SIT724 Research Report: Comprehensive Framework for Deepfake Detection using custom CNN

Submitted as Research Report in SIT724

SUBMISSION DATE

T1-2024

Aryan Bagoria

STUDENT ID 221071501

COURSE - Bachelors of Software Engineering Honours (S464)

Supervised by: Dr Adnan Anwar

Abstract

Deepfakes, initially were introduced in 2017 as a simple face-swapping technique [1], have evolved into a sophisticated technology capable of generating highly realistic multimedia content that can deceive human perception. This paper presents a comprehensive deepfake detection solution that combines the strengths of two distinct approaches: a Multi-Task Cascaded Convolutional Neural Network (MTCNN) for precise facial feature identification and a custom Convolutional Neural Network (CNN) model adept at extracting intricate features. The MTCNN accurately locates facial landmarks like eyes, nose, and mouth, while the custom CNN model leverages its deep learning architecture to capture subtle irregularities indicative of manipulation. By integrating these methodologies, the proposed solution amplifies its ability to detect even the most sophisticated deepfakes, offering higher accuracy and resilience in identifying fabricated media. The unified framework's robustness and reliability make it a promising solution for mitigating the growing threat of deepfakes in various domains.

1. Introduction

1.1 Problem Description

The rise of deepfake technology poses a serious threat to the authenticity of online visual content. Current methods like manually inspecting videos and using basic deep learning models just aren't cutting it against the diverse and rapidly advancing deepfake techniques out there. However, a promising solution has emerged in the form of multi-modal analysis. This approach combines various technologies like face detection with MTCNN and customised CNN models. By integrating different methods, multi-modal analysis provides a more comprehensive defence against deepfakes. It improves accuracy, resilience, and scalability compared to techniques used individually. As deepfake methods continue evolving, multi-modal analysis can adapt and overcome the limitations of any single technique. It represents a crucial step forward in our ability to detect and mitigate deepfake manipulations.
Relying solely on manual inspection is impractical for large-scale use due to the time and resources required. Not to mention the subjectivity that can lead to inconsistencies in identifying deepfakes. Simple deep learning has potential but gets held back by using extensive, unbiased datasets to work well. And focusing only on facial artefacts misses alterations to other parts of images or video. Multi-modal analysis solves these issues by putting multiple technologies together into one holistic solution. It addresses scalability concerns, enhances accuracy through cross-verification, and provides comprehensive protection that goes beyond just facial recognition. This integrated approach simply works better against deepfakes, enhancing accuracy, robustness, and scalability.

1.2 Aims of the Report

The key aims of this report are as follows:

- Deep Learning Model :- Developing an effective deep learning model for detecting deepfake media content (images or videos) by accurately classifying input media as real or deepfake.
- Architectures and Techniques :- Exploring different deep learning architectures, techniques (e.g., convolutional neural networks, transfer learning, data augmentation), and optimizations to improve the model's performance in terms of accuracy, precision, recall, and F1-score.
- Multi-modal Integration :- Integrating multi-modal analysis approaches, such as combining a custom CNN model with the MTCNN (Multi-Task Cascaded Convolutional Networks) algorithm for precise facial feature identification, to enhance the overall deepfake detection capabilities.

- Minimizing overfitting :- Minimizing false positives and false negatives in deepfake detection by fine-tuning the model based on performance metrics from the training set, ensuring continuous improvement and adaptability to evolving deepfake techniques.
- Evaluating Performance :-Evaluating the proposed solution's performance through rigorous testing, data analysis, and metrics such as accuracy, confusion matrices, and ROC curves, to validate the model's effectiveness and identify areas for further improvement.

1.3. Deepfake technology and its implications

Deepfake technology marks a notable advancement in artificial intelligence and digital media. It involves using AI-driven algorithms, particularly GANs, to produce highly convincing but altered images and videos. This technology has evolved from simple face-swapping techniques to producing almost indistinguishable fake content, as detailed in "A Comparative Analysis of Deep Fake Techniques"[2]. Such progress has enabled applications for entertainment, yet it also raises serious concerns like misinformation and identity manipulation.

The potential misuse of deepfake technology for criminal purposes has made the development of effective detection methods a critical area of research. The challenge lies not only in the sophistication of the technology itself but also in the ever-evolving nature of the techniques used to create deepfakes. As deepfake creation methods become more advanced, the task of detection becomes increasingly complex. Detection methods must evolve in parallel to effectively identify and mitigate the threats posed by deepfakes.The need for these detection methods becomes even more critical when considering how deepfakes can be used for spreading misinformation and manipulating facts. Being able to reliably spot these fakes is essential for preserving the trustworthiness and integrity of digital media and information.

In response to this challenge, researchers and technologists have developed a range of detection techniques, leveraging advancements in machine learning, image processing, and computer vision. These techniques vary in approach and complexity, from analysing visual inconsistencies to employing deep neural networks trained to recognize the subtle signs of digital manipulation. The ongoing research in this area, as reflected in the academic papers provided, highlights a dynamic field striving to keep pace with the rapid advancements in deepfake technology.

2. Overview of the proposed approch

I'll be using a custom-built deep learning framework de- signed specifically for processing input images. This archi- tecture is intricately crafted to effectively identify deepfakes by incorporating a series of different layers, each with its own unique role within the model. I've carefully selected and arranged these layers to optimise how relevant features get extracted from the input data. In the next few sections, I have explained each layer used in my customised CNN model [3].

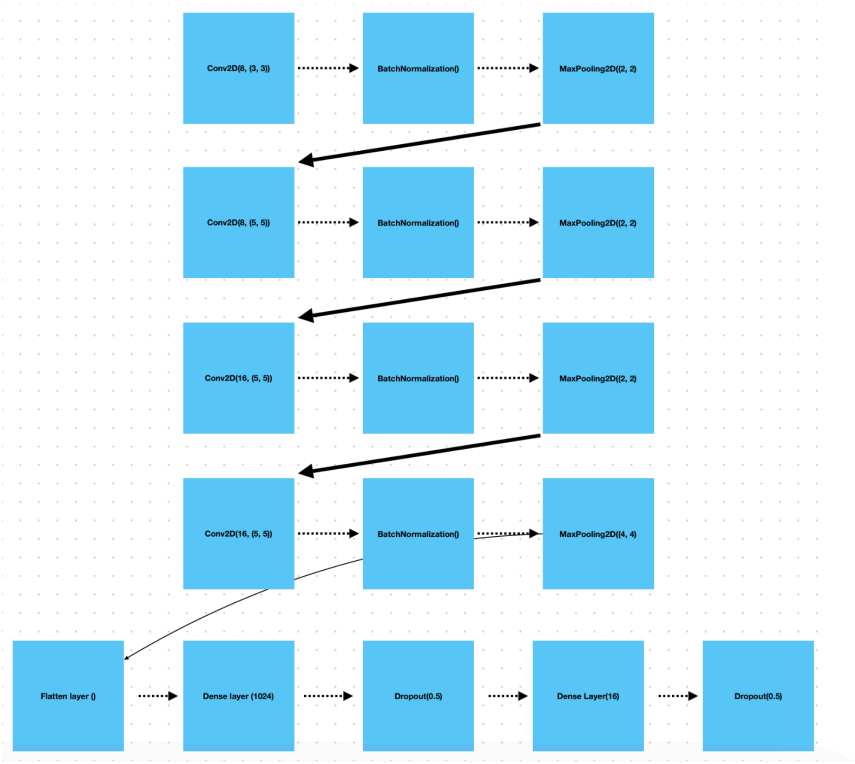

Figure 1: Deakin Campus Building.

2.1 Input Layer

I set the input layer to expect images of size 224x224 pixels with 3 colour channels (RGB). This size is commonly used in image classification tasks, providing a balance between computational efficiency and information retention.

2.2 First Convolutional Block

In the first convolutional block, I initiated with 8 filters of size 3x3. These filters are designed to detect basic patterns and features present in images, which is crucial for identifying manipulated content in deepfake images. Following the convolutional layer, I added batch normalisation to stabilise and accelerate the training process. This helps the

model converge faster and improves its ability to determine subtle differences in the input images.
Finally, I included a max-pooling layer with a 2x2 window to downsamples the feature maps, reducing their spatial dimensions while retaining essential information. This operation enhances computational efficiency and helps the model focus on the most relevant features.

2.3 Second Convolutional Block

Moving on to the second convolutional block, I introduced another set of 8 filters, this time with a larger size of 5x5. This allows the model to capture more complex patterns and features present in the images. Similar to the first block, The inclusion of batch normalisation and max-pooling layers after the convolutional layer ensures that the model remains stable during training and effectively captures important features while discarding irrelevant information.

2.4 Third Convolutional Block

In the third block, I increased the number of filters to 16 to capture even more intricate details in the images. The larger number of filters enables the model to extract higher-level features necessary for discriminating deepfake content. As before, batch normalisation and max-pooling were applied to stabilise the training process and further reduce the dimensionality of the feature maps, facilitating efficient feature extraction.

2.5 Fourth Convolutional Block

For the final convolutional block, I maintained the number of filters at 16 but introduced a larger max-pooling window size of 4x4. This larger window allows the model to capture broader spatial information, aiding in the detection of manipulated regions in the images. Batch normalisation is applied to stabilise the activations, while max-pooling further reduces the dimensionality of the feature maps before passing them to the dense layers.

2.6 Flatten and Dense Layers

Following the convolutional blocks, I flattened the feature maps into a 1D vector to prepare them for input into the dense layers. The first dense layer consists of 1024 neurons with ReLU activation, allowing the model to learn complex relationships between the extracted features.
To prevent overfitting, I introduced dropout with a rate of 0.5 after the first dense layer. Dropout randomly drops connections between neurons during training, preventing the model from becoming overly reliant on specific features or patterns present in the training

data. Another dense layer with 16 neurons and ReLU activation was added to further enhance the model's ability to capture intricate patterns present in deepfake images.

2.7 Output Layer

Finally, the output layer consists of a single neuron with sigmoid activation. This neuron outputs the probability that the input image is a deepfake, with values closer to 1 indicating a higher likelihood of manipulation.

2.7 MTCNN integration

To enhance the overall deepfake detection capabilities, the solution integrates a multimodal analysis approach by combining the custom CNN model with the MTCNN (Multi-Task Cascaded Convolutional Networks) algorithm [3][4]. The MTCNN algorithm is employed for precise facial feature identification, including the accurate localization of key facial features like the eyes, nose, and mouth.

Figure 2: MTCNN Detection.

This multimodal approach not only identifies human faces in digital photographs but also provides a confidence score for each detected face, quantifying the model's certainty about the prediction. This confidence score offers a quantitative measure of the model's performance. Additionally, the model generates bounding boxes around the recognized faces, outlining the relevant regions within the image. These bounding boxes are crucial for later stages of the process, such as cropping the face regions for data preparation and further analysis.

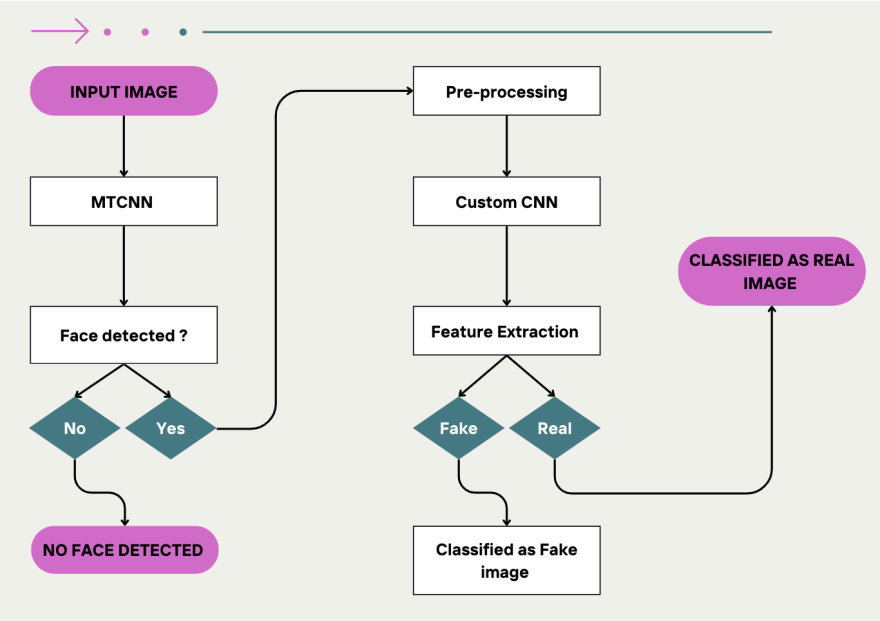

Figure 3: Model UML diagram.

By combining the custom CNN model and the MTCNN algorithm, the proposed solution leverages the strengths of both approaches, enabling accurate facial feature detection while simultaneously extracting intricate features from the input images. This synergy enhances the overall capability to detect even subtle manipulations, leading to improved accuracy and reliability in deepfake detection.

3. Data Preprocessing and Augmentation

The dataset selected for this project is sourced from Kaggle and consists of around 141,000 images, including both authentic and manipulated images. The initial phase focuses on image preprocessing, a crucial step to prepare the images for subsequent stages in model development. Image preprocessing serves as a foundational process, ensuring the optimal utilisation of the dataset for training the deep learning model. By undertaking image preprocessing, several key objectives are accomplished. Firstly, it standardises the image formats and dimensions, ensuring consistency across the entire dataset. This step typically involves resizing the images to a uniform resolution, which facilitates efficient processing and training.

3.1 Image Preprocessing

The preprocessing pipeline for the deepfake detection dataset involves several crucial steps to prepare the images for model training. Firstly, the images are loaded and resized to a standard resolution of 224x224 pixels using the Python Imaging Library (PIL). This resizing ensures consistent input dimensions across the dataset, which is essential for developing an effective deep learning model [5][7].
Next, the resized images are converted into NumPy arrays, which represent each image as a grid of pixels with intensity values ranging from 0 to 255. This conversion facilitates efficient numerical processing by the deep learning model. To optimise the model's learning process and enhance its generalisation capability, the pixel intensity values are then normalised by dividing each value by 255.0, scaling them to a range between 0 and 1.

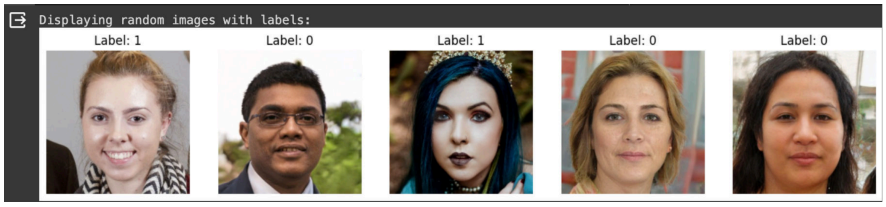

Figure 4: Merged dataset with labelling.

Data augmentation techniques are also applied to artificially increase the size and diversity of the training dataset. These include operations such as random rotations, flips, crops, and brightness/ contrast adjustments. Data augmentation helps expose the model to more variations of the data during training, reducing over-fitting and improving generalisation.
The dataset is subsequently divided into two distinct sets: real images and fake (manipulated) images. Each image is assigned a label, with real images labeled as 0 and fake images labeled as 1. This labelling scheme allows the model to distinguish between genuine and manipulated images during training and evaluation.
Finally, the real and fake image sets, along with their corresponding labels, are combined into a unified dataset. This combined dataset serves as the foundation for training the

deepfake detection model, enabling it to learn the patterns and features necessary to accurately classify images as real or fake.

3.2 Data Splitting

Data is split into three distinct sets: the training set, the validation set, and the test set. This strategic division ensures that the model is exposed to diverse data scenarios, enhancing its ability to generalise well to unseen data and avoid overfitting.

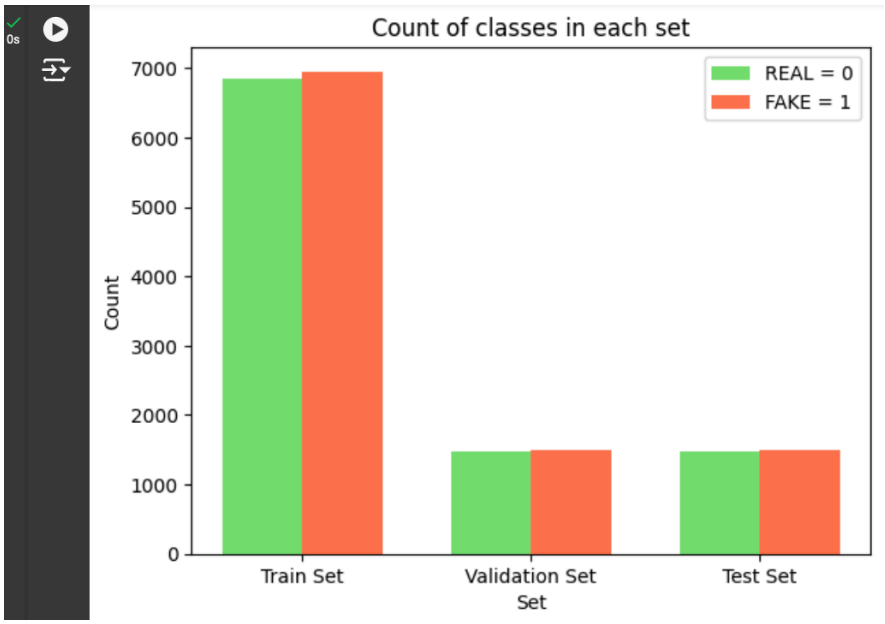

Figure 5: Dataset splitting Bar Chart.

First, I allocated 70% of the data for training and 15% for validation. This split ensures that a substantial portion of the data is dedicated to training the model, while a smaller but significant portion is reserved for validation during the training process. Given the massive size of the dataset, comprising around 141,000 images, I employed a batching approach to efficiently manage and process the data. Instead of loading the entire dataset into memory at once, which could overwhelm system resources, I created a combined generator that combine generators for both real and fake images. This generator allowed me to collect data in manageable batches during the training process.
I systematically collected data from the combined generator in batches and assembled it into a single dataset. Batching the data not only conserved system resources but also enabled more efficient processing and training of the deep learning model. Utilising the train-test-split function from scikit-learn, I split this combined dataset into two segments: one for training and validation (train-val-data), and another for testing (test-data).
The test-size parameter was carefully adjusted to ensure that the test set contained the remaining data after allocating 70% for training.
Further splitting of train_val_data took place using train_test_split again, dividing it into separate train_data and val_data sets. The test_size parameter was set to match the

desired proportion of validation data relative to the training data. Finally, I unpacked the data into separate arrays for images and labels in each set: train_images, train_labels, val_images, val_labels, test_images, and test_labels.

3.3 Procedures used to capture the data

To capture the data for the deepfake detection model, extensive research was conducted to find a suitable dataset. This involved thorough examination of various datasets to ensure high data quality. One of the primary considerations was that the images were of high quality, as this is essential for accurate model training [6].
Additionally, the dataset needed to have a manageable pixel size to facilitate ease of upload and processing. Large datasets pose challenges in terms of storage and computational requirements, so selecting images with a smaller pixel size helped alleviate these issues.
One major challenge was the difficulty in compiling and preprocessing the dataset due to its size. Processing and handling such a large dataset demanded significant computational resources, including memory and processing power. Adequate resources were necessary to efficiently preprocess and analyse the data.
Furthermore, the preprocessing stage involved tasks such as resizing images, converting them into numerical arrays, and normalising pixel values. These tasks were computationally intensive and required careful optimisation to ensure efficiency.

3.4 Data Augmentation

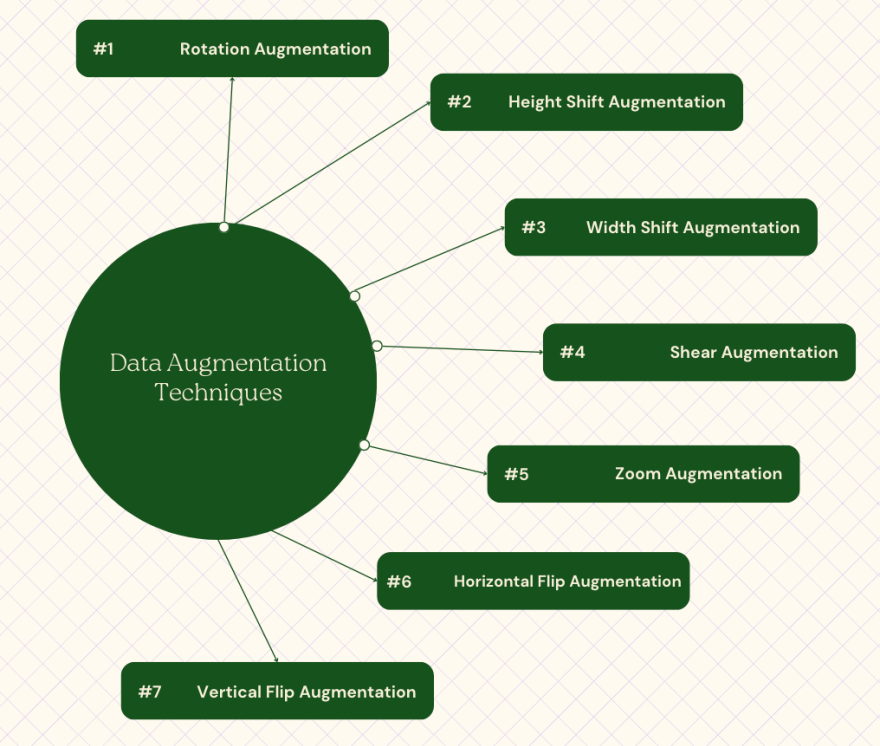

Figure 6: Data Augmentation.

The incorporation of data augmentation techniques significantly enhances the quality of the data used to train the model. By applying various transformations to the images, such as rotation, shifting, shearing, zooming, flipping, and adjusting brightness, the dataset becomes more diverse and representative of real-world scenarios. This augmentation introduces a broader range of variations and distortions to the images, effectively enriching the dataset with instances that the model may encounter in practical applications.
By exposing the model to this augmented dataset during training, it learns to recognize and adapt to these diverse variations, thereby improving its ability to generalise and make accurate predictions on unseen data. Essentially, data augmentation helps thelearn more robust and invariant features, reducing the risk of overfitting to specific patterns present in the original dataset. As a result, the model becomes more adept at distinguishing between real and fake images, enhancing its overall performance and reliability in real-world deepfake detection scenarios.

4. Efficiency Analysis

Cost-effectiveness: While the deep learning models require significant computing power, accurate deepfake detection provides substantial long-term cost benefits by mitigating risks like misinformation, financial losses, and reputational damage from deepfakes. Integrating MTCNN can also reduce computational costs compared to using only complex CNN models.
Resource utilization: Techniques like batching and parallelization allow efficient use of available resources. Processing the large dataset in batches optimizes memory allocation. The solution can leverage cloud computing and GPUs for scalable, distributed processing too.
Time savings: Combining MTCNN's efficient facial detection with the custom CNN can accelerate the overall process compared to using only complex models or manual inspection. MTCNN quickly identifies relevant facial regions, reducing CNN computation time. The optimized architecture also enables faster training and inference for timely threat detection.

5. Model Evaluation

After compiling the model, the next step is to evaluate how well it can distinguish between real and fake images. For this assessment, a random image is selected from the testing dataset. The trained model is then applied to this chosen image to predict whether the image shows a real or fabricated representation.

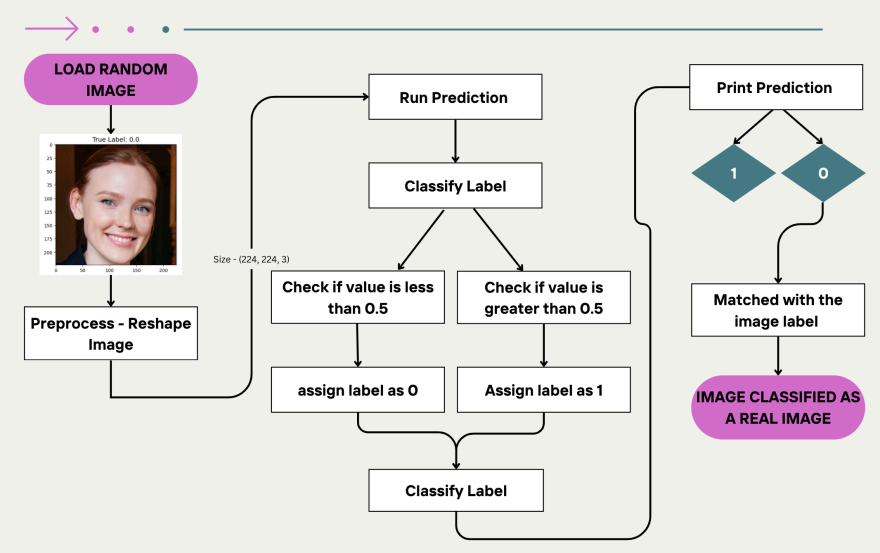

Figure 7: Model Testing.

In this specific case, the randomly selected image has a label of 0, indicating it is a genuine image. Impressively, the model's prediction matches the true label, accurately classifying the image as 0, recognizing it as an authentic image. This successful classification demonstrates the model's ability to discriminate between real and manipulated images.

5.1 Model Accuracy

After evaluating the performance of my model, the results demonstrate a test accuracy of 0.8483, reflecting an improvement from my previous experiments. This metric offers valuable insights into the model's proficiency in accurately classifying instances as either real or deepfake images. Specifically, the accuracy value denotes the proportion of samples from the test dataset that were correctly predicted by the model. An accuracy of 0.8483 means that out of all the test images, approximately 84.83%were correctly classified as either real or fake by the model.

This high accuracy rate indicates that the model has learned to effectively distinguish between genuine and manipulated images, leveraging the features and patterns it was trained on during the model training process. The improvement in accuracy compared to previous experiments suggests that the modifications made to the model architecture, training techniques have positively impacted its performance.

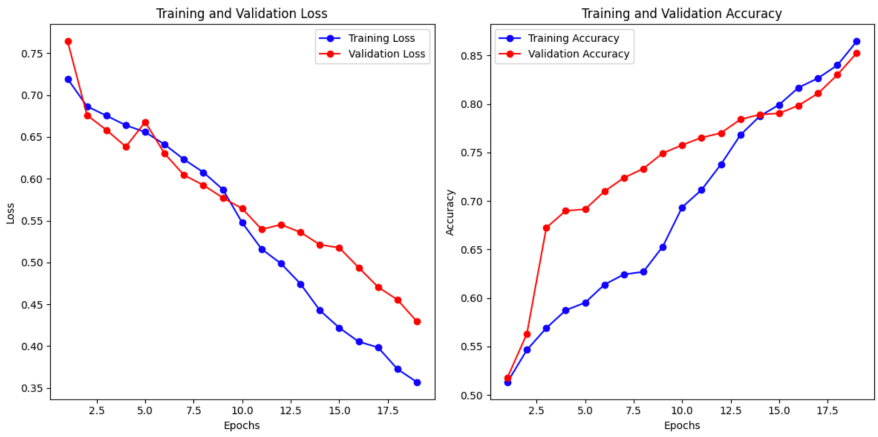

Figure 8: Model loss and Accuracy.

5.2 Confusion Matrix

To gain a more comprehensive understanding of the model's performance, a confusion matrix was prepared. This matrix provides a detailed overview of how the model performed on the testing dataset, breaking down the correct and incorrect classifications across the two classes: real images and fake (deepfake) images. The confusion matrix offers valuable insights that allow for further analysis and potential adjustments to improve the model's capabilities. The breakdown of the confusion matrix is as follows :

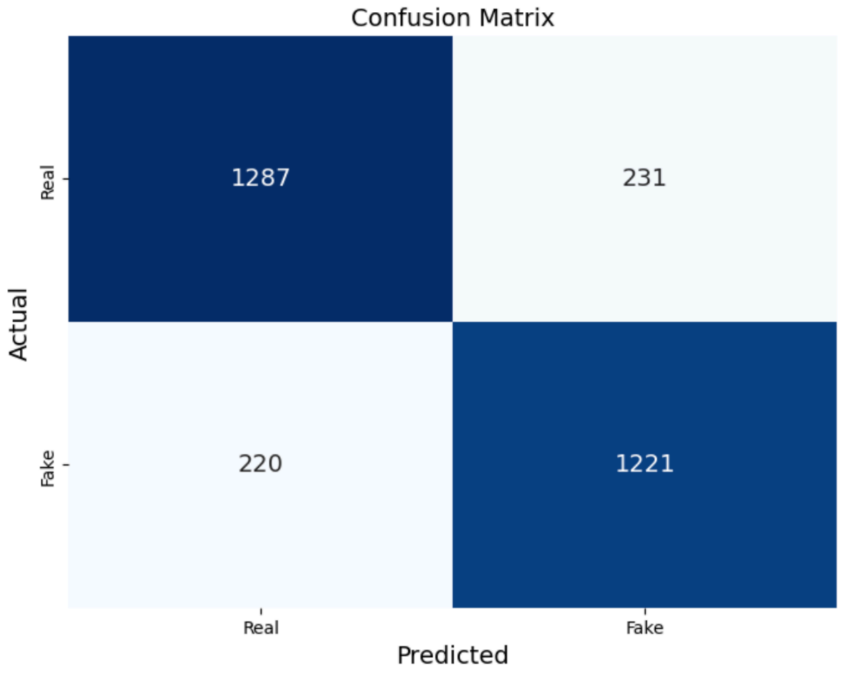

Figure 9: Confusion Matrix.

- True Negatives (TN): 1287 - This represents the number of fake (deepfake) images that were correctly classified by the model as fake.

- True Positives (TP): 1221 - This represents the number of real images that were correctly classified by the model as real.
- False Positives (FP): 231 - This represents the number of fake (deepfake) images that were incorrectly classified by the model as real.
- False Negatives (FN): 220 - This represents the number of real images that were incorrectly classified by the model as fake.

The high number of true negatives (1287) and true positives (1221) indicates that the model performed well in correctly identifying both real and deepfake images in a significant portion of the test dataset. However, the presence of false positives (231) and false negatives (220) suggests that there is still room for improvement in reducing misclassifications.

5. Challenges Faced and Limitations

5.1 Dataset Limitations

- Challenge: While utilising a very large and diverse dataset is advantageous, it also presents challenges. Acquiring an extensive dataset comprising both authentic and manipulated images is crucial for training a model to identify a broad spectrum of deepfake variations accurately. However, the sheer size of the dataset made it difficult to upload the data to the cloud and load/ preprocess it efficiently.
- Impact: The dataset's quality and diversity play a pivotal role in the model's performance. However, managing such a sizeable dataset required continuous experimentation and adjustments to enhance the model's accuracy over time. It also requires significant computational resources and efficient data handling techniques to load and preprocess the large dataset effectively during training.

5.2 Over-fitting Challenge

- Challenge: When developing a deep learning model, a formidable challenge is over-fitting, where the model may overly memorise the training data, leading to poor generalisation on new data. Over-fitting poses a threat to the deepfake detection model's accuracy, potentially resulting in false positives or false negatives.
- Impact: To address over-fitting, I implemented regularisation techniques such as integrating dropout layers during model training. However, striking the right balance between preventing over-fitting and maintaining accuracy required meticulous attention and fine-tuning.

5.3 Computational Trade-offs

- Challenge: Designing a deep learning architecture on a massive dataset inherently demands substantial computational resources, including powerful computing power, ample memory, and accelerated GPUs. The intricate process of model training often exceeds the capabilities of standard hardware.

- Impact: To address these resource challenges, I opted for cloud computing solutions equipped with accelerated GPUs and expanded CPU memory. While enhancing computational efficiency, this decision introduced dependencies on external services like Google Colab and incurred additional costs.

6. Conclusion

In conclusion, the process of building this deepfake detection solution taught me several important lessons. First, having high-quality and diverse datasets is extremely important. Carefully managing the data, processing it properly, and allocating enough resources is crucial for getting good model performance.
Secondly, overfitting continues to be an ongoing challenge that requires using robust techniques like regularization and tuning the model's hyperparameters. Efficient use of resources, especially when using cloud computing, is also essential for developing solutions that can scale up.
Lastly, taking an iterative development approach and continuously evaluating the model ensures you can evolve and improve the deepfake detection capabilities over time. These key lessons serve as guiding principles as work in this field moves forward.

References

1. “Deepfake technology,” Organization for Social Media Safety, Available at: https://www.socialmediasafety.org/advocacy/deepfake-technology/#:~:text=Deepfake%20technology%20first%20appeared%20in,to%20create%20realistic%20fake%20videos. (accessed May 16, 2024).

2. S. Chaudhary, R. Saifi, N. Chauhan and R. Agarwal, ”A Comparative Analysis of Deep Fake Techniques,” 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 2021, pp. 300-303, doi: 10.1109/ICAC3N53548.2021.9725392.

3. Ying Zhang, Linyu Guan, Jingyan Lu, and Songyang Wu. 2022. Face Forgery Detection of Deepfake Based on Multiple Convolutional Neural Networks. In Proceedings of the 2022 5th International Conference on Signal Processing and Machine Learning (SPML ’22). Association for Computing Machinery, New York, NY, USA, 211–217. https://doi.org/10.1145/3556384.3556416

4. D. Shah, D. Shah, D. Jodhawat, J. Parekh and K. Srivastava, ”Xception Net Vision Transformer: A comparative study for Deepfake Detection,” 2022 International Conference on Machine Learning, Computer Systems and Security (MLCSS), Bhubaneswar, India, 2022, pp. 393-398, doi: 10.1109/MLCSS57186.2022.00077.

5. Aniruddha Tiwari, Rushit Dave, and Mounika Vanamala. 2023. Leveraging Deep Learning Approaches for Deepfake Detection: A Review. In Proceedings of the 2023 7th International Conference on Intelligent Systems, Metaheuristics amp; Swarm Intelligence (ISMSI ’23). Association for Computing Machinery, New York, NY, USA, 12–19. https://doi.org/10.1145/3596947.3596959

6. Xhlulu, “140k real and fake faces,” Kaggle, Available at: https://www.kaggle.com/datasets/xhlulu/140k-real-and-fake-faces/data (accessed May 16, 2024).

7. Ruby Chauhan, Renu Popli, Isha Kansal, ”Summary of ’A Comprehensive Review on Fake Images/Videos Detection Techniques’,” in 2022 10th International Conference on Reliability, 2022. (accessed May 16, 2024).