**Detecting AI-Altered Media with Deep Learning**

By

Rhys Chua

Jim Fang

Jonathan Huff

Dr. Arnab Bose, Advisor

(Independent Thesis Project)

A Capstone Project Submitted to the University of Chicago in partial fulfillment of the requirements for the degree of Master of Science in Analytics.

Graham School of Continuing Liberal and Professional Studies

(March 2021)

The Capstone Project committee for Rhys Chua, Jim Fang, and Jonathan Huff Certifies that this is the approved version of the following capstone project report:

Detecting AI-Altered Media with Deep Learning

APPROVED BY

SUPERVISING COMMITTEE:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Supervisor’s name

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Program Director’s name

# **Abstract**

As the capabilities of machine learning grow at an unprecedented pace, the benefits these models provide also come with a cost. Deep learning media creation methodologies such as the autoencoder and Generative Adversarial Network (GAN) are being used across a wide array of industries with great impact. However, these innovations can be easily exploited for nefarious purposes to create fraudulent image, audio, and video impersonations known as deepfakes. Due to the reputational, financial, and geopolitical instability that deepfake media can cause, an increasingly vital need exists to mitigate these risks through advanced detection methodologies. Having surveyed the state-of-the-art techniques used in deepfake generation and detection, we introduce a webapp consisting of advanced deep learning networks that is not only competitive with state-of-the-art on in-distribution and out-of-distribution lab results, but also extends detection capabilities to YouTube-sourced in-the-wild deepfake videos.

*Keywords: Generative Adversarial Networks (GANs), Machine Learning, Artificial Intelligence, deepfake, forgery detection, deep learning*

# **Executive Summary**

In recent years, fake news has become a critical issue that poses extreme threat to national security, human society, and democracy. Advanced artificial intelligence has allowed users to create hyper-realistic videos that include features such as changes in facial expressions, complete face swapping, and generation of new faces without leaving large traces of manipulation behind. The sophistication of this technology to be used for malevolent intent is deeply concerning. Tied in with the rampant usage of social media, the rate at which false information disseminates continues to increase, where it can impact millions of users in a short amount of time. Misuse of such technology could lead to political sabotage, market manipulation, and ill-informed decision making.

While spreading false videos is easy, detecting them is much harder. To do so, one must understand the various methods used to create fake images and videos, the imperfections resulting from such techniques, and the deep learning frameworks that would be most effective at identifying them. We were able to incorporate an ensemble meta-learner that leverages the power of two computer vision architectures, XceptionNet and EfficientNet, that when combined with an LSTM model used to analyze sequences, outputs a final prediction. We also curated a composite dataset consisting of several open source deepfake datasets to diversify our training samples and subsequently increase the generalizability of our solution. As a result, we successfully achieve an Area Under the Curve (AUC) score of 0.843 and 0.669 on two unseen out-of-distribution (OOD) datasets – Deepfake Detection (DFD) and Deepfake Detection Challenge (DFDC).

Table of Contents

[Abstract i](#_Toc66224261)

[Executive Summary ii](#_Toc66224262)

[List of Figures iii](#_Toc66224263)

[List of Tables iv](#_Toc66224264)

[List of Appendices iv](#_Toc66224265)

[1. Introduction 1](#_Toc66224266)

[1.1 Definition of Deepfakes 1](#_Toc66224267)

[1.2 Problem Statement 2](#_Toc66224268)

[1.3. Research Purpose and Analysis Goals 3](#_Toc66224269)

[1.4 Variables and Scope 3](#_Toc66224270)

[2. Literature Review 4](#_Toc66224271)

[2.1 Generative Adversarial Networks (GANs) 4](#_Toc66224272)

[2.2 Detection Methods 5](#_Toc66224273)

[2.3 State-of-the-Art Detection Methods 6](#_Toc66224274)

[3. Methodology 8](#_Toc66224275)

[3.1 Data 8](#_Toc66224276)

[3.2 Descriptive Analyses 10](#_Toc66224277)

[3.3 Modelling Framework 10](#_Toc66224278)

[11](#_Toc66224279)

[3.3.1 Data Pre-Processing 11](#_Toc66224280)

[3.3.2 Model Experimentation 14](#_Toc66224281)

[3.3.3 Model Interpretability 17](#_Toc66224282)

[3.3.4 Web Application 20](#_Toc66224283)

[5. Conclusion 27](#_Toc66224284)

[6. Recommendations and Future Work 28](#_Toc66224285)

[7. References 31](#_Toc66224286)

[8. Appendix 35](#_Toc66224287)

# **List of Figures**

|  |  |
| --- | --- |
| Figure 1 | 1 |
| Figure 2 | 1 |
| Figure 3 | 1 |
| Figure 4 | 1 |
| Figure 5 | 11 |
| Figure 6 | 15 |
| Figure 7 | 17 |
| Figure 8 | 17 |
| Figure 9 | 19 |
| Figure 10 | 19 |
| Figure 11 | 20 |
| Figure 12 | 23 |
| Figure 13 | 24 |
| Figure 14 | 25 |
| Distribution 1 | 13 |
| Distribution 2 | 13 |

# **List of Tables**

|  |  |
| --- | --- |
| Table 1 | 9 |
| Table 2 | 10 |
| Table 3 | 21 |
| Table 4 | 26 |
| Table 5 | 27 |

# **List of Appendices**

|  |  |
| --- | --- |
| Appendix 1 | 35 |

# **1. Introduction**

## **1.1 Definition of Deepfakes**

Deepfakes are the product of deep learning models designed to generate fake media. More specifically, these models systematically replace one person's likeness with another, ultimately to create realistic counterfeit videos. Deepfakes rely on neural networks that analyze large sets of data samples to learn how to mimic a person's facial expressions, mannerisms, voice, and inflections (Westerlund, 2019).

A picture containing text, person, posing

Description automatically generatedA group of women smiling

Description automatically generated with low confidenceA picture containing text, posing

Description automatically generatedA collage of a person

Description automatically generated with medium confidenceDeepfakes are generated predominantly by the autoencoder and generative adversarial network (GAN). The GAN architecture consists of a generator and discriminator which employs game theory to iteratively improve on the fake image synthesis of the generator until the discriminator can no longer identity the forgery as fake (Goodfellow, et. al. 2014). Examples of types of generation include reenactment (using facial or body movements from a subject to

**Figure 4.** Facial synthesis

**Figure 1.** Facial Reenactment

**Figure 2.** Facial Replacement

**Figure 3.** Facial editing

place it on a targeted image) (Figure 1), replacement (mapping a subject’s face to a target) (Figure 2), editing (altering facial attributes of a person) (Figure 3), and synthesis (generating completely new materials with no targeted person) (Figure 4) (*The State of Deepfakes in 2020*, n.d.).

Early example of deepfakes focused on fake video evidence in courts, blackmails, and fake news (Maras & Alexandrou, 2019). Consequences of the misuse of deepfakes are detrimental to society and may be used to manipulate public opinion for deceitful purposes. However, deepfake technologies has its benevolent uses in industries such as healthcare, fashion, digital communication, education, gaming, and entertainment (Westerlund, 2019). In a healthcare setting, synthetic patients with true to life data could be used for research purposes to evaluate the outcome of treatments for global pandemics and infectious and chronic diseases (“AI in Healthcare,” 2020). Deepfakes also allow game developers and movie makers to create virtual worlds, recreate classic scenes, and use face editing to improve quality post-production (Westerlund, 2019).

## **1.2 Problem Statement**

Technological breakthroughs in the field of machine learning over the past five years have allowed deep learning researchers to manufacture synthetic data representative of the underlying authentic data. These advances can be seen in improved corporate profitability, increased flexibility in the healthcare sector regarding the Health Insurance Portability and Accountability Act (HIPAA) regulations, enhanced consumer experiences, among other benefits. However, these techniques for generating data can be and have been commandeered for nefarious purposes to generate media in the form of images, audio, and video, collectively known as deepfakes. Deepfakes used to create fake news has become so common that it is now a household term. Such false information can now be propagated virally across social media platforms in the form of realistic Artificial Intelligence (AI) generated deepfake media by perpetrators who exploit this technology for malicious purposes. While traditional fake news from news outlets may expose companies and high-profile figures to reputational and monetary damage, deepfake media can be generated by anyone and is especially harmful as it provides false evidence of unfavorable speech and/or actions.

## **1.3. Research Purpose and Analysis Goals**

Deepfake detection methodologies have been abundant since the invention of GANs. Our research purpose is to understand several existing state-of-art detection methods and improve on those algorithms. To achieve our goals, we intend to first focus on detecting fakes in images and videos containing faces, followed by exploring different neural network architectures and use an ensemble method to determine the best detection approach, and finally develop a webapp that will be used to detect deepfake artifacts in user-uploaded media and add interpretability to the model predictions as well. If successful, the application will be made available for academic or commercial use in the future.

## **1.4 Variables and Scope**

To date, the scientific community has focused its efforts around generating fake content of human faces and speech to push the boundaries of what is possible while also challenging current state-of-the-art discriminators by offering more realistic videos with fewer easily identifiable salient features. Hence, most of these fake content alongside real data is available for the public to use. We acquired open-sourced datasets such as Celeb-DF (CDF), FaceForensics++ (FF+), and Deepfake Detection Challenge (DFDC) to conduct our research. These were chosen since they were featured in recent detection challenges and were often cited by top authors as being particularly difficult to train an effective model on when validating within the same dataset, while also being difficult to generalize to as an out-of-distribution dataset (OOD) (Tolosana, Vera-Rodriguez, et al., 2020). These datasets were generated to tackle specific challenges of deepfake such as forgery detection, reduced visual quality gap in videos, and face swapping.

# **2. Literature Review**

## **2.1 Generative Adversarial Networks (GANs)**

The GAN framework was introduced by Ian Goodfellow and his colleagues in 2014. As mentioned in Section 1.1 of this paper, the GAN has a two-pronged network, the ‘generator’ and the ‘discriminator.’ The generator learns to generate synthetic and realistic materials while the discriminator attempts to decipher if generated material is real or fake. After many iterations, the quality of the generated data and the ability to detect the forged data both increase substantially to the point where the discriminator can no longer determine whether the data is real or fake. This this point, the model is fully trained and able to generate very realistic data of various types (Goodfellow et al., 2014). Face manipulation techniques can be broken down into one of four categories: Entire Face Synthesis, Identity Swap, Attribute Manipulation, and Expression Swap (Tolosana, Vera-Rodriguez, et al., 2020). The first category, known as Entire Face Synthesis, is capable of constructing entirely new faces that currently do not exist, through the use of an architecture called StyleGAN. Another technique, known as attribute manipulation, consists of altering parts of the face through the addition or subtraction of certain features such as wrinkles, eyeglasses, or skin tone, typically using a StarGAN approach. This is commonly seen in the FaceApp mobile app and other tools that allow a user to change his/her appearance and can be used to deceive the general public or for other nefarious purposes as well. The last two techniques, Identity Swap and Expression Swap, have been highlighted by recent papers such as Rössler of the FF++ dataset as posing a more immediate threat to high profile and powerful figures such as politicians or celebrities (Rössler et al., 2019). This method allows users to impose the face of one person in a video with the face of another person, and potentially make it appear as though algorithmically generated person of interest is actually speaking. Identity swaps can be carried out by classic graphics-based techniques such as FaceSwap or more modern techniques such as DeepFakes through the ZAO mobile app. Expression Swaps, on the other hand, are capable of manipulating image frames of a video through GAN models to replace the facial expressions of a person with the facial expressions of another person in any video. This particular technique is carried out by Face2Face or the NeuralTextures method, and to-date represents quite possibly the most believable and dangerous technique that modern detection methods must surmount in order to prevent the spread of misinformation. As an example, a recent video from Mark Zuckerberg was edited to portray him speaking negatively about the business model of his own company (Tolosana, Vera-Rodriguez, et al., 2020).

## **2.2 Detection Methods**

In the past three years alone, a multitude of advancements have been made in the field of fake video detection by researchers from all over the world. These detection techniques have advanced from detection of blemishes or inconsistencies (such as missing detail in reflections of teeth, eye color, or pixel inconsistencies) of still-frame captures through algorithmic or unsupervised learning approaches (such as K-means), to entirely deep-learning based approaches that involve heavy use of Convolutional Neural Network (CNN) or Long-Short Term Memory models (LSTM) (Kupyn et al., 2018)(Tolosana, Vera-Rodriguez, et al., 2020)

## **2.3 State-of-the-Art Detection Methods**

In 2018, research began to focus more on detecting features using deep learning methods such as CNN's and LSTMs in conjunction with mesoscopic feature analysis and steganalysis. The application of steganalysis, a study of hidden messages, to the field of deepfake detection was first proposed by Fridrich et al. (Fridrich & Goljan, 2002). The method seeks to detect potential modifications in videos by developing handcrafted features based on noise components and pixel co-occurrence matrices that can be extracted from an image and used within a CNN or Support Vector Machine (SVM). It should be noted this did not work well for low-quality (high compression) images and is one of the main flaws of this technique. Authors such as Zhou et al. in 2018 were able to make use of these features in combination with deep learning features to produce a model with 85.1% AUC on the UADFV dataset but scored an AUC of only 54.8% on the newer CDF dataset (Tolosana, Vera-Rodriguez, et al., 2020).

Afchar et al. discovered mesoscopic features and the resulting MesoNet model, which intentionally uses shallow neural networks to detect smaller, more elementary patterns with a reduced number of convolutions of the image during a forward pass (Afchar et al., 2018). This method combined with a CNN resulted in generalizable models with high accuracy (98.4%) on their own in-house dataset and greater computational efficiency but was unable to generalize well to unseen videos in other datasets such as DFDC (AUC 75.3%), or CDF (AUC 54.8%). It should be noted that as of current date, the CDF large scale training dataset presents the biggest challenge for most SoTA detection methods (Tolosana, Vera-Rodriguez, et al., 2020).

The research community saw the discovery of Temporal Features by Guera et al., wherein a combination of CNN-generated feature maps (frame by frame images from video) was passed into an LSTM to process temporal data sequences. This enabled learning of specific movement-based behaviors of its subjects to achieve 97%+ accuracy on only 40 frames of data (Güera & Delp, 2018). Marra et al. discovered that GAN models produce fingerprints or recurring quasi-periodical patterns of their own that can be reliably detected and suggested newer training datasets remove these fingerprints to produce more robust detection models (Marra et al., 2019).This was later remedied by the creation of GANprintR (GAN fingerprint removal) and the creation of the iFakeFaceDB which set a higher bar and challenge for detection models. Most notably, Tolosana et al. devised a method using face warping regions which, in conjunction with a CNN architecture, was able to achieve an AUC of 83.6% on the Celeb-DF dataset, the best in class for Identity Swap detection models (Tolosana, Vera-Rodriguez, et al., 2020). However, this AUC only applied to within distribution data, meaning that its performance on out-of-distribution data was unknown.

Altogether, impressive strides have been made in recent years in the field of fake video detection. However, one common trend is that these models are severely limited by the datasets that they use, with a good mixture of actors, context, and levels of quality/compression that would allow a detector to generalize well to "wild" or unseen videos found on YouTube, as an example. Researchers at Dessa have made a note of this in a recent publication, wherein they recommended that fake and real videos be sampled across multiple databases (CDF, DFDC, FF++, etc.) in addition to videos found on YouTube, to generate a detection model that can more reliably detect spoofs made by videos not similar to the ones it has already seen. These video qualities and attributes have yet to be defined, however warrant a thorough and comprehensive analysis in our implementation process.

Moreover, while some researchers have attempted ensembling a few models at a time to see if performance can be improved, a large-scale study that examines the combination of several or all SoTA models mentioned in the most recent Deepfake Detection Survey has yet to be implemented (Bonettini et al., 2020). Furthermore, multimodal models that use image/video, text, audio, and micro-expression features in a unified CNN architecture have shown promise on smaller datasets but have yet to be implemented at scale on a wider database and presents a further opportunity (*FaceForensics++ & Survey of Multi-Modal Techniques | by Wer Deepfakers | Medium*, n.d.). In a similar vein, researchers borrowing from the field of Psychology have proposed the usage of Siamese networks, which combine the visual and audio modalities of a video to search for emotional cue inconsistencies (sad voice, happy facial expressions) and present an interesting area of further research as well (Mittal et al., 2020). A few additional burgeoning areas of interest in this field include the application of capsule networks and defensive measures against adversarial white-box or black-box attacks, wherein the attacker has prior knowledge of the model parameters that a detection model was trained and seeks to make slight alterations to their GAN generated images/videos to avoid detection.

# **3. Methodology**

## **3.1 Data**

Among the three datasets the latest generation DFDC (considered a third-generation dataset) is the largest public available dataset with 25 terabytes (TB) of data, approximately 129k videos, and more than 10 million number of frames per second (Dolhansky et al., 2020). At the time of data collection, we had access to the previous generation DFDC dataset which generated deepfakes using face swapping. We selected each dataset with the intent of diversifying the types of deepfake manipulation our model trains on for later detection (Table 1). FF++, a second-generation dataset focuses on different forgery methods by utilizing GAN and autoencoders to perform four manipulation techniques which are, FaceSwap (graphic based technique), Deepfakes (autoencoder technique), Face2Face (facial reenactment technique), and Neural Textures (neural texture-based technique) (Rössler et al., 2019). Included in FF++ are two external datasets called FaceShifter which utilizes GANs for manipulation, and Deepfake Detection (DFD) which consisted of deepfakes on paid actors. CDF a second-generation dataset, was generated using CAE. CDF’s manipulation technique includes increasing resolution pixels, color transfer algorithm, face masking, and temporal flickering (Li et al., 2020). We curated custom train, validation, and test datasets to diversify our data pool which contains a mixture of FF++ and CDF. DFDC and DFD were reserved for the OOD testing.

**Table 1.** Attributes of datasets used for modelling

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DFDC**  **(preview)** | **FaceForensics++ (FF++)** | **Celeb-DF (CDF)** | **Mixed Dataset** |
| **Main Focus** | Large and diverse corpus | Different forgery methods | Reduce Visual Quality Gap | FF++ and CDF |
| **Generation** | 2nd | 2nd | 2nd | Created by Us |
| **Size (# Videos)** | 400GB | 39GB (5K) | 34GB (6K) | 280GB (8.4K) |
| **Train/Val/Test %** | 58/21/21 | 72/14/14 | 68/16/16 | 70/15/15 |
| **Real/Fake Ratio** | 1:8 | 1:4 | 1:9 | 1:1 |
| **Method** | Convolution Autoencoder | Generative  Adversarial Network | Convolution Autoencoder | - |
| **Technique** | FaceSwap | FaceSwap, Deepfakes, Face2Face,  NeuralTextures,  FaceShifter,  DFD | Increase resolution pixels, color transfer algorithm, face masking, temporal flickering | - |

Initially, deepfake datasets were collected and stored on Google Drive. After data were being pre-processed and split into train, validation, and test they were stored in University of Chicago’s Resource Computing Center’s (RCC) Midway Cluster as HDF5 files.

## **3.2 Descriptive Analyses**

CDF comprehensive documented the different demographics present in their dataset. The videos contain mostly males (56.8%), people between the ages of 50 to 59 years (30.5%), 30 to 39 years (28.0%), and 40 to 49 years (26.6%). The racial distribution for CDF is 88% Caucasians, 6.8% African Americans, and 5.1% Asians (Table 2) (Li et al., 2020).

|  |  |
| --- | --- |
|  | **Celeb-DF (CDF)** |
| **Sex** |  |
| Male (%) | 56.8 |
| Female (%) | 43.2 |
| **Age (years)** |  |
| > 30 (%) | 6.4 |
| 30 – 39 (%) | 28.0 |
| 40 – 49 (%) | 26.6 |
| 50-59 (%) | 30.5 |
| < 60 (%) | 8.5 |
| **Race** |  |
| Asians (%) | 5.1 |
| African Americans (%) | 6.8 |
| Caucasians (%) | 88.1 |

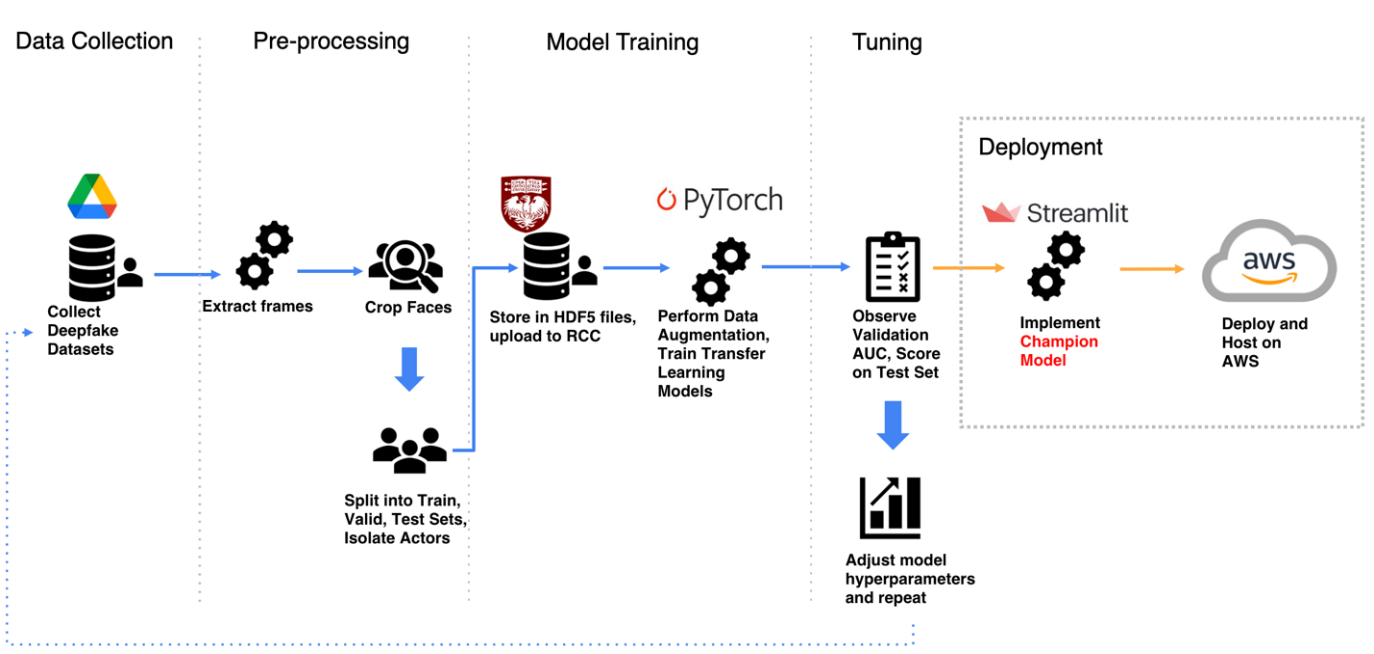
**Table 2.** Demographic breakdown for CDF data

## **3.3 Modelling Framework**

Our methodology for a deepfake detection webapp can be broken down into four core components: data pre-processing, model experimentation, model interpretability, and webapp design as can be seen in Figure 5.

### 

**Figure 5.** Data pipeline illustration

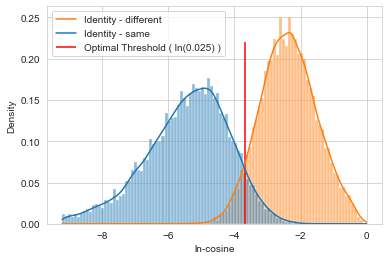
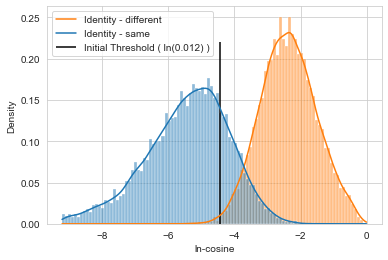


### **3.3.1 Data Pre-Processing**

Though all data was sourced from readily available opensource datasets, the videos were offered in raw format and required extensive processing in order to be modeled effectively. Additionally, though we implemented pre-defined and pre-trained deep network image classification architectures, we performed requisite modifications and additional training for ensemble learning and anomaly detection tasks. All Python code in our implementation was either written by this team or modified extensively from pre-existing repositories.

Our intention initially was to detect deepfake videos by analyzing both still-frame pixels as well as temporal features across multiple frames. The first challenge we faced involved parsing all video frames to extract individual faces. The library we used for this task, OpenFace 2.0(*OpenFace*, n.d.), required manually specifying whether to extract one or multiple faces. While most of the videos across our collection involved single subjects, there existed enough videos containing multiple subjects to merit a robust flexible solution. Applying multiple face detection to videos containing only one face yielded a prevalence in erroneous detections. At first, we considered applying single face detection to every dataset, at the expense of slight sparsity in the amount of data we would collect. This was infeasible for two reasons. First, applying single face detection to a video containing multiple faces would cause the algorithm to arbitrarily detect the faces of different subjects between frames. As stated, our intent was to model the sequential nature of deepfake features, and therefore mixing faces of different subjects would yield extremely unclean data. Manually reviewing all videos and selecting only the ones with single subjects was highly impractical, as we processed over 10,000 videos. More importantly, as this would be deployed as a production-ready solution, we had to account for the possibility that users would upload videos containing multiple faces.

The objective at this stage was to determine how to extract clean sequences of faces. The solution we arrived at involved parsing each video frame, and then sorting the faces post-extraction. The sorting process involved two primary methods. First, using the bounding box of the detected face (the coordinates of the box outlining the detected face in the video frame, as returned by the FastMTCNN(*Fast MTCNN Detector (~55 FPS at Full Resolution)*, n.d.) algorithm), we required that a certain intersection-over-union (IOU) threshold be met. That is, the consecutive bounding boxes between frames must overlap by a certain amount. This makes sense, as most videos are recorded at a rate of at least 30 frames-per-second, and therefore the subjects’ faces will typically be in a very similar location from one frame to the next. However, there was a flaw in this method. In the case of a one-on-one interview for example, the camera may shift from the interviewer to the interviewee, leaving the bounding box of the detected face in the same location, while the identity of the subject indeed changes. To make face extraction more robust, we employed the pre-trained FaceNet algorithm which returned an embedding vector, or an array of values, which concisely described each face identity. Our justification was that faces of the same person would naturally be very similar from one frame to the next. To quantify the similarity of the subjects’ identities to determine if the face detection algorithm was viewing the same person, we used a metric called cosine distance and divided up sequences where the distance failed to meet a specified threshold. This threshold was arrived at by performing statistical analysis on same- and different-identity faces and was iteratively relaxed until the threshold was equal to the optimal as seen in the Distribution 1 and Distribution 2 below. This optimal value maximized same-identity faces while minimizing different-identity faces and coincided with 95% confidence in the same-identity distribution and 5% in the different-identity distribution. This custom sorting algorithm, written in Python, was made highly performant by compiling to machine code using the Numba library. Once the faces were extracted into clean sequences, we could then feed them into the OpenFace algorithm (*OpenFace*, n.d.) which would further process them into the format we desired. All processed data were stored in HDF5 file format, totaling over 3 terabytes of data.



**Distribution 1**: Initial strict cosine distance

**Distribution 2**: Relaxed optimal cosine distance

To ensure there was no data leakage between train, validation, and test sets, we isolated the subject IDs to one of the three subsets, such that they appeared in their respective data subset and that set only. This was done for DFDC and CDF, while for FF++, we followed the convention set forth by the authors of the dataset and other researchers (Dessa, 2019). All datasets were balanced evenly between real and fake videos.

### **3.3.2 Model Experimentation**

The next stage involved experimental modeling. Upon initially putting together a model training script, we found that data load times were extremely slow. Our datasets were too large to load fully into memory, so to solve this issue, we built a fully custom multithreaded dataloader which assembled randomly shuffled video frames into minibatches and maintained a steady background stream of data to the models during the training process. Our first step, inspired by the work in (Tolosana, Romero-Tapiador, et al., 2020), was to meet or exceed their results predicting deepfakes in the CDF dataset using the Xception model architecture (Chollet, 2017) for in-distribution testing, where the training dataset and validation/test datasets are extracted from the same source data. Initial results were lackluster but were vastly improved once we rescaled our images to a higher resolution and introduced random mirror and rotation transformations. In the interest of generating a robust solution, we experimented with a vast array of network hyperparameters (options for fine-tuning) but found the defaults to be optimal using the AdaBelief optimizer(Zhuang, 2020/2021) and adding weight decay regularization. Continuing to use CDF as our benchmark dataset due to its superior quality, we then experimented with ensemble methods by eliminating the model prediction component in Xception, and feeding the pre-output values into an LSTM (Figure 6), a neural network designed for analyzing sequences. We achieved stellar results, boosting our AUC test metric from 0.935 to 0.981 for in-distribution testing.

A picture containing graphical user interface

Description automatically generated

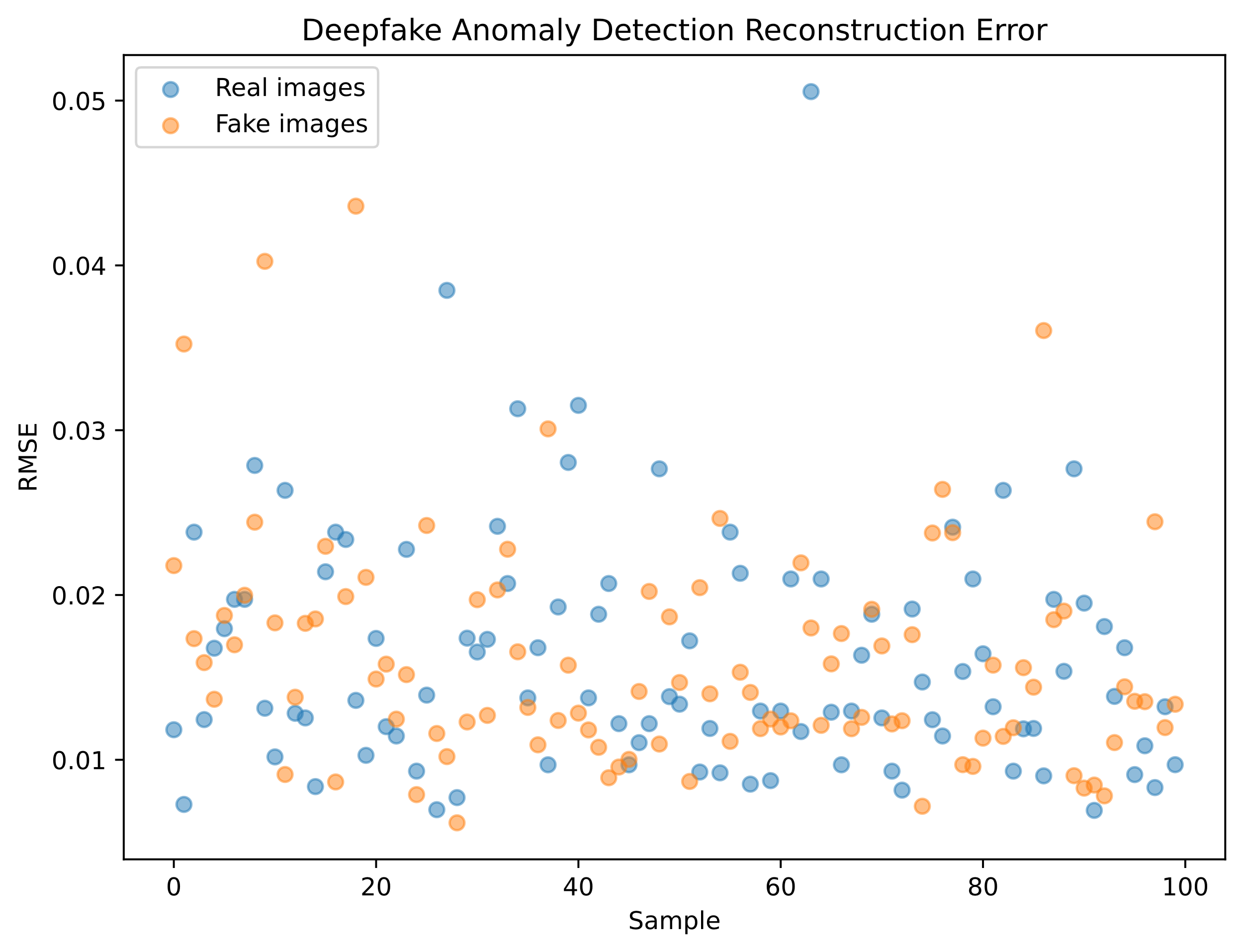
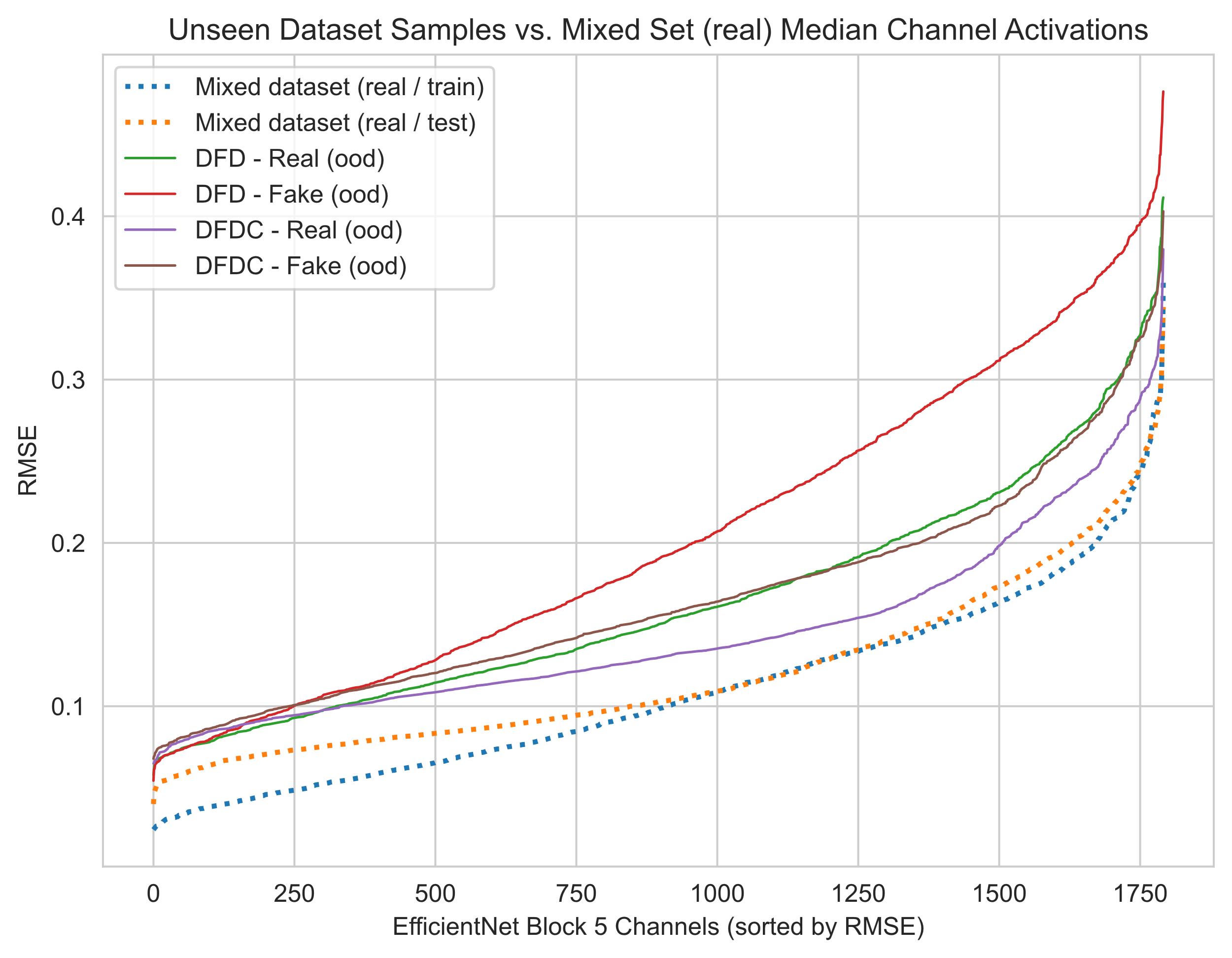
**Figure 6.** Illustration of Ensemble Metalearner

Expanding on this, we experimented with three other state-of-the-art image classification architectures, EfficientNet(Tan & Le, 2020), ResNeXt (Xie et al., 2017), and ResNeSt (Zhang et al., 2020). EfficientNet (b4) became our reference network going forward, as it outperformed Xception on both CDF and a mixed dataset consisting of CDF and FF++ video frames. While EfficientNet did add overhead in terms of training time and overall number of parameters, the impact was inconsequential. Conversely, ResNeXt and ResNeSt added significant overhead while underperforming EfficientNet on the mixed dataset, and so were therefore abandoned. All models were loaded with pretrained ImageNet weights, and all layers except for the very bottom were unfrozen. This was done intentionally in attempts to allow the network to discover salient features while preventing overfitting to the train data. We believe we succeeded as training with only the final layer unfrozen prevented the model from achieving acceptable results, while training with most of the network unfrozen, in many cases, resulted in superior test metrics over validation metrics.

It has been shown empirically that combining the power of two or more networks boosts overall performance, as witnessed prior with our success combining Xception and an LSTM. To test this further for our use case, we attempted two different ensemble architectures. The first ensemble consisted of six different EfficientNet modules, each trained on their respective dataset. The models were then fully frozen, output layers removed, and two fully connected layers were trained on the aggregated EfficientNet output embeddings. The second ensemble consisted of an Xception module and an EfficientNet module, both trained on the mixed dataset, consisting of images from all six datasets. These models were then frozen, and outputs were aggregated in a metalearner, similar to the first architecture. The second model was more lightweight and yielded slightly better results, at which point we moved forward with this architecture to ensemble with an LSTM module. The ensemble model generated embeddings which were then fed into the LSTM in 30-frame sequences. This boosted our classification performance significantly on in-distribution testing.

Throughout this experimentation stage, we discovered that while the models performed exceptionally well, they were unable to generalize to unseen forgery methods. While, to the best of our knowledge, no researchers have successfully provided an adequate solution to this challenging problem (Du et al., 2020) (Ganiyusufoglu et al., 2020), we attempted to do so through several avenues. Our first course of action, necessary for any machine learning model, was to calibrate the model’s output probabilities via temperature scaling. While doing so was imperative, we did not realize any gain in detecting out-of-distribution deepfakes. A second approach was to analyze the values generated within the model to determine if it was being subjected to anomalous or out-of-distribution data. The values for this analysis were calculated as follows:

where is the mean of the EfficientNet block 5 endpoint layer activation map for each sample , for each channel in the layer, and is the median of all mean activations per channel from real in-distribution samples. This showed some positive results, as we can see in figure 7 that the DFD-fake and DFDC-fake both exhibit higher RMSE values than their real counterparts. However, the DFD-real and DFDC-fake values nearly perfectly overlap, indicating that this analysis requires further work to be included in our solution. Finally, we attempted to identify deepfakes using a convolutional autoencoder. The theory is that the model learns how to effectively deconstruct and reconstruct real face images. When passing a fake image through the model, the hope is that it reconstructs poorly, at which point it can be flagged as an outlier. Though many variations in model complexity and dimensionality reduction were trained and



**Figure 8.** Autoencoder anomaly detection

**Figure 7.** Neural activation analysis

tested, no model seemed to provide any value in this space. As we can see in figure 8, the results from reconstructing both real and fake images completely overlap, and therefore there is no way to distinguish real from fake.

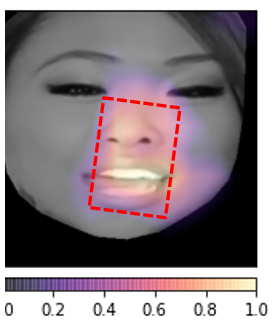
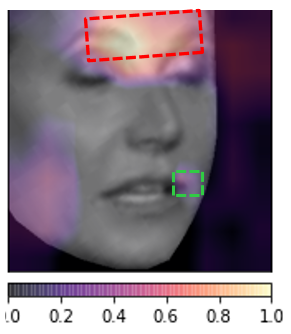
### **3.3.3 Model Interpretability**

To better understand our models, it was crucial to implement model interpretability algorithms such as GradCAM that offer visibility as to what regions of pixels positively

contributed to our final model predictions. This also gives us the ability to potentially improve on areas of weakness where we have False Positives or False Negatives that can be easily amended. We ultimately decided to pursue the Captum library to address this need as it was the most developed and well-documented package available online that also happened to be built on the PyTorch framework, and so would fit seamlessly into our own detection pipeline in RCC.

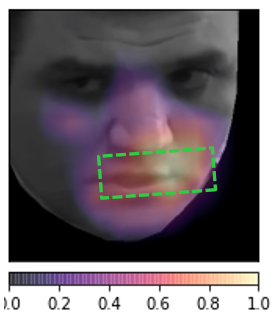
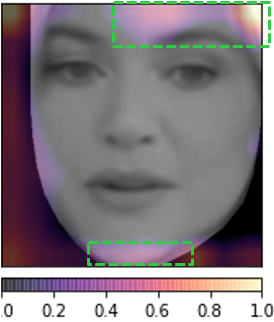
We initially experimented with several algorithms offered in Captum, including but not limited to Integrated Gradients, GradientSHAP, DeepLift, Layer GradCAM, Guided GradCAM, and Saliency. After extensive testing with sample data, we decided that Layer GradCAM produced the most visually informative outputs within the most reasonable amount of time. As a general rule, we believed that any interpretability algorithm that required several minutes to generate results on a single sequence of frames would not be conducive for research purposes

nor would it be viable as a webapp made available to others. Layer GradCAM works by assessing the magnitude of the gradients of each logit of class fake or real that feeds into the final convolutional layer of a neural network model such as XceptionNet or EfficentNet, where it then produces a coarse localization map highlighting the important regions. Large gradients generally indicate that the weights and biases of a particular neuron absolutely must move in a particular direction in order for the model to converge towards the minimum based on its optimization score, which in our case binary cross entropy. The algorithm converts the gradients associated with region of pixels to a score between 0 and 1, where 1 represents a very high contribution towards the prediction of either a fake or real image, and 0 represents a very low or absence of contribution (*A Review of Different Interpretation Methods in Deep Learning (Part 1: Saliency Map, CAM, Grad-CAM) | by Mohammadreza Salehi | Medium*, n.d.).



**Figure 9.** Example of True Positive (left) and True Negative (right) outputs from GradCAM based on EfficientNet prediction

**Figure 10.** Example of False Negative (left) and False Positive (right) outputs from GradCAM based on EfficientNet prediction



As we see in Figure 9 above on the left side, our detector performs well in identifying the blending artifacts created by Face2Face around the outside edges for the True Positive case. In the example of a True Negative in Figure 9 on the right, the model seems to focus on the nose and mouth, which makes sense given that is where most of the movement occurs in a video.

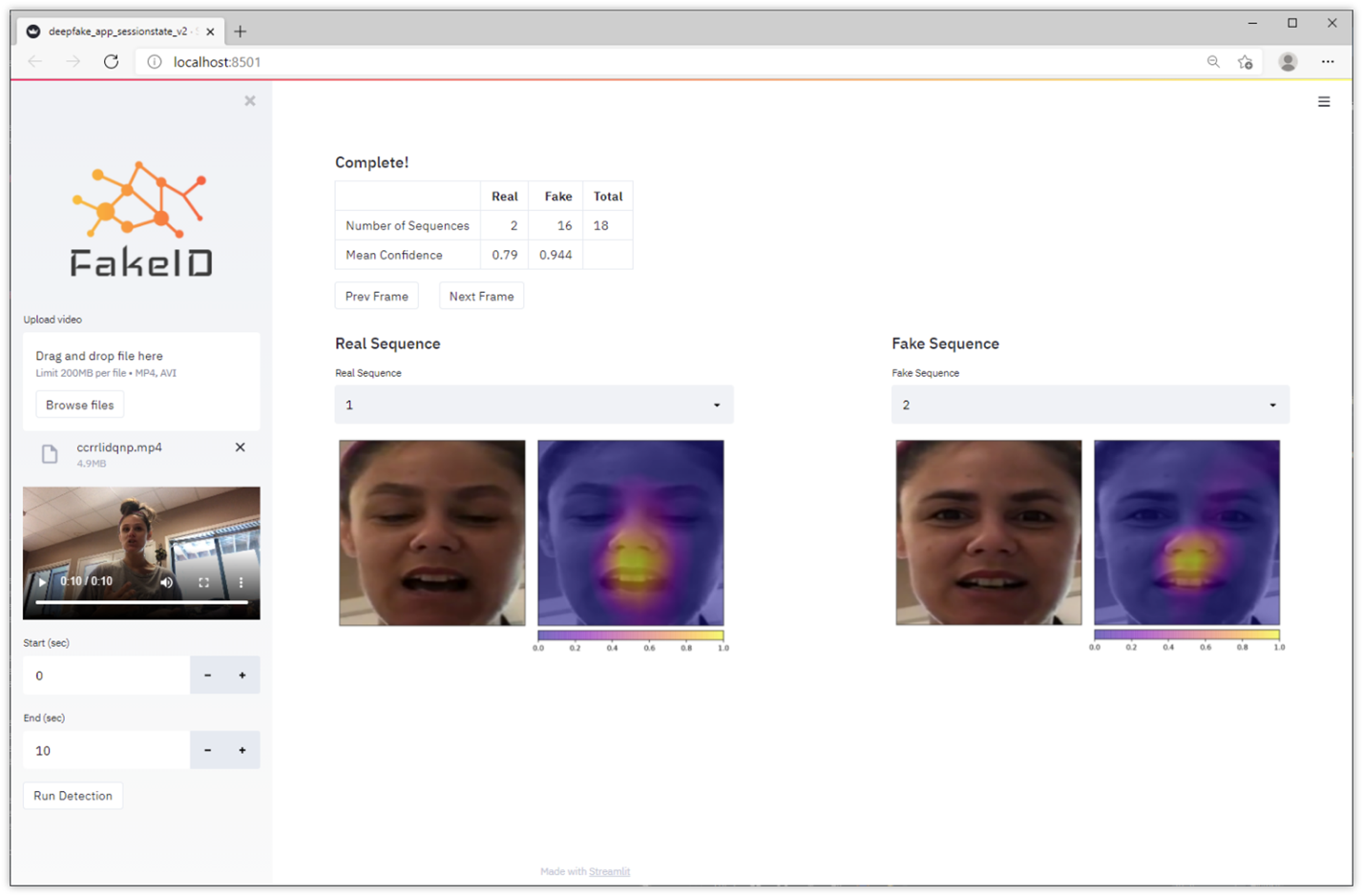
But the next two examples shown in Figure 10 are the most interesting - on the left we have an example of a failure to detect a fake image where the model did not place enough importance on the artifact at the bottom left edge of the face. On the right side of Figure 10, the model failed to detect a real image by focusing too much on the outside edge which could have turned out to be a natural blur or low-resolution frame of the video.

It should be evident now that GradCAM can help to illuminate areas where our detector can improve. We can find examples of frames where we perform the worst and augment our dataset with images of faces with these regions blacked out for better performance down the road.

### **3.3.4 Web Application**

Our front-end solution, titled FakeID, was built on the Streamlit platform. Streamlit was chosen for this task as it is built with data science dashboards in mind. Though it is relatively straightforward to design the dashboard, implement certain features, and incorporate all code, it is still a young platform and certain user-interface elements we required were not readily supported. This involved implementing some custom architecture to achieve operability, however the performance at present remains undesirable until the Streamlit team rolls out future functionality.

**Figure 11.** “FakeID” with detection results displayed



**4. Findings**

**In-Distribution Model Results:**

**Table 3.** Top in-distribution image classifier results

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Dataset** | **Unfrozen** | **Epochs** | **Best Epoch** | **Best Val (AUC)** | **Val AUC (mu)** | **Val AUC (sd)** | **Test AUC** |
| XceptionNet | CDF | 5: | 56 | 16 | 0.927 | 0.840 | 0.057 | 0.9248 |
| EfficientNet-b4 | CDF | 1: | 28 | 5 | 0.934 | 0.879 | 0.039 | 0.9332 |
| EfficientNet-b4 | CDF | 1: | 28 | 8 | 0.940 | 0.912 | 0.022 | 0.9351 |
| **EfficientNet-b4** | **CDF** | **1:** | 26 | **4** | **0.943** | - | - | **0.9380** |
| **XceptionNet** | **Mixed** | **5:** | **38** | **22** | **0.929** | **0.916** | **0.009** | **0.9442** |
| **EfficientNet-b4** | **Mixed** | **1:** | **25** | **10** | **0.948** | **0.934** | **0.009** | **0.9547** |

In Table 3, we find a summary of the most performant transfer learning models for a given training dataset and set of hyperparameters (most of which are not listed for easier viewing). For most models that were trained and tested on an out-of-sample dataset, we used either Adam or AdaBelief as our optimizer, with a learning rate of either 0.001 or 0.0001. L2 (weight decay) was typically set to 0.01, 0.001, or 0.0001, while Gamma was kept mostly constant at 0.9. Step sizes varied between 1 and 3. One of the more interesting hyperparameters that we tuned on was how many layers of the transferred learning model were unfrozen and retrained to pick up on a more optimized set of weights to detect GAN/autoencoder artifacts and other latent features that would be predictive of an AI altered image/video. Generally speaking, we found models to be most performant when all layers were unfrozen except for first meta-block in EfficientNet and the first five layers in Xception.

It is evident that the EfficientNet-b4-adv model was the most performant when trained and tested on the CDF dataset, currently the most state-of-the-art and prolific dataset in terms of manipulation techniques used and sheer volume of deepfake videos. We see that the model quickly converges to a minimum during the training process after only several epochs, with a best validation AUC of 0.943 and test AUC of 0.9380. This actually beats the AUC reported by Tolosana et al. (2020) of 0.836 when using XceptionNet (Tolosana, Vera-Rodriguez, et al., 2020). An even more surprising result is that an instance of XceptionNet that was trained for 56 epochs yielded a test AUC of .9248 which also outperforms the reported metric from Tolosana et al. (Tolosana, Vera-Rodriguez, et al., 2020). While not indicated, it is possible they were training and testing on a previous version of the Celeb-DF dataset which would explain their significantly inferior results.

We proceeded to test different models on a mixed dataset composed of the top deepfake datasets such as CDF and FF++ to see if there were ways to improve the generalizability of our discriminators on unseen deepfake videos from the “wild”. We saw increased performance in the Test AUC for both XceptionNet and EfficientNet-b4, with the former increasing by nearly 0.02 points from 0.928 to 0.9442, and the latter increasing by nearly 0.02 points from 0.9380 to 0.9547. This is an impressive result considering the current research landscape and performance of models cited in our initial literature review across several authors. It is worth noting that the average Validation AUC across all epochs was also the highest, and the standard deviation of the Validation AUC was lowest for the EfficientNet-b4 architecture when tested on either CDF or the Mixed dataset, suggesting the superior stability of predictions when trained over many epochs and confidence of the accuracy provided by using this architecture compared to others.

**Figure 12.** Summary of Test AUC for XceptionNet, EfficientNet, Ensemble, and Ensemble+LSTM for In-Distribution Data. Mixed: (Train/Test/Validation, in-distribution) Dataset comprised of randomly sampled videos from curated CDF and FF++ full datasets (Deepfakes, Face2Face, FaceSwap, FaceShifter, Neural Textures). DFDC: (Holdout, out-of-distribution) Deepfake Detection Challenge, random sample from full dataset. DFD: (Holdout, out-of-distribution) FF++ Deepfake Detection; curated full dataset (original new actors different from FF++, not YouTube videos) ​

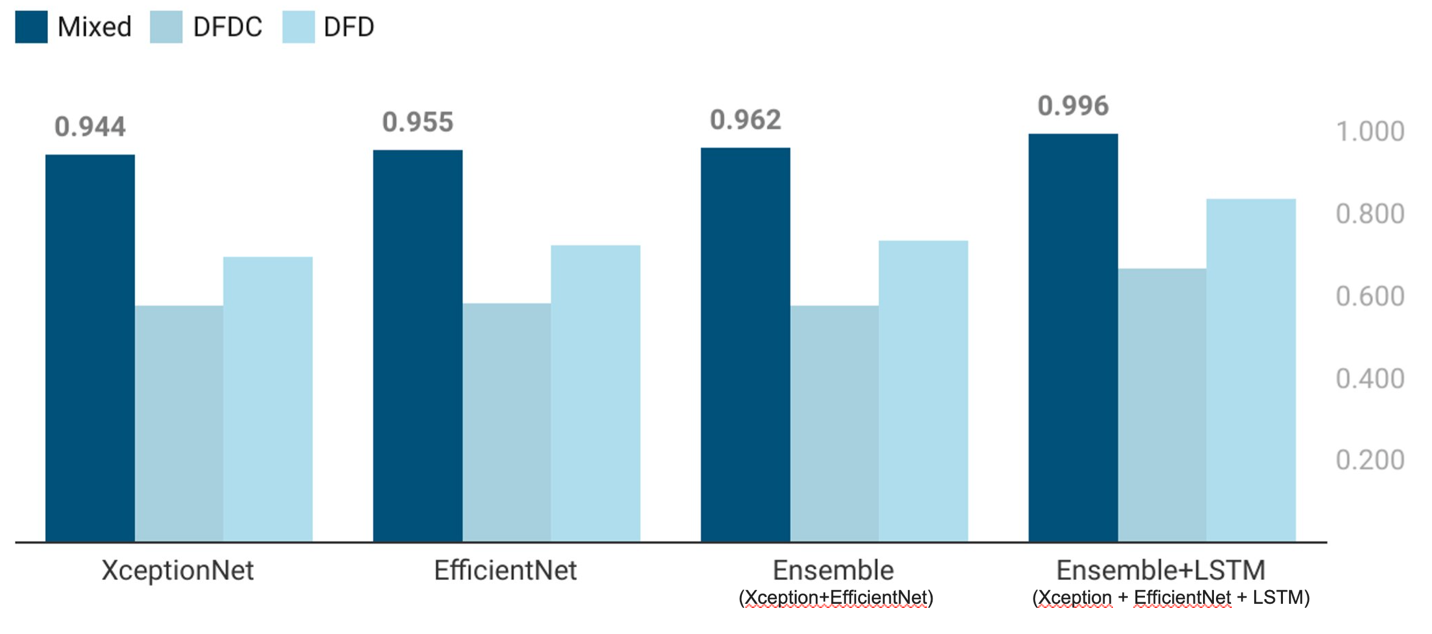


Figure 12 illustrates the results of our implementations of XceptionNet, EfficientNet, Ensemble, and Ensemble combined with LSTM on In-Distribution Test Sets composed of a custom curated Mixed dataset that we created, comprised mainly of sample videos from the FF++ and CDF datasets. The AUC performance quite expectedly learned all of the salient features of the underlying datasets after extensive testing, similar to what we have witnessed in prior published works as noted by Tolosana et al (2020) (Tolosana, Romero-Tapiador, et al., 2020). We achieved a precision of 0.982, recall of 0.963, and F1 of 0.972, with a false positive rate of 0.04. While we were able to achieve a satisfactory result after introducing our ensemble metalearner as well as the ensemble + LSTM solution, it was imperative to confirm whether our best detectors could stand up to in-the-wild (ITW) datasets that have never been seen before by these models during the training process.

**Figure 13.** Summary of Train and Validation AUC for each model tested on out-of-distribution data without data augmentation and adding Flickrfaces.

Mixed: (Train/Test/Validation, in-distribution) Dataset comprised of randomly sampled videos from curated CDF and FF++ full datasets (Deepfakes, Face2Face, FaceSwap, FaceShifter, Neural Textures)​. DFDC: (Holdout, out-of-distribution) Deepfake Detection Challenge, random sample from full dataset.DFD: (Holdout, out-of-distribution) FF++ Deepfake Detection; curated full dataset (original new actors different from FF++, not YouTube videos).

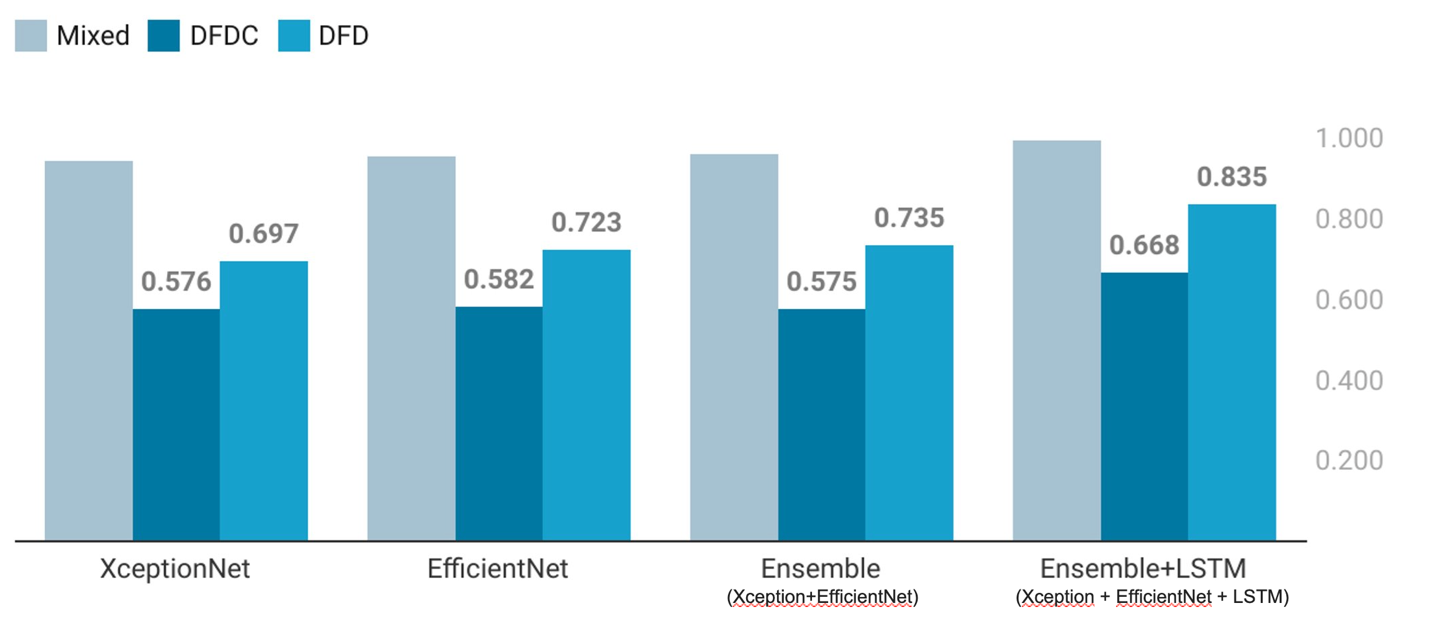
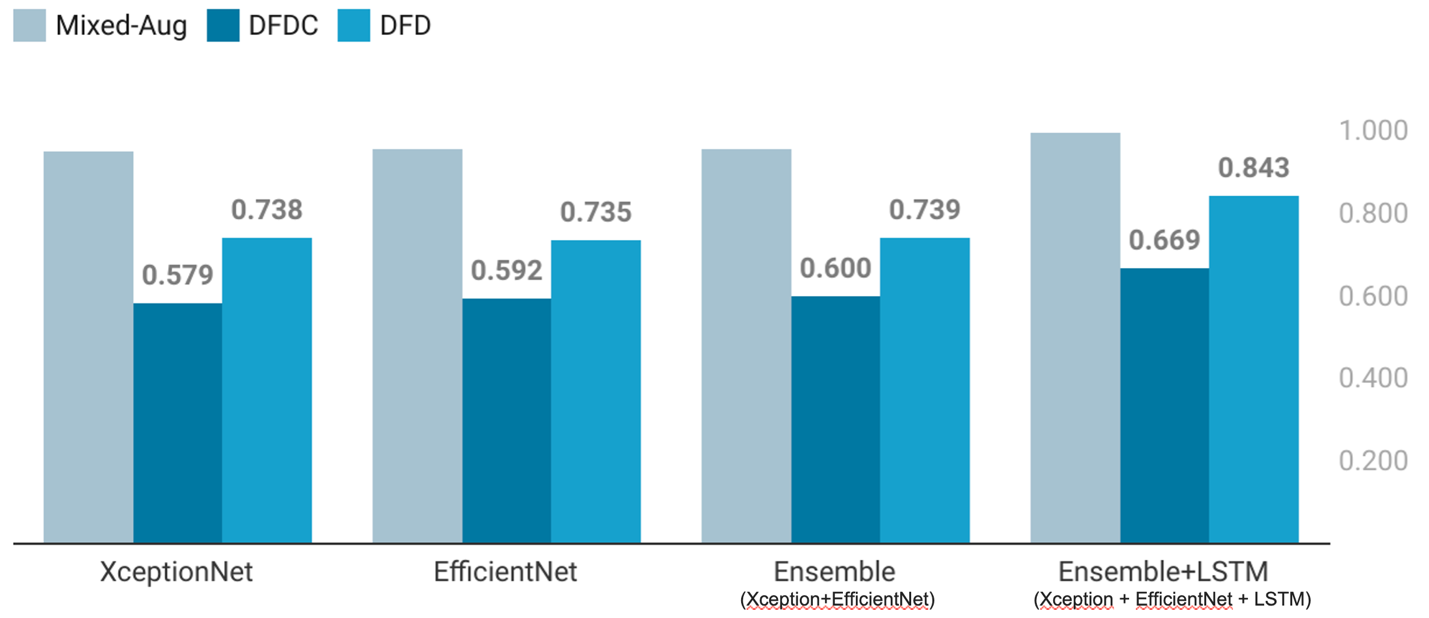


  Figure 13 illustrates the results of our implementations of XceptionNet, EfficientNet, Ensemble, and Ensemble combined with LSTM on two OOD datasets – DFDC, and DFD. These datasets are fully curated with videos never seen before by our models and performance suffered when compared to the initial In-Distribution test performance (Figure 10). Due to the inferior construction of the DFDC dataset, we focused solely on DFD for our analysis. Most notably, the false positive rate was 0.46, which indicates that the model misclassified nearly half of all real videos as fake. We achieved a precision of 0.588, recall of 0.869, and F1 of 0.701, with a false positive rate of 0.46. We understood this to mean that quality of the real videos in DFD was significantly different from the quality of the real videos in our training data. That is, our authentic video training data was not varied enough in quality.

**Figure 14.** Summary of train and validation AUC for each model tested on out-of-distribution data after data augmentation and adding Flickrfaces.

Mixed: (Train/Test/Validation, in-distribution) Dataset comprised of randomly sampled videos from curated CDF and FF++ full datasets (Deepfakes, Face2Face, FaceSwap, FaceShifter, Neural Textures)​. DFDC: (Holdout, out-of-distribution) Deepfake Detection Challenge, random sample from full dataset. DFD: (Holdout, out-of-distribution) FF++ Deepfake Detection; curated full dataset (original new actors different from FF++, not YouTube videos).



To combat this, we combined our mixed dataset with high quality real face images from a dataset called FlickrFaces (*NVlabs/Ffhq-Dataset*, 2019/2021). We generated 6 versions of the FlickrFaces dataset, all varying in image compression levels ranging from quality 100 to quality 10. We were able to achieve a precision of 0.651, recall of 0.848, and F1 of 0.736. While AUC only increased slightly, precision increased from 0.588 to 0.651 at the expense of a slight decrease in recall, and our false positive rate decreased from 0.46 to 0.34. This was a notable achievement because while having a high recall is important for this application, the prevalence of false deepfake detections would render the solution impractical. In this case, a 0.011 decrease in recall to 0.848 while simultaneously increasing precision by nearly 0.07 points to 0.651, with F1 score increasing by almost 0.035 points to 0.736 is justifiable. Compared to other SoTA detection models that we have found during our literature review, our solution appears to be quite robust and generalizable to unseen datasets, although it is unfair to make a true comparison due to our inclusion of CDF in our train set, while most authors elected to use CDF as their test set. The fact remains that our performance is comparable to other authors.

**Table 4.** Summary of In-Distribution and Out-of-Distribution test performance by SoTA papers compared to our results.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Set** | **OOD** | **AUC** |
| **Tolosana et al (2020)** | CDF |  | 0.999 |
| **Oscar et al (2020)** | CDF |  | 0.997 |
| **Ours (ID)** | Mixed |  | 0.996 |
| **Ours (OOD)** | DFD | 🗸 | 0.843 |
| **Lingzhi et al (2020)** | CDF | 🗸 | 0.806 |
| **Yuval et al (2020)** | CDF | 🗸 | 0.660 |
| **Dessa (2019)** | FF++ | 🗸 | 0.630 |

To take this one step further, we have noted a few authors that have attempted ensembling these models as well to see if we can get further improved performance.

One last check we wanted to make was to see how generalizable our top discriminator models truly were when presented deepfake videos from the “wild”, and so we proceeded to test the performance of our top models against videos found on YouTube that were created in either 2020 or 2021 using state-of-the-art manipulation techniques and enhancements such as those found in the DeepFaceLabopen-source tool created for consumer use.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Tolosana et al (2020)** | **1st Place DFDC Winner** | **Dessa (2019)** | **Wang et al (2020)** | **Our Method** |
| 1.Utilizes CNN’s and image preprocessing techniques (augmentation, cropping) | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 |
| 2.Trained on datasets that use Face2Face, FaceSwap, Deepfake, and NeuralTextures techniques | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 |
| 3.Transfer learning using Xception | 🗸 |  | 🗸 | 🗸 | 🗸 |
| 4.Transfer learning using EfficientNet |  | 🗸 |  |  | 🗸 |
| 5.Trained on Mixed Datasets for greater generalizability |  |  | 🗸 |  | 🗸 |
| 6.Transfer learning using 3D CNNs |  |  |  | 🗸 |  |
| 7.Utilizes LSTM to process sequences of frames |  |  |  |  | 🗸 |
| **8.Ability to detect multiple subjects per frame** |  |  |  |  | **🗸** |
| **9.Uses Ensemble Meta-learner that increases performance and allows plug/play new models** |  |  |  |  | **🗸** |
| **10.Developed a functional app with built-in model interpretability algorithms** |  |  |  |  | **🗸** |

# **5. Conclusion**

**Table 5**. Achievement highlights in comparison to SoTA methods

Table 5 depicts a list of highlighted features from four SoTA journals (no. 1-7) and the highlights this group has achieved (no. 8-10). Throughout this project, our group was able to 1) develope a brand-new cropping technique that accurately tracks the identity of two different faces in the same video​ 2) incorporate EfficientNet in transfer learning process​ 3) create a strong ensemble meta-learner that feeds the feature vectors of multiple network models into a final fully connected layer as opposed to traditional soft/hard majority voting featured in packages such as DeepStack 4) use mixed datasets to allow for greater generalizability during evaluation phase​ 5) achieve an AUC of 0.843 and 0.669 on 2 out of OOD datasets for using an Ensemble + LSTM solution.

# **6. Recommendations and Future Work**

Detection of deepfake contents is increasingly challenging as the synthesis of these materials are getting more sophisticated. To combat this problem, 3D CNN and convolutional LSTM models that could capture spatial-temporal features of inputs have been proven to add a layer of robustness into detection methods (Ganiyusufoglu et al., 2020). The addition of spatio-temporal detection features in 3D CNN models allowed the authors to detect new unseen fake image sequences compared to sequential and spatial methods such as image feature exactors coupled with LSTM modules (Ganiyusufoglu et al., 2020). The authors trained XceptionNet, EfficientNet, and RNN models as a baseline to their 3D CNN model on FF++ datasets. To ascertain the precision of 3D CNN in capturing temporal abnormalities, the authors compared a drop in precision scores between models that were trained on all five generation methods of FF++ and models where one single method is left out (Ganiyusufoglu et al., 2020). The 3D CNN model has a drop of 0.049% in precision score whereas the other three methods had a drop in 0.069% to 0.10% in precision score (Ganiyusufoglu et al., 2020).

Recurrent All-Pairs Field Transforms (RAFT) an Optical Flow architecture is another deep learning method that could potentially be used for future works. Optical flow is used to detect fast moving objects, motion blurs, occlusions, and textureless objects in videos as it optimizes the estimation per-pixel motion between frames (Teed & Deng, 2020). Teed and Deng introduced the RAFT architecture to increase generalizability in model training, they were able to achieve a 40% error reduction from prior deep network trained on the same data. Additionally, RAFT has an error rate of 5.10% and has the capability to optimize training speed, it trains with ten times fewer iterations compared to other architectures (Teed & Deng, 2020).

Another proposed model for future exploration is Siamese Network. This model is used in facial recognition systems to detect face likeness between two real face images or one real and one fake face images (Hao & Pei, 2019). The model checks for “two real” facial features from the input, it detect facial uniqueness. Fourth deep learning technique that would aid in future studies is Online Deep Learning (ODL) (Sahoo et al., 2017). This deep learning model allows the optimization of predictive models over a stream of data instances in a sequential manner (Sahoo et al., 2017). This enables online learning to be highly scalable and memory efficient (Sahoo et al., 2017).

Lastly, future works should include a wide range of datasets that represents different groups of people. Demographics in CDF dataset contains majority of Caucasians and people from 50 – 59 years of age. Diversifying race and age for model training would be helpful in teaching the algorithm to pick up on different types of skin tones, facial features, and facial aging. An addition of datasets like Deeper Forensics and latest-generation DFDC with higher quality deepfakes more diverse augmentation such as full-face swapping would help with model predictive power.

# **7. References**

A Review of Different Interpretation Methods in Deep Learning (Part 1: Saliency Map, CAM, Grad-CAM) | by Mohammadreza Salehi | Medium. (n.d.). Retrieved March 8, 2021, from https://medium.com/@mrsalehi/a-review-of-different-interpretation-methods-in-deep-learning-part-1-saliency-map-cam-grad-cam-3a34476bc24d

AI in healthcare: How artificial intelligence can help us fight pandemics. (2020, April 9). ThinkAutomation. https://www.thinkautomation.com/bots-and-ai/ai-in-healthcare-how-artificial-intelligence-can-help-us-fight-pandemics/

Bonettini, N., Cannas, E. D., Mandelli, S., Bondi, L., Bestagini, P., & Tubaro, S. (2020). Video Face Manipulation Detection Through Ensemble of CNNs. ArXiv:2004.07676 [Cs, Eess]. http://arxiv.org/abs/2004.07676

Chollet, F. (2017). Xception: Deep Learning with Depthwise Separable Convolutions. ArXiv:1610.02357 [Cs]. http://arxiv.org/abs/1610.02357

Dessa. (2019, November 24). Towards Deepfake Detection That Actually Works. Dessa. https://www.dessa.com/post/deepfake-detection-that-actually-works

Dolhansky, B., Bitton, J., Pflaum, B., Lu, J., Howes, R., Wang, M., & Ferrer, C. C. (2020). The DeepFake Detection Challenge (DFDC) Dataset. ArXiv:2006.07397 [Cs]. http://arxiv.org/abs/2006.07397

Du, M., Pentyala, S., Li, Y., & Hu, X. (2020). Towards Generalizable Deepfake Detection with Locality-aware AutoEncoder. ArXiv:1909.05999 [Cs]. http://arxiv.org/abs/1909.05999

FaceForensics++ & Survey of Multi-Modal techniques | by wer deepfakers | Medium. (n.d.). Retrieved March 8, 2021, from https://medium.com/@werdeepfakers/faceforensics-survey-of-multi-modal-techniques-7b637fc161d0

Fast MTCNN detector (~55 FPS at full resolution). (n.d.). Retrieved March 8, 2021, from https://kaggle.com/timesler/fast-mtcnn-detector-55-fps-at-full-resolution

Fridrich, J., & Goljan, M. (2002). Practical steganalysis of digital images: State of the art. 4675, 1–13. https://doi.org/10.1117/12.465263

Ganiyusufoglu, I., Ngô, L. M., Savov, N., Karaoglu, S., & Gevers, T. (2020). Spatio-temporal Features for Generalized Detection of Deepfake Videos. ArXiv:2010.11844 [Cs]. http://arxiv.org/abs/2010.11844

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative Adversarial Networks. ArXiv:1406.2661 [Cs, Stat]. http://arxiv.org/abs/1406.2661

Güera, D., & Delp, E. J. (2018). Deepfake Video Detection Using Recurrent Neural Networks. 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 1–6. https://doi.org/10.1109/AVSS.2018.8639163

Hao, H., & Pei, M. (2019). Face Liveness Detection Based on Client Identity Using Siamese Network. ArXiv:1903.05369 [Cs]. http://arxiv.org/abs/1903.05369

Kupyn, O., Budzan, V., Mykhailych, M., Mishkin, D., & Matas, J. (2018). DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks. ArXiv:1711.07064 [Cs]. http://arxiv.org/abs/1711.07064

Li, Y., Yang, X., Sun, P., Qi, H., & Lyu, S. (2020). Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics. ArXiv:1909.12962 [Cs, Eess]. http://arxiv.org/abs/1909.12962

Maras, M.-H., & Alexandrou, A. (2019). Determining authenticity of video evidence in the age of artificial intelligence and in the wake of Deepfake videos. The International Journal of Evidence & Proof, 23(3), 255–262. https://doi.org/10.1177/1365712718807226

Marra, F., Gragnaniello, D., Verdoliva, L., & Poggi, G. (2019). Do GANs Leave Artificial Fingerprints? 2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), 506–511. https://doi.org/10.1109/MIPR.2019.00103

Mittal, T., Bhattacharya, U., Chandra, R., Bera, A., & Manocha, D. (2020). Emotions Don’t Lie: An Audio-Visual Deepfake Detection Method Using Affective Cues. ArXiv:2003.06711 [Cs]. http://arxiv.org/abs/2003.06711

NVlabs/ffhq-dataset. (2021). [Python]. NVIDIA Research Projects. https://github.com/NVlabs/ffhq-dataset (Original work published 2019)

OpenFace. (n.d.). Retrieved February 5, 2021, from https://cmusatyalab.github.io/openface/

Rössler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., & Nießner, M. (2019). FaceForensics++: Learning to Detect Manipulated Facial Images. ArXiv:1901.08971 [Cs]. http://arxiv.org/abs/1901.08971

Sahoo, D., Pham, Q., Lu, J., & Hoi, S. C. H. (2017). Online Deep Learning: Learning Deep Neural Networks on the Fly. ArXiv:1711.03705 [Cs]. http://arxiv.org/abs/1711.03705

Tan, M., & Le, Q. V. (2020). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. ArXiv:1905.11946 [Cs, Stat]. http://arxiv.org/abs/1905.11946

Teed, Z., & Deng, J. (2020). RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. ArXiv:2003.12039 [Cs]. http://arxiv.org/abs/2003.12039

The State of Deepfakes in 2020. (n.d.). Skynet Today. Retrieved March 8, 2021, from https://www.skynettoday.com/overviews/state-of-deepfakes-2020

Tolosana, R., Romero-Tapiador, S., Fierrez, J., & Vera-Rodriguez, R. (2020). DeepFakes Evolution: Analysis of Facial Regions and Fake Detection Performance. ArXiv:2004.07532 [Cs]. http://arxiv.org/abs/2004.07532

Tolosana, R., Vera-Rodriguez, R., Fierrez, J., Morales, A., & Ortega-Garcia, J. (2020). DeepFakes and Beyond: A Survey of Face Manipulation and Fake Detection. ArXiv:2001.00179 [Cs]. http://arxiv.org/abs/2001.00179

Westerlund, M. (2019). The Emergence of Deepfake Technology: A Review. Technology Innovation Management Review, 9(11), 40–53. https://doi.org/10.22215/timreview/1282

Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated Residual Transformations for Deep Neural Networks. ArXiv:1611.05431 [Cs]. http://arxiv.org/abs/1611.05431

Zhang, H., Wu, C., Zhang, Z., Zhu, Y., Lin, H., Zhang, Z., Sun, Y., He, T., Mueller, J., Manmatha, R., Li, M., & Smola, A. (2020). ResNeSt: Split-Attention Networks. ArXiv:2004.08955 [Cs]. http://arxiv.org/abs/2004.08955

Zhuang, J. (2021). Juntang-zhuang/Adabelief-Optimizer [Jupyter Notebook]. https://github.com/juntang-zhuang/Adabelief-Optimizer (Original work published 2020)

# **8. Appendix**

**Appendix 1**

One of the key innovations in this implementation that deserves to be recognized is the custom PyTorch dataloader which decreased train time for one epoch on Xception from 17 hours to about 2. Every dataset we trained on was well over 100gb, and therefore would simply not fit into memory. For this reason, we had to read in data over the network on the fly, an extremely inefficient operation. This dataloader addressed efficiency in two primary ways. First, it was designed with true multiprocessing architecture so that parallel reads from the HDF5 dataset files occurred. A second level of multiprocessing was enacted so that additional minibatches could be buffered in the background while the model trained on the data that was already in memory. The buffered data was stored in a buffer queue and the training data was stored in a minibatch queue. Once the minibatch queue emptied out, the buffer queue would reload the minibatch queue after which the buffer queue would fetch new data from disk while the model trained on the refreshed in-memory minibatch data. This module was critical to the outcome of our solution.