are expected to lead to superior results.

8.2.2 Optimiser Type

We consider two optimisers in the random search: Adam (Kingma and Ba, 2014) and stochastic gradient descent, also known as SGD (Kiefer et al., 1952). The former is an adaptive learning algorithm which monitors model performance on the training dataset and adjusts the learning rate accordingly. While adaptive learning algorithms are well suited to fast results with very little tuning required, Karpathy (2019) suggests that well-tuned SGD optimisers almost always outperform Adam.

8.2.3 Learning Rate Schedule

During training, the random initialisation of weights typically results in these weights being far off the optimal solution. For this reason, a large learning rate can lead to numerical instability in the early stages of training. We found that a sudden increase from a constant low learning rate to the initial learning rate leads to results with a significant amount of instability. To stabilise the process, we include a linearly increasing warmup strategy in the random search, proposed by Goyal et al. (2017), which starts at 0 and increases linearly after each batch until it reaches the initial learning rate parameter by the specified epoch.

Once the learning rate has been warmed up to its initial value, we implement a decaying learning rate to improve performance. Traditionally, exponential and step decays proposed by Szegedy et al. (2016) have been used during training. Instead, we experiment with the cosine annealing strategy proposed by Loshchilov and Hutter (2016):

$$\eta_t = \frac{1}{2}\left(1 + cos(\frac{t\pi}{T})\right) \cdot \eta \tag{1}$$

where η is the initial learning rate, t is the current batch, η_t is the learning rate for the current batch and T is the total number of batches to be used for training after the warmup period.

The primary difference between cosine and step decays is that the former leads to a higher learning rate during the later epochs. We believe that cosine decay's superior performance can be explained by the fact that substantial accuracy improvements are still possible during the later epochs and these gains are not realised by a very low learning rate. The cosine annealing strategy's smooth, gradually decreasing nature ensures these gains are attained. In addition to this cosine annealing strategy, we also include a constant learning rate in the random search, as suggested by Karpathy (2019).

8.2.4 Momentum

As mentioned in Section 8.2.2, SGD performs well provided its hyperparameters are well-tuned. This is particularly true in the case of momentum, where the correct parameter choice is central to dealing with curvature issues on the loss surface (Sutskever et al., 2013). We include three momentum values in the random search: 0, 0.9 and 0.99.

8.3 Batch Size

The use of mini-batches to train neural networks is a common theme among all high-performing models, as they allow for an increased parallelism and lower communication costs. Small batch sizes have also been shown to exhibit better generalisation to unseen data as well as faster optimisation convergence (Wilson and Martinez, 2003; LeCun et al., 2012; Keskar et al., 2016). One possible reason for the former advantage is that most loss surfaces of deep neural networks are fraught with local minimisers (Choromanska et al., 2015) and the low noise convergence with

large batches is incapable of escaping sharp local minima. Contrastingly, the high noise from small batches pushes the network out of these sharp local minima, obtaining a lower, smoother point on the loss surface and, hence, leading to better generalisation. Additionally, smaller batch sizes have a significantly lower memory requirement. On the other hand, large batch sizes allow for increased parallelism during the training process by improving efficiency on each processor as well as scaling up processing to use more nodes (Dean et al., 2012; Das et al., 2016).

There is no universally accepted batch size proven to be effective in all scenarios. Finding the optimal value is often a key factor in realising substantial accuracy improvements since batch size does not only affect cost and efficiency considerations, but can also heavily impact obtained results. To investigate which batch sizes should be included in our random search, we train four EfficientNet-B0s varying the batch size (8, 16, 32, 256). The results, shown in Figure 17, display a negative correlation: a batch size of 256 achieves the lowest accuracy (98.57%), while other batches all obtain accuracies 0.43% higher, using a simple mean aggregation. These results are in line with the findings of Masters and Luschi (2018), who argue that large batch sizes lead to a small range of learning rates that provide stable convergence and high test accuracy. Furthermore, according to Masters and Luschi (2018), the frequent gradient updates of smaller batches (less than 32) allow for greater stability and more margin for error when tuning learning rate values. Smaller batch sizes can be thought of as enforcing stronger regularisation because the empirical batch mean and standard deviations are approximate or wiggly versions of the full mean and standard deviations. In addition, this 'wiggliness' often forces the network out of local minima, reducing the likelihood of obtaining sub-optimal weights.

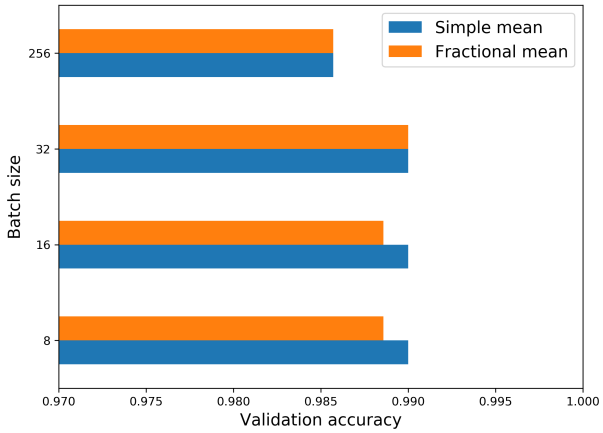

Figure 17: Video validation accuracies for the batch sizes of 8, 16, 32 and 256

Encouraged by our results for smaller batch sizes and Masters and Luschi's 2018 findings, we include an even smaller batch size of 4 in the random search. We assign probabilities of 0.15, 0.3, 0.25, 0.25 and 0.05 to batch sizes 4, 8, 16, 32 and 256 respectively, to favour smaller batch size

options, in line with the findings above.

8.4 Label Smoothing Regularisation

In the past, supervised methods typically used training data where label assignments followed a hard allocation. In other words, training samples are assigned to a particular class with full certainty. A common hurdle stemming from hard label assignment is the model's high level of training prediction confidence, resulting in overfitting and poor generalisation to different datasets. A useful solution to implement are soft label assignments, where the positive class is assigned the largest probability and all other classes are given a probability slightly larger than zero. This technique was first proposed by Szegedy et al. (2016) who modify cross entropy loss by replacing the hard label assignments with a weighted mixture of these hard assignments and a uniform distribution.

Below we give a brief mathematical motivation for the use of label smoothing. Let $(x_i, y_i), i \in 1, ..., n$ be samples and their corresponding labels from the training dataset. Let p be the ground truth distribution over labels which we have and q be the predicted label distribution that the model will train. Assuming we use cross-entropy loss to obtain the model parameters, we aim to minimise the loss function given in Equation 2.

$$L = -\sum_{i=1}^{n}\sum_{y=1}^{2} p(y|x_i) \cdot \log q(y|x_i) \tag{2}$$

Using the probability distribution in Equation 3 (hard allocation), we show the loss simplifies to Equation 4.

$$p(y|x_i) = \begin{cases} 1 & \text{if } y = y_i \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$L = -\sum_{i=1}^{n}\sum_{y=1}^{2} p(y|x_i) \log q(y|x_i) = -\sum_{i=1}^{n} p(y_i|x_i) \log q(y_i|x_i) = -\sum_{i=1}^{n} \log q(y_i|x_i) \tag{4}$$

The function in Equation 4 can easily be minimised to achieve a loss of almost 0 thereby predicting training data perfectly. This can result in severe overfitting.

Contrastingly, if we manipulate the ground truth distribution p by introducing a small positive number, ε, as well as a uniform noise distribution $u(y|x)$, we obtain a new ground truth label for sample x_i, shown in Equation 5.

$$
\begin{aligned}
p'(y|x_i) &= (1-\varepsilon)p(y|x_i) + \varepsilon u(y|x_i) \\
&= \begin{cases} 1-\varepsilon+\varepsilon u(y|x_i) & \text{if } y = y_i \\ \varepsilon u(y|x_i) & \text{otherwise} \end{cases}
\end{aligned} \tag{5}
$$

Substituting Equation 5 into the loss function from Equation 2 we get:

$$
\begin{aligned}
L' &= -\sum_{i=1}^{n}\sum_{y=1}^{2} p'(y|x_i)\log q(y|x_i) \\
&= -\sum_{i=1}^{n}\sum_{y=1}^{2} \Big[(1-\varepsilon)p(y|x_i) + \varepsilon u(y|x_i)\Big]\log q(y|x_i) \\
&= \sum_{i=1}^{n}\Bigg\{(1-\varepsilon)\Big[-\sum_{y=1}^{2} p(y|x_i)\log q(y|x_i)\Big] + \varepsilon\Big[-\sum_{y=1}^{2} u(y|x_i)\log q(y|x_i)\Big]\Bigg\} \\
&= -\sum_{i=1}^{n}\Big[(1-\varepsilon)CE_i(p,q) + \varepsilon CE_i(u,q)\Big]
\end{aligned} \tag{6}
$$

Equation 6 proves that by adopting a soft label assignment approach, the loss function becomes the sum of the cross entropy between the hard label distribution and the prediction distribution, $CE_i(p,q)$, and the cross entropy between the uniform distribution and the prediction distribution, $CE_i(u,q)$. The primary difference between this loss and the loss from Equation 4 is the additional $\varepsilon CE_i(u,q)$ term which increases dramatically with high values of prediction confidence on the training dataset. This additional penalty prevents overfitting.

Put simply, the above motivates the use of label smoothing as a form of regularisation. We integrate three values of label smoothing into our random search (0, 0.01 and 0.05) with the aim of achieving better generalisation to test datasets as well as increased model calibration.

The above-mentioned model confidence calibration is an additional benefit of label smoothing. Confidence works as follows: given 100 predictions, each with confidence of 0.7, we expect that 70 should be correctly classified. A model's confidence calibration defines how the asserted model's confidence reflects its true correctness likelihood. Modern networks are not particularly good at achieving high levels of confidence calibration (Guo et al., 2017), and therefore label smoothing provides one possible solution. Müller et al. (2019) demonstrate that label smoothing implicitly allows for better alignment between prediction confidence and prediction accuracy. Hence, label smoothing improves the networks ability to 'know when it doesn't know' and significantly increases ensemble performance. This is particularly important when constructing the final model in Section 10 where ensembles are used.

8.5 Additional Hyperparameters and Considerations

Weight Initialisation The computational and time complexities pose a major obstacle when training neural networks on large datasets. Large architectures can take months to reach acceptable levels of convergence. Pratt et al. (1991) provide a solution in the form of transfer learning – a technique that significantly reduces training time at a very small accuracy cost. Transfer learning involves extracting the features learned by a trained neural network, typically over several months of training, and leveraging these features for a different task. This is particularly useful when applied to computer vision as the convolutional layers store a variety of commonly occurring features – edges, corners, faces etc. While transfer learning still performs with high accuracy when the datasets are different, the best results are achieved when the image content of both datasets are of a similar nature.

Typically, transfer learning follows the workflow below (Chollet, 2017a):

1. Extract weights from a trained model.
2. Freeze these weights so they are not updated during training.
3. Add some dense layers on top of the frozen base – only these weights update during the training.
4. Fine-tune the model: Once the added layers are trained to convergence, unfreeze the base and re-train using a very low learning rate.

For the random search, we use transfer learning for all experiments due to computational constraints. We train all models using an EfficientNet-B0 architecture initalised with noisy-student weights, shown to produce more precise results than *ImageNet* initialisations (Xie et al., 2020).

Weight Decay Following the advice of Karpathy (2019), we include weight decay as a parameter in the random search. However, instead of regularising weights and bias, we only decay weights to avoid overfitting as suggested by Jia et al. (2018).

Dropout Large neural networks can be prone to overfitting. We control this tendency using dropout, where random units are dropped during training to prevent co-adaptation of units. We also take into account that more complex models require a larger dropout parameter (Srivastava et al., 2014). Values of 0, 0.2 and 0.5 are included in the random search.

Class Weights The FaceForensics++ dataset is imbalanced containing four manipulated videos for each authentic one. This data imbalance builds bias into the trained model leading to an

inclination to predict test videos as fake most of the time. We introduce class weights to combat this bias. These class weights adjust the loss function by placing a heavier penalty on incorrect minority class predictions. This ensures the data imbalance has a negligible effect on model accuracy and generalisation.

8.6 Results

We present the results of the random search in Table 5. The random search is conducted using the hyperparameter considerations discussed throughout this section. We train fifty end to end models until convergence. Combinations that produced the highest validation accuracies are included in Table 5. These nine configurations will be used when training the final model ensemble, discussed in Section 10.

Table 5: Top Performing Hyperparameter Configurations for Random Search

	Batch	**LR type**	**LR**	**Decay**	**Optimiser (Momentum)**	**Label S.**	**Dropout**	**Warm-up**	**Val acc**
1	8	cosine	0.0001	0.01	Adam	0.05	0.5	0	0.9900
2	8	const.	0.00001	0.01	Adam	0.01	0.2	2	0.9871
3	8	const.	0.00001	0.0001	Adam	0.05	0.2	2	0.9800
4	16	cosine	0.00001	0	Adam	0.05	0.5	0	0.9886
5	16	cosine	0.00001	0.01	Adam	0	0.2	4	0.9800
6	16	const.	0.001	0.01	SGD (0)	0.01	0	4	0.9886
7	16	const.	0.00001	0	SGD (0.99)	0.01	0.5	4	0.9857
8	32	cosine	0.0001	0.0001	SGD (0.9)	0	0	4	0.9814
9	32	cosine	0.00001	0.0001	SGD (0.99)	0.01	0.2	4	0.9900

9 LSTM and Transformer

There are multiple inherent weaknesses in the generation of deepfake videos. A CNN attempts to exploit the differences between deepfakes and authentic images on a frame-basis. More often than not, a well-trained CNN should be able to distinguish between the nature of both entities. However, there is more relevant information that a model could process to produce an even better distinction between these two categories. An important weakness of deepfakes that is not exploited when using frame-level CNNs is inherent to the generation process of the final deepfake video itself. In the deepfake generation process, the auto-encoder is used on a frame-by-frame basis, and as a consequence it is unaware of any previous or future frames that are generated. Such a lack of temporal awareness can cause several anomalies. One such anomaly is a phenomenon that causes flickering in the face region that is common to a large proportion of deepfake videos, caused by different lighting in the source and target videos (Güera and Delp, 2018).

To incorporate this extra information, we have created two different temporal-aware pipelines, which first extract frame-level features, and then feed these features into either a Long Short-Term Memory (LSTM) unit (Hochreiter and Schmidhuber, 1997) or Transformer blocks (Vaswani et al., 2017) which classifies 20 frames as a collection rather than individually. Essentially, the CNN is used for frame feature extraction and the LSTM or Transformer for temporal sequence analysis.

9.1 Theory

9.1.1 LSTM

LSTM networks are a particular type of RNN, first introduced by Hochreiter and Schmidhuber (1997). RNNs perform well in situations where a network needs to learn short to medium-term dependencies in data sequences. LSTMs also perform well at both these tasks, but are additionally equipped to learn long-term dependencies. LSTMs and CNNs go well together for video tasks. A convolutional LSTM has multiple properties that make it suitable for such tasks, namely that it performs well at visual inference, can learn long-term temporal information, and can be trained in an end-to-end fashion. Such convolutional LSTM architectures have been studied in detail for activity recognition or object detection in videos (Donahue et al., 2015), and this combination has advanced our performance at those tasks significantly.

9.1.2 Transformer

In 2017, Vaswani et al. proposed a new type of neural network to process sequential data. This architecture forgoes recurrence and instead relies entirely on an attention mechanism to identify global dependencies between input and output. This model achieved a new state-of-the-art in translation quality after being trained for only half a day, outperforming models that had been trained for several days. The mechanism underpinning this architecture is called self-

attention, which relates different positions of a single sequence to compute a representation of that sequence.

As with RNNs, Transformers consist of an encoder and decoder. The encoder maps an input sequence to a continuous representation $z = (z_1, \ldots, z_n)$. Given z, an output sequence $(y_1, \ldots, y_m)$ is constructed by the decoder one element at a time. The model is auto-regressive at each step, taking all previously generated symbols as additional input.

The Transformer's essential function is the attention function, which maps a query and a set of key-value pairs to an output. In this situation the query, keys, values, and outputs are all vectors. The output is a weighted sum of all these values, where the weights are computed and upgraded in each step of the network's training. The attention function is computed on a set of queries simultaneously. All of these are in a matrix Q. The values and keys of dimension d_k are all combined into matrices V and K respectively. The matrix of outputs is then computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Combining multiple attention functions creates a multi-head attention, which concatenates the output of a number of attention functions. In its original design, 8 parallel attention layers are combined into one multi-head attention.

The encoder contains 6 identical layers, each containing two sub-layers. The first is a multi-head self-attention mechanism, the second is a simple feed-forward network. The decoder also contains 6 identical layers and is very similar in design. However, in addition to the two sub-layers in the encoders, the decoder adds a third sub-layer, which performs multi-head attention over the output of the encoder stack. The entire process is demonstrated in Figure 18.

RNNs (including LSTMs) have an inherently sequential nature that precludes the possibility of parallelising the computational process. Parallelisation becomes essential at longer sequence lengths, as otherwise memory constraints prevent the learning of truly long-term dependencies. RNNs perform worse than the Transformer for three reasons. Firstly, the computational complexity per layer is smaller for the Transformer. Secondly, the Transformer allows for strong parallelisation. Thirdly, the Transformer can learn longer-range dependencies more easily than RNNs can. The third point is not relevant for the current application as the model we implement only works with sequences of 20 in length.

Although convolutional LSTMs have been studied thoroughly, we found no implementation of a convolutional Transformer in the literature, making this implementation novel.

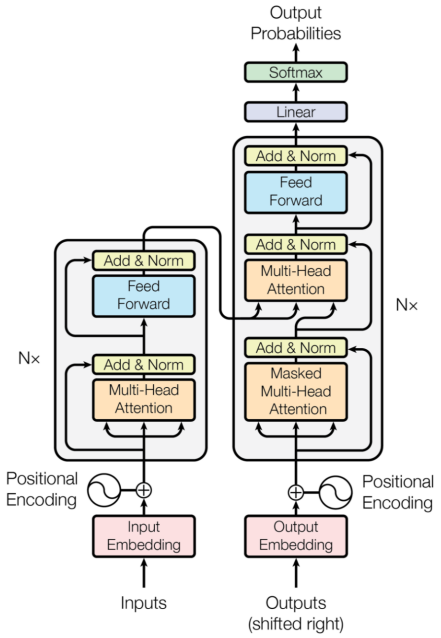

Figure 18: The Transformer (Vaswani et al., 2017)

9.2 Experiment Design

For each video, the first twenty extracted frames are selected. If the face-extraction script did not extract twenty frames, the missing frames are chosen starting from the first frame in consecutive order. The CNN extracts features for each frame, and these features are then concatenated and passed to an LSTM or Transformer.

Inspired by its success in the DFDC and its winning entry in the *ImageNet* competition, and as in the rest of this paper, we employ an EfficientNet-B0 architecture. The top of this model is removed, which gives the model an output of shape $(7, 7, 1280)$, a height and width of 7, and 1280 channels. This convolutional base is kept unfrozen throughout the training process, inspired by Sabir et al. (2019). Using a 1D Global Average Pooling layer, the output of the convolutional base is transformed into a vector of shape 1280. These vectors are then, for the first experiment, passed into an LSTM layer, and for the second experiment into a Transformer model. After this transformation, a dropout layer is added, after which two fully-connected layers are added, of width 512 and 128 respectively. After these two layers, another dropout layer is added after which the output is finally passed to a layer with two neurons with a softmax-activation. This softmax layer computes the probability of the sequence of frames being either authentic or fake. Neither the LSTM nor the Transformer are in need of auxiliary loss functions, as they are employed within the model. The pipeline is shown in Figure 19.

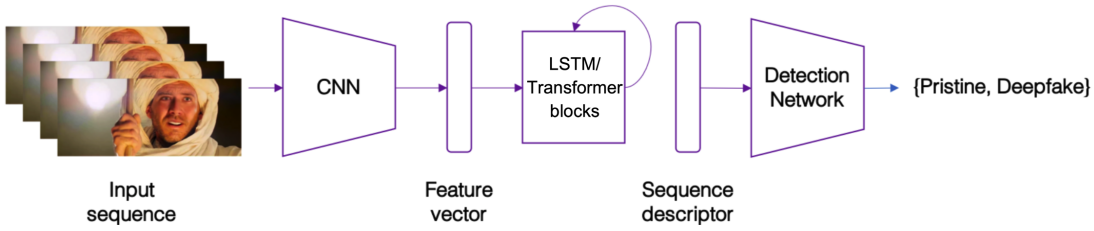

Figure 19: Pipeline LSTM/Transformer

9.3 Results

The results are presented in Table 6.

Table 6: Temporal Model Results

	LSTM			**Transformer**		
	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6
Structure	512-wide	2048-wide	2048-wide	One-layer	Two-layer	Six-layer
# params	6.2M	32.4M	32.4M	7.6 M	20.5M	52M
Loss	0.0943	0.1211	0.0907	0.0423	0.0561	0.1212
Accuracy	0.9743	0.9814	0.9814	0.9914	0.9886	0.9800

For the LSTM, changing the model parameter settings did not affect the results strongly. Switching to EfficientNet-B0 from MobileNet leads to a slight increase in validation accuracy (compare Model 2 and 3 to 1). Furthermore, a 2048-wide LSTM outperforms a 512-wide LSTM in terms of validation accuracy. For validation loss, however, these results do not hold as the validation loss of Model 1 is smaller than that of Model 2.

For the Transformer model, a smaller model appears to outperform larger models. With a growing number of parameters, the validation accuracy only decreases while the validation loss increases. Model 4 achieves a validation accuracy of 99.14%.

The two simpler Transformers outperform the LSTM models in terms of validation accuracy and loss. The most complex Transformer model (Model 6), however, did not achieve impressive results. A comparison to previous literature is hard to make, as RNNs have been implemented in the case of deepfakes but on a wholly different dataset. Furthermore, Transformers have, as far as we know, yet to be combined with CNNs. The results are promising. These model architectures perform just about as well as the CNNs from Section 8 on the validation set. However, their generalisation ability might be larger than that of the simpler models. We choose not to include Model 4 in the final ensemble due to the structure of the data needed for the CNNs in the ensemble.

10 Final Model

This section presents our final model design and proceeds to compare its performance against other networks on the FaceForensics++ raw and low quality (LQ) test set.

10.1 Model Design

For our final model, we adopt several different EfficientNet bases. We do not include a simple features model or temporal architectures as the former did not achieve an acceptable level of validation accuracy, while the latter requires the data to be structured differently.

One obstacle that remains is the high variability of neural networks. This means the results are highly sensitive to initial weights and statistical noise in the dataset. Moreover, the stochastic nature of the learning algorithm leads to different models each time a model is trained. The discrepancies between models can be dramatic in some cases. Finding a way to reduce this variance is key to more robust predictions.

Ensemble methods provide the solution to lowering the above variance and almost always produce more accurate predictions than those from a single network. The idea behind ensemble methods is that different networks will make different errors – averaging these out counters the variance from each network and allows errors specific to a particular model to be mitigated. Bishop et al. (1995) succinctly explain that a 'committee' of networks always outperforms a single member. Ensemble models ensure that higher accuracies and better generalisation are accompanied by stable predictions.

Our final model comprises an ensemble of nine networks, with different EfficientNet bases and four dense layers stacked on top of the convolutional base. These dense layers have 1024, 256 and 128 neurons with the final layer consisting of only two neurons. We use the top nine hyperparameter configurations presented in Table 5. Our motivation behind using different EfficientNet architectures and hyperparameter configurations is to ensure lower correlation between members of the ensemble. Weakly correlated models tend to make errors on different parts of the dataset. Including multiple accurate models in the ensemble allows members to correct each other's mistakes, thereby increasing overall accuracy.

The final ensemble is evaluated on both raw and LQ test data. To prepare the data for the LQ test set, we perform a train-time compression augmentation. This augmentation randomly compresses an image using a JPEG compression algorithm with a quality factor between 0 and 90, 0 indicating strong compression, shrinking the image size strongly while also losing a great deal of information, and 90 indicating a compression without much loss. We choose to augment the data in this fashion to simulate the compressed LQ data available from FaceForensics++, while intently not training on these data. We do not train on these data as we want to see how well our model generalises to unseen data of a different nature.

We train the nine models to convergence using the final augmentation setup, after which we switch all secondary augmentations for JPEG compression augmentation with a varying level of compression. We then once again retrain our nine models until convergence. To determine the weights of each model in the ensemble, we use a stacked approach training a meta-learner on validation data. The meta-learner acts as a classifier finding the best way to combine each model in the ensemble to get the most accurate image predictions. These image predictions are then combined with aggregation methods. Our final packaged end-to-end deepfake detector is presented in Figure 20.

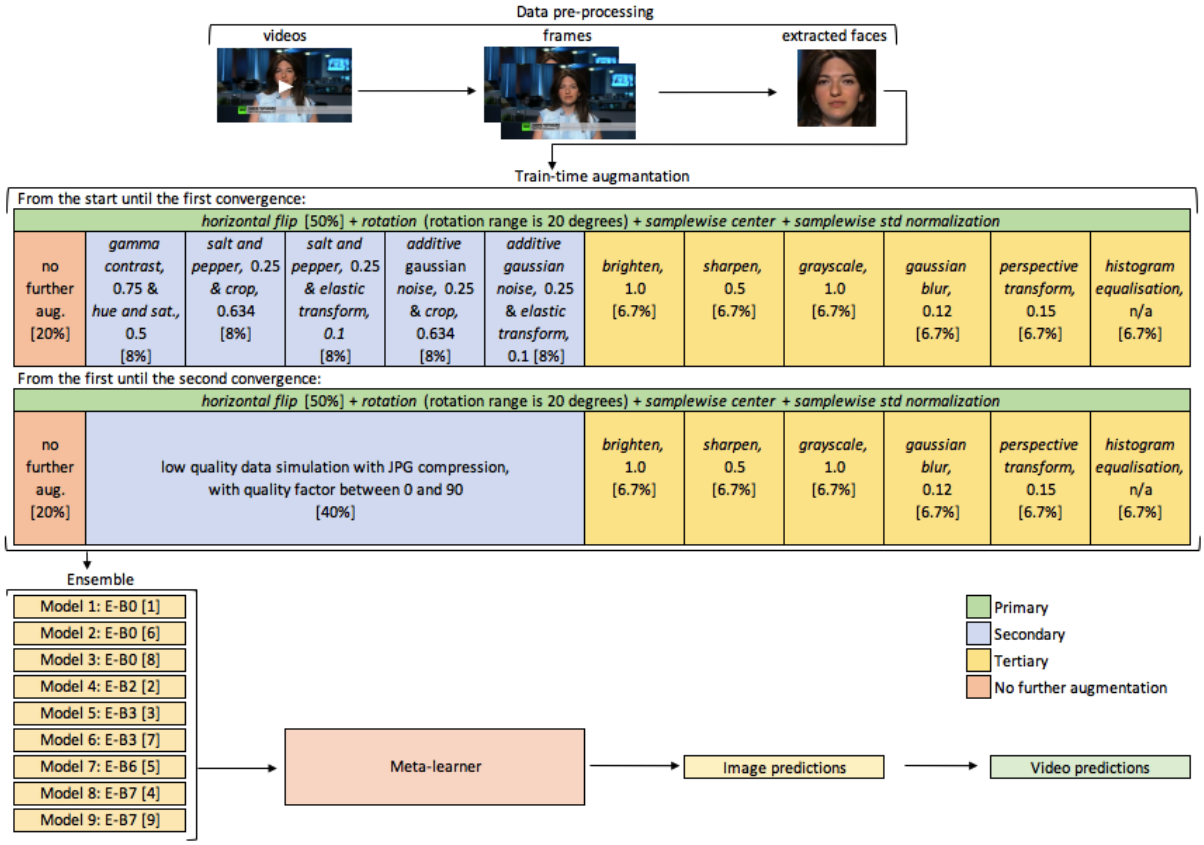

Figure 20: The Final Model: the first stage is our pre-processing pipeline including a custom developed face-extraction algorithm. It is followed by the final augmentation set up, discussed in detail in Section 6.3. After the models have converged, the secondary augmentations are switched to low quality data simulation with JPEG compression and the models are trained further until convergence. Augmented data are passed to the ensemble of nine EfficientNets trained with different hyperparameter combinations from Table 5 (indices are specified in brackets). Lastly, a meta-learner outputs image predictions which are aggregated into final video predictions.

10.2 Results

We present the results of our final ensemble on raw and low quality data in Table 7, along with results reported for Xception and the best performing model from the ensemble.

Table 7: Ensemble Results: test accuracies for Xception (reported results from FaceForensics++), the best performing model from the ensemble, and the final ensemble model. The aggregation method used to achieve the highest test accuracy is fractional mean in all instances.

	Xception	**Top model**	**Ensemble**
Raw data	0.9926	0.9900	0.9900
Low quality data	0.8100	0.8457	0.8571

While the ensemble obtains the same test accuracy of 99.00% as the top model in the ensemble on raw data, the ensemble outperforms the top model on the LQ data by 1.14%. These differences provide evidence to support the earlier statement that a collection of models almost always produces more accurate predictions than a single network.

It is important to note that the raw test set is constructed under the same conditions using the same manipulation methods and so resembles the training set. The final ensemble performs slightly worse than the Xception network on raw data. The majority of the baseline networks in Section 5 achieve a higher accuracy than our final model because these models are trained without augmentation and therefore generalise extremely well to the test set, but not so to any other unseen data. The introduction of various complex augmentations during training reduces the risk of overfitting and improves generalisation, but does not lead to a higher test accuracy due to the nature of the test data. Furthermore, gains could only have been marginal from a 99.26% test accuracy. The results on LQ data are therefore much more important. This ensemble outperforms the FaceForensics++ highest accuracy by 4.71%, achieving 85.71%, without ever being trained on their LQ data. This shows that our model possesses an essential trait in deepfake detection: generalisation. This test accuracy on LQ data is higher than that of any other paper mentioned in related work.

Table 8: Ensemble Test Accuracy per Manipulation Method

	Deepfakes	Face2Face	FaceSwap	NeuralTextures	Original
Raw data	1.0000	0.9857	1.0000	0.9786	0.9857
Low quality data	0.9929	0.9643	1.0000	0.8571	0.4714

There is one important limitation to the LQ results. While the fractional mean produces the highest test accuracies for this dataset, it is slightly unfair in the context of the test set. By design, fractional mean favours detecting fakes, which is advantageous as 80% of the test set is manipulated. However, the model which relies on this aggregation will be less accurate at classifying real videos, which can be observed in Table 8.

11 Conclusion

Motivated by the ongoing success of digital face manipulations, this paper provides a comprehensive analysis of solutions to detecting deepfake manipulated content. Its intention is to optimise a deepfake detector trained on the FaceForensics++ dataset. Thereafter, this model could be employed and productionised by Samsung to detect deepfakes on social media platforms.

The first sub-objective of this study was to reproduce the FaceForensics++ results. In the process of reproducing these results, we discovered that an EfficientNet is a better convolutional base than Xception. These results, along with an extensive literature review and the DFDC models, demonstrated that performance could be increased by researching augmentations, hyperparameters, and temporal models. Therefore, this paper studies augmentations including a classical frequency analysis and a class of information-dropping approaches, state-of-the-art hyperparameter optimisation for EfficientNets and building convolutional LSTMs and Transformers to incorporate the temporal information within videos into the analysis.

For the second sub-objective, our first finding suggests that information-dropping augmentations provide no accuracy improvements on FaceForensics++ video data. Nevertheless, information-dropping augmentation approaches show promise in the field of deepfake detection. We propose that our information-dropping augmentations warrant further research as they have a potential to greatly improve generalisation while reducing the risk of overfitting on other, more challenging, datasets. Additionally, we suggest an automated augmentation selector for augmentation optimisation for each dataset as augmentations are never universally optimal. We also advise integrating a large variety of augmentations, for greater generalisation ability. Regarding classical frequency analysis as an alternative to traditional augmentations, we found that simple features do not provide satisfactory results on the FaceForensics++ validation data, achieving a 87.90% accuracy. It is not fit for larger datasets and more traditional forms of augmentation are a better approach for deepfake detection.

Searching for further accuracy gains, we found that hyperparameter selection played a crucial role in detection ability. It became clear that learning rate was a key hyperparameter where values of 0.00001, 0.0001 and 0.001 led to good results. While tuned constant learning rate schedules performed well, decaying the learning rate using a cosine function often led to more promising accuracies. We also found that batch sizes of less than 32 allowed for stronger regularisation and avoidance of sub-optimal minima. Additional techniques including label smoothing, dropout, weight decay and class weights also improved results. With a clearer understanding of promising hyperparameter values, we conducted a random search by training 50 EfficientNet-B0 models to convergence, finding ten models with high validation accuracies.

We group these models in an ensemble to extract further accuracy gains.

We also experimented with temporal models, hoping to detect characteristic inter-frame features of deepfakes. These models performed slightly worse than the baseline models on the validation data, achieving a 98.14% and a 99.14% accuracy for the best convolutional LSTM and convolutional Transformer respectively. However, these models may be capable of better generalisation than their non-temporal counterparts and we encourage further research on this.

Lastly, combining all information from the analyses above, we arrive at a final model that can be pushed into production. This is an ensemble of nine networks, with different bases of EfficientNet. We test this model on both raw and low quality data. The final model's test accuracy on the raw data is 99.00%, just below that of Xception and other baseline networks. This accuracy is a consequence of the similarity between training data and the test set. Therefore, baseline models trained without augmentation generalise extremely well to the test set and perform marginally better than our final model. However, the final model's test accuracy on the low quality data is 85.71%, outperforming Xception by 4.71% and all other papers reporting results on this test set, without ever being trained on the FaceForensics++ LQ data. This further provides evidence that our model is robust to unseen data, and generalises well to data of a different nature.

Although our final model is fully trained and ready to be deployed for deepfake detection purposes, developers can also use our findings and propositions to tailor their models to specific requirements and other datasets. To attain better generalisation, we recommend retraining on a more complex, larger, third generation dataset as FaceForensics++ data, which was the most sophisticated dataset available at the beginning of our research, is now considered to be first generation.

Our first recommendation involves using our pre-processing pipeline. The custom face-extraction algorithm we developed, designed to retain as much information as possible for each video, leverages the latest face detection software and is packaged and ready to be applied to any video dataset. Secondly, we advise experimenting with sophisticated augmentations including our proposed custom information-dropping methods which provide good generalisation to other datasets. Thirdly, we highly recommend incorporating an automatic augmentation selector to tailor augmentations for each dataset. Finally, with regard to training, we find random search essential for successful hyperparameter selection. The above provide guidelines for a deepfake detector approach should the end user need tailoring to a specific dataset. However, if the FaceForensics++ dataset is sufficient, we provide guidelines on using our packaged final model below.

Our final product packages the pre-processing pipeline and trained ensemble. Using the product is simple: first pre-process new data by executing the *extract_compressed_videos.py* and *retinaface-cropper.py* scripts. Then run the output through our model, which can be loaded directly into Keras. This product is designed to be leveraged on social media platforms or other websites containing frequently uploaded visual material. We suggest incorporating our model as an online tool that flags manipulated content to platform users, or automatically removes this content if a significant threat is detected. We propose running the model on posts at the content-uploading stage to minimise screen time of manipulated content. Our final recommendation involves monetising the product. We suggest providing an API service to other companies where our technology can be used to detect manipulated content on their platforms. Our product not only contributes to minimising the threat that deepfakes pose, but also contributes to preserving visual material's online credibility. We look forward to seeing its value in practice.

Finally, it is important to note that there exists a gap between academic research related to deepfake detection and its real-life applications. Most current face manipulations are easily detected under controlled conditions, that is, when fake detectors are evaluated in the same conditions they are trained for. Academic literature, including this paper, often achieves extremely high test accuracies. Such academic scenarios are, however, not very realistic as they do not replicate the social media environment where manipulated content is typically shared. Deepfake videos on social media have often undergone heavy transformations, with different compression levels, resizing, noise and more. DFDC proves this, as the best model only achieves a 65% test accuracy on an unseen, private test set, which simulates a real-world scenario. With this nuance in mind, further research should focus on detectors' ability to accurately spot deepfakes under a larger variety of settings, which align more closely with the social media environment. This new focus will pave the way towards solving an issue that is likely to become one of the most pressing problems of our time.

Bibliography

Afchar, D., Nozick, V., Yamagishi, J. and Echizen, I. (2018). Mesonet: a compact facial video forgery detection network, *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, IEEE, pp. 1–7.

Ajder, H., Patrini, G., Cavalli, F. and Cullen, L. (2020). The state of deepfakes: Landscape, threats, and impact, *DeepTrace Labs* .

Amerini, I., Galteri, L., Caldelli, R. and Del Bimbo, A. (2019). Deepfake video detection through optical flow based cnn, *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 0–0.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures, *Neural networks: Tricks of the trade*, Springer, pp. 437–478.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization, *The Journal of Machine Learning Research* **13**(1): 281–305.

Bishop, C. M. et al. (1995). *Neural networks for pattern recognition*, Oxford University Press.

Chen, P. (2020). Gridmask data augmentation, *arXiv preprint arXiv:2001.04086* .

Chollet, F. (2017a). Deep learning with python.

Chollet, F. (2017b). Xception: Deep learning with depthwise separable convolutions, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258.

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B. and LeCun, Y. (2015). The loss surfaces of multilayer networks, *Artificial intelligence and statistics*, pp. 192–204.

Citron, D. K. and Franks, M. A. (2014). Criminalizing revenge porn, *Wake Forest L. Rev.* **49**: 345.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V. and Le, Q. V. (2018). Autoaugment: Learning augmentation policies from data, *arXiv preprint arXiv:1805.09501* .

Dang, H., Liu, F., Stehouwer, J., Liu, X. and Jain, A. K. (2020). On the detection of digital face manipulation, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5781–5790.

Dang, L. M., Hassan, S. I., Im, S. and Moon, H. (2019). Face image manipulation detection based on a convolutional neural network, *Expert Systems with Applications* **129**: 156–168.

Das, D., Avancha, S., Mudigere, D., Vaidynathan, K., Sridharan, S., Kalamkar, D., Kaul, B. and Dubey, P. (2016). Distributed deep learning using synchronous stochastic gradient descent, *arXiv preprint arXiv:1602.06709* .

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., Le, Q. et al. (2012). Large scale distributed deep networks, in 'advances in neural information processing systems'.

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

Deng, J., Guo, J., Zhou, Y., Yu, J., Kotsia, I. and Zafeiriou, S. (2019). Retinaface: Single-stage dense face localisation in the wild, *arXiv preprint arXiv:1905.00641* .

DeVries, T. and Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout, *arXiv preprint arXiv:1708.04552* .

Dolhansky, B., Bitton, J., Pflaum, B., Lu, J., Howes, R., Wang, M. and Ferrer, C. C. (2020). The deepfake detection challenge dataset, *arXiv preprint arXiv:2006.07397* .

Dolhansky, B., Howes, R., Pflaum, B., Baram, N. and Ferrer, C. C. (2019). The deepfake detection challenge (dfdc) preview dataset, *arXiv preprint arXiv:1910.08854* .

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

Durall, R., Keuper, M., Pfreundt, F.-J. and Keuper, J. (2019). Unmasking deepfakes with simple features, *arXiv preprint arXiv:1911.00686* .

Farid, H. (2011). Photo tampering throughout history, *Hany Farid, Research (Image Science Group, Dartmouth College Computer Science Department)* .

Geitgey, A. (2020). face_recognition.

Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep learning*, MIT press.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y. and He, K. (2017). Accurate, large minibatch sgd: Training imagenet in 1 hour, *arXiv preprint arXiv:1706.02677* .

Güera, D. and Delp, E. J. (2018). Deepfake video detection using recurrent neural networks, *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, IEEE, pp. 1–6.

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

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

He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J. and Li, M. (2019). Bag of tricks for image classification with convolutional neural networks, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 558–567.

Hochreiter, S. and Schmidhuber, J. (1997). Lstm can solve hard long time lag problems, *Advances in neural information processing systems*, pp. 473–479.

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

Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K. Q. (2017). Densely connected convolutional networks, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708.

Jia, X., Song, S., He, W., Wang, Y., Rong, H., Zhou, F., Xie, L., Guo, Z., Yang, Y., Yu, L. et al. (2018). Highly scalable deep learning training system with mixed-precision: Training imagenet in four minutes, *arXiv preprint arXiv:1807.11205* .

Jung, A. B., Wada, K., Crall, J., Tanaka, S., Graving, J., Reinders, C., Yadav, S., Banerjee, J., Vecsei, G., Kraft, A., Rui, Z., Borovec, J., Vallentin, C., Zhydenko, S., Pfeiffer, K., Cook, B., Fernández, I., De Rainville, F.-M., Weng, C.-H., Ayala-Acevedo, A., Meudec, R., Laporte, M. et al. (2020). imgaug. Online; accessed 01-Feb-2020.

Karpathy, A. (2019). A recipe for training neural networks.

Karras, T., Aila, T., Laine, S. and Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation, *arXiv preprint arXiv:1710.10196* .

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M. and Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima, *arXiv preprint arXiv:1609.04836* .

Kiefer, J., Wolfowitz, J. et al. (1952). Stochastic estimation of the maximum of a regression function, *The Annals of Mathematical Statistics* **23**(3): 462–466.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980* .

Kornblith, S., Shlens, J. and Le, Q. V. (2019). Do better imagenet models transfer better?, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2661–2671.

Krogh, A. and Hertz, J. A. (1992). A simple weight decay can improve generalization, *Advances in neural information processing systems*, pp. 950–957.

LeCun, Y. A., Bottou, L., Orr, G. B. and Müller, K.-R. (2012). Efficient backprop, *Neural networks: Tricks of the trade*, Springer, pp. 9–48.

Li, Y. and Lyu, S. (2018). Exposing deepfake videos by detecting face warping artifacts, *arXiv preprint arXiv:1811.00656* .

Loshchilov, I. and Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts, *arXiv preprint arXiv:1608.03983* .

Masters, D. and Luschi, C. (2018). Revisiting small batch training for deep neural networks, *arXiv preprint arXiv:1804.07612* .

Matern, F., Riess, C. and Stamminger, M. (2019). Exploiting visual artifacts to expose deepfakes and face manipulations, *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, IEEE, pp. 83–92.

Mo, H., Chen, B. and Luo, W. (2018). Fake faces identification via convolutional neural network, *Proceedings of the 6th ACM Workshop on Information Hiding and Multimedia Security*, pp. 43–47.

Müller, R., Kornblith, S. and Hinton, G. E. (2019). When does label smoothing help?, *Advances in Neural Information Processing Systems*, pp. 4694–4703.

Nguyen, H. H., Fang, F., Yamagishi, J. and Echizen, I. (2019). Multi-task learning for detecting and segmenting manipulated facial images and videos, *arXiv preprint arXiv:1906.06876* .

Nguyen, H. H., Yamagishi, J. and Echizen, I. (2019). Capsule-forensics: Using capsule networks to detect forged images and videos, *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, pp. 2307–2311.

Nguyen, T. T., Nguyen, C. M., Nguyen, D. T., Nguyen, D. T. and Nahavandi, S. (2019). Deep learning for deepfakes creation and detection, *arXiv preprint arXiv:1909.11573* .

NTechLab (2020). Dfdc solution description.

Ozmen, B. (2019). Deepaugment, `https://doi.org/10.5281/zenodo.2949929`.

Paris, B. and Donovan, J. (2019). Deepfakes and cheap fakes, *United States of America: Data & Society* .

Pratt, L. Y., Mostow, J., Kamm, C. A. and Kamm, A. A. (1991). Direct transfer of learned information among neural networks., *Aaai*, Vol. 91, pp. 584–589.

Rossler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J. and Nießner, M. (2019). Faceforensics++: Learning to detect manipulated facial images, *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1–11.

Sabir, E., Cheng, J., Jaiswal, A., AbdAlmageed, W., Masi, I. and Natarajan, P. (2019). Recurrent convolutional strategies for face manipulation detection in videos, *Interfaces (GUI)* **3**(1).

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O. (2017). Proximal policy optimization algorithms, *arXiv preprint arXiv:1707.06347* .

Seferbekov, S. (2020). Deepfake detection (dfdc) solution.

Shaoanlu (2019). Face parser.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition, *arXiv preprint arXiv:1409.1556* .

Singh, K. K., Yu, H., Sarmasi, A., Pradeep, G. and Lee, Y. J. (2018). Hide-and-seek: A data augmentation technique for weakly-supervised localization and beyond, *arXiv preprint arXiv:1811.02545* .

Song, Y., Zhu, J., Li, D., Wang, X. and Qi, H. (2018). Talking face generation by conditional recurrent adversarial network, *arXiv preprint arXiv:1804.04786* .

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting, *The journal of machine learning research* **15**(1): 1929–1958.

Sutskever, I., Martens, J., Dahl, G. and Hinton, G. (2013). On the importance of initialization and momentum in deep learning, *International conference on machine learning*, pp. 1139–1147.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. (2016). Rethinking the inception architecture for computer vision, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826.

Tan, M. and Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks, *arXiv preprint arXiv:1905.11946* .

Thies, J., Zollhöfer, M. and Nießner, M. (2019). Deferred neural rendering: Image synthesis using neural textures, *ACM Transactions on Graphics (TOG)* **38**(4): 1–12.

Thies, J., Zollhofer, M., Stamminger, M., Theobalt, C. and Nießner, M. (2016). Face2face: Real-time face capture and reenactment of rgb videos, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2387–2395.

Tolosana, R., Vera-Rodriguez, R., Fierrez, J., Morales, A. and Ortega-Garcia, J. (2020). Deepfakes and beyond: A survey of face manipulation and fake detection, *arXiv preprint arXiv:2001.00179* .

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I. (2017). Attention is all you need, *Advances in neural information processing systems*, pp. 5998–6008.

Verdoliva, L. (2020). Media forensics and deepfakes: an overview, *arXiv preprint arXiv:2001.06564* .

Wang, S.-Y., Wang, O., Zhang, R., Owens, A. and Efros, A. A. (2020). Cnn-generated images are surprisingly easy to spot... for now, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 7.

Wang, Y. and Dantcheva, A. (2020). A video is worth more than 1000 lies. comparing 3dcnn approaches for detecting deepfakes, *FG'20, 15th IEEE International Conference on Automatic Face and Gesture Recognition, May 18-22, 2020, Buenos Aires, Argentina*.

Wilson, D. R. and Martinez, T. R. (2003). The general inefficiency of batch training for gradient descent learning, *Neural networks* **16**(10): 1429–1451.

Xie, Q., Luong, M.-T., Hovy, E. and Le, Q. V. (2020). Self-training with noisy student improves imagenet classification, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10687–10698.

Xie, R. (2020). Gridmask.

Yang, S., Luo, P., Loy, C.-C. and Tang, X. (2016). Wider face: A face detection benchmark, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5525–5533.

Zhang, H., Cisse, M., Dauphin, Y. N. and Lopez-Paz, D. (2017). Mixup: Beyond empirical risk minimization, *arXiv preprint arXiv:1710.09412* .

Zhong, Z., Zheng, L., Kang, G., Li, S. and Yang, Y. (2017). Random erasing data augmentation. arxiv 2017, *arXiv preprint arXiv:1708.04896* .

Appendix

AI Platform Setup

How to recreate experiments using an AI platform notebook instance on GCP:

1. Create a Tensorflow2.2 instance with GPUs (in the same region as the bucket).

2. SSH into the virtual machine and mount the a bucket to the instance (as intermediate epoch weights and final models will be saved to the bucket) by running the code below:

```
export GCSFUSE_REPO=gcsfuse-'lsb_release -c -s'
echo "deb http://packages.cloud.google.com/apt $GCSFUSE_REPO main" | sudo tee
/etc/apt/sources.list.d/gcsfuse.list
curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -sudo
    apt-get update
sudo apt-get install gcsfuse
cd /home/jupyter/
sudo -s su jupyter
mkdir all_faces_bucket
/usr/bin/gcsfuse --implicit-dirs all_faces all_faces_bucket
```

3. Copy all data to the virtual machine's persistent disk (only when creating the persistent disk for the first time):

```
cd /home/jupyter/
sudo gsutil -m cp -r gs://all_faces/forensics_split/test .
sudo gsutil -m cp -r gs://augmented_all_faces .
```

4. To avoid having to copy data to the persistent disk each time an instance is created, edit the virtual instance's boot disk through the GCP console. Next time you create an instance, you can just attach this persistent disk. Note that you can only attach a persistent disk to multiple instances if you restrict each instance's disk access to read only. To mount the read only persistent disk, run the following after you have attached it to an instance:

```
cd /home/jupyter/
sudo mkdir all_faces_disk
sudo mount -o ro /dev/sdb1 all_faces_disk
```

Note: Be careful once the bucket and disk are mounted. Any changes you make from within the JupyterLab GUI will reflect on the actual disk/bucket.

5. Proceed to link the notebook instance to the GitHub repository as follows:

- Open JupyterLab from the AI Platform notebook dashboard
- Select Git -> Open Git Repository in Terminal

- Select Git -> Clone and paste the repository link:
  https://github.com/lse-st498/DeepFake-2019-20.git
- Enter your GitHub credentials
- The repository should appear as a folder on the left pane
- Select the git tab/icon on the far left
- Click the plus sign on the files in the changed and untracked sections
- Commits and pushes to the repository can be made from this tab

5. Notebooks can now be run. Note that it is necessary to prevent your local computer from sleeping to ensure notebooks continue to run. Alternatively, Kubeflow Fairing can be used to package and deploy notebooks eliminating the need to leave your laptop/desktop running.

Hyperparameter Combinations

Table 9: Hyperparameter Combinations for Respective Sections

Section	Batch	LR	Decay	Optimiser	Label S.	Patience	Dropout
5	32	0.0002	0	Adam	0	7	0.5
6.1	32	0.0002	0	Adam	0.01	6	0.5
7	32	0.005	10^{-6}	Adam	0	-	0.5
8.2.1	32	Varied	0	SGD	0	-	0.5
8.3	Varied	0.0002	0	Adam	0	-	0.5
9	4	0.0001	Multiple	Adam	Multiple	-	0.5

For the augmentation experiment in Section 6.1, each model is initially trained with a frozen convolutional base and then is fine-tuned (with learning rate decreased by a factor of 10).