Final Report

Comparative Analysis of Deepfake Detection Methods

Adam Zureiki

Summary

This project investigates the escalating challenge posed by deepfake technology. These sophisticated artificial intelligence-generated forgeries of real human images or videos, present a growing threat to information authenticity and personal security.

This report delves into the different methods of detecting deepfake content. It starts off by understanding human capabilities in detecting fake content through user testing. Subsequently, previous image classification methods are tested, including a TensorFlow Convolutional Neural Network and a ResNet50 based model. Using this as inspiration, a PyTorch Convolutional Neural Network image classification model is created. All deepfake detection models tested in this research are image classification models. They are trained to differentiate between real and fake images. After that, a comparative analysis is conducted to understand their strengths and weaknesses.

The report concluded that humans are unjustifiably overconfident in their deepfake detection capabilities, with the ability of detecting around half of the content presented to them. On the other hand, the image classification models all proved to have good capabilities in detecting deepfakes. The PyTorch and ResNet50 models proved to be the most reliable, with the ResNet50 model taking the edge in its generalization capabilities. With that said, all three models can be improved and enhanced further for future utilization.

Chapter 1
Introduction and Motivation

This chapter serves as an introduction to this project, offering an overview to the motivation behind the topic.

1.1 Introduction

Artificial intelligence (AI) is arguably the most transformative tool of our generation, as well as for future generations - a technology that has proven to significantly simplify our lives. With that said, AI still possesses the power and potential of significant harm. The recent advancements in AI have raised concerns over several questions: Can AI outsmart the human brain? What are the consequences? Where do we draw the line between the good and the bad of AI? In addition to many concerns that continue to be investigated every day. We cannot be sure of such future aspects. Nevertheless, we certainly can address the current dilemmas faced by AI in our current age and time.

This project centers on the significance of detecting deepfake AI, a branch of AI that generates convincingly deceptive fake audio, images, and videos. The project's initial segment delves into the dangers posed by deepfakes, the background research behind it, and compares different detection methods. In addition to this, it also explores how well deepfakes are detected by humans through user testing. The latter portion of the project focuses on experimenting these methods and explores the weaknesses and limitations. Moreover, this portion will evaluate the performance of these methods, and how to possibly enhance them. Throughout the project, a combination of primary and secondary research methodologies was employed. The overall objective of this project is to understand the dangers of deepfakes, how they are created and detected, and attempting to enhance the way in which we detect them.

1.2 Motivation

The motivation behind this research paper on deepfake detection stems from the need to address the increasingly concerning sophistication of deepfake detection. As the fake content we encounter daily becomes more convincing, it becomes more complex to differentiate between what is reliable and what cannot be trusted. Moreover, a personal interest in the technology behind deepfakes, coupled with a desire to contribute to critical societal safeguards against the misuse of deepfakes, further fuels my motivation.

Chapter 2
Background Research and Related Work

This chapter delves into the origins of deepfakes, how they are made and how they have been detected in previous work. It expands into discussions on artificial intelligence, machine learning, deepfake algorithms, and the implications of deepfake content.

2.1 Artificial intelligence

Artificial intelligence (AI) is a branch of computer science that focuses on creating powerful tools capable of completing tasks typically performed by human beings. AI tools are powered by algorithms, which are essentially statistical models. AI can be divided into two types: weak and strong. Weak AI tends to perform specific tasks like image recognition and search engines. Strong AI (which does not yet exist) would have the ability to understand and apply knowledge like a human being, requiring human cognitive abilities (Copeland, 2022).

Undoubtedly, the emergence of AI related techniques in all fields are becoming so advanced that humans fear the loss of their jobs. While AI can prove to create new jobs, it can also make some redundant. Unfortunately, this fear of becoming redundant within your workspace is rapidly becoming justified by the advancement of AI tools like GitHub Copilot, ChatGPT, and Amazon CodeWhisperer. All of which represent remarkable capabilities in programming, data analysis, and task completion. This potential job displacement poses a threat of exacerbating economic inequality and social unrest.

2.2 Deepfake AI

The AI threat addressed in this project is the emergence of deepfake AI technologies, which in recent years has raised significant concerns in the authenticity, integrity, and reliability of the media we view daily. Deepfake AI technologies encompasses several tools and methods that strive in creating realistic fake audio, images, and videos. The malicious utilization of these images and videos, or even audio clips, has evolved rapidly in recent years. In its foundation stage, deepfakes emerged from decades worth of research in machine learning and computer vision. Initially, the focus was on face recognition tasks, image manipulation and speech synthesis. These foundational deepfakes paved the way for advanced deepfakes, which are sophistically produced using several algorithms that utilize neural networks.

2.3 Machine learning and neural networks

Machine learning is a subset of artificial intelligence (AI). It enables computers to learn from data and make decisions. This subset of AI develops algorithms that process and analyze data, then use these analyses to make predictions about unseen data. Understanding the concepts of machine learning and neural networks is crucial to understanding how deepfakes work and how they can be detected.

Essentially, a neural network is a machine learning model that makes decisions like the human brain. Neural networks do this by utilizing processes that mimic the way biological neurons work together to identify phenomena, compare options, and get to conclusions (IBM, 2023). Neural networks recognize patterns in data, by utilizing nodes/neurons that pass information to each other, they consist of several layers of nodes:

- Input layer: This layer receives the input data; each node here will represent a feature in the data that is inputted.
- One or more hidden layers: Computations are performed in this layer.
- Output layer: Outputs the final part of the network, such as a class label in classification problems. For instance, in detecting deepfakes, a label of REAL or FAKE would be outputted from the neural network algorithm used. The number of nodes in the output layer would be equal to the number of classes given in the dataset.

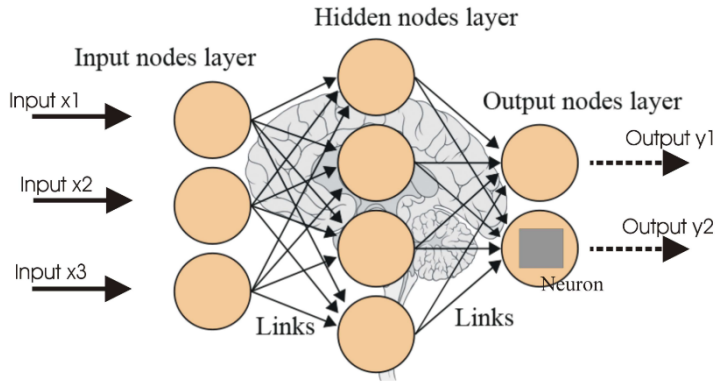

Figure 1 Simple neural network (Ali and Ahuja, 2016)

As seen in figure 1 above, neural networks are composed of interconnected nodes, and each one has its own weight and threshold. Weights are parameters within the network that impact its ability in making predictions If the weight of a node is higher than its threshold (indicating a strong signal), it activates and passes the message along to the next layer, if not, it will not send anything. The links between the layers represent the weights and biases, the biases here are an indication of how easily a neuron activates (Pramoditha, 2022). An example

of how a neural network functions involves processing an image input. is As the features of the image (input nodes) move through the layers, the network will progressively refine its understanding of what the image represents. At the output layer, the network will give its best guess based on the computations that took place. For instance, it might conclude, “This is a car”.

Neural networks rely on training data over time to improve accuracy and performance, once they are accurate enough, they become powerful tools that can complete tasks in minutes compared to hours by manual human methods (IBM, 2023).

2.3.1 Training neural networks

Before a neural network is trained, there must be a preparation of clean datasets. Almost all neural networks are trained through an iterative process that can be divided into three main parts: forward propagation, calculation of the loss function, and backward propagation (Pramoditha, 2022). Figure 2 below represents how a neural network is trained; the arrows are the links between the layers (determined by weights and biases of the neurons):

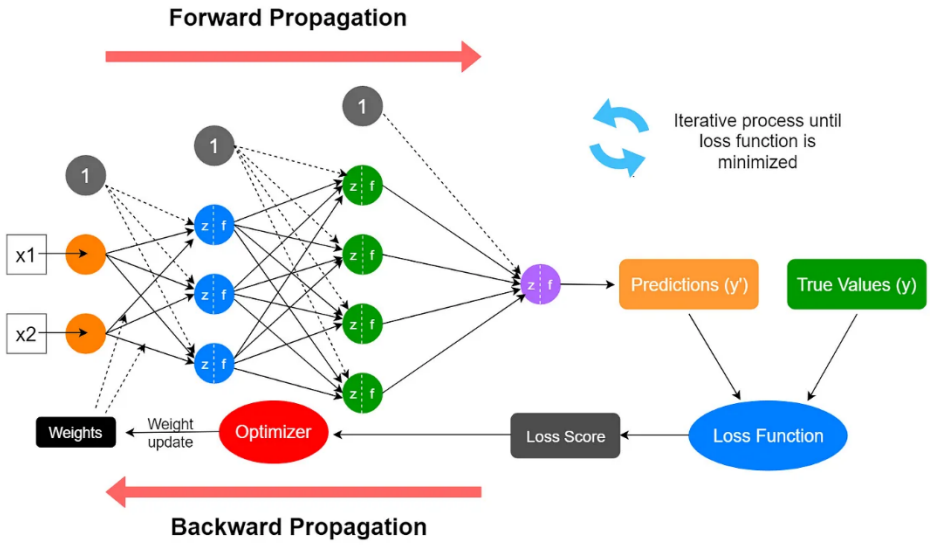

Figure 2: Neural network training process

1. Forward propagation: This is the initial phase where input data, which could be anything from numbers to pixel values (x1 and x2 as per figure 2), passes through the network. As the data goes forward through the layers, it is transformed using weights and activation functions. Every time data is inputted into the network, it is multiplied by the weights as it moves between layers. This mechanism allows the network to learn complex patterns by introducing non-linearity. On the other hand, the activation function mathematically determines the output of every node, depending on the input, Lastly, the output layer produces the

network's prediction. Non-zero random values are assigned during parameter initialization of the weights and biases, based on this, two calculations are done in each node of the network. The first calculation is of the node's linear function, the second is of its activation function.

2. Calculating the loss function: The output value from the first iteration of forward propagation (predicted value) is compared with the real value through a loss function (known as the error of the model). The represents how well the neural network is performing. A loss function is computed in every iteration, and it provides us with feedback for updated parameters to be used in backpropagation. The goal is to minimize the loss function as much as possible, which will create a more realistic model. For instance, in a network that creates deepfake images, a minimized loss function produces images that seem quite realistic to the human eye. There are several loss functions that can be used (Pramoditha, 2022):

Performance of regression problems:

- Mean Absolute Error
- Mean Squared Error
- Huber Loss

Performance of binary classification problems:

- Log Loss (Binary Cross-entropy)

3. Backward propagation: This is essentially the method that reduces the loss function by changing the parameters of the weights and biases based on a calculated gradient. The gradient of the loss function is how much the error alters as the network's biases and weights change. Eventually, this aids in optimizing these parameters by utilizing an optimization algorithm such as gradient descent (Al-Masri, 2022).

The process of training neural networks can also introduce problems, such as the vanishing gradient problem. This occurs when the gradients, which are used to reduce the loss function, become very small during backward propagation. When the number of layers in a network are high, the training process becomes more difficult as the gradient becomes very small, which causes slower rates of learning. Therefore, at a certain point of adding more layers to a network, optimization is more complex (Wang, 2019). There are several ways to combat this, one will be explained in the implemented methods in Chapter 3.

2.4 Types of neural networks that create deepfakes

2.4.1 Generative adversarial networks (GANs)

The most common way in which deepfakes are generated is through generative adversarial networks (GANs). These consist of two neural networks (generator and discriminator). The generator creates a deepfake image or video to look real, and the discriminator evaluates the deepfake by comparing it to a database of real content. This

process between the generator and discriminator continues until the discriminator cannot distinguish between the generated deepfake and the real image/video (Payne, 2023). The neural networks used as generators and discriminators depend on the deepfake that is being created.

2.4.2 Autoencoders

Autoencoders are utilized by learning how to encode a face into a lower-dimensional representation. This representation could be a small set of numerical values or any other type of data. The process of encoding regularly goes through layers of gradually downsizing variables until they reach a bottleneck layer, which contains the target number of variables. This compression is later decoded to a representation that is modified in recreating the original image. This method is usually used in face swap deepfakes.

2.4.3 Deep convolutional neural networks (CNNs)

These are neural networks that learn and extract features, like edges and corners, from inputs like images. By learning the low-level features, the network is trained to understand complex features. CNNs can be utilized for face recognition, image classification, object detection, and several other applications. Since CNNs can learn the details of images, they can be used in the creation of deepfake images and videos. For example, in a face swap deepfake, the CNN learns the full details of several pictures of both faces. After this, the CNN can create a digital mask that is identical to a persons face (to replace with another face), that can fit onto the head of the other face. During this process, the CNN must ensure that the digital mask it creates fits perfectly onto the head of the other face, it is important to create a very realistic look. CNNs will be crucial in this research, they will be used to create models that can detect deepfakes through image classification.

2.4.4 Recurrent neural networks (RNNs)

Recurrent neural networks handle sequential data. This type of network is commonly utilized for language translation and speech recognition. They are typically incorporated into voice recognition applications such as Apple's Siri, and Google Translate (IBM, 2023). RNNs can also to use their memory (their stored knowledge from previous training of older inputs) to aid them in computing new outputs of the sequence (Saeed, 2023). Essentially, they handle sequential data one element at a time. However, within this, the RNN updates its memory with every step for every element, based on the current input and the previous output. In terms of deepfakes, RNNs are useful for synthesizing realistic-sounding speech. For instance, RNNs can generate audio that matches a target speaker by receiving a sequence of phonetic units and acoustic features.

2.5 Related work

Since the rise of deepfakes, there have been several attempts in creating systems that can detect them. Many of these systems revolved around using a neural network, such as CNNs for image classification, and RNNs for synthesizing speech. In December of 2019, a deepfake detection challenge started on Kaggle, which is an online platform that hosts data science competitions. The competition was an initiative by AWS, Facebook, Microsoft, and others. The primary objective was to challenge researchers to create technologies that can be used in deepfake detection systems.

One of the submissions for this competition was a “Deepfake Starter Kit” implemented by Gabriel Preda (Preda, 2020), the programming language used in it was Python, and it utilized several deep learning frameworks and libraries such as OpenCV, which is used for image processing tasks. The notebook examines datasets of videos and images in several stages. Firstly, it analyzes subsets of fake and real videos to identify patterns or anomalies. Moreover, it examines deepfake videos with the same origin. Essentially, this reviews several fake videos that are derived from the same real video, which helps understand how deepfakes can come in different variations. The notebook also investigates the integrity of the videos by checking for missing data. Another submission from this Kaggle competition is by Nirmal Gaud (Gaud, 2024), this implementation looks at using image classification models like CNNs to detect deepfakes. It looks at how accurate these models are, and examines the use of architectures that combat the vanishing gradient problem. This implementation will be examined and tested in the second sprint of the methodology implementation.

Furthermore, an article from 2022 explores several deepfake detection methods (Guarnera et al., 2022). The article investigates methods including EfficientNet architecture, transformer functions, discrete cosine transform (DCT), and vision transformers combined CNNs. The EfficientNet architecture is a scalable neural network architecture, it scales up models efficiently so that width, depth, and image resolution is scaled up in a balanced manner, which improves the model’s overall performance (Sarkar, 2021). Transformer networks are a type of neural network architecture that handles sequential data, but processes the data simultaneously (Ankit, 2022). This helps in tasks that require understanding context. Discrete cosine transform (DCT) is a technique which separates images into sections of importance (Marshall, 2001). The article suggests that the EfficientNet models had the highest accuracy, achieving an accuracy of 93.61% with the best model (Guarnera et al., 2022). Moreover, these models were highly effective with images that are likely to occur in digital media platforms. Nonetheless, transformer networks and DCT methods also performed well, just not as good as the EfficientNet models. The research concluded that even though these

detection methods are capable of detecting deepfakes, there are still challenges faced, especially with images that undergo complex transformations.

It is also important to understand how well humans can detect deepfakes, to fully understand the scope of the danger faced by allowing them on social media platforms. A behavioral experiment completed on sample of 210 people concluded that humans cannot reliably detect deepfakes, however, they are confident that they can, overestimating their abilities (Köbis et al., 2021). Another experiment conducted on 529 participants by University College London showed that only 73% of people can detect fake artificially generated speech (Farah, 2023). Lastly, an online public quiz showed that only 1.5% of participants correctly identified all deepfake videos (8 videos), and just 56% were able to correctly identify over half of the video clips (McKenna, 2023). Additionally, the quiz shows that the deepfake clips of the UK's most popular politicians (Boris Johnson and Jeremy Corbyn) were the most difficult to identify. Interestingly enough, the quiz also showed that older generations scored better in deepfake detection in comparison to the younger participants, which just shows how much people have become used to misleading content.

2.6 Ethical, social, and security challenges of deepfake AI

Deciding whether something is ethical or not is more complex than it may seem. From a very basic view, something is considered ethical depending on how well it aligns with the moral principles accepted and valued within a society. Determining these principles comes down to how well they emphasize the well-being of others and how well they highlight honesty, fairness, and respect. In the world of deepfakes, two important pillars of ethics are often compromised, honesty and respect. A video clip that falsely represents someone, not only undermines their character and honesty, but also shows a lack of respect towards them.

The existence of deepfakes today creates a significant ethical and social challenge. They essentially mimic a person's looks, voice, and identity without their permission. In most cases, this theft of identity can cause a catastrophic impact on a person, especially if the deepfake creates a bad reputation for them. For example, when deepfake technology is used to create explicit videos, in most cases it is without consent. This exploitation of deepfake technology for sexual gratification is purely humiliating, yet unfortunately very popular. As of 2020, one study found that 96% of existing deepfakes are in the form of adult content (Goodwine, 2020). The trauma that this content can leave on its victims is disastrous, ranging from depression to suicidal thoughts (Comar, 2021).

Deepfakes can also be dangerous in the context of politics. Content that portrays a controversial political message or depicts a politician spreading a questionable speech can cause serious harm to their integrity. In addition to the potential impact on social safety since

deepfakes can be used maliciously by framing an individual for a certain crime. The aftermath of this not only impacts the individual in question but also creates an “unsafe” reputation of the environment they are in. At the same time, deepfakes can also be used in failing to convict the guilty, in forms of using the content to blackmail, which causes a detrimental impact on the justice system.

A video of Mark Zuckerburg in 2019 portrayed him saying that he is glad to have control over people’s data (Goodwine, 2020). This sort of misinformation creates a lack of trust between Facebook users and the firm. Additionally, it creates a global mistrust in social media platforms. Another example to consider can be in the corporate world. If a fake video is released of the CEO of a company stating that there has been major losses in quarter x, the stocks of this company will crash, impacting employees. This shows that the impact can very much be on an economic level as well. With deepfakes, sky is the limit.

Deepfake technology can also impose a threat on cybersecurity. This threat can range from impacting the security of people’s data to threatening the digital security of nations worldwide. Unauthorized access to private data is possible with the utilization of fake images (Face ID) and fake voices. Trafficking and illegal immigration become easier when passports can have deepfake photographs (Townsend, 2022). Online blackmailing is also linked to embarrassing fake images or videos. In addition to this, phishing, which is a cybercrime where malicious actors trick individuals into revealing sensitive information (Phishing.org, 2019), is now more convenient for attackers as they can use deepfake images to trick the recipient into thinking they are someone else.

As deepfake technology improves, the confidence and trust we have in the media diminishes, along with our trust in sharing our images and voices. This not only creates a world that lacks faith in those who have control over our digital security, but can also significantly ruin the reputation of individuals targeted by deepfakes.

Chapter 3
Methodology

This chapter outlines the methodologies employed in this project, there are three overall approaches to this research. The first method is a quiz designed to test how well humans can detect deepfake content. The subsequent methods are neural network models, two are adopted from a Kaggle competition, while the other is developed using PyTorch.

3.1 Approach

The approach taken in this research is an agile approach that revolves around understanding how well we can detect deepfakes, there are three methods that will be used in this research as shown below:

1. Deepfake detection quiz (one week)
2. Kaggle image classification models: TensorFlow CNN and ResNet50 (Gaud, 2024) (two weeks)
3. Creating my own PyTorch CNN image classification model (two-three weeks)

These methods will be utilized to understand the reality of the danger in deepfakes. The methods will be implemented in three sprints.

3.2 Sprint 1: Human detection

The most concerning type of deepfake content that we view as social media consumers are deepfake videos and images. This content can trick many into believing incorrect information, therefore, it is important to understand how well humans detect this content, which can be deceiving and misleading in its meaning.

3.2.1 Aims of sprint

1. To explore the effectiveness of human deepfake detection using a quiz.
2. To investigate the types of content that influence detecting deepfakes.
3. To examine the psychological factors behind human recognition of deepfakes.

3.2.2 Implementation

To gauge the accuracy of human perception, a quiz for deepfake detection is created using Google Forms and can be viewed under Appendix C.1. This quiz is the first method utilized in exploring the concept of deepfake detection and represents a primary research approach. The quiz includes three types of questions. The first two display a short clip from a video or an image, which could be real or fake, and the participant had to determine its

authenticity. The second type showed two versions of the same clip, one is the real video and the other was fake, the participant must identify which was the real clip. Moreover, with each question, the participant must state the confidence level in their answer from 1-5 (5 being the most confident), in addition to stating if they have seen the video before. The images and videos, can be seen under Appendix C, used for this quiz were taken from three online quizzes. The first is by a group from Northwestern University (Kamali et al., 2019). The second is by the University of Washington, and the third is another online quiz (McKenna, 2023).

3.2.3 Sprint 1 review

Sprint 1 took approximately one week to complete, this included collecting a database consisting of deepfake images and videos, the development of a quiz, and collecting the results. The most crucial part of this sprint was collecting the quiz content because the goal was to investigate real life examples of deepfakes that you see around the media. It was important to have a variety of political videos, social media related content, and images that ranged in difficulty. Some of the content displayed in the quiz was relatively easy for anyone to detect, the goal of including these was to see if everyone is actually doing the quiz, rather than just guessing the answers. On the other hand, some deepfakes were very well done, and therefore their purpose was to challenge the participants. The results from this quiz can be seen in Chapter 4 under section 4.1.

3.3 Sprint 2: Deepfake detection notebook from Kaggle

Another approach is taken from a Deepfake Detection Challenge. The notebook, made by Nirmal Gaud (Gaud, 2024), attempts deepfake detection through image classification models. There are two models created in this notebook, one is a CNN created from scratch using various layers and the second is based on the ResNet-50 architecture (explained below). The notebook can be viewed under Appendix B.1.

3.3.1 Aims of sprint

1. Learn and understand both models and the differences in their structures.
2. Grasp the fundamental concepts of constructing a CNN.
3. Train, validate, and test the models in this notebook to understand how well they can detect deepfakes (tests and results are shown in Chapter 4).

3.3.2 Language, framework, and libraries/modules used

Table 1: Libraries used in the Kaggle Deepfake Detection Challenge notebook

Model training	Visualizations	Image processing	Data processing	Functional programming
Sklearn	Matplotlib.pyplot	Cv2 (OpenCV library)	Pandas	Functools
Tensorflow.keras	Seaborn		Numpy	
Tensorflow.keras.applications (class ResNet50)	Scikitplot			
Tensorflow				
Sklearn.model_selection				

The language used in this method is Python. The main deep learning framework used is TensorFlow. TensorFlow is an open-source machine learning framework developed by Google. The model training modules are used for performance metrics, training the model, and for access to pre-trained models like the ResNet-50 model. The notebook also takes advantage of image processing libraries like the OpenCV library, in addition to using pandas and NumPy for data analysis.

3.3.3 Implementation

3.3.3.1 Preprocessing dataset

The notebook originally operates on the dataset of the deepfake detection challenge, however, for this research, the images from the FaceForensics-1600 videos-preprocess database from Kaggle (Hasan and Khan, 2022) will be used, the link to this dataset can be accessed from Appendix B.4. 15448 images were used from this dataset, half were fake, and the other half were real. 70% of the images will be used in the training phase, 15% in the validation phase, and the remainder goes to the testing phase.

3.3.3.2 Model 1: Built from scratch using TensorFlow

The notebook creates a two models for the images dataset. The first is a CNN, a sequential neural network model consisting of convolutional layers is defined using the TensorFlow and Keras libraries/modules, it can be seen under Appendix B.2. The model is built from scratch using layers like “Conv2D”, “MaxPooling2D”, “BatchNormalization” and “Dense”. The convolutional layers (Conv2D) are fundamental in applying a convolution operation to the input data, and they are designed to automatically learn spatial hierarchies of image features, from simple features to more complex ones in deeper layers. The max pooling layer is there to reduce computational load on the network and to control overfitting, it does

this by reducing spatial dimensions of the input feature maps. The batch normalization error normalizes the output of the previous layer, by subtracting the batch mean and dividing by the batch standard deviation. This stabilizes the learning process by reducing the number of training epochs needed. Finally, the dense, a layer fully connected (each input node is connected to each output node), receives input from the neurons of the previous layer and computes the weighted sum plus the bias, which is passed through an activation function.

In addition to this, the model uses the Adam optimizer (Adaptive Moment Estimation), which is an algorithm used to minimize the loss function of neural networks during training (Vishwakarma, 2023). The threshold for model predictions is 0.5, above 50% the prediction is converted to 1 (binary) indicating a fake video, otherwise the prediction is 0. The accuracy of the model is then analyzed through different tools (discussed further in Chapter 4). The structure of the model can be seen under Appendix B.2.1.

3.3.3.3 Model 2: ResNet50 model

After the first model, the ResNet50 model is trained and tested in the same way, using the same optimizer, and analyzed through the same tools. The ResNet-50 is a variant of the Residual Network architecture, a widely used architecture for image classification. The residual network architecture introduces a solution to solve the vanishing gradient problem (as discussed in section 2.3.1). To briefly explain the residual network architecture, it can be viewed as an architecture that allows layers to skip over some network layers. This is done by the utilization of skip connections (Sharma, 2023), these connections add the input of a block of layers to the output of a block of layers. The 50 in ResNet-50 refers to the number of layers contained. The structure of the model can be seen under Appendix B.3.

3.3.4 Sprint 2 review

Sprint 2 took approximately two weeks to complete, with all aims successfully completed. The image classification notebook investigated in this sprint is a large notebook consisting of multiple libraries, performance indicators, and two image classification models. The notebook was highly informative and studying it has given an inspiration for the next sprint, which is creating an image classification model using PyTorch.

3.4 Sprint 3: Creating a CNN on image classification using PyTorch

The method created in this research is inspired by the notebook from sprint 2. The model implemented below is trained to detect deepfake images from the FaceForensics-1600 videos-preprocess database from Kaggle (Hasan and Khan, 2022). The aim of the written code (explained in section 3.4.3) is to essentially create an image classification CNN model that can distinguish between real and fake images, and to compare this model to the other

two models discussed in sprint 2. The split between the training, validation, and testing datasets is the same as the split used in sprint 2. This method can form the backbone of a deepfake detection system, it can be a vital tool in detecting images and videos.

3.4.1 Aims of sprint

1. Learn about PyTorch libraries and modules.
2. Find and utilize a large image dataset.
3. Build a CNN model using PyTorch to train, validate and test.

3.4.2 Language, framework, and libraries/modules used

Table 2: Libraries used for this CNN model

Model training	Visualizations	Image processing	Data processing	Functional programming
Torch	Matplotlib.pyplot	Torchvision.transforms	Torch.utils.data. DataLoader	Torch.nn.functional
Torch.nn	Seaborn	Torchvision	Torch.utils.data. random_split	
Torch.optim	Scikitplot		Torchsummary	
Tensorflow				
Sklearn.metric				

The language used in this method is Python, and the deep learning framework is PyTorch, which is known as an open-source library used for machine learning purposes. This is different to what has been used in the method retrieved from the Kaggle competition above in section 3.3, where TensorFlow is used. It is known for its ease of use and for being capable of handling operations in an intuitive matter, making it a useful platform for research and production-grade development in AI. The modules used for model training like “torch.nn” and “torch.optim” are crucial in defining neural network layers and activation functions, they also contain the optimizers used for updating the model parameters in the training process. “matplotlib” and “seaborn” are used for visualizing performance metrics. In terms of image processing, the modules used can preprocess images, by resizing and converting to tensor format. Additionally, the “DataLoader” class provides the functionality required to iterate over the dataset, it can do automatic batching, sampling, and multiprocessing. All the components used in this implementation are taken from the PyTorch 2.3 documentation (PyTorch, 2019).

3.4.3 Implementation

3.4.3.1 Preprocessing dataset

The implemented approach can be viewed in the GitHub repository. The implementation starts off by preprocessing the dataset to ensure that they are all resized to one size for fairness. Additionally, the images are converted to tensor format and normalized using ImageNet mean and standard deviation values (PyTorch, 2019). ImageNet is a large dataset with diverse range of images. This is recommended by the documentation provided for PyTorch for several reasons. Firstly, normalization can help achieve a faster rate of convergence, as it can help the neural network learn faster. Normalization can help in stabilizing gradient descent optimization as it can make sure that each input feature contributes to the learning process equally. The utilization of of ImageNet is useful to maintain optimal training conditions for a neural network. In addition to this, using ImageNet statistics means that any performance or result of the neural network can be compared to other models, and it allows for benchmarking, as ImageNet is a standard in the field of neural networks. Code snippet 1 below preprocesses all the input images that are loaded in.

```
preProcessTransform = transforms.Compose([
   # resizing to keep everything the same
   transforms.Resize((224,224)),
   # normalizing, ToTensor is for converting image dimensions and pixel intensity
   transforms.ToTensor(),
   # normalize to ImageNet parameters
   transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
```

Code snippet 1: Preprocessing dataset

The inputted dataset is split into three different variables, for training, validation, and testing. After the dataset is split into three datasets, instances of DataLoader (PyTorch, 2019) will be used. DataLoader instances are powerful in automating loading batches of data, the aim of the DataLoader in this code is to iterate over each dataset (training, validation, and testing). For each dataset the DataLoader will prepare and return batches of 32 samples at a time, a choice that balances performance and memory usage. Each DataLoader instance has the option of shuffle, only in the training dataset will the DataLoader instance have shuffle set to true to prevent the model from learning unwanted patterns, which may lead to a certain bias and inaccuracy.

3.4.3.2 The model

Creating the CNN model consists of utilizing several components offered by PyTorch. Firstly, a sequential container is initialized. Containers are components that encapsulate layers in a neural network. This container defines the order of operations for the layers it contains. Each layer would pass its output to the layer directly following it and will receive input from the layer directly preceding it. The first layer initialized is the convolutional layer (Conv2d). This layer performs a mathematical operation which slides a filter over the input to produce future maps. The process extracts any important features from the images that are needed for image classification, such as edges and corners.

In addition to this, a non-linear activation function is used (ReLu), this non-linearity is added after linear components, like Conv2d, and it allows the model to learn more complex patterns from the images (Krishnamurthy, 2022). ReLu takes in an input x, if the input is positive, it returns the same value x. However, if x is negative, the function returns 0, which blocks negative input from passing through to the next layers (PyTorch, 2019). Moreover, the ReLu function supports solving the vanishing gradient problem, explained in section 2.3.1.

Furthermore, a pooling (MaxPool2d) layer is added. The pooling layer decreases the dimensions of the input feature maps so that they are more manageable (DeepAI, 2019). Another non layer component used is a flattening component, which turns 2D feature maps output into a 1D vector. This component is crucial to transition from the convolutional layers. Lastly, a linear layer is used, this essentially connects every input to every output by a learned weight. The code for the model can be seen below in code snippet 2. The structure of the model can be seen under Appendix D.1.

```
myModel = neuralModel.Sequential(
   #below is the code for the first two conv layers with the non-linear activation function and the maxpooling
   neuralModel.Conv2d(3, 32, kernel_size=3, padding=1),
   neuralModel.ReLU(),
   neuralModel.MaxPool2d(kernel_size=2, stride=2),
   neuralModel.Conv2d(32, 64, kernel_size=3, padding=1),
   neuralModel.ReLU(),
   neuralModel.MaxPool2d(kernel_size=2, stride=2),
   neuralModel.Flatten(),
   #third part of model. fully connected layer
   neuralModel.Linear(64 * 56 * 56, 128),
   #non-linearity
```

```
neuralModel.ReLU(),
    neuralModel.Linear(128, 2)
)
```

Code snippet 2: Initializing a CNN model

Before training, an optimizer and loss function must be initialized. The optimizer utilized is the same as in the first model from the Kaggle notebook, which is the Adam optimizer (explained in section 3.3.3.2). The loss function used is one that is particularly good for classification tasks, which is the Cross-entropy loss function. This loss function measures the performance of the model, with an output as a probability value between 0 and 1. A probability closer to 1 indicates higher certainty that the prediction of the model is correct. Moreover, the model gets trained in a loop structure of 10 epochs. The training loop processes batches of images and their labels from the training dataset. In every epoch, the prediction for every input batch is presented, followed by the loss function, which is calculated using the predictions. After this, backpropagation is performed using the .backward() function method provided by PyTorch (GeeksforGeeks, 2022). Finally, the model's weights are updated using the gradients that were calculated during backpropagation. The optimizer completes this step to minimize the loss using the .step() method from PyTorch (Muhammad, 2022).

As explained in section 2.2.1, during the backward propagation phase of training a neural network, a gradient is calculated to alter the parameters of the weights and biases, which in return reduces the loss function. Therefore, in this training loop, existing gradients are cleared in every iteration to prevent any accumulation, which would naturally occur every time backward propagation takes place. After training the model, the model is tested on the testing set, the testing set only includes unseen data, to properly assess the generalization of the model. The results from the testing are shown in Chapter 4 under section 4.2.

3.4.4 Sprint 3 review

At the end of this sprint, all the aims and objectives were successfully completed in roughly 2-3 weeks. This sprint took the longest as it required building a model from scratch, using PyTorch. The first week was essentially learning how to use PyTorch for image classification, understanding what modules will be required or/and useful. In addition to finding a dataset of images that is large enough for training, validating, and testing the model. The second week was for creating the actual model, while the last week was to test the model and run performance indicators on it.

Chapter 4
Results and Discussion

This chapter presents the results of the methods of this study. It starts off with the interpretation of the results from the quiz. Then, it assesses the effectiveness of the image classification models by running performance indicators to understand what model is best at detecting deepfakes. The last part of the chapter concludes with a discussion of the implications and improvements for the detection models.

4.1 The quiz

The full quiz can be seen under Appendix C.1. The content shown in the quiz ranged in difficulty, some images were very easy to detect as a deepfake, this was done to ensure participants were answering the quiz and not just guessing. On the other hand, some content required a bit of focus, while others were very difficult to detect. Examples of some images (Kamali et al., 2022) ranging in difficulty are shown below:

Figure 3: Medium difficulty

Figure 4: Easy difficulty

Figure 3 is not very easy to detect as a deepfake, the quality is very high, not much is shown in the picture, and there are no missing parts in the faces of the man and the kid. Whereas with figure 4, the parts circled in red (original image in the quiz does not have the circles), are missing from the image, making it much easier to detect as a deepfake. In figures 5 and 6 below, the background is blurred, taking away from the details of the picture, which makes it more difficult to detect, the human eye will most likely ignore the background and focus on the main character.

Figures 5 and 6: Blurred background (Kamali et al., 2022)

4.1.1 User testing

The quiz was sent out to 56 participants, the initial results of the points scored out of 11 is shown below in figure 7. All results can be viewed under Appendix C.3.

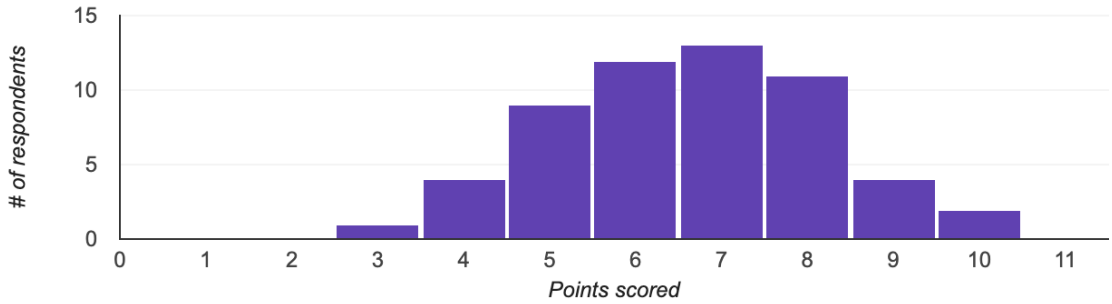

Figure 7: Total points distribution from the quiz

When looking into the results of different questions from the quiz, the most difficult deepfake to detect was the video from question 2, which can be accessed under Appendix C.2. This is a high-quality video, making it difficult to detect, with only 32% of participants correctly identifying it as a deepfake, yet almost 90% of participants were confident in their answers. Additionally, 98% of the participants stated that they have never seen the video from question 2. Figure 8 below shows the confidence levels of the answers to question 2.

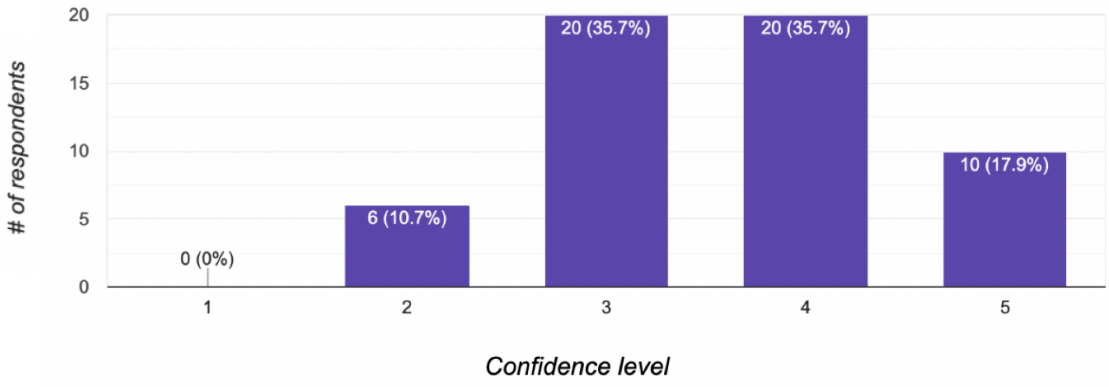

Figure 8: Confidence levels of participants from question 2 (1 being the lowest)

In terms of what question the participants were least confident in answering, this is undoubtedly question 3. Here the participants were asked to pick the fake video out of two identical videos, where the deepfake is done incredibly well. Almost all participants were not confident in answering this question, with 30% of participants picking a 1/5 confidence level. The two videos for this question can be seen under Appendix C.2. The question with the highest confidence level was question 11, which also showed two videos, where the deepfake had to be identified. This question was made to be very easy, to ensure that participants were focused. Over 60% of participants picked a 5/5 confidence level, 90% of participants correctly answered this question.

Over 10% of participants claim that they do not trust in the authenticity of the images and videos they view on social media, whereas 66% only “sometimes” trust them. Over 60% of participants claim that they have been tricked by deepfakes in the past. Finally, over 75% of participants claimed that the quiz was on the difficult side.

4.1.2 Evaluation of results

The bar chart from figure 7 shows the distribution of the total points scored by all participants. The most common score was 7/11 with 13 participants achieving this. Scores of 6 and 8 were also quite common, indicating a cluster of scores around the median value, which suggests that the quiz was of moderate difficulty. A small number of participants achieved a low score of 3-4 or a high score of 9-10. Moreover, no participants scored in the 1-2 region or the full score (11) region. This also indicates that the quiz was not very difficult, but also not very easy. This is backed up by the fact that 28% of participants claimed that the quiz had a 6/10 difficulty level. Additionally, the results show that the participants had a similar ability in detecting deepfakes.

Furthermore, the trend of high confidence in the wrong answers from question 2, aligns with the research paper explained in Chapter 2 under section 2.5, which claims that humans are not reliable when it comes to detecting deepfakes, yet they overestimate their abilities and are confident that they can (Köbis et al., 2021). Additionally, the fact that 98% of the participants did not see the video before yet still were confident in their incorrect answers further proves this point.

Another key observation is the psychological factor, which plays a role in the choices picked in the quiz. Some participants found that they started to intentionally look for deepfakes in every image/video, because they know the quiz is about detecting deepfakes, which leads to two different interpretations. This can impact the decision-making process for each

participant, making them think that everything is a deepfake, which can slightly take away from the credibility of these answers.

4.2 Comparative analysis: Results of the three models

All the models are image classification models, they classify images between two classes “fake” and “real”. In terms of the performance of the models, all three were trained on 70% of the dataset, validated on 15%, and tested on the remainder (unseen data). The accuracy of the CNN TensorFlow model on the unseen testing dataset averaged 61.43%, the training accuracy averaged as high as 71.98%. On the other hand, the ResNet50 model averaged a testing accuracy of 71.72%, and a training accuracy of 76.35%. The CNN PyTorch model achieved a training accuracy of 96.70% after 10 epochs. In terms of the testing accuracy, the model achieved an average accuracy of 67.32%. The changes in training accuracies for the three models can be seen below.

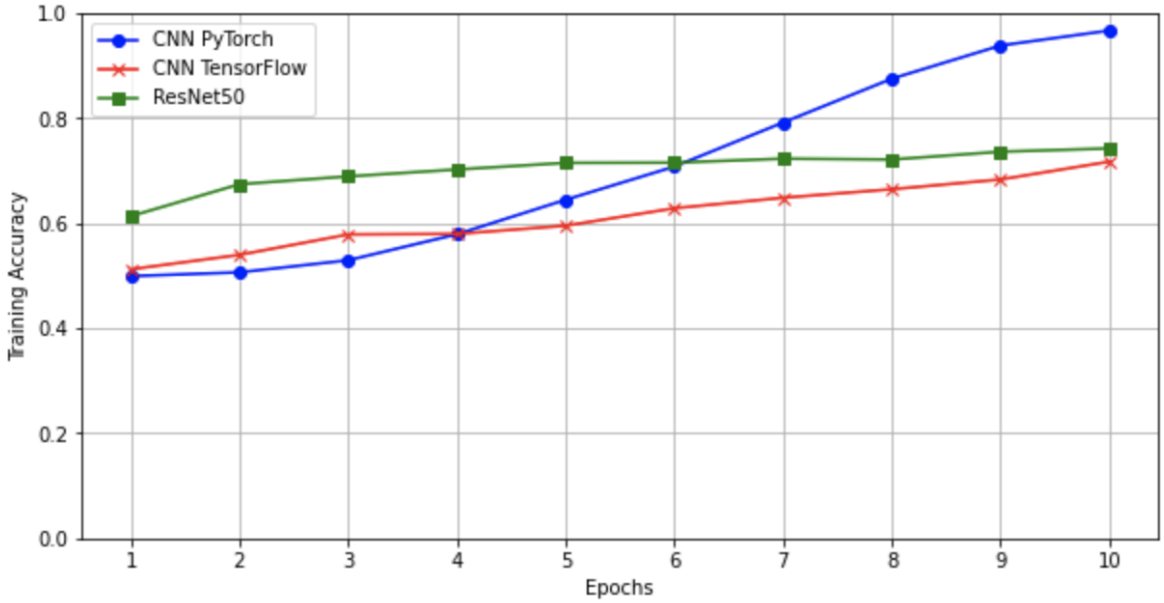

Figure 9: Training accuracy over 10 epochs

Each training phase goes through 10 epochs, an epoch is one complete pass through the entire training dataset by the learning model. The CNN PyTorch ends with the highest training accuracy, while the CNN TensorFlow ends with a much lower accuracy. That said, the ResNet50 maintains a consistent training accuracy over the 10 epochs.

Figure 10 below shows the difference in changes of the loss function throughout the training phase. The CNN PyTorch model shows a significantly lower loss function after 10 epochs, while also showing the highest validation loss. From a starting point perspective, all

models started with a loss function and validation loss of around 0.6, whereas the CNN TensorFlow model started a validation loss of around 1.35.

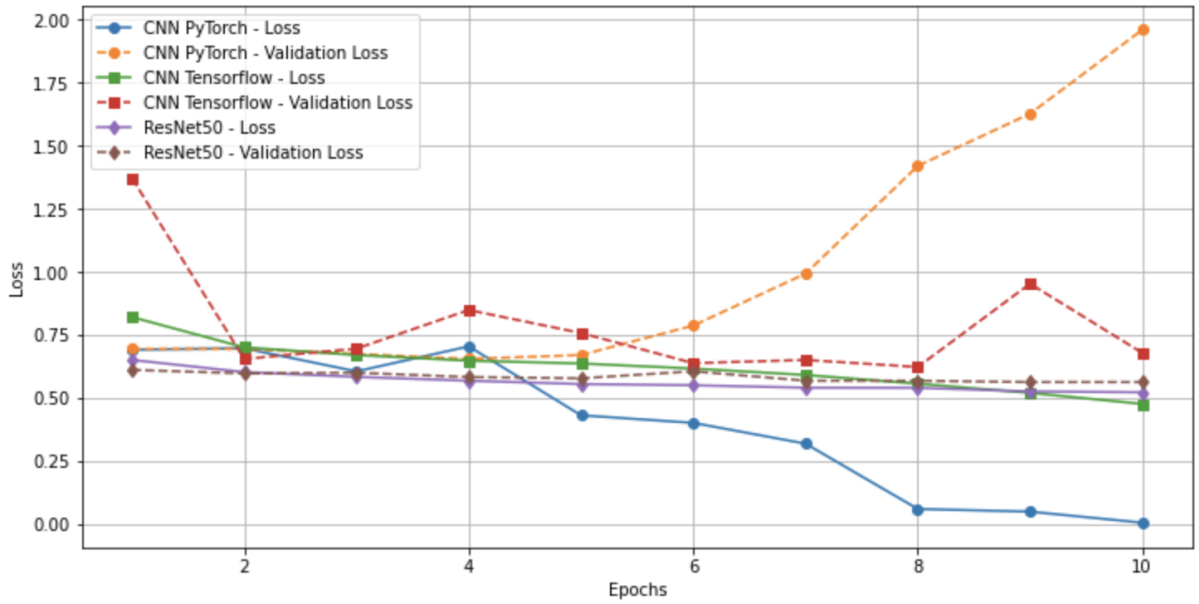

Figure 10: Changes in loss functions over 10 epochs

The confusion matrices below provide more detailed insights into the performance of the models. The top left box is a representation of the true negatives, which shows the rate at which the model correctly predicted the fake images as fake. The top right box is the false positives rate, where the model incorrectly predicted fake images. The bottom left is the false negatives rate, where the model incorrectly predicted real images as fake, and the bottom right is the false positives rate, where the model correctly predicted the real images.

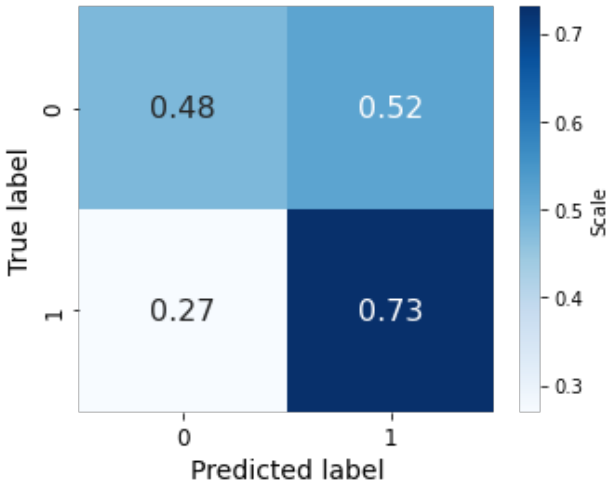

Figure 11: TensorFlow CNN

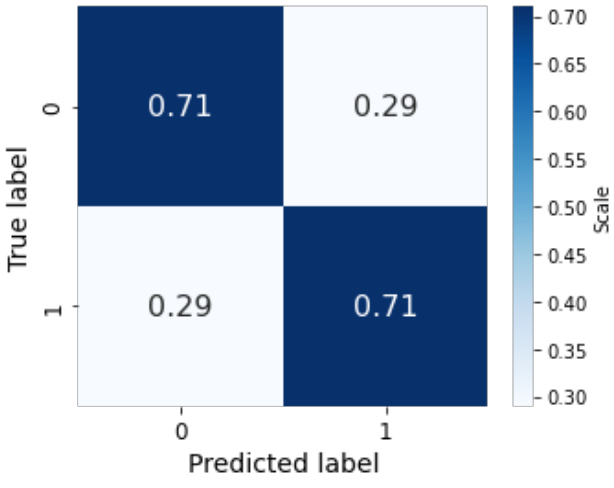

Figure 12: ResNet50 model

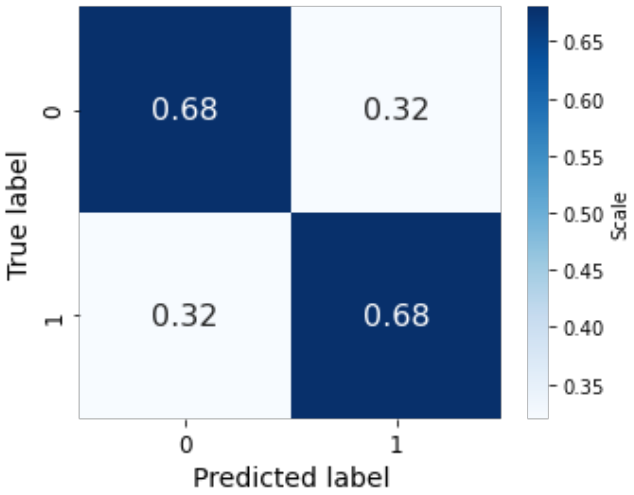

Figure 13: PyTorch CNN

Figure 14 below represents the performance of the three models using the true negative rate (specificity), which in this case evaluates how well the model identifies fakes, and the false negative rate as metrics.

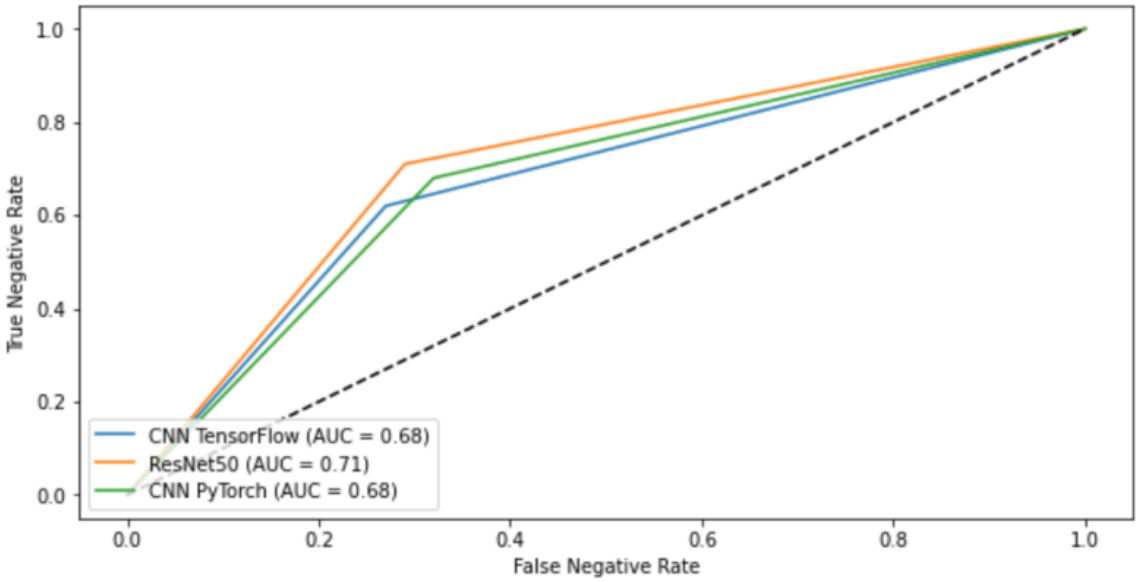

Figure 14: ROC curve comparison

Furthermore, the performance indicators of the three models can be seen in table 3 below for a comparison. The table shows the precision, recall, and F-1 scores for both classes for each model. Precision is a measure of how precise the model is with its predictions, whereas recall refers to the model's capability of capturing all the relevant instances in the dataset. Finally, F-1 score provides a balance between those two measures. The equations below show how each of these measures are calculated (Kundu, 2022) :

$$Precision = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Positives\ (FP)}$$

$$Recall = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Negatives\ (FN)}$$

$$F1\ Score = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)}$$

Table 3: A comparison of the average performance of the three models (rounded to two decimal places)

Model	Training accuracy (%)	Testing accuracy (%)	F1 score (%)		Precision (%)		Recall (%)	
			fake	real	fake	real	fake	real
CNN TensorFlow	71.98	61.43	52.21	64.12	63.41	59.25	48.28	73.12
ResNet50	76.35	71.72	63.39	72.32	71.21	73.48	71.18	71.19
CNN PyTorch	96.70	67.32	66.14	67.18	65.19	68.22	68.21	68.42

The statistics above are indicative of how well the models classified the images. For example, the average overall precision of the CNN TensorFlow model in the "real" class indicates that 59% of the instances that the model identified as real were real. Moreover, the overall average recall for that same model in the "real" class means that the model correctly identified 73% of the real instances.

Lastly, figures 15-17 below represents a few examples of the images (Hasan and Khan, 2022) that were commonly misclassified.

Figure 15: PyTorch CNN misclassified images

4.2.1 Evaluation of results

4.2.1.1 Accuracies

When looking at the accuracies of the models, the TensorFlow CNN model shows a noticeable decrease in accuracy from training to testing, which can imply a few things. Either the model may be overfitting the training data, or the model's architecture is too complex for data. If the model is overfitting for training data then it is too closely fitting the nuances and noise of the training data. Nonetheless, when looking at the structure of the model (can be

seen under Appendix B.2.1), the high number of parameters in the dense layers suggests that the model is indeed too complex, especially in the first dense layer with over 51 million parameters. Additionally, all the layers in this model are trainable, meaning that the model is learning features from scratch, this can lead to learning patterns that do not actually exist in the unseen testing data. Therefore, the complexity potentially increases the susceptibility to overfitting.

In comparison, the ResNet50 model shows a smaller drop from the training accuracy to the testing accuracy, which indicates a more robust model, generalizing better to unseen data. This robustness is supported by the simpler structure of the ResNet50 model (as seen under Appendix B.3) with very few trainable parameters. This reliance on pre-trained weights leverages learned features that generalize well.

Meanwhile, the accuracy of the CNN PyTorch incredibly decreases from the training phase to the testing. Similar to the CNN TensorFlow model, this implies a strong likelihood of overfitting. There are many factors intrinsic to the model structure that can contribute to this. When looking at the structure of the model under Appendix D.1, the linear layer is very large, with 25 million parameters. This layer potentially learns excessive amounts from the learning data, which increases the risk of memorizing the data rather than learning to generalize. Another observation here is the similarity between this model and the capability humans have in detecting deepfakes. Humans showed high confidence when classifying the images as fake or real, even when being incorrect. Similarly, the PyTorch CNN model leads with a very high training rate, implying high confidence, although the testing accuracy significantly drops.

4.2.1.2 Detailed performance indicators

The confusion matrices above from figures 11, 12 and 13 give more insight onto the performance of the models. The CNN TensorFlow model has a true negative rate of 48%, indicating that it can correctly identify 48% of the fake images, whereas it incorrectly classifies the remaining 52% of the fake images as real. Moreover, this model incorrectly classifies 27% of the real images as fake, and correctly identifies the real images 73% of the time. These statistics support the low accuracy in the testing phase, as correctly identifying only 48% of the fake images shows that the patterns learnt during the training phase may be misleading and do not apply to the unseen data. The model is decent at identifying the real images as it was able to identify 73% of them. Nevertheless, the model lacks overall accuracy as it poorly identifies the fake images.

Conversely, the ResNet50 model shows strength in its identical accuracy in both classes, with a 71% true negative rate and true positive rate. This implies that the model has better

overall accuracy, making it more reliable. Additionally, these rates also suggest that the model suffers from less overfitting, which is supported by its structure.

On the other hand, the PyTorch CNN model shows a true positive rate of 68%, indicating that 68% of the actual real images were correctly identified. Identically, the true negative rate shows that 68% of the actual fake images were correctly identified. These values show that the model was consistent in how accurate it was during the classification. Additionally, it suggests that there is no bias towards predicting one class more accurately, unlike the TensorFlow CNN. The accuracy of the model is quite good, however, with some improvements, better accuracy can be achieved, while reducing overfitting.

Furthermore, table 3 provides a good visual comparison between the performance of the three models. In terms of precision and recall, the ResNet50 model shows the best performance, with high precision and recall in both classes, indicating few false positives and negatives. This gives the model an advantage when needed for scenarios where both types of errors would have significant implications. The PyTorch CNN model also has balanced precision and recall, although slightly lower than the ResNet50 model. In terms of the TensorFlow CNN model, the low precision in the “fake” along with the high recall in the “real” class implies that it misclassifies many fake images as real. The F-1 scores indicate that the PyTorch CNN model and the ResNe50 model are well balanced, this balance indicates that these two models are more effective in generalization.

Looking back at figure 10 with the loss functions, there are a few pointers to discuss. Firstly, the very low training loss of the PyTorch CNN model indicated a very effective learning rate for the model. This is supported by the high training accuracy of the model as seen in figure 9. In comparison, the learning loss in the TensorFlow CNN model and the ResNet50 model gradually decreases as the epochs increase. This indicates a more stable learning rate in those models. At the same time, the validation loss rate in the PyTorch CNN model increases significantly after around 5 epochs, indicating overfitting. Similarly, in the TensorFlow CNN model, there is some increase in the validation loss after epoch 8, also a sign of overfitting. On the other hand, the ResNet50 maintains a stable validation loss, suggesting a strong capability to generalize.

In addition to this, figure 14 with the ROC curve comparison shows that the ResNet50 has the best AUC value, and higher specificity, suggesting it can better detect fake images.

To conclude these metrics, the TensorFlow model suffers from overfitting due to the complexity and large amount of learning parameters found in its structure. Whereas the PyTorch model overfits due to its potentially aggressive learning rate and less effective regularization techniques in comparison to the ResNet50 model. Possible improvements for the CNN models include more robust regularization techniques, refining learning rate adjustments, lowering the complexity of the models by reducing the learning parameters. In

terms of the ResNet50, fine tuning more layers and allowing them to take place may allow for better adaptation of the learning techniques in specific tasks like deepfake detection. In addition to this, all models can benefit from larger datasets and different techniques of testing like cross validation. Different training, validation, and testing splits may also aid in viewing different performance from all the models.

4.2.1.3 Misclassified images

A sample of incorrectly classified fake images by the PyTorch CNN model were reviewed in figure 15. Several factors can cause the models to incorrectly classify the images such as:

- Low resolution and blurred out images.
- Unnatural facial expressions such as facial hair, shut eyes, and alignment.
- Background, lighting, and visual inconsistencies.

The images from figure 15 suggest that faces with complex expressions like shut eyes and facial hair are more difficult to correctly classify as deepfakes by CNN models, as they make the person's face seem unnatural. This is because they are both prone to overfitting on complex detailed images. Additionally, they lack the sophisticated built-in features of the ResNet50 architecture, which is renowned for its deep architecture as it is pretrained on ImageNet. As mentioned in Chapter 3, this a dataset with a diverse range of images. This pretraining helps in developing a robust initial understanding of image features, which is useful for deepfake detection. That said, images that are partially blurred out or come in low resolution can trick the model into thinking they are deepfakes as seen in the first image from figure 15.

4.3 Concluding remarks

4.3.1 The quiz

The quiz served as an invaluable tool for gauging human ability in detecting deepfakes. With a sample size of 56 participants, it is a reliable indicator of how well or bad humans can detect deepfakes, and how different their decision making is based on the type of content (images and videos). The biggest observation is the disconnect between confidence and accuracy, excluding a few extreme examples. This discrepancy points to an expected cognitive bias in human judgement, or in other words "overconfidence". The high confidence did not correlate with the correct answers, which aligns with the previous research discussed in Chapter 2. Moreover, participant feedback revealed a tendency to anticipate deepfakes in every question, which in return lead to a high false negative rate – classifying real content as deepfakes. That said, the judgement criteria were influenced by the types of videos shown as well. Participants were more suspicious of political videos, likely due to its common association

with deepfakes. Most participants also claimed that the quiz was difficult, although the difficulty of the content was moderate. Likewise, participants claimed that their trust in the authenticity of the content they view is quite low, which emphasizes the urgency for effective solutions.

Several enhancements could refine the effectiveness of the quiz. One improvement could be adding a time limit for viewing the content based on the difficulty, which would prevent overthinking, incorporating audio-only deepfakes further expands the understanding of human detection beyond visual content. In addition to this, a brief training or tutorial session on what to expect from deepfakes can help participants prepare for the quiz better. Another adjustment for more accurate results could be a larger sample size.

The results from the quiz underscore the need for more comprehensive training, which does not just educate people about deepfakes, but also raises their self awareness of their own biases. Ultimately, the results reinforce the need for AI-driven solutions in detecting deepfakes, the need of implementing a system that can protect users from this exposure.

4.3.2 Image classification neural networks

The comparative analysis of the PyTorch CNN, TensorFlow CNN, and ResNet50 models provides a robust framework for understanding the strengths and limitations of image classification neural networks in deepfake detection systems. The three models reveal significant findings, and critical insights. All in all, the results from the three models provide us with several conclusions:

1. PyTorch CNN and TesnsorFlow models suffer from overfitting.
2. ResNet50 architecture has the best performance and reliability on complex images.
3. PyTorch CNN has the quickest learning rate, yet still suffers from overfitting.
4. PyTorch CNN and ResNet50 models are more accurate in correctly classifying deepfakes in comparison to TensorFlow CNN.
5. Images with more complex features and images with less shown features are more prone to misclassification.

Every model showed strengths and weaknesses. The PyTorch CNN model proves to be useful with its fast-learning rate as it can quickly adapt to the training data. Despite that, it suffered from overfitting as its testing accuracy did not match the high training accuracy. These observations can be advantageous in rapidly changing content, where the detection model needs to adapt quickly. Moreover, the model is well balanced in its decision-making accuracy, therefore, it is an attractive option for situations where the accuracy in classifying both real and fake images is critical. This model can come in very useful for online platforms that are time sensitive. For instance, any live streaming platform will immediately show unassessed content. Implementing this type of model is beneficial due to its fast learning and adaption

capabilities, the model can adjust to new deepfakes as they are developed. That said, companies can employ this type of model to maintain brand integrity and consumer trust.

Similarly, the TensorFlow CNN model benefits from somewhat good training accuracy, although just like the PyTorch CNN model, it suffers from overfitting issues. That said, the TensorFlow CNN model proved to have lower capabilities in detecting fake images, and showed less balance in the image classification accuracy between both classes “real” and “fake”. Nonetheless, the TensorFlow CNN model has a more versatile complex model than the PyTorch CNN model, which is more capable of capturing intricate details. Both CNN models are advantageous in scenarios where overfitting is less of a concern.

In spite of that, the ResNet50 proves to be the best overall model. This model illustrated more stable performance and superior generalization. The overall structure is simpler and more efficient, and it relies heavily on learned features from pre-trained models. The model does not show any signs of overfitting or underfitting, and proves to be consistent in its performance across different datasets. ResNet50 architecture outperformed the other two models in precision and recall, proving its reliability. Although both the PyTorch CNN model and the ResNet50 model are reliable, the ResNet50 architecture takes the edge due to its ability of correctly identifying complex sets of images. The ResNet50 model is also the best for general purpose image recognition due to its ability to scale. That said, the model can be utilized in security systems in identifying manipulated content, such as altered surveillance footage. These scenarios require models of high accuracy and reliability, as mistakes can be critical.

4.3.4 Summary

This research has been successful in achieving its aims, and in demonstrating the importance of emphasizing ways of detecting deepfakes. The quiz gave a very clear idea of how capable humans are at detecting deepfakes. The lack of correlation between the overconfidence and the correct results shows how unaware people can be when viewing fake content around the internet. In terms of the image classification models, selecting the better model heavily depends on the requirements and constraints. In a controlled environment where overfitting can be limited, the CNN models can excel. More specifically, the TensorFlow CNN model can excel in precision-focused tasks, and the PyTorch CNN model in environments that require flexibility and rapid adaptability, whereas the ResNet50 model provides a more overall robust and reliable performance. After experimenting with different types of image classification neural networks, it is clear that they each benefit from a unique factor, however, the PyTorch CNN, and the ResNet50 architecture take the edge in their effectiveness and reliability.

4.4 Ideas for future work

The conducted research on deepfake detection methods proposes several ideas for future work. The ultimate goal of extending this project would be to create an effective deepfake detection system. The quiz from this research can be utilized in developing education tools and in shaping awareness campaigns. Researchers can identify the patterns from deepfakes that trick humans and use this knowledge to improve the use of AI in detecting deepfakes. Additionally, comparing the performance of humans against AI techniques, can lead to hybrid systems that combine the strengths of both. Moreover, the quiz can serve as valuable data for psychological or neurological research, as the results can be an indication of what cognitive biases or heuristics humans rely on when making decisions.

Despite the several improvements that can be made to the three image classification models, including reducing model complexity, optimizing training process, and data augmentation, they can still be highly influential for future work. Reducing complexity of the TensorFlow CNN model will help mitigate overfitting, such as by adding additional dropout layers after the dense layers, to drop units during training. Additionally, all models can benefit from implementing a learning state schedule, or adaptive learning rates, which will enhance training efficiency. Moreover, the optimization of the training process will benefit all models, methods like early stopping and cross validation are beneficial. Early stopping when validation performance stops improving prevents overfitting. While cross validation provides better estimates of model performance and can aid in fine-tuning hyperparameters. Both CNN models will benefit from methods that aid in generalization, and the ResNet50 model will benefit from finetuning in its last few pre-trained layers to specify in tasks like deepfake detection. Lastly, enhancing the models to detect videos is a great idea for future work.

Furthermore, these image classification models can be used in several ways for deepfake detection. The integration of an image classification model on social media platforms can undoubtedly enhance user awareness and mitigate the harmful impacts of deepfakes. If every upload of a deepfake includes a deepfake label, no one will be deceived into believing false information, and the trust of users in their platforms will increase. This integration can be beneficial for social media platforms, streaming platforms, and in browsers. For instance, these models can be beneficial in browsers in cases where the user wants to fact check an image or a video.

For future work, exploring other neural networks can be beneficial for creating a deepfake detection system capable of detecting all types of deepfakes. Using RNNs is useful for detecting audio, investigating transformers, hybrid networks, and GANs can lead to promising deepfake detection capabilities.