Identifying StyleGAN images

Neil Foxcroft

orcid.org/0000-0002-8389-8826

Thesis submitted for the degree *Bachelors Honours* in Computer Science and Information Technology at the **NWU**

Supervisor: Dr. R. Serfontein

2021

School of Computer Science and Information Systems

North-West University

Student number: 28418077

Acknowledgements

I would like to thank the following persons for their support and guidance throughout this project that enabled me to complete it successfully.

Dr. Rudi Serfontein, Thank you for your insight and support in my research project and always being there to answer any questions I had. Your guidance enabled me to complete the project successfully and professionally.

Prof Tiny Du Toit, Thank you for the knowledge you shared with me regarding anything related to artificial intelligence. As a master in your craft it was helpful to learn from you. Thank you for making available your GPU that helped me train a portion of my artifact on.

Manre' van Zyl who I worked alongside in 2021 while both of us where conducting our research

Abstract

The creation of a Style-Based Generator Architecture for Generative Adversarial Networks (StyleGAN) introduced the world to the possibility that images can be generated in such a way that it is difficult for a human to detect that these images are not real. A study conducted on StyleGAN and Convolutional Neural Networks helped in the identification of these images. A was created artefact that demonstrate the proposed neural network solution method. The created CNN proved that this method of detection resulted in promising results. To improve the detection accuracy further the hyperparameter optimization framework Optuna was implemented on the neural network and dataset to increase the model's accuracy drastically. Hyperparameter optimization is a powerful tool in machine learning that can structure a neural network architecture in such a way that massive gains are made in the networks predictions. To allow for easy interaction a Front-end web application formed part of the artefact and the Flask framework was used for the model implementation and the user interface. Uploading images to the application will provide users with feedback whether or not a specific image was generated using StyleGAN or if the image is that of a real human being.

Keywords: Convelutional Neural Network, Flask, Hyperparameter Optimization, Machine Learning, Neural Networks, Optuna, StyleGAN

Opsomming

Die skepping van ’n stylgebaseerde skeppende argitektuur vir generatiewe teëstanderige netwerke (StyleGAN) het die wêreld bekendgestel aan die moontlikheid dat beelde op so ’n manier gegenereer kan word dat dit moeilik is vir ’n mens om vas te stel dat hierdie beelde nie werklik regte beelde is nie. ’n Studie wat op StyleGAN en Konvolusiese Neurale Netwerke gedoen is, het gehelp met die identifisering van hierdie beelde. ’n Artefak was geskep wat die voorgestelde neurale netwerk oplossing metode demonstreer. Die geskepte KNN het bewys dat hierdie metode van identifikasie belowende resultate tot gevolg gehad het. Om die identifikasie akkuraatheid verder te verbeter, is die hiperparameter optimeringsraamwerk Optuna op die neurale netwerk en datastel geïmplementeer om die model se akkuraatheid drasties te verhoog. Hiperparameter-optimering is ’n kragtige instrument in masjienleer wat ’n neurale netwerkargitektuur op so ’n manier kan struktureer dat massiewe winste gemaak word in die netwerkvoorspellings. Om maklike interaksie moontlik te maak was daar ’n webtoepassing deel van die artefak ontwikkel en die Flask-raamwerk is gebruik vir die modelimplementering en die gebruikerskoppelvlak. Die oplaai van beelde na die toepassing sal gebruikers terugvoer gee of ’n spesifieke beeld met StyleGAN gegenereer is en as die beeld dié van ’n regte mens is.

Sleutelwoorde: Flask, Hiperparameter-optimering, Konvolusiese Neurale Netwerk, Masjienleer, Neurale Netwerke, Optuna, StyleGAN

Table of Contents

List of Figures

List of Tables

Table of abbreviations

A table containing a list of abbreviations in the order of appearance, that will be used throughout text.

StyleGAN	a Style-Based Generator Architecture for Generative Adversarial Networks
GAN	Generative Adversarial Network
AI	Artificial Intelligence
DSRM	Design Science Research Methodology
SWOT	Strengths, Weaknesses, Opportunities and Threats
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
VS Code	Visual Studio Code
px	pixles
AI	Artificial Intelligence
AI	Artificial Intelligence
AI	Artificial Intelligence
AI	Artificial Intelligence
AI	Artificial Intelligence

Chapter 1

Introduction

The creation of a Style-Based Generator Architecture for Generative Adversarial Networks (StyleGAN) and similar technologies introduced the world to the possibility that images can be generated in such a way that it is difficult for a human to detect that these images are not real and instead generated by a neural network. With this proposed research project, the StyleGAN technology will be thoroughly studied to determine how an approach can be developed to identify artificially generated images to enable the detection of these falsified identities that used the StyleGAN technology. An artefact will demonstrate the proposed solution methods function and convey the successful method in a practical environment with the use of StyleGAN generated images and images retrieved from the Flickr Faces dataset to evaluate the method. The following section is the proposal for the research project and will form the first chapter of this proposed project.

1.1 Project Description

StyleGAN is an open-source Generative Adversarial Network (GAN) that can be used to generate faces of people that do not exist (such as those shown on thispersondoesnotexist.com). This means that fraudsters can use StyleGAN generated faces that normally would pass a visual inspection conducted by a human inspector as part of false identities. The detection of such images with the use of artificial intelligence will be useful because of the factors that currently lead to misidentification.

1.2 Project Background

In this modern world with humanity currently in its 4th industrial revolution, the additions of innovative technologies require original approaches to implement and maintain these technologies. The change from the digital age to the automation age is accelerated by breakthroughs in the fields of Artificial Intelligence (AI) and security (Skilton and Hovsepian, 2017).

One of the big advances in AI is the creation of StyleGAN, this new approach to a GAN allowed for more control in an image than its predecessors (Karras et al., 2019). StyleGAN uses the principles of a GAN to create new images derived from input that specifies what "styles" need to be included in the image. According to Karras et al. (2019) a style is defined in this context as a set of parameters that modifies the input of the image to result in different outputs. If the input received is that the image needs to be in the style of a person with glasses, red hair and must be female, StyleGAN will then generate that image based on images used of that similar styles in the initial training of the model. The resulting image will thus be that of the "styles" required in the input. While this functionality can be utilized for various positive use cases – this breakthrough also creates various challenges and setbacks in the field of security, more specifically the aspects of facial recognition and identity verification as malicious use of this GAN might aid in the creation of fraudulent identities (Mitra et al., 2021). Figure 1.1 demonstrates how an image can be altered with the use of StyleGAN by specifying what style changes should be made on that image.

The styles applied in figure 1.1 demonstrate the capabilities of StyleGAN and further illustrate the possible difficulties of detecting that these images were generated with a GAN (Karras et al., 2019). Possible misidentification of StyleGAN images is a reality that needs to be addressed. Humans in the role of identifying artificially generated human faces may be susceptible to external factors hindering their capabilities and increasing the rate of error in which they identify fraudulent images (Fysh and Bindemann, 2018). Fysh and Bindemann (2018) also noted that in the specific use case of passport officers that were tested on passport images captured on the same day against the "traveller" presenting that image, that the officers made substantial errors in a controlled environment when comparing the picture identity to that of the traveller. These results enforced their original statement that humans struggle with unfamiliar face identification.

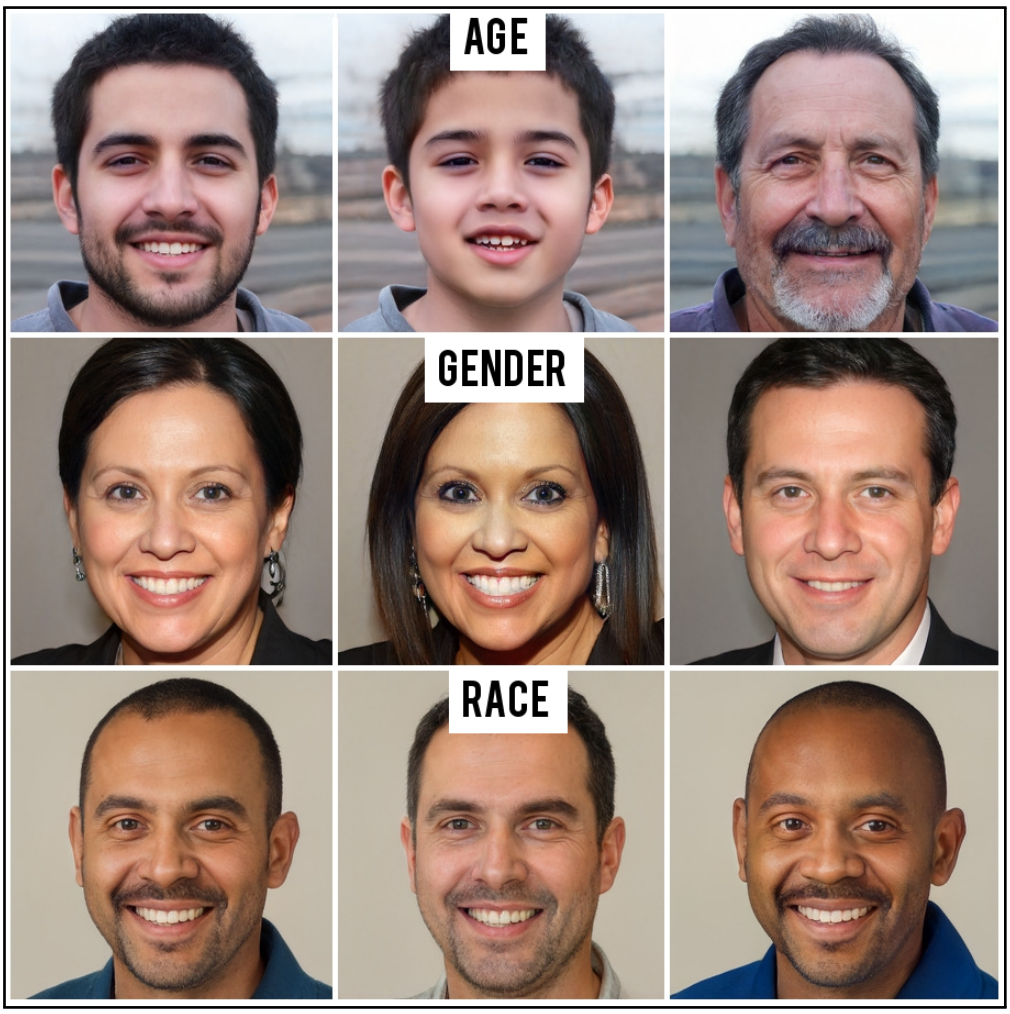


Figure 1.1: Applied "styles" on images using StyleGAN that demonstrates styles and the resulting changes adapted from Karras et al. (2019)

Opportunities for fraud escalated with the recent boom in online applications. This bigger threat for fraud is exaggerated in the mobile market due to the high volume and demographic variety of current mobile users (Mitra et al., 2021). Most applications require user accounts and the unique user's identity is commonly the specific value the application requires. Examples of this in practice is online dating applications namely Bumble, Tinder and Hinge. These free to use applications allow multiple users to connect and interact where interaction mostly starts because of the user's interaction with the images displayed on a profile. This emphasis on the gallery of a user profile creates a unique opportunity where StyleGAN can help cyber criminals fake identities within these apps. Fraudulent use of StyleGAN in this environment can directly lead to an increase in catfishing incidents. Catfishing is the act of deceiving an individual with the use of a specific fictional identity and persona to gain assets that are usually in the form of financial gains or personal information (Chandler et al., 2016). These apps usually have various systems in place to combat such fraudulent actions by requiring

account verification. The verification process requests that a user submit one or multiple images where they pose in specific ways. StyleGAN can successfully counter these forms of verification as an image can be generated wherein a person perform specific poses (Karras et al., 2019).

Because of StyleGAN's ease of use and availability more fraudsters used it to create false accounts on popular social media platforms since its creation. Facebook took down an undisclosed number of accounts in December 2019 that had reportedly made use of StyleGAN to generate realistic profile pictures for establishing false identities on their site (Gallagher and Calabrese, 2019). This emphasized the possible threat StyleGAN poses to the security of individuals and the IT industry. Because of the threat, StyleGAN poses the need for a method to detect these generated images is identified.

StyleGAN was released to the public in December 2018 with all its packages and source code. This novel approach to GAN's that was developed by Nvidia demonstrated its possible capabilities by the accompanying portraits of convincing human faces. The big leap forward in this technology was the realism brought to the GAN's generated faces dataset that is close to real human beings and not easily detected by humans (Fleishman, 2019; Karras et al., 2019).

StyleGAN was popularised because of a former engineer at Uber, Phillip Wang's website thispersondoesnotexist.com that was released in February 2019. Phillip's main goal when publishing this website was to educate the public on how GAN's work and the dangers that they might pose to the average user. Phillip achieved this by specifically emphasizing StyleGAN and its realistic human face generation capabilities Fleishman (2019). The goal of making more people aware of StyleGAN and possible fake identities was also promoted by another website. Two members of the University of Washington created the website whichfaceisreal.com Fleishman (2019). This website allows users to select an image between a StyleGAN image and a verified human image. This online tool helped users realise that the differences between computer-generated images and real images are only decreasing with the advancement of technology.

The big advancement in StyleGAN that increased the need for this proposed project was the release of StyleGAN2 in February 2020 (Karras et al., 2020). Following the release of version 2, there were improvements in the generation process of these images with this updated version of StyleGAN. StyleGAN2 saw that images were no longer subverted with artefacts or traces left on the image because of the processing method used in the previous iterations (Karras et al., 2020). These traces were easy to identify and clearly showed out of place in the context of the image. With the removal of the traces, usually, in the form of drop-like spots, the difficulty in detecting the generated images increased and similarly the need to detect StyleGAN generated images used maliciously in security verification processes.

By looking at how the technology has been used since its release, the possible use-cases for GAN generated images and the always growing cybercrime industry the possible detection of these images is identified as a crucial function in the 4th industrial revolution. With these identified factors will the proposed project aim to solve the problem of detecting StyleGAN images by using an artificial intelligence approach to solve the problem.

1.3 Research Question

With the identified need for detection of StyleGAN images and the discussed security implications that the invention of StyleGAN and similar methods introduced the proposed project aims to detect these images with the use of a trained neural network. The main research question that this proposed project aims to answer is: How can StyleGAN generated images be detected?

1.4 Aims and Objectives

1.4.1 Aims

The main purpose of the proposed project is to develop a method that can detect fraudulent human faces created by StyleGAN with relative accuracy. Various techniques and approaches to the detection of GAN generated images will be researched and the simplest implemented approach that can still detect these types of images with relative surety will be selected for the artefact.

1.4.2 Objectives

The success of the project will be weighed against the completion of the secondary objectives that have been identified as listed below.

- Perform a literature study on GAN's and specifically analyse the architecture and function of StyleGAN to understand the technology.
- Develop an approach to the successful identification of generated images.
- Develop an artefact that will use the selected method to detect a fake identity.

The successful completion of the above-identified objectives will aid the researcher in satisfying the aim of the proposed research project.

1.5 Procedures and Methods

This section will describe the research paradigm and the methodologies that will be used to complete the proposed project. By examining the chosen paradigm, methodology and artefact life cycle in an academic viewpoint with specific reference to information systems design and development will the most applicable approaches in these sections be identified and selected. The data that will be captured in the development phase and reviewed in the testing phase will be discussed.

1.5.1 Paradigm

Positivism is a research paradigm that is focused on the world view that "factually accurate" knowledge is gained through the observations made by the observer. In positivistic studies, the research is confined to only the collection of data and the interpretation of this data to gain knowledge and insight into the problem. Positivism requires the researcher to only make quantifiable observations that can directly lead to statistical analysis. With the use of this paradigm, the researcher must reject intuitive knowledge because it cannot be justified by sensory experiences and is thus subjective to the researchers own interpersonal influences. Positivism as a philosophy is justifiable by empiricist views that knowledge stems from a human experience (Collins, 2018).

The positivistic research paradigm is suitable for the proposed project because with the creation of the artefact, the data collected will be examined and an unobjective interpretation of the data is necessary. The data collected will be the results of the artefact's successful identification of StyleGAN generated images.

1.5.2 Methodologies

Methodologies determine the structure of completion for a specific project. With this proposed project the researcher will study the technology that enables StyleGAN to generate images. To aid in the research process and the development of the artefact the Design Science research methodology (DSRM) will be used throughout the proposed project (Peffers et al., 2007).

DSRM is an information systems specific methodology that focuses on research and iterative design (Peffers et al., 2007). Because the researcher is studying the field and technologies in which they want to solve the specific problem the background knowledge of the problem will be explored parallel to the design of the problem solution. DSRM will enable iterative design and development throughout the completion of the proposed project.

1.5.3 Artefact Life Cycle

With the use of the DSRM, the researcher will amend the project with the acquisition of new knowledge. The artefact will be developed while research is conducted on the technologies required, thus the artefact development will similarly be conducted in increments. The implementation of the Agile methodology for the development of the artefact will be suitable as Agile accommodates changes in artefact scope, planning and incremental deployments of preliminary artefacts (Weiyin et al., 2011).

The artefact that will be developed will aim to detect StyleGAN generated images between a data set of real images of human faces and StyleGAN generated images. The artefact can be hosted online if needed as this increase its compatibility and deployment reach across multiple platforms. Because of these factors, Agile will be the most suitable methodology for artefact development.

1.5.4 Data Capture

The data captured in this project will be collected and displayed to the user in the web application to aid in the demonstration of the artefact, and method of detections success in the finalization phase of the project. Data that will be captured is the number of images used in the training of the models, the number of images that were used throughout the identification testing phase and the accuracy of the chosen identification method. These statistics will help the researcher determine the success of the chosen method of detection.

1.6 Project Management and Project Plan

1.6.1 Scope

The research of the technologies implemented in StyleGAN and the specific architecture of StyleGAN is important for the design of a suitable approach for the detection of these images. The scope of this research is the identification of images, the use of artificially generated human faces, security concerns when determining identities based on profile images and the research of neural network's that will be a possible technology used for the solution to the problem.

Because of the time of StyleGAN3 and 2 individual releases and the initialization of the proposed project the scope of this project will only focus on the detection of StyleGAN1 generated images.

1.6.2 Limitations

Possible limitations that can be faced during the proposed project will hold back advancements that the researcher aims to complete in this study. The limitations thus need to be identified and addressed to ensure the completion of the proposed project within the specified period. The proposed project will make use of neural network's and image processing technologies to answer the research question. These technologies require large computing capabilities for the training of neural network models. This will be addressed by using cloud services instead of traditional hardware. Cloud services provide a more cost-effective approach to large computing needs.

1.6.3 Risks

Possible risks that can be identified that this proposed project is susceptible to include the risk that the scope of the project is not adhered to. The scope defines the boundaries in which this proposed project will take place, therefore careful adherence to the scope will ensure that the research is relevant to the initial project that was proposed. The responsibility of staying within the scope of the project lies in the researcher proposing this project.

1.6.4 Strengths, Weaknesses, Opportunities and Threats

The Strengths, Weaknesses, Opportunities and Threats (SWOT) of this proposed project can be evaluated using a SWOT-analysis table. The risks and limitations mentioned previously is also mentioned when conducting a SWOT analysis. Figure 1.2 shows the SWOT analysis of the proposed project.

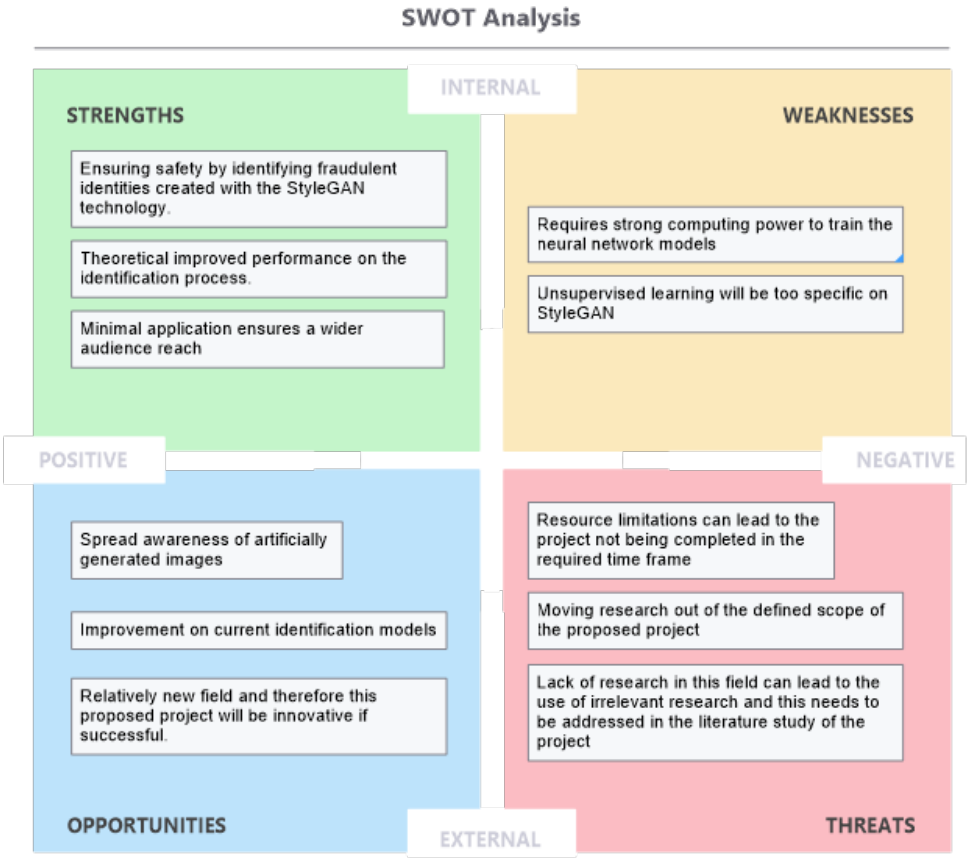


Figure 1.2: SWOT analysis in the Identification of StyleGAN images

1.6.5 Timetable

The proposed project will start on the 16th of February 2021 and will be finalized and completed on the 8th of November 2021. This project will be subdivided into 3 phases each with separate deadlines. The first phase is the research proposal that will be concluded on the 18th of April 2021. The second phase involves research that will require an in-depth analysis of the problem, possible solutions, and the discussion of StyleGAN in a literature study that must be completed on the 13th of June 2021. The last phase is the development of an artefact to practically demonstrate the solution to the identified problem. Phase 3 of the proposed project will also require a demonstration of

the developed artefact and a video demonstration of the project on the 1st of November 2021. The submission of the whole project must be completed and the final documentation will take place on the 8th of November 2021. The Gantt chart in figure 1.3 graphically displays the preliminary project planning with the tasks in the required sequential order for the successful completion of the proposed project. The adherence to the project planning and the preliminary Gantt chart will enable the researcher to effectively divide the tasks into manageable time frames. The prerequisite tasks are required to begin with a dependent task and are displayed graphically in figure 1.3.

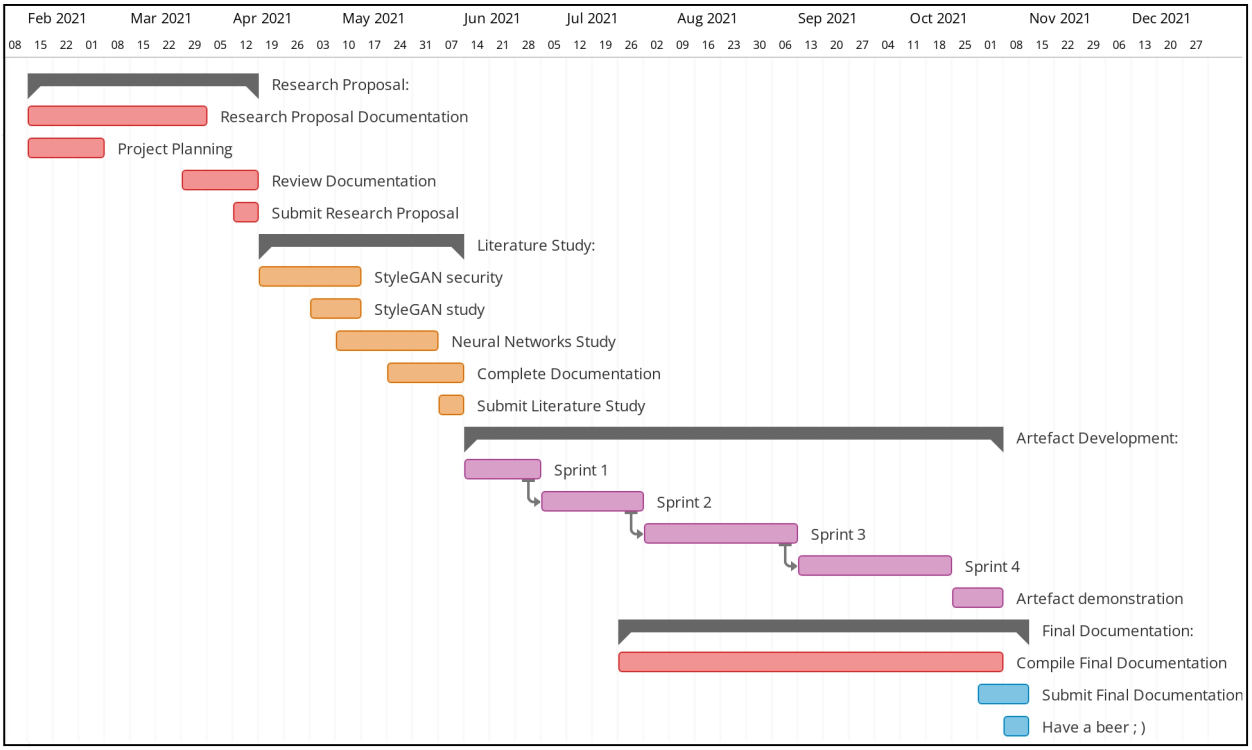


Figure 1.3: Gantt Chart graphically demonstrating the planned schedule of the proposed project

In figure 1.3 time management and planning is clearly shown to be crucial for the completion of the proposed project. The artefact development planning is subject to change based on the research outcomes and findings of the researcher in this proposed project.

1.7 Development Platform, Resources and Environment

The development of the artefact will be conducted in parallel with the research on the subject. Development of the GUI will be conducted at the same time as the literature study. The development of the final artefact and method of detection will only be conducted after the literature study based on the knowledge acquired in that phase. With the knowledge gained in the literature study will the specific method to detect StyleGAN images be implemented into the artefact. The artefact development platforms, resources and environments will be discussed in this section.

1.7.1 Web Application

The artefact will be a web application that can be scaled and hosted through a cloud services provider. The language that will be used for the front-end web application is Python-based using the scaled-down Django framework, Flask. For the implementation of the method for detection of StyleGAN generated images the language used in the back end of the artefact will be Python. The reasoning behind the selection of the above-mentioned technologies is discussed in the following section.

Developing a web-based application allows for further reach and compatibility compared to historical installation software (Murugesan et al., 2011). Developing a simple web application with a minimalistic front end will users more intuitively be able to use the application and will improve complex implementations ease of use (Murugesan et al., 2011). Web apps opened in the browser natively scale to be available on various devices and screen sizes. For the StyleGAN artefact, improved availability will provide the application with a wider reach and therefore more artificially generated images can be detected.

The web application will allow a user to load a new image where the artefact will then classify that image as a StyleGAN generated image or a real human image. The web application must keep track of basic statistics and provide them to the user. The basic statistics that will be provided in the artefact of this proposed project will be the neural network's confidence in its prediction. An uncomplicated design will be implemented where the user can intuitively navigate the web page.

1.7.2 Python

For the backend of the proposed project Python will be used. Python is an intuitive programming language that due to its active community has an abundant set of resources available to use in the artificial intelligence field. Python is widely used for artificial intelligence applications and thus is a suitable programming language to code the chosen method of detection.

1.7.3 Flask

For the front-end of the artefact's web application Flask will be used. Flask is a framework based on the larger framework Django that is a web framework using the Python language. Flask front-end features can use CSS-styling and the use of bootstrap will make the front-end uniform that enables faster development without the need for extensive visual design. Flask will enable the artefact to be used on multiple devices with a lightweight package without the explicit code and design to accommodate those devices. The framework natively scales assets to fit a wide range of devices.

1.7.4 Jupyter Notebook

Jupyter Notebooks is a Docker-like implementation of python coding. In a Jupyter notebook cells can be run individually allowing for more control when developing neural network's. Most cloud services use Jupyter notebooks when implementing python code on their platforms.

1.7.5 Cloud Services

Because of the possible processing resource requirements that the detection of StyleGAN images require, the training of the neural network model will be conducted on the platform and infrastructure that is provided by Google Colab.

PaaS and IaaS

In cloud services, there are multiple forms in which the cloud can be implemented for use by an individual or organisation. These implementations can range from Software to Infrastructure offered to the entity from the cloud services provider (Pfleeger and Pfleeger, 2002). For this proposed project there is a need for a platform and infrastructure as the training using large image datasets require processing power. Platform as a Service (PaaS) is a service model where the client develops software by using the languages and tools offered by the cloud services provider (Pfleeger and Pfleeger, 2002). Infrastructure as a Service (IaaS) is the second service model this proposed project will require. In IaaS the cloud services provider offers the use of processing resources and storage to name a few, to the client (Pfleeger and Pfleeger, 2002). A combination of IaaS and PaaS will be required for the completion of the artefact in this proposed project.

Google Colab

Google Colab is a free-service that Google offers to data scientist to use for the development of neural network's. Google Colab workbooks is the development platform in which programming is conducted that is similar to Jupyter Notebook development on local machines. The difference in using Google Colab is the increased system resources of the web based virtual machine. Using Google Colab for the development in this project will have the benefit of the GPU attached development environment where a Nvidea K80 GPU with 12GB of RAM can be used.

Google Drive

Google Drive is a storage solution offered by Google that is integrated in Google Colab. The large datasets can be uploaded to a NWU google accounts drive, where storage is unlimited. The dataset can be mounted in Google Colab using Google Drive.

1.7.6 Visual Studio Code

Visual Studio Code (VS Code) will be the selected IDE for the development of the artefact. VS Code allows for extensions that enable easier programming practices. ESLint is an automated code formatting and standardizing tool that was developed by Nicholas C. Zakas in June 2013. With the use of ESLint all code will be the same structure and allow for easier review of code. VS Code built-in terminal is a powerful addition to the IDE and will allow for Git statements to be run in the development environment.

1.7.7 Git

For version control and deployment, the Git language will be used and the repository for the development of the artefact will be hosted on GitHub. GitHub is integrated with various cloud services and this allows the developer to easily deploy and evaluate the code.

1.8 Ethical and Legal Implications

With the development of this artefact, certain resources will be used to train the neural network. The training requires images that were generated by StyleGAN. For the comparison in the artefact images of real humans will be used.

The StyleGAN generated images are available on the official StyleGAN GitHub repository. Included in this repository is trained StyleGAN models and multiple datasets of StyleGAN generated images. The licencing of these images is stated on the GitHub repository and is a Creative Commons license by NVIDIA Corporation (Karras et al., 2019).

For the verified human faces, the preliminary dataset that will be used is the Flickr Faces dataset that was initially used to benchmark StyleGAN. The individual images were published in Flickr by their respective authors under either Creative Commons, Public Domain. All of these licenses allow free use, redistribution, and adaptation for non-commercial purposes (Karras et al., 2019).

1.9 Provisional Chapter Division

For this whole project, the following flow of the final document can be expected. These chapters will logically flow to aid in the understanding of the topic at hand and to ultimately ensure a successful artefact.

Chapter 1: Introduction

This chapter will be concluded in the project proposal phase. It will include the research question, the project description and background. It will include the proposed project plan for the entire project.

Chapter 2: Literature Study

This chapter will be comprised of all the necessary research to understand the project and fulfil the project aims and objectives. Mostly in this project will be focusing on the specific workings of StyleGAN to effectively detect fake images.

1. StyleGAN: The history of StyleGAN and direct technologies that lead to this breakthrough in the field of GAN's will be provided as well as the specific architecture of StyleGAN.
2. Neural Network: This project will make use of Machine Learning and neural network's to detect StyleGAN generated images. A study will be completed on the field of artificial intelligence and how neural network's can aid in the detection of images.
3. Hyperparameter Optimization will prove useful in improving the created neural network model to increase the detection accuracy for StyleGAN images. A Study on hyperparameter optimization methods will aid in the creation of a hyperparameter artefact.

The study of these three sub-topics will provide the relevant background and base knowledge to develop a successful artefact and will provide the relevant foundation to improve and expand on this project in the future.

Chapter 3: Development of the artefact

Chapter 3 will apply the chosen methodologies that were identified and discussed in Chapter 1 to enable the successful development of the proposed project's artefact. This chapter will document the artefact development phase including unfamiliar problems that are identified within the development stage. The success of the artefact will be compared to the Aims and Objectives of Chapter 1 and if they are met.

Chapter 4: Results

The results of the Artefact will be introduced in this Chapter and the successful identification of StyleGAN images will be determined. The testing in this Chapter will identify the success of the artefact with the comparison in Chapter 4.

Chapter 5: Reflection

Chapter 5 will summarize the entire proposed project and conclude if the problem was solved with the successful completion of the objectives of the proposed project that allowed it to fulfil the aim of the project. The limitations that impeded the proposed project will be discussed in this section. The future expansion of the project will be discussed and explained in the context of the limitations faced.

1.10 Summary

Chapter 2

Literature Study

The aim of this proposed project is to develop a method to detect images created by a style-based generator architecture for generative adversarial networks in a set of artificial images (generated images of human faces contained in the StyleGAN dataset) and authentic images (Flickr-Faces-HQ dataset of human faces used as a benchmark for StyleGAN). Before the aim of the project can be satisfied, a literature study is required on neural networks and StyleGAN. The basic structure of neural networks and the different types of neural networks will be researched and discussed in context with the StyleGAN technology, and the detection of StyleGAN generated images. Current detection methods will be investigated and examined to evaluate their workings and relevancy towards a StyleGAN application. In conclusion, all knowledge acquired from the literature study will be used when creating the artefact to detect StyleGAN generated images.

2.1 Neural Networks

In recent times, neural networks have been regarded as a fast-growing field offering powerful tools for most types of problem-solving (Albawi et al., 2017). The increased use results from the neural network's capabilities to function even with large data sets as input effectively. Therefore, a study into neural networks is required as it is a tool to use in the detection of StyleGAN generated images because of the large datasets accompanying StyleGAN.

2.1.1 History of Neural Networks

Artificial intelligence is a modern growing field within information technology rooted in historical discoveries leading to new advances. Artificial intelligence has been in the development stages since the mid-20th century. However, early on, most advances made in artificial intelligence were developed in mathematics and computational model theory (Müller et al., 1995).

Warren McCulloch and Walter Pitts initialised the now big field of artificial intelligence when they proposed a new general theory in information processing that artificial neurons can mimic the neurons present in the human brain (Müller et al., 1995). Müller et al. (1995) also notes that the neurons Warren McCulloch and Walter Pitts proposed were more simplified than biological neurons and still promised reliable computational power. The artificial neuron that could be implemented in a network to mimic a single neuron cell in the human brain and mimic the whole brain as a network is the foundation of early artificial intelligence and machine learning theory.

The next significant advance in this field came in the 1960s by researchers Caianiello and Rosenblatt. This advance resulted from focusing on two aspects of Artificial Neural Network's (ANN): 1st being the aim to mimic their biological counterparts in their design and the 2nd is that different structures proposed different advantages (Müller et al., 1995). This difference of structure leading to a difference of perception is present in nature, where mammals all possess a biological brain, but the shape and network design allow mammals to think differently based on their specific computational needs(Müller et al., 1995). Rosenblatt coined the differing structure of ANN's as the perceptron of the network, and in modern artificial intelligence theory, the perceptron is called the perception of the neural network.

The historical, theoretical initialization of neural networks is the foundation of modern neural networks. Modern neural networks' development focuses on practical applications in modern times. Neural networks are still relatively new in artificial intelligence, with theory founded in historical mathematics

2.1.2 The Function of Neural Networks

Neural networks are present in most forms of biological computation. For example, the human brain is a neural network of neurons that allow us to compute our daily tasks. An ANN mimics the human brain in its structure (Krenker et al., 2011). Like a human brain thinking through the impulses sent and received between biological neurons, an ANN sends and receives input and output through the artificial neurons in the network (Krenker et al., 2011). The neurons that form the network allow neural networks to imitate a biological brain's basic structure and computation.

An artificial neuron is a single node within a neural network. This neuron is an independent node residing in a neural network that receives data and applies the neuron's mathematical model to this input. The output is the result of a calculation and the operation of activation and deactivation of the node.

Krenker et al. (2011) states that the neuron structure consists of three separate stages. The first stage applies the weight to the input with multiplication. The second stage is a sum function of all the previous stage weights and biases applied to the initial input. The final stage of the artificial neuron determines if the neuron should be activated or deactivated (Krenker et al., 2011). Artificial neurons activated or deactivated artificially enable the neural network to "think" and adaptively apply its computation on inputs.

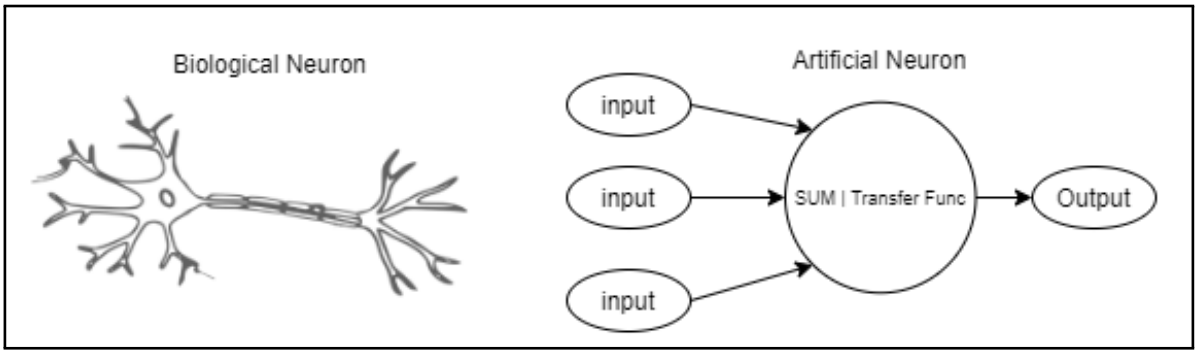


Figure 2.1: Biological Neuron and Artificial Neuron Krenker et al. (2011)

Figure 2 shows the similarities between the biological neural network and the ANN. The basis of biological neural network for the design foundation of an ANN is

The set of neurons functioning together to form a network is called a neural network. In a biological brain, the whole cellular neurons connected in the brain and artificial intelligence are the ANN with the artificial neurons working together. ANN's are capable of processing information of real-world problems that requires a more complex approach because they can distribute their neurons in a non-linear, parallel structure (Krenker et al., 2011). Multiple neurons simultaneously are the neural network structure, and neurons do not have to flow their output into the next.

Figure 3 illustrates a primary ANN with connected neurons that mimics the structure of connected biological neurons connected inside the human brain. Each neuron within the structure of the ANN in Figure 3 can be compared to a human single brain cell in the biological brain structure. The artificial neuron present in the network only apply basic computational calculation on the data, and as output, the node only activated or deactivated itself.

2.1.3 Neural Network Learning Paradigms

Neural networks can learn in different ways, and these differences pose their respective advantages and disadvantages—the manner of learning in a neural network the network's learning paradigm. The three different learning paradigms for neural networks used in the training of neural networks are supervised learning, unsupervised learning, and reinforcement learning (Krenker et al., 2011). The learning paradigms for neural networks are also specific to the type of data the network use in training. Concerning image classification and computer vision, only two of the three learning paradigms apply, namely supervised learning and unsupervised learning (O'Shea and Nash, 2015). Different neural network topologies use different learning paradigms. The specific choice of learning paradigm is essential in the implementation of the detection method of StyleGAN images.

Supervised Learning

The supervised learning paradigm sets the parameters of the neural network based on the training data set it receives. The critical difference in supervised learning is that the input data is labelled before training the neural network (Krenker et al., 2011; O'Shea and Nash, 2015). The already labelled input is then compared to the output of the neural network to determine its error and accuracy. Training is applied to the network based on comparing the output the network created against the labelled input. If this paradigm is applied to the StyleGAN image detection problem, the training set of data consist of a set of images of human faces and a set of StyleGAN generated images. This input training set requires parameters or labels that identify a single image as either a human face or a StyleGAN generated image. The prediction made by the neural network will then be validated against these know classifiers on the images, and training will take place based on how the neural network predictions compare to the valid identifiers. In deep neural networks, supervised learning also deals explicitly with the labelled data. The advantages of supervised learning applied in deep neural networks such as CNN and

GAN are that training can be conducted without initial knowledge about the differences between two different data inputs. The fallback to this advantageous method is that when there is outlying data present within the dataset, An image of a dog in the set of authentic images, the training can overstrain the boundary of decision (Alzubaidi et al., 2021).

The supervised learning paradigm is subdivided into two separate fields determined by the specialisation required while the neural network learns in the training phase. Semi-supervised learning and self-supervised learning selection are determined by how the input data is labelled to provide the neural network with information (Zhai et al., 2019). For a supervised learning approach, when aiming to identify StyleGAN generated images, the subsections of supervised learning must be evaluated in context to this project.

Semi-supervised learning is a good choice of training algorithm if the input data is both labelled and unlabelled. In most cases, the learning algorithm assumes the label of input data if the data originated from the same distribution (Zhai et al., 2019). In the case of analysing StyleGAN images, the 2 labelled data sets are authentic images and generated images. When selecting a semi-supervised learning algorithm, the initial labels in the dataset must be standardised, and the data set is then altered that only a portion of the labels is kept with the data set. The algorithm treats the rest of the dataset as unlabelled data (Zhai et al., 2019). Semi-supervised learning was based initially on different neural network architectures, and one prominent algorithm used was the GAN that is also the basis of StyleGAN. An advantage of semi-supervised learning is that it applies inductive learning through generalisation, mapping the inputs to the outputs classify the data that have the most significant impact on the output of the network.

Self-supervised learning uses only the unlabelled data to formulate mundane tasks within the network. It is commonly used to label datasets that have no classification of data (Zhai et al., 2019). Self-supervised learning algorithms can be implemented on the StyleGAN identification utilizing it labelling the dataset. The input dataset can then be a combination of StyleGAN generated images and authentic images that the self-supervised learning algorithm will then aim to solve by labelling the data as either generated by StyleGAN or is a unique actual human image. One problem with implementing this learning algorithm is that the data set containing StyleGAN images and authentic human faces are very similar. Differences are minimal between the two images, and Self-supervised learning might struggle to distinguish between the two different images, and miss labelling might occur.

Unsupervised Learning

The difference with the unsupervised learning paradigm compared with supervised learning is that with unsupervised learning, the data does not have the added information that can aid the neural network in determining its correctness of prediction. The unsupervised learning paradigm entails learning based on a set cost function and minimising the goal's cost (Krenker et al., 2011). With a StyleGAN application, this paradigm will not necessarily help the neural network to learn effectively. This inefficiency results from an image's initial problem, either an actual image or a fake image. And a cost function to how fake an image might not necessarily lead to the neural network deciding that the image was generated using StyleGAN. This particular case is substantiated by the artefacts embedded in StyleGAN images. In specific cases, the neural network can receive a StyleGAN generated image close to an actual image with only one unique artefact created by the StyleGAN inefficiency. The cost function might still pass that image as an actual image, yet a person will quickly identify the artefact in the image with ease. More on StyleGAN artefacts will follow further in the StyleGAN analysis.

The different learning paradigms discussed each poses their own set of advantages and disadvantages. In the context of identifying StyleGAN generated images, supervised learning could be the chosen learning paradigm, more specifically semi-supervised learning, because of the two labels present in images used to train the neural network. On the other hand, self-supervised learning will not be used because of the minor differences between a StyleGAN image and an actual human face image.

2.1.4 Hot and Cold Learning

Hot and Cold learning is one of the most straightforward approaches to determining the optimal weights for machine learning problems. In a StyleGAN problem hot and cold learning will be randomly guessing the initial hyper parameters and changing them in training to increase the accuracy of the neural networks prediction.

Hot and cold learning is the process of increasing and decreasing the weights after a prediction and then training the model again. In theory, the continuous repetition of this process will lead to an error value of 0. Based on the increased or decreased error value, the changes in weights should either increase or decrease. (Trask, 2019) However, hot and cold learning is not efficient. A developer must repeat a process

manually numerous times until they finally stumble on a perceived perfect combination of weights. This implementation also does not ensure that optimal values are found. A developer might start by changing a parameter to its "*optimal values*" and then changing another to the previous and ultimately closing in on a false optimal set of parameters.

Hot and cold learning is a simple form of machine learning that is not optimal and might not get the best values for training a model. Hot and Cold learning falls behind because its implementation may lead to false positives where changes in the hyper parameters leads to reduced prediction values and show that the data is optimal yet there may be an even better set of parameters. However, it is useful when implemented on minor scope problems and in the initialization in developing a neural network where hyper parameter optimization can improve the mode further. Hot and cold learning will aid in the understanding of hyperparameters and how it connects to the machine learning model and improvements in training.

2.1.5 Neural Network Architecture

The structure in which the artificial neurons are presented within a neuron network is the neural network architecture. There are multiple different neural network architectures with different benefits and disadvantages in specific practical applications. StyleGAN is a generative adversarial neural network that applies specific styles to create a new unique image (Karras et al., 2019). A technique of identifying neural network generated images such as those generated from ProGAN, StarGAN and Deepfakes focused on the shared base convolutional neural network architecture of these technologies (Wang et al., 2020). Therefore GANs and convolutional neural networks are identified as relevant architectures that require deep analysis in this study to ensure a thorough understanding of these architectures that will be interacted with in detecting StyleGAN images.

Convolutional Neural Network

Neural networks perform exceptional at identifying patterns hidden in different large datasets, but specific neural network architectures perform better than others when implemented to detect these patterns within specific data set consisting of different data types (Liu et al., 2017). Convolutional neural networks (abbreviated as CNN) commonly used for computer vision are great neural network architecture options for identifying patterns in image datasets (Albawi et al., 2017; Yosinski et al., 2015). When trying to detect StyleGAN generated images, a large dataset containing images generated

through StyleGAN and a large dataset containing images of humans will be used to train the neural network. A CNN could be an appropriate neural network architecture to use in the detection method because of the specific large image dataset required to solve this problem.

CNN's are similar to the more basic ANN's, with the only difference being the advances a CNN has towards image classifications and pattern recognition within computer vision (O'Shea and Nash, 2015). In a CNN, the primary neuron within the network improves throughout learning, and the network still takes a single weight and apply it throughout. CNN's consists of layers that interact with the raw image data and declassify it into raster data where subsequent layers preside over the fragmented raster's consecutively. CNN's are chosen for image application because ANN's struggle with the computation power needed when attempting image classification (O'Shea and Nash, 2015).

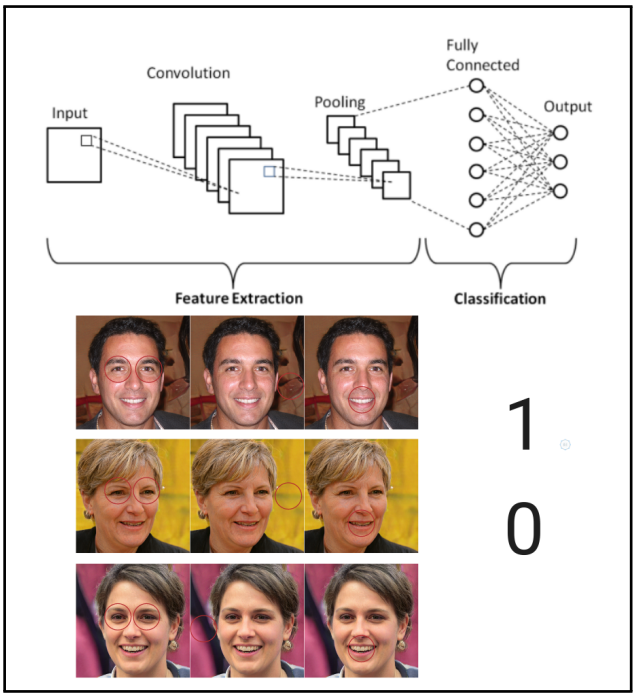


Figure 2.2: Possible Architecture of CNN applied to the StyleGAN problem (Karras et al., 2019; O'Shea and Nash, 2015)

The network architecture of a CNN is divided into three dimensions categorised as the convolutional layer, pooling layer and fully-connected layer (O'Shea and Nash, 2015). The dimensionality created by the stacked layers sets CNN's apart from ANN's in image classification. The first layer in a CNN distinguishes it from the standard ANN and produces improved performance when it is implemented for image classification.

When the input is passed through this layer, the convolution applies various filters on the data to activate two-dimensional maps (O'Shea and Nash, 2015). These 2D maps determine if features are present in the image based on pixels activated when filtered on the activation maps. The convolutions within this network can get large in dimensionality, and the pooling layer is responsible for reducing the complexity of the calculated data (O'Shea and Nash, 2015). The function of the pooling layer to reduce the dimensionality of the network can be detrimental to the data because any dimensional reduction in the architecture of the NN further reduces the data dimension simultaneously. Finally, neurons gather the data directly from nodes in the previous layer and, without connecting the preceding layers, conclude the neural network (O'Shea and Nash, 2015).

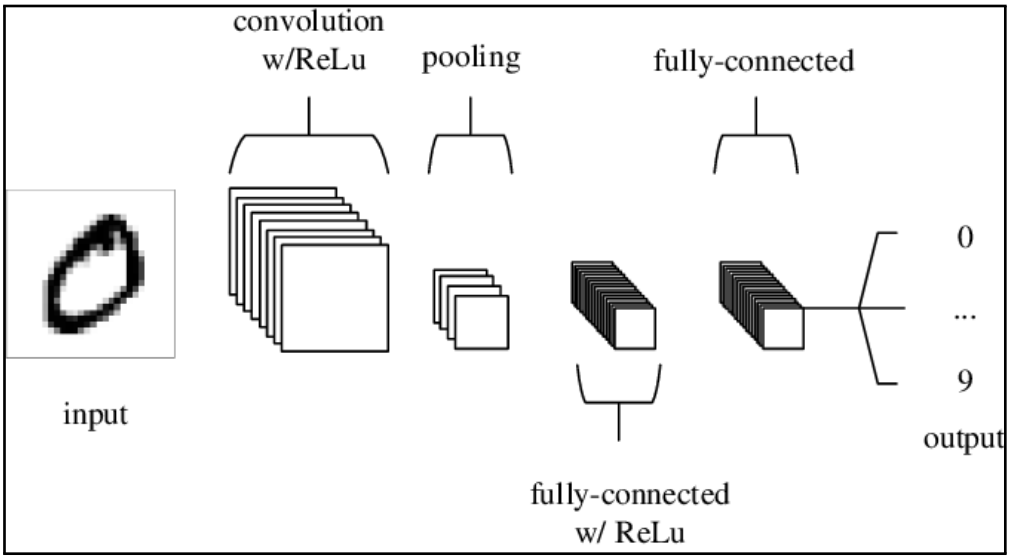


Figure 2.3: CNN used in the classification of digits O'Shea and Nash (2015)

Figure 2.3 visualizes the hidden convolution layer, pooling layer, and the fully connected layers within a CNN. Various iterations of the layers pose different advantages and disadvantages in a practical application.

The benefit when using a CNN on an image processing problem is the improved accuracy in detecting objects within the image (Albawi et al., 2017; Alzubaidi et al., 2021). The activation maps present in a CNN enables the neural network to robustly identify objects within an image through the localisation of the raster data. CNN share weight between its parameter, and this reduces the number of required nodes in training. The reduction in training nodes improves CNN generalisation and reduces overfitting created by the training process. The localisation and object detection of CNN requires extensive calculations, and the process is computationally expensive. This drawback means that without the necessary graphical processing hardware, the training of the CNN will take a long time compared to most other neural networks (Alzubaidi et al., 2021)

Generative Adverserial Networks

Competition increases the performance of sports athletes, students, and businesses (Burguillo, 2010; Hays et al., 2009; Medvedev and Zemplinerová, 2005). Neural networks can train themselves with the appropriate datasets as identified earlier. With the analogy that a neural network aims to mimic the human brain in its structure, the assumption can be made that competition might increase a neural network's performance. This assumption that competitive neural networks competing against one another led to the formulation of GANs (Creswell et al., 2018).

GAN's use a discriminator and generator to produce two neural networks competing against one another (Creswell et al., 2018). The result of pitting two networks to compete against each other leads to the network being able to improve itself more effectively and further increase application possibilities with these types of networks (Goodfellow et al., 2014). The generator constantly tries to fool the discriminator and the discriminator, in turn, try to identify when its input originated from the generator. The generator can be characterised as a criminal trying to falsify identification documents and the discriminator as a customs officer checking a passport. The criminal constantly tries to fool the customs officer. When he succeeds, the officer remembers the fake passport he missed in his identification of fake passports, and the process starts again. The officer learned from his mistake and can in future detect fake passports better. When the criminal fails, the process restarts and similar to the officer the criminal also learns from his mistakes and adapts how he generates fake documents (Goodfellow et al., 2014). The generator constantly creates, and the discriminator constantly describes. This cat and mouse game of fooling and detection in training is how a GAN evolves into the robust networks present in the field of fake images created by neural networks.

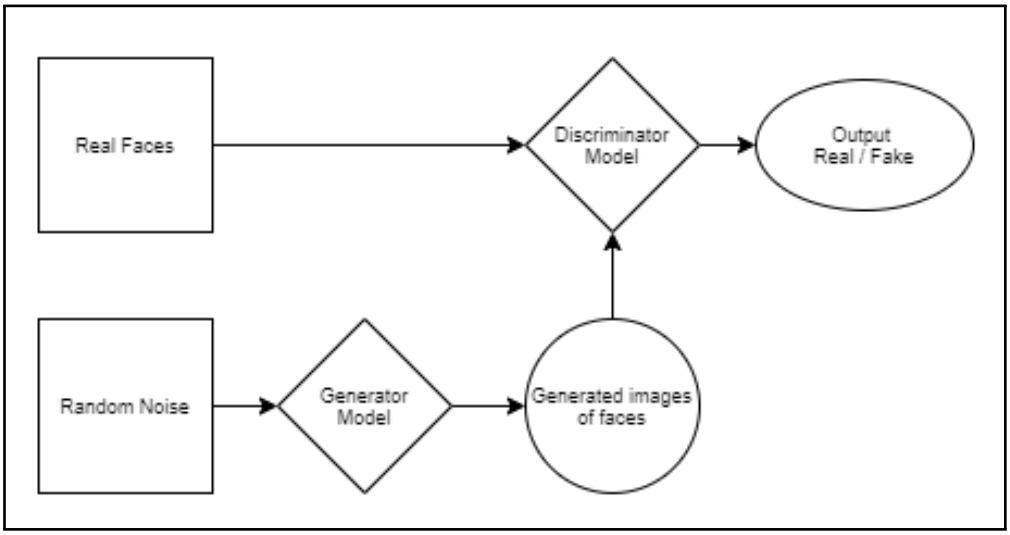


Figure 2.4: GAN architecture adapted from Creswell et al. (2018)

A demonstration of how the structure of a GAN allow the discriminator to be critical of the generator output is evident in Figure 2.4. The advantage brought on by this neural network architecture is the improved data processing performance. The GAN can process images with sharper resolution than the standard ANN that require images with reduced sharpness for model integrations (Goodfellow et al., 2014). The main disadvantage of implementing GAN is increasing complexity to keep the generator and discriminator synchronised while training the models.

StyleGAN and StyleGAN2 are GANs with an improved generator that applies specific "styles" on images, and the combination leads to the generation of new images (Karras et al., 2019, 2020). A convolutional GAN is a GAN implemented where the generator creates images with convolutional neural network architecture. This combination of the GAN checking against itself while creating images with the architecture of CNN enables improvisation in image generation (Karras et al., 2019; Wang et al., 2020).

2.1.6 Activation Functions in Neural Networks

Activation functions is a crucial part of neural networks and their processing capabilities. Without the presence of activation functions in the nodes of a neural network, the learning of the network will be reduced and the maps between the input and the output will not reach the required complexity. In this literature study an analysis on activation functions, why they are necessary and the different types of activation functions available must be conducted.

Activation functions are crucial for neural network learning

Activation functions are the processes on the nodes within the layers of neural networks that allow information to be derived from the data in specific ways. An activation function determines in what way the node will set itself on/off and allow its weight to influence the output of the network. Activation functions are used in an ANN to transform the input signal on a node to the output signal and sequentially add the output of the previous layer to the input of the next layer Sharma et al. (2017).

The inputs and weights are calculated first and then before the output is sent to the next layer the activation function is applied on the node (Sharma et al., 2017). Accuracy within neural networks can fluctuate greatly and is influenced by the number of layers within the network and the types of activation functions used. The types of activation functions within the neural network however has a more significant influence on the accuracy of the

prediction of the neural network Sharma et al. (2017). There is no clear way to determine the best number of layers a neural network architecture must consist of, but there is a clear consensus between data scientists that a minimum of two layers must be used Sharma et al. (2017).

In neural networks, there are different types of activation functions but the most common set of these functions is the non-linear activation functions (Sharma et al., 2017). In neural network activation functions, there are boundaries present and these boundaries describe the type of activation function a specific approach consist of, in a linear activation function this boundary is linear (Sharma et al., 2017). Because of these linear boundaries in linear activation functions, the neural network will only be able to change its perception of the data in linear increments. The problem however faced with these activation functions is that real-life scenarios and problems the errors present consist of non-linear characteristics (Sharma et al., 2017). Therefore data scientists opt for non-linear activation functions over linear activation functions in their functional neural network implementations.

In the development of the artefact the use of non-linear neural networks will ensure that complex information will be processed from the initial dataset. By adding non-linear activation functions to the neural network the output and steps toward the output will be non-linear in the result.

The most important aspect of using activation functions is that the functions are differentiable so that backpropagation optimization can be implemented. When backpropagation can be implemented with the use of gradient descent will the neural network have the capability of calculating the errors and losses based on the calculated weights its uses within its layers (Sharma et al., 2017).

Different types of activation functions

Different types of activation functions can be used and implemented on the layers within the neural network. The type of activation function that can be used depends on the data set and properties present in the data, and the output required from the data. As an example for binary image classification the layers in the network will have to consist of ReLU activation layers, yet the final output layer must be a Sigmoid activation layer.

Table 2.1: Different Activation Functions in Neural Networks (Sharma et al., 2017)

Linear	Sigmoid	Tanh	ReLU	SoftMax	ExpoLU

In the review on activation functions it is apparent that activation functions is a crucial aspect of neural networks and the successful output that neural network can present. Activation functions have a bigger impact on the prediction accuracy of a neural network than the number of layers present within the neural network. There is still some guessing involved into which functions will result in the best accuracy predictions of the neural network but to some degree, there is a clear consensus as to what function will work best with specific types of data. Image classification is improved with the use of ReLU functions throughout the neural network layers and a Sigmoid activation functions as it finals output layer.

2.1.7 Summary

With the evaluation of CNN and GAN, an understanding of neural networks in the context of the aim of this project was conducted. CNN provides enhanced capabilities in the generation and classification of image data. While GAN provides specific alterations to images and specialized focus on own output validation and improvement. For the neural network training on a dataset containing authentic images and StyleGAN generated images, supervised learning is the appropriate choice. The architecture and learning paradigms choices for neural networks should be influenced by the type of application and datasets used within training, and because of this, a CNN neural network with semi-supervised learning will be used in the development of the project's artefact.

2.2 StyleGAN

StyleGAN brought new developments in applying styles on images and changing these styles to morph images into new unique style based changes (Karras et al., 2019). Fraudsters are empowered to better create fraudulent identities by applying these styles to images of faces. A fraudster can use StyleGAN to apply styles on their faces to change how they are perceived at checkpoints and more. StyleGAN still introduced advantages to the field; however, the detection of these images is necessary in today's media-centric world. The technology of StyleGAN generated images must first be understood to create a successful implementation technique to detect the generated images.

The StyleGAN addition to the vast sets of different neural networks implemented on different problems originated sequentially with the advances made in the fields of convolutional neural networks and GANs. Because of the computational advances of CNN discussed earlier StyleGAN improved in its generation of new images. StyleGAN2 further improved on this by reducing artefacts brought on in combining styles using the GAN from StyleGAN (Karras et al., 2020).

2.2.1 StyleGAN Architecture

The structure of StyleGAN differentiates it from normal GANs and allows for the combination of "styles" to create new images of non-existing humans (Karras et al., 2019). StyleGAN initializes with an insertion of the base image and input parameter of what styles need to be applied to that image, such as age, sex, and race. Applying all these parameters in the neural network leads to an image being created that changes the styles in the base image. StyleGAN architecture altered the architecture of the generation model to allow for more control over generating with different styles.

Figure 2.5 illustrates the original StyleGAN network architecture and compares the StyleGAN architecture to the architecture of a traditional GAN. The mapping of the styles added to the architecture of traditional GANs present and a clear indication of how StyleGAN can apply the styles within images can be seen.

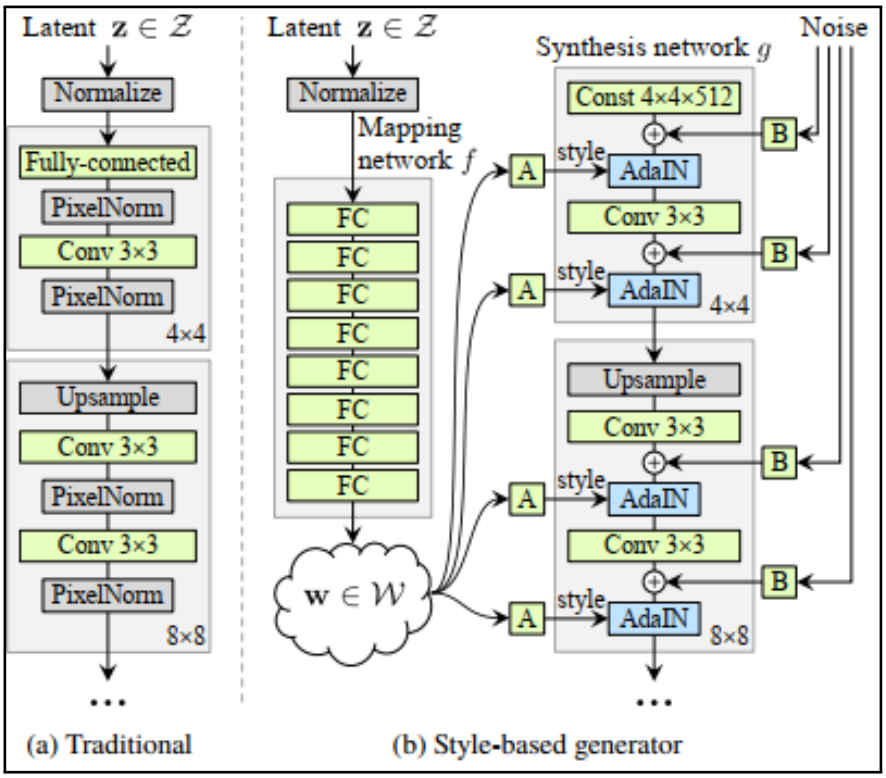


Figure 2.5: Network architecture of StyleGAN vs Traditional GAN's (Karras et al., 2019)

Differences in authentic images and StyleGAN images are minimal. The most noticeable differences are the artefacts that StyleGAN generates due to deficiencies within the StyleGAN technology. These artefacts vary from rain-drop blotches in images to hair strands not showing realistic definition (Karras et al., 2020). In the second iteration of StyleGAN, the artefacts were addressed with minor improvisations but remain present in StyleGAN2 images (Karras et al., 2020).

Figure 2.6: Artefacts present in StyleGAN generated images adapted from Karras et al. (2020)

Figure 2.6 illustrates artefacts present in StyleGAN generated images. This water drop effect present in these images could be more accessible for humans to identify than for a neural network due to the generalization of raster data (Karras et al., 2020).

2.2.2 Detection of CNN Generated Images

To detect StyleGAN generated images, the discriminator of the initial neural network can, in theory, be isolated and used to detect images created by the generator. The generator recurrently aimed to fool the discriminator, and the discriminator constantly adapted to the generator (Karras et al., 2020). The isolation of the discriminator is an invalid approach to detection because of how learning takes place within the creation of the StyleGAN network and when stoppage occurred in the training phase of the initial technology. The discriminator stops training before the generator, and thus an isolated discriminator will not be able to detect the output of the improved generator. In StyleGAN, the discriminator is not available to the public and to isolate it would require retraining of the GAN with the initial datasets of Flickr Faces images. A basic technique of supervised training to detect CNN generated images was implored by the researchers, and their results proved successful in detecting generated CNN images (Wang et al., 2020).

Wang et al. (2020) proved that by employing a neural network to train labelled images as either real or faked, CNN-generated images could be detected. Table 1 results from their application of different image generation or changing neural networks. The success that Wang et al. (2020) achieved demonstrated that a semi-supervised neural network can detect CNN-generated images. Therefore a simple neural network approach is validated, and in the context of detecting StyleGAN images, a similar approach will be taken.

Table 2.2: Results of Wang et al. (2020) detecting various CNN's generating images

CNN-image generator	Detection accuracy of (Wang et al., 2020)
ProGAN	98.8%
StyleGAN	99.6%
BigGAN	66.4%
CycleGAN	88.7%
StarGAN	87.3%
Deepfake	58.1%

The basic architecture and structure of the generator of StyleGAN discussed previously gives more understanding of how StyleGAN can create such precise replication images of human beings. The success of a previous form of detection was evaluated, and StyleGAN images can be detected. By evaluating the work conducted by Wang et al. (2020), a primary neural network with semi-supervised training is identified as a solid foundation for further detecting StyleGAN generated images.

2.3 Summary

Through this literature study, research and developments surrounding StyleGAN, neural networks, and different network architectures and learning paradigms were evaluated and discussed. CNN can create or detect images utilizing the strong computational power provided. GAN can change or initially create derived images, and StyleGAN can implement further control over the changes in the generated images. The types of neural networks involving StyleGAN and other surrounding technologies were evaluated, and an approach was discovered. A simple neural network utilizing semi-supervised learning will train on StyleGAN generated images and authentic images of human faces contained in the Flickr faces dataset will it be possible to detect these generated images as initially discovered by (Wang et al., 2020). The knowledge gained throughout this literature review will be used to develop the artefact to detect StyleGAN images. The development of the final artefact that detects these images will form the third chapter in this project

Chapter 3

Development of the Artefact

The problem of StyleGAN generated faces aiding in the malicious creation of false identities confirmed in the crackdown of Facebook profiles as shown by Chandler et al. (2016) and in the new addition of StyleGAN3 where the creator Nvidea also emphasized the problem and research aiming to detect these images (Karras et al., 2021).

In this projects literature study it was identified that a machine learning deep neural network approach will be the simplest solution to the problem. A neural network that can identify images as either real human images or images generated by StyleGAN with relative accuracy as a method for detection must be implemented in an easy to use front end for the aims of this project to be satisfied. The proposed method of detection should be the easiest solution to the problem as substantiated by Rasmussen and Ghahramani (2001) with their findings concluding that a simple neural network is usually the best neural network.

3.1 Artefact Description

The artefact for this proposed project can be divided into two main aspects namely the neural network model that will train on 2 separate datasets to classify new images between the characteristics of the data classes it learned in trained. The second aspect is the minimal front-end that will allow users to easily interact with the neural network model and receive simple output on the identification of their uploaded images. The neural network was created using the python programming language in a Jupyter notebook that was compiled and executed on the Google Colab virtual cloud-based environment. To create the neural network popular machine learning packages in the

python language was used namely Keras and TensorFlow. The neural network was evaluated and determined to be sufficient, but improvisations could be made using the new technology Optuna for hyperparameter optimization. The neural network was then implemented in a minimalistic front end web app using the python web framework Flask.

3.2 Artefact Life Cycle

The development of the artefact was conducted with the use of DSRM and an Agile combination as stated in Chapter 1. The DSRM

3.3 Description of the Development of the Artefact

As stated in the methodologies used in Chapter 1 and in the Artefact Life Cycle the development of the artefact in this project was subdivided into Sprints. Each sprint handled a subset of tasks to allow for manageable section of development. This partitioning of the overall workload aided in the development of the artefact and the subsequent testing of the implementations to ultimately ensure a successful project compared to the initial project aims and objectives. Therefore the overall artefact development will be discussed in terms of these sprints. The software used and implemented as well as the final implementation of the method of detection will be structured out in these sprints.

3.4 Sprints

In the development of the artefact the overall workload was subdivided into 4 separate sprints. In the 1^{St} Sprint the datasets where retrieved and compiled and an initial neural network was created. The 2^{Nd} Sprint entailed training the neural network and evaluating the accuracy of the model in detecting StyleGAN generated images. The 3^{Rd} Sprint used the results of the evaluation in the 2^{Nd} Sprint and it was determined that improvements could be made. Optuna was used to improve the neural network model. The 4^{Th} and final sprint completed the method of detection by implementing the neural network into a minimalistic front-end web application using the Flask framework.

3.4.1 Sprint 1

In the 1St Sprint of the development of the artefact the StyleGAN dataset was downloaded from the official StyleGAN GitHub repository where various datasets are included of StyleGAN generated images. For the images of real human faces the FlickrFaces dataset was retrieved from the FlickrFaces GitHub repository. The original datasets contained 100 000 images each and for the detection of StyleGAN images a subset was created of these datasets that reduced them in size by number of images and included them together in a dataset that can be used in the training of the neural network model (Karras et al., 2019).

Downloading and the Dataset

The datasets that is provided by StyleGAN and the FlickrFaces dataset each contain 100 000 high-quality images. The problem however that was faced is that these high-quality image datasets was also very large. Conventionally datasets used and sometimes contained in software such as Google Colab and Kaggle is small in total size. The StyleGAN high quality generated faces image dataset size was a total of 175GB and the FlickrFaces dataset total size was 140GB.

The two datasets was made available on their respective GitHub repositories, redirecting to the file buckets hosted in Google Drive. To access the data locally the drives was shared to an NWU institution Google Drive account which does not have a storage limit. In Google Drive shortcuts to the original datasets could be added to the home directory of the account. the datasets could then be downloaded from the home directory.

The problem however faced when using this method is the internet limitations posed on campus at the North-West University's Potchefstroom campus. When downloading through a browser on using on-campus internet infrastructure the connection will be throttled if a single file larger than 1GB is downloaded. Google Drive automatically compresses downloads into zip files when downloading and individual files cannot be downloaded sequentially using the internet browsers. Speeds up until 1GB downloaded would stay constant and at high-capacity then after reaching 1GB would be throttled down to low inconsistent speeds. A better way to retrieve the datasets locally had to be used.

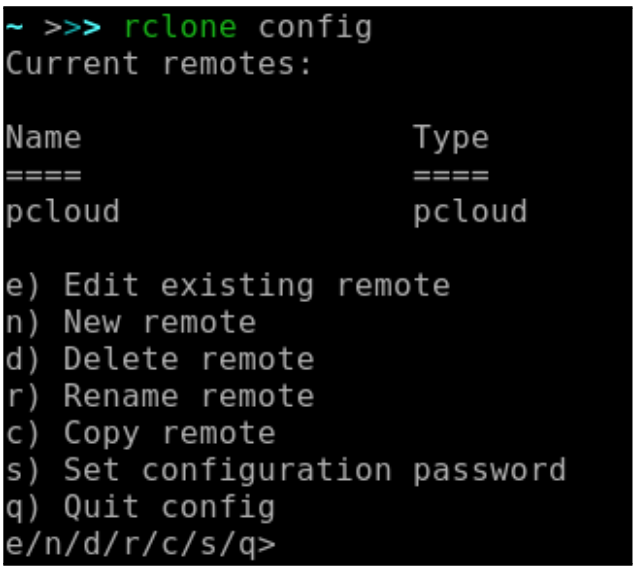


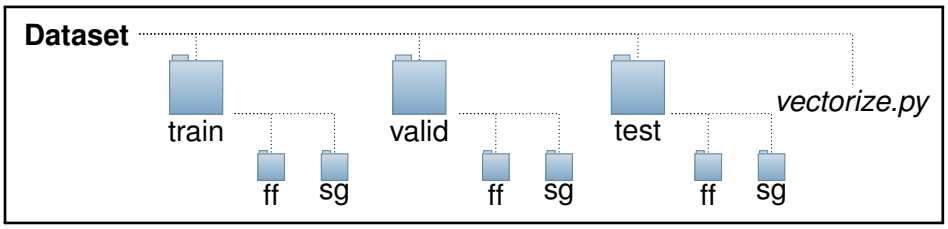
Figure 3.1: rclone setup config example (Craig-Wood, 2021)

To address the problem of internet speeds being throttled when downloading the datasets the command line software rclone was used. rclone, a Linux native package, was installed on the windows local system using the dedicated windows installer. As demonstrated in Figure 3.1 rclone could be opened up in the command line and after mounting a connection to the Google Drive account could the whole dataset be downloaded sequentially. Each file is downloaded individually from the Google Drive dataset to the local machine, and the total download times of both datasets completed overnight.

Dataset Preparation

The datasets had to be complied together into one dataset that was used within training the neural network. When training a neural network an binary image classification problem the dataset directory structure is a important aspect of the data. When the neural network trains and validates its training it compares its prediction with the original image file directory in both training and validation. If the network predicts a image to be StyleGAN generated it checks it's prediction against the folder in which the file is located. The sub directories for the datasets created in this project followed a similar structure to that of the Cat-vs-Dog problem that is used when learning about binary image classifications in neural networks. Table 3.1 shows the structure of the dataset, this directory structure remained the same for all subsequent datasets as they where created to address the resources problems that was identified in Sprint 2 and for the hyperparameter optimization trails in Sprint 3.

Table 3.1: Folder Structure of the dataset used in identifying StyleGAN images



Initially the subset created from the StyleGAN and FlickrFaces datasets retrieved consisted of 20 000 images. The number of images used for the identification of StyleGAN images was decided based on the findings of Nasr-Esfahani et al. (2016), that concluded the more images a neural network can use in image classification the better its prediction will be. To an extend this is true for the problem on StyleGAN generated images versus images of real human faces due to the fine differences between the two types of images. The neural network won't classify the images based on the shape as in the Cat-vs-Dog problem but will rather focus on the small artefacts or features that are present in StyleGAN images for its classification.

Using too large a dataset can lead to the neural network overfitting the data (Trask, 2019). Overfitting occurs when the created neural network model becomes exceptionally good at being classifying the dataset it trained on, but performs much worse when classifying data that it was not trained on. Because of the scope of the proposed project the neural network that is created does not have to be the "*perfect*" model, and will in a sense overfit facial recognition features as that is the premise for the creation of the model. What this means is that if the model created gets an input image of a car and classifies it as a real human or a StyleGAN generated image, this form of overfitting is acceptable in the context of this projects scope. Figure 3.2 illustrates the acceptable amount of overfitting that can be expected from the neural network. The network however must not overfit on the images included in the training set of real human faces retrieved from the FlickrFaces dataset, as then it will classify all images as StyleGAN if it was not in the FlickrFaces training set. This issue is addressed in the second sprint when the initial neural network is created.

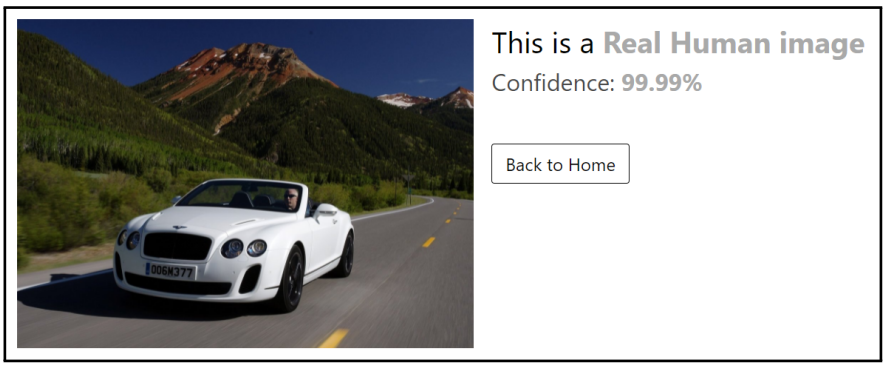


Figure 3.2: Acceptable level of Overfitting for the StyleGAN identification model

A subset of these 2 datasets will improve the development of the artefact as working with a large dataset while testing and implementing hot and cold learning will unnecessarily increase the development time. Training times is influenced by the number of epoch in training and the number of samples used. When developing neural network works it is standard to use a small subset of the training set to keep the times in testing lower, and after the development of the neural network is completed move over to the full dataset. The Full dataset that will be used in the final training and optimization of the neural network will be larger than the small dataset used in development to ensure that the final model can identify the features present in StyleGAN generated images. Table 3.2 states the sizes of the datasets and the sub directories that will be used in development and in the final optimization of the neural network model.

Table 3.2: Total amount of images used in the Artefact creation

Directory	Development		Training		Optimization	
	sg	ff	sg	ff	sg	ff
Train	1000	1000	2000	2000	4000	4000
Validation	500	500	1000	1000	2000	2000
Test	500	500	1000	1000	2000	2000
Total Dataset Size		4000		8000		16000

Table 3.2 and Table 3.1 shows the layout of the directories that must be consistent throughout the training of the neural network model in the artefact creation. In binary image classification, the process of training a neural network to classify two sets of images into two separate classes, the neural network references the directory in which an image resides for its check on its prediction. This means that in the training stages the neural network will makes its prediction on a image, and check if that image is in the

real human directory (ff) or in the StyleGAN generated images directory (sg) and then change its weights according to the correlation between its prediction and the directory path of the image receives. In short the neural network will use the directories to "*know*" what type of image the received training image is.

The validation folder in the directory is the set of images the neural network uses to check its accuracy after each training step, while training its weights. The neural network changes its weights after each image in the training set and just uses the validation set as a benchmark for each checkpoint in training. The Test set is important as it will be used to evaluate the final model. The Test set must be kept aside and never accessed in the training phase of the model. Thus a Test set was created early with unique images the will not be accessed again in the training phases.

Creating the First Neural Network

For the creation of the Artefact it was identified that the problem of identifying StyleGAN generated images required binary image classification. To understand how binary image classification could be applied to a StyleGAN problem the Cat-vs-Dog example was used to understand binary image classification. The Cat-vs-Dog problem can be seen as a Hello World exercise that will teach the base fundamentals of binary image classification and the knowledge gained in this problem will be applied to the StyleGAN problem (Albaradei et al., 2014). The Cat-vs-Dog problem is a basic exercise to create a neural network to classify images as either a cat or a dog, and the dataset used in this exercise contains 4000 images of both cats and dogs with image sizes of 200px by 200px.

The StyleGAN problem was applied to this exercise by using the StyleGAN dataset instead of Cat-and-Dog images. A problem was identified with this implementation as StyleGAN images are very large compared to other datasets used in CNN neural network implementations, namely the MINST handwriting dataset and the Cat-vs-Dog dataset. StyleGAN images and FlickrFaces images native resolution is 1024px by 1024px, which could not be passed to a neural network catered for the classification of Cat-vs-Dog images as the input layer of the neural network must match the image resolution in CNN's (Albaradei et al., 2014; Wang et al., 2020). Figure 3.3 shows the input layers and image sizes in identifying StyleGAN images and the Cat-vs-Dog exercise.

```
input_layer = layers.Input(shape=(200, 200, 3))
input_layer = layers.Input(shape=(512, 512, 3))
input_layer = layers.Input(shape=(1024, 1024, 3))
```

Figure 3.3: Input layer of the neural network according to the image sizes

In Figure 3.4 a comparison between a dog image retrieved from the Cat-vs-Dog dataset and the dataset used for the identification of StyleGAN images in this project with the original images sizes (1024x1024px) the scaled down (200x200px) images to fit into the input layer of the Cat-vs-Dog problem.

Figure 3.4: Image sizes of Cat-vs-Dog dataset, rescaled artefact dataset and StyleGAN original sizes

When the StyleGAN images was passed into the Cat-vs-Dog neural network with a change in the input layer of the network to accommodate full resolution StyleGAN images a usage limit was reached on Google Colab. This identified that some image processing will be required on the dataset. When the images was scaled down to

200x200px and the Cat-vs-Dog neural network trained on StyleGAN images an accuracy of 61% was achieved. The accuracy achieved with this basic implementation was used as a proof of concept and showed that it was possible to identify StyleGAN images with a neural network. The accuracy of the network however was not ideal and thus a network specific to the StyleGAN problem had to be created.

Image Processing

As mentioned in the previous section, the dataset of images gathered for the identification of StyleGAN images had to be processed to allow the neural network to train on these images. Image processing is an entire field on its own with theory relating to it. For the image processing needs required in this stage of the artefact development the python library OpenCV was identified to be the simplest solution to the problem being faces.

OpenCV is a programming library geared mostly at real-time computer vision. It was created by Intel and then supported by Willow Garage and Itseez. Under the open-source Apache 2 License, the library is cross-platform and free to use (Culjak et al., 2012). A python script was created to change the entire dataset sequentially and the need to manually scale the image was avoided. The script was used to create different datasets for later use in this proposed project. The image resolutions of the different datasets created included a 200x200 pixels, 512x512px and 1024x1024px.

Summary

The first sprint in the development of the artefact required the dataset to be retrieved, structured and the images processed. The dataset was downloaded using the rclone program to avoid the throttling limitation experienced. The layout of the directories in the dataset that was compiled for the identification of StyleGAN images was structured to enable the dataset to be used in training. The Cat-vs-Dog exercise proved that a StyleGAN identification neural network model could be created and showed that images had to be processed for a neural network to be trained on the limited resources available. For the processing of the images a python script was created using the open-source image processing library OpenCV.

3.4.2 Sprint 2

In the second sprint of the artefact development the first neural network was created based on the findings in the first sprint and using that dataset gathered and processed in the first sprint. The initial neural network that was created in this phase of the artefact development just required hot and cold learning and was not optimized in any form. Hot and Cold learning as analysed in Chapter 1 is an uninformed guessing game in hyperparemeter optimization.

Creating the First Neural Network

To create the neural network the TensorFlow package was used. TensorFlow is a machine learning and artificial intelligence software library that is free and open-source. It may be used for a variety of applications, but it focuses on deep neural network training and inference (Abadi et al., 2016). TensorFlow was used within the Jupyter Notebooks in Google Colab and locally, and was run in the python environment.

Keras is an open-source software library for artificial neural networks that includes a Python interface (Ang et al., 2017). Keras serves as a user interface for TensorFlow. Keras supports the TensorFlow packadge and is used for creating the neural network in the development of the artefact (Ang et al., 2017).

The initial neural network was then created using TensorFlow and Keras in Google Colab. The Code for the notebooks used can be found in Appendix C and D. As seen in Figure 3.5 the neural network consisted of 3 convolutional layer, a single dense layer and had a total amount of trainable parameters exceeding 17 million. The large amounts of trainable parameters would result in the resources available in Google Colab to be exhausted in training. A hot and cold learning approach was taken and with the smaller test dataset the model was trained and changed until some improvements could be seen. The trained accuracy of this neural network was 51% at this stage but could be improved more.

Layer (type)	Output Shape	Param #
input_3 (InputLayer)	[(None, 200, 200, 3)]	0
conv2d_13 (Conv2D)	(None, 198, 198, 16)	448
max_pooling2d_12 (MaxPooling	(None, 99, 99, 16)	0
conv2d_14 (Conv2D)	(None, 97, 97, 32)	4640
max_pooling2d_13 (MaxPooling	(None, 48, 48, 32)	0
conv2d_15 (Conv2D)	(None, 46, 46, 64)	18496
max_pooling2d_14 (MaxPooling	(None, 23, 23, 64)	0
flatten_1 (Flatten)	(None, 33856)	0
dense_2 (Dense)	(None, 512)	17334784
dense_3 (Dense)	(None, 1)	513

Total params: 17,358,881
Trainable params: 17,358,881
Non-trainable params: 0

Figure 3.5: Summary of Cat-vs-Dog network applied to the StyleGAN problem

More layers and dropout layers where added to the different convolutions using hot and cold learning. Figure 3.6 is a summary of the neural network architecture of a model created for the identification of StyleGAN images using the principles of hot and cold learning applied on the hyperparameters of the model.

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 200, 200, 3)]	0
conv2d_7 (Conv2D)	(None, 198, 198, 16)	448
max_pooling2d_6 (MaxPooling2	(None, 99, 99, 16)	0
dropout_6 (Dropout)	(None, 99, 99, 16)	0
conv2d_8 (Conv2D)	(None, 97, 97, 32)	4640
max_pooling2d_7 (MaxPooling2	(None, 48, 48, 32)	0
dropout_7 (Dropout)	(None, 48, 48, 32)	0
conv2d_9 (Conv2D)	(None, 46, 46, 64)	18496
max_pooling2d_8 (MaxPooling2	(None, 23, 23, 64)	0
dropout_8 (Dropout)	(None, 23, 23, 64)	0
conv2d_10 (Conv2D)	(None, 21, 21, 64)	36928
max_pooling2d_9 (MaxPooling2	(None, 10, 10, 64)	0
dropout_9 (Dropout)	(None, 10, 10, 64)	0
conv2d_11 (Conv2D)	(None, 8, 8, 128)	73856
max_pooling2d_10 (MaxPooling	(None, 4, 4, 128)	0
dropout_10 (Dropout)	(None, 4, 4, 128)	0
conv2d_12 (Conv2D)	(None, 2, 2, 128)	147584
max_pooling2d_11 (MaxPooling	(None, 1, 1, 128)	0
dropout_11 (Dropout)	(None, 1, 1, 128)	0
flatten (Flatten)	(None, 128)	0
dense (Dense)	(None, 200)	25800
dropout_12 (Dropout)	(None, 200)	0
dense_1 (Dense)	(None, 1)	201

Total params: 307,953
Trainable params: 307,953
Non-trainable params: 0

Figure 3.6: Summary of created Neural Network

The final self created neural network trained on the training dataset could identify the StyleGAN generated images with 81% accuracy using the test data. The relatively high true accuracy showed promising result but because of the randomness in hot and cold learning and what was identified in the literature review of Chapter 2 the hyperparameters had to be optimized.

3.4.3 Sprint 3

Hot and Cold learning is a viable option for setting neural network parameters when developing neural networks and learning how they can be used to solve problems (Trask, 2019). But for the identification of StyleGAN images hot and cold learning proved to be an inefficient manner in which to create the neural network architecture. Optuna was defined as a possible technology to omit the hot and cold learning process and create a highly optimized neural network.

Optuna: A hyperparameter optimization framework

Optuna is a software framework for automated hyperparameter optimization that is specifically developed for machine learning. It has an imperative, define-by-run user interface. The code built with Optuna has a high level of flexibility thanks to its define-by-run Interface, and the user of Optuna may dynamically design the search spaces for the hyperparameters. The phrases "study" and "trial" are used as follows: A Trial is a single execution of the objective function, whereas a Study is optimization based on an objective function (Akiba et al., 2019).

When the Optuna hyperparameter process was started in this sprint the trail function and study functions was created based on the documentation of the Optuna framework. The trail is a single iteration in the larger study and the study is a collection of trails where different combinations of hyperparameters are used to reach the goal of the trail. The study goal for identifying StyleGAN images was to increase the accuracy and Akiba et al. (2019) terms it was to move the accuracy metric in a maximum position. The trail function was set to suggest a neural network layer count ranging from a single layer network to a 7 layer network. The optimizer was suggested from a pool of optimizers that perform well with image classification problems as was identified by Bera and Shrivastava (2020) and Kandel et al. (2020). The Study could be changed to also train full neural networks on each trail but it was decided to train the networks the minimal amount just to evaluate the improvements of the models. When the study completed the resulting model was then trained on the full dataset.

When the Optuna study was executed on Google Colab a problem was faced regarding available resources. Google Colab provides a free service but can throttle users resources based on the amount of resources used. The policies that Google use to dictate what will amount in throttling is unclear and Google can throttle any account based on their discretion. When the Optuna study started the account used on Google Colab for the development of the artefact was throttled down and exceeded the acceptable usage dictated by Google. To work around the problem the jupyter notebooks was downloaded and run on a physical computer provided by Prof. Tiny du Toit. The computer used for the study included a GPU, namely a GTX 1080 with 8GB dedicated graphics memory.

Graphics cards differences: comapre colab and nvidea gpu

Image of Optuna output in Study

Train

Image of training

The model created using the Optuna framework for hyperparameter optimization was trained on the full dataset on the local machine as previously stated and training times was increased due to the improved machine learning factor of the improved GPU. The trained hyperparameter optimized model could identify StyleGAN generated images with an accuracy of 97.3% and a true accuracy of 97.6% as tested with the python script. The results of the Optuna network compared to the network created in Sprint 1 will be further discussed on the Results chapter in this document.

3.4.4 Sprint 4

With the neural network model completed and trained to a exceptional accuracy a front-end application was created to allow users to easily and intuitively interact with the model as stipulated as one of the objectives of this project. By allowing users to interact with the model easily the aim of identifying StyleGAN images can be satisfied as users will be able to pass images to the neural network that in turn will be able to identify if these images was generated by StyleGAN or if the images are that of real human beings.

For the front end of the artefact a web application, that will allow for easier deployment and wider reach had to be created to satisfy the objectives of this proposed project. In the final sprint of the artefact development various frameworks where considered and while developing the artefact it was realised that a python-based web framework will enable the implementation of the neural network.

Adding the model to the Web App

Because the neural network model was created using the python libraries for machine learning it was realised that a python based framework will allow the neural network to be implemented in the web app. If another front end framework like React or Angular was used the neural networks implementation would be unnecessarily complex. React and Angular are JavaScript based frameworks and for the Python model to communicate with the front end of a JavaScript framework an API had to be implemented. The neural network requires the python machine learning libraries TensorFlow and Keras to pass an image through the model and provide the identification as feedback. Thus the libraries must reside in a python environment where TensorFlow and Keras is installed.

To avoid this added complexity an analysis into python web based frameworks where conducted and the Django framework stood out. The Django framework however is a very large and clunky framework when compared to the implementation required for the artefact. The child framework Flask, that was derived from Django but with a lightweight footprint was used for the creation of the front-end of the artefact. The Flask app was developed as a minimalistic interface that would guide users intuitively on how to use the app with clear instructions and a simplified design.

python requires backend

we use tensorflow and keras

django

flask better

artefact looks

home page

UI UX error and 404

Identified StyleGAN

Identified FF

3.5 Summary

Chapter 4

Results

The Artefact that was created in Chapter 3 could identify StyleGAN images with a very high accuracy. The various results achieved in the development of the Artefact will be discussed and evaluated. The initial neural network accuracy will be discussed and example output from the artefact making the predictions will be demonstrated. For the hyperparameter optimized neural network that acts as this projects final model will be compared to the initial model and the model created by Wang et al. (2020) to demonstrate the large gains made to the model performance using optimization in a constrained resources environment.

4.1 The First Neural Network

The neural network created in Sprint 1 produced a detection accuracy of 61% by just replacing StyleGAN generated and real human images with the Cat-vs-Dog images in the introduction to CNN's exercise. The accuracy achieved acted as a proof of concept. Using Hot and Cold learning techniques improved the model to a tested accuracy of 81%. This increase in accuracy and already high starting accuracy shows how powerful CNN's can be in a world with increasing artificially generated images. Figure shows the self created model implemented in the front-end artefact predicting on StyleGAN images and real human images from the FlickrFaces dataset. Although the model's prediction confidence is low, the predictions still act advantages compared to a random guess. A random guess and a neural network accuracy of 50% can be seen as providing the same value. If a model predicts with only 50% accuracy then code that mimics a coin flip prediction by random selection will provide the same results to the user. Therefore with the first neural network model starting with a accuracy higher that 60% and the then reworked model providing a accuracy of 80% was the value added by this project successful from the first iteration of the artefact development.

Self created model accuracy

Self classifying real and fake

the self created model miss classifying neeils

The optuna model accuracy

Optuna calasifying a human and a SyleGAN

Optuna Miscalssifying

Improved classification neels

Optuna on StyleGAN2

Table of all network accuracies

Conclusion

Chapter 5

Reflection

Bibliography

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283.

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631.

Albaradei, S., Wang, Y., Cao, L., and Li, L.-J. (2014). Learning mid-level features from object hierarchy for image classification. In *IEEE Winter Conference on Applications of Computer Vision*, pages 235–240. IEEE.

Albawi, S., Mohammed, T. A., and Al-Zawi, S. (2017). Understanding of a convolutional neural network. *IEEE*, pages 1–6.

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., and Farhan, L. (2021). Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8(1):1–74.

Ang, L., LI, Y.-x., and LI, X.-h. (2017). Tensorflow and keras-based convolutional neural network in cat image recognition. *DEStech Transactions on Computer Science and Engineering*, cmsam(cmsam).

Bera, S. and Shrivastava, V. K. (2020). Analysis of various optimizers on deep convolutional neural network model in the application of hyperspectral remote sensing image classification. *International Journal of Remote Sensing*, 41(7):2664–2683.

Burguillo, J. C. (2010). Using game theory and competition-based learning to stimulate student motivation and performance. *Computers & education*, 55(2):566–575.

Chandler, D., Munday, R., and Oxford University, P. (2016). *A dictionary of social media*. Oxford University Press.

Collins, H. (2018). *Creative research: the theory and practice of research for the creative industries*. Bloomsbury Publishing.

Craig-Wood, N. (2021). rclone documentation.

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., and Bharath, A. A. (2018). Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65.

Culjak, I., Abram, D., Pribanic, T., Dzapo, H., and Cifrek, M. (2012). A brief introduction to opencv. In *2012 proceedings of the 35th international convention MIPRO*, pages 1725–1730. IEEE.

Fleishman, G. (2019). How to spot the realistic fake people creeping into your timelines. *Fast Company*.

Fysh, M. C. and Bindemann, M. (2018). Human–computer interaction in face matching. *Cognitive Science*, 42(5):1714–1732.

Gallagher, F. and Calabrese, E. (2019). *Facebook's latest takedown has a twist - AI-generated profile pictures*. ABC News.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks. *arXiv preprint arXiv:1406.2661*.

Hays, K., Thomas, O., Maynard, I., and Bawden, M. (2009). The role of confidence in world-class sport performance. *Journal of sports sciences*, 27(11):1185–1199.

Kandel, I., Castelli, M., and Popovič, A. (2020). Comparative study of first order optimizers for image classification using convolutional neural networks on histopathology images. *Journal of imaging*, 6(9):92.

Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., and Aila, T. (2021). Alias-free generative adversarial networks. In *Proc. NeurIPS*.

Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. *IEEE transactions on pattern analysis and machine intelligence*, PP.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2020). Analyzing and improving the image quality of stylegan. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8107–8116.

Krenker, A., Bešter, J., and Kos, A. (2011). Introduction to the artificial neural networks. *Artificial Neural Networks: Methodological Advances and Biomedical Applications. InTech*, pages 1–18.

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., and Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26.

Medvedev, A. and Zemplinerová, A. (2005). Does competition improve performance? evidence from the czech manufacturing industries. *Prague Economic Papers*, 14(4):317–330.

Mitra, A., Mohanty, S. P., Corcoran, P., and Kougianos, E. (2021). A machine learning based approach for deepfake detection in social media through key video frame extraction. *SN Computer Science*, 2(2).

Müller, B., Reinhardt, J., and Strickland, M. T. (1995). *Neural networks: an introduction*. Springer Science & Business Media.

Murugesan, S., Rossi, G., Wilbanks, L., and Djavanshir, R. (2011). The future of web apps. *IT Professional*, 13(5):12–14.

Nasr-Esfahani, E., Samavi, S., Karimi, N., Soroushmehr, S., Jafari, M., Ward, K., and Najarian, K. (2016). Melanoma detection by analysis of clinical images using convolutional neural network. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1373–1376.

O'Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

Peffers, K. E. N., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3):45–77.

Pfleeger, C. and Pfleeger, S. L. (2002). *Security in Computing*. Prentice Hall.

Rasmussen, C. E. and Ghahramani, Z. (2001). Occam's razor. *Advances in neural information processing systems*, pages 294–300.

Sharma, S., Sharma, S., and Athaiya, A. (2017). Activation functions in neural networks. *towards data science*, 6(12):310–316.

Skilton, M. and Hovsepian, F. (2017). *The 4th industrial revolution : responding to the impact of artificial intelligence on business*. Springer International Publishing AG.

Trask, A. (2019). *Grokking Deep Learning*. Manning Publications Co.

Wang, S.-Y., Wang, O., Zhang, R., Owens, A., and Efros, A. A. (2020). Cnn-generated images are surprisingly easy to spot... for now. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8695–8704.

Weiyin, H., Thong, J. Y. L., Chasalow, L. C., and Dhillon, G. (2011). User acceptance of agile information systems: A model and empirical test. *Journal of Management Information Systems*, 28(1):235–272.

Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., and Lipson, H. (2015). Understanding neural networks through deep visualization. *arXiv e-prints*, page arXiv: 1506.06579.

Zhai, X., Oliver, A., Kolesnikov, A., and Beyer, L. (2019). S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1476–1485.

Appendix A

Ethics Form

Title of project:

Identifying StyleGAN images

Name: Neil Foxcroft
Supervisor: Dr. R. Serfontein

Starting date: 16/02/2021
End date: 08/11/2021

1. Have you read the information available related to research ethics (Chapter 5 of Researching Information Systems and Computing; BJ Oates and Chapter 13 of Writing for computer science, J Zobel; Manual for postgraduate studies, available on eFundi)?	Yes ✔	No

2. Do you make use of people as the source of data in your project (for example the completion of questionnaires or evaluation of products)?	Yes	No ✔

3. Are there any aspects of your research that you need permission from another party to use (for example use of property or tools)? If yes, provide more detail.	Yes	No ✔

4. Describe your research question and give a brief description of your plans for the collection of data.

With the development of this artefact, certain resources will be used to train the neural network. The training requires images that were generated by StyleGAN. For the comparison in the artefact images of real humans will be used.

The StyleGAN generated images are available on the official StyleGAN GitHub repository. Included in this repository is trained StyleGAN models and multiple datasets of StyleGAN generated images. The licencing of these images is stated on the GitHub repository and is a Creative Commons license by NVIDIA Corporation (Karras *et al*., 2019).

For the verified human faces, the preliminary dataset that will be used is the Flickr Faces dataset that was initially used to benchmark StyleGAN. The individual images were published in Flickr by their respective authors under either Creative Commons, Public Domain. All of these licenses allow free use, redistribution, and adaptation for non-commercial purposes (Karras *et al*., 2019).

With the identified need for detection of StyleGAN images and the discussed security implications that the invention of StyleGAN and similar methods introduced the proposed project aims to detect these images with the use of a trained neural network. The main research question that this proposed project aims to answer is: How can StyleGAN generated images be detected?

Data that will be used in this proposed project is free to use data sets provided in the StyleGAN repository and the Flickr Faces dataset. These datasets stated in their respective repositories that they are included with a creative commons license that will allow me to use them for my research purposes.

5. Describe how you plan to provide information about yourself and the goals of your research to participants.

This proposed project will not require any participants

6. Describe what methods you will use to get permission from participants in your study.

This proposed project will not require any participants

7. Will you be able to ensure that participants' information will be used in an anonymous, private, and confidential way? How? This proposed project will not require any participants	Yes ✔	No

8. Are there any foreseeable risks of damage (physical, social, or psychological) to participants or the environment? If you answer yes, give detail of the preventative measures you will follow.	Yes	No ✔

9. Are there any foreseeable risks to the NWU, for example, lawful actions that may follow the research or damage the image of the university? If yes, give detail.	Yes	No ✔

10. Are there any other ethical issues that may occur during the execution of the research (for example conflicting interests)? If yes, provide detail and explain how you plan to manage them.	Yes	No ✔

I hereby declare that the information contained in this form is accurate. I have attempted to identify the risks that may arise in conducting this research and acknowledge my obligations and the rights of the participants. I confirm that the research will be conducted in line with all University, legal and ethical standards.

Name of student: Neil Foxcroft

Signature:_____
Date: 18/04/2021

Name of study leader: Dr R. Serfontein

Signature: _____
Date:

Name of an additional moderator:

Signature: _____
Date:

Appendix B

Research Proposal

SUBJECT GROUP COMPUTER SCIENCE AND INFORMATION SYSTEMS

Research Proposal for an Honours project

The student and the supervisor must consult the *Manual for Postgraduate Studies* before writing the research proposal. The *Manual for Postgraduate Studies* explains in detail what is expected at each of the subheadings below. The proposal should not be longer than 5 pages.

The Subject Group requires that the research proposal will be submitted through the use of this form and in the format below. Please complete using a computer.

1 Student initials, surname, and student number

Initials	N	Surname	Foxcroft	Student number	28418077

2 Degree for which student is registered

BSc Honours in Computer Science and Information Technology

3 Name of supervisor

Initials and surname	Dr R. Serfontein

4 Proposed title

Title (preferably not more than 12 words)	Identifying StyleGAN images

5 Problem statement and substantiation

Provide the theme and link with gaps in the literature and recent research in the area. Indicate the research question, its actuality and how the research will endeavour to answer the question.

StyleGAN is an open-source Generative Adversarial Network (GAN) that can be used to generate faces of people that do not exist (such as those shown on thispersondoesnotexist.com). This means that fraudsters can use StyleGAN generated faces that normally would pass a visual inspection conducted by a human inspector as part of false identities. The detection of such images with the use of artificial intelligence will be useful because of the factors that currently lead to misidentification.

Possible misidentification of StyleGAN images is a reality that needs to be addressed. Humans in the role of identifying artificially generated human faces may be susceptible to external factors hindering their capabilities and increasing the rate of error in which they identify fraudulent images (Fysh & Bindemann, 2018). Fysh and Bindemann also noted that in the specific use case of passport officers that were tested on passport images captured on the same day against the "traveller" presenting that image, that the officers made substantial errors in a controlled environment when comparing the picture identity to that of the traveller. These results enforced their original statement that humans struggle with unfamiliar face identification.

By looking at how the technology has been used since its release, the possible use-cases for GAN generated images and the always growing cybercrime industry the possible detection of these images is identified as a crucial function in the 4th industrial revolution. With these identified factors will the proposed project aim to solve the problem of detecting StyleGAN images by using an artificial intelligence approach to solve the problem.

6 Research aims and objectives

Provide the different general as well as the specific aspects which will form part of the research.

Aims:

The main purpose of the proposed project is to develop a method that can detect fraudulent human faces created by StyleGAN with relative accuracy. Various techniques and approaches to the detection of GAN generated images will be researched and the simplest implemented approach that can still detect these types of images with relative surety will be selected for the artefact.

Objectives:

The success of the project will be weighed against the completion of the secondary objectives that have been identified as listed below.

- Perform a literature study on GANs and specifically analyse the architecture and function of StyleGAN to understand the technology.
- Develop an approach to the successful identification of generated images.
- Develop an artefact that will use the selected method to detect a fake identity.

The successful completion of the above-identified objectives will aid the researcher in satisfying the aim of the proposed research project.

7 Basic hypotheses (where applicable)

The use of Neural Networks will aid in the successful detection of StyleGAN generated images.

8 Method of investigation

8.1 Literature study

Provide an indication only of which literature will be used in the study with key references. A summary of the literature is not required here.

In this modern world with humanity currently in its 4th industrial revolution, the additions of innovative technologies require original approaches to implement and maintain these technologies. The change from the digital age to the automation age is accelerated by the breakthroughs in the fields of artificial intelligence (AI) and security. (Skilton & Hovsepian, 2017)

One of the big advances in artificial intelligence is the creation of StyleGAN, this new approach to a generative adversarial network allowed for more control in an image than its predecessors. (Karras et al., 2019) StyleGAN uses the principles of a GAN to create new images derived from input that specifies what “styles” need to be included in the image. According to Karras et al. (2019) a style is defined in this contexts as a set of parameters that modifies the input of the image to result in different outputs. If the input received is that the image needs to be in the style of a person with glasses, red hair and must be female, StyleGAN will then generate that image based on images used of that similar styles in the initial training of the model. The resulting image will thus be that of the “styles” required in the input. While this functionality can be utilized for various positive use cases – this breakthrough also creates various challenges and setbacks in the field of security, more specifically the aspects of facial recognition and identity verification as malicious use of this GAN might aid in the creation of fraudulent identities. (Mitra *et al.*, 2021)

The big advancement in StyleGAN that led to the conception of this proposed project was the release of StyleGAN2 in February 2020. (Karras, Laine, Aittala*, et al.*, 2020) Following the release of version 2, there were improvements in the generation process of these images with this updated version of StyleGAN. StyleGAN2 saw that images were no longer subverted with artefacts or traces left on the image because of the processing method used in the previous iterations. (Karras, Laine, Aittala*, et al.*, 2020) These traces were easy to identify and clearly showed out of place in the context of the image. With the removal of the traces, usually, in the form of drop-like spots, the difficulty in detecting the generated images increased and similarly the need to detect StyleGAN generated images used maliciously in security verification processes.

8.2 Methods of investigation

The proposed design, data acquisition, procedures, data processing, funding sources (but not a budget), mathematical methods, computer methods.

The positivistic research paradigm is suitable for the proposed project because with the creation of the artefact, the data collected will be examined and an unobjective interpretation of the data is necessary. The data collected will be the results of the artefact's successful identification of StyleGAN generated images.

DSRM is an information systems specific methodology that focuses on research and iterative design. (Peffers *et al.*, 2007) Because the researcher is studying the field and technologies in which they want to solve the specific problem the background knowledge of the problem will be explored parallel to the design of the problem solution. DSRM will enable iterative design and development throughout the completion of the proposed project

The artefact that will be developed will aim to detect StyleGAN generated images between a data set of real images of human faces and StyleGAN generated images. The artefact will be hosted online as this will simplify the development of user interaction. Because of these factors, Agile will be the most suitable methodology for artefact development.

Neural networks and computer vision will be used once an extensive literature study identifies these methods suitable for StyleGAN image detection.

9 Provisional chapter division

Chapter 1: Introduction - This chapter will be concluded in the project proposal phase. It will include the research question, the project description and background. It will include the proposed project plan for the entire project.

Chapter 2: Literature Study - This chapter will be comprised of all the necessary research to understand the project and fulfil the project aims and objectives. Mostly in this project will be focusing on the specific workings of StyleGAN to effectively detect fake images.

Chapter 3: Development of the artefact - Chapter 3 will apply the chosen methodologies that were identified and discussed in Chapter 1 to enable the successful development of the proposed project's artefact. This chapter will document the artefact development phase including unfamiliar problems that are identified within the development stage. The success of the artefact will be compared to the Aims and Objectives of Chapter 1 and if they are met.

Chapter 4: Testing and Results - The results of the Artefact will be introduced in this Chapter and the successful identification of StyleGAN images will be determined. The testing in this Chapter will identify the success of the artefact with the comparison in Chapter 3.

Chapter 5: Conclusion - Chapter 5 will summarize the entire proposed project and conclude if the problem was solved with the successful completion of the objectives of the proposed project that allowed it to fulfil the aim of the project. The limitations that impeded the proposed project will be discussed in this section. Future expansion of the project will be discussed and explained in the context of the limitations faced

10 Literature references

Provide complete references to the literature referenced in this proposal only.

Gallagher, F. and Calabrese, E. (2019).Facebook's latest takedown has a twist - AI-generated profile pictures. ABC News. 31 December 2019. https://abcnews.go.com/US/facebooks-latest-takedown-twist-ai-generated-profile-pictures/story?id=67925292 Date of access: 24 March 2021.

Fysh, M. C. and Bindemann, M. (2018). Human–computer interaction in face matching.Cognitive Science, 42(5):1714–1732.

Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture forgenerative adversarial networks.IEEE transactions on pattern analysis and machineintelligence, PP.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2020). Analyzingand improving the image quality of stylegan.2020 IEEE/CVF Conference on ComputerVision and Pattern Recognition (CVPR), pages 8107–8116.

Mitra, A., Mohanty, S. P., Corcoran, P., and Kougianos, E. (2021). A machine learningbased approach for deepfake detection in social media through key video frameextraction.SN Computer Science, 2(2)

Peffers, K. E. N., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. (2007). Adesign science research methodology for information systems research.Journal ofManagement Information Systems, 24(3):45–77.

Skilton, M. and Hovsepian, F. (2017).The 4th industrial revolution : responding to theimpact of artificial intelligence on business. Springer International Publishing AG.

Boxcroft

.....

Student

18 / 04 / 2021

Date

.....

Supervisor

.....

Date

Appendix C

Jupyter Notebook: 1

Mount, Download and Unzip the datasets from Google Drive

[1]:
```
import zipfile
from google.colab import drive

drive.mount('/content/drive/')

#poc_DATASET
#zip_ref = zipfile.ZipFile("/content/drive/My Drive/sg_ff_filtered_red.zip", 'r')

#ISGI_20000_200gray_DATASET
# zip_ref = zipfile.ZipFile("/content/drive/My Drive/ISGI_dataset_200g.zip", 'r')

#ISGI_20000_200rgb_DATASET
zip_ref = zipfile.ZipFile("/content/drive/My Drive/ISGI_dataset_200rgb.zip", 'r')

zip_ref.extractall("/tmp/")
zip_ref.close()
```

```
Drive already mounted at /content/drive/; to attempt to forcibly remount, call
drive.mount("/content/drive/", force_remount=True).
```

Set the directory paths to the subfolders

[2]:
```
import os

base_dir = '/tmp/ISGI_dataset_200rgb'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'valid') #valid or validations
```

```
# Directory with our training FlickerFaces pictures
train_ff_dir = os.path.join(train_dir, 'ff')

# Directory with our training StyleGAN pictures
train_sg_dir = os.path.join(train_dir, 'sg')

# Directory with our validation FlickerFaces pictures
validation_ff_dir = os.path.join(validation_dir, 'ff')

# Directory with our validation StyleGAN pictures
validation_sg_dir = os.path.join(validation_dir, 'sg')
```

[3]:
```
train_ff_fnames = os.listdir(train_ff_dir)
train_ff_fnames.sort()
print(train_ff_fnames[:10])

train_sg_fnames = os.listdir(train_sg_dir)
train_sg_fnames.sort()
print(train_sg_fnames[:10])
```

```
['00000.png', '00001.png', '00002.png', '00003.png', '00004.png', '00005.png',
'00006.png', '00007.png', '00008.png', '00009.png']
['000000.png', '000001.png', '000002.png', '000003.png', '000004.png',
'000005.png', '000006.png', '000007.png', '000008.png', '000009.png']
```

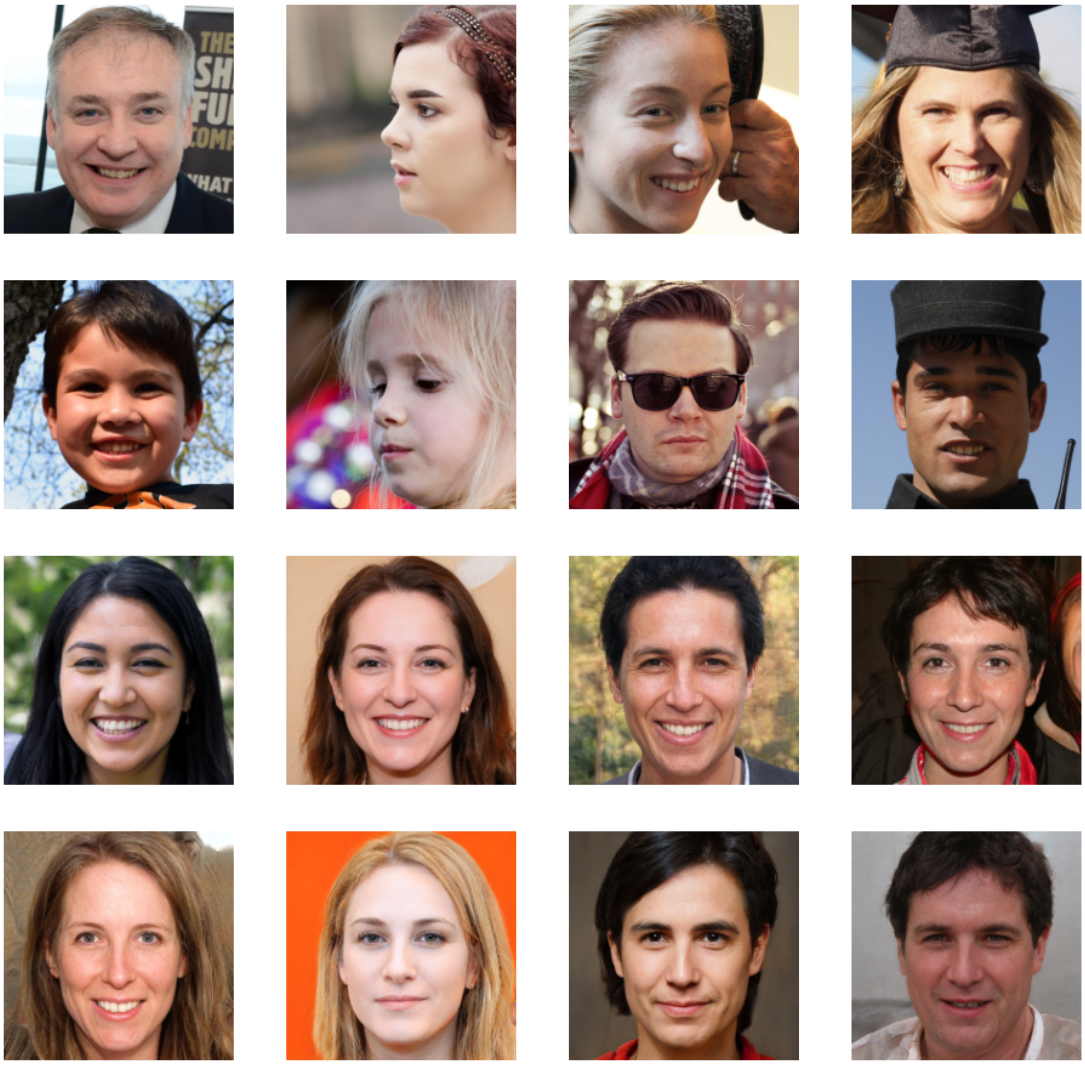
[4]:
```
print('Training_FlickerFaces images total: \t', len(os.listdir(train_ff_dir)))
print('Training_StyleGAN images total: \t', len(os.listdir(train_sg_dir)))
print('Validation_FlickerFaces images total: \t', len(os.
 ↪listdir(validation_ff_dir)))
print('Validation_StyleGAN images total: \t', len(os.listdir(validation_sg_dir)))
```

```
Training_FlickerFaces images total:      8000
Training_StyleGAN images total:          8000
Validation_FlickerFaces images total:    1000
Validation_StyleGAN images total:        1000
```

[5]:
```
%matplotlib inline

import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import random

#params for graph
nrows = 4
ncols = 4

#index for iteration
```

```
pic_index = random.randint(0, 990)
```

[6]:
```
fig = plt.gcf()
fig.set_size_inches(ncols * 4, nrows * 4)

pic_index += 8
next_ff_pix =  [os.path.join(train_ff_dir, fname)
                for fname in train_ff_fnames[pic_index-8:pic_index]]
next_sg_pix =  [os.path.join(train_sg_dir, fname)
                for fname in train_sg_fnames[pic_index-8:pic_index]]

for i, img_path in enumerate(next_ff_pix+next_sg_pix):
  sp = plt.subplot(nrows, ncols, i + 1)
  sp.axis('Off')

  img = mpimg.imread(img_path)
  plt.imshow(img)

  plt.show
```



[7]:
```
from tensorflow.keras import layers
from tensorflow.keras import Model
from tensorflow.keras.layers import BatchNormalization, Dropout

#from tensorflow.keras.layers import Input, Flatten, Dense, Conv2D,␣
 ↪BatchNormalization, LeakyReLU, Dropout, Activation
```

[9]:
```
input_layer = layers.Input(shape=(200, 200, 3))

x = layers.Conv2D(16, 3, activation='relu')(input_layer)
x = layers.MaxPooling2D(2)(x)
x = Dropout(rate = 0.2)(x)
```

```
x = layers.Conv2D(32, 3, activation='relu')(x)
x = layers.MaxPooling2D(2)(x)
x = Dropout(rate = 0.3)(x)

x = layers.Conv2D(64, 3, activation='relu')(x)
x = layers.MaxPooling2D(2)(x)
x = Dropout(rate = 0.4)(x)

x = layers.Conv2D(64, 3, activation='relu')(x)
x = layers.MaxPooling2D(2)(x)
x = Dropout(rate = 0.5)(x)

x = layers.Conv2D(128, 3, activation='relu')(x)
x = layers.MaxPooling2D(2)(x)
x = Dropout(rate = 0.5)(x)

x = layers.Conv2D(128, 3, activation='relu')(x)
x = layers.MaxPooling2D(2)(x)
x = Dropout(rate = 0.5)(x)

x = layers.Flatten()(x)

x = layers.Dense(200, activation='relu')(x)
x = Dropout(rate = 0.5)(x)

output_layer = layers.Dense(1, activation='sigmoid')(x)

model = Model(input_layer, output_layer)

model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 200, 200, 3)]     0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 198, 198, 16)      448
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 99, 99, 16)        0
_________________________________________________________________
dropout_6 (Dropout)          (None, 99, 99, 16)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 97, 97, 32)        4640
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 48, 48, 32)        0
_________________________________________________________________
dropout_7 (Dropout)          (None, 48, 48, 32)        0
```

```
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 46, 46, 64)        18496
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 23, 23, 64)        0
_________________________________________________________________
dropout_8 (Dropout)          (None, 23, 23, 64)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 21, 21, 64)        36928
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 10, 10, 64)        0
_________________________________________________________________
dropout_9 (Dropout)          (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 8, 8, 128)         73856
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 4, 4, 128)         0
_________________________________________________________________
dropout_10 (Dropout)         (None, 4, 4, 128)         0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 2, 2, 128)         147584
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 1, 1, 128)         0
_________________________________________________________________
dropout_11 (Dropout)         (None, 1, 1, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 200)               25800
_________________________________________________________________
dropout_12 (Dropout)         (None, 200)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 201
=================================================================
Total params: 307,953
Trainable params: 307,953
Non-trainable params: 0
_________________________________________________________________
```

[ ]:

[10]:
```
input_layer = layers.Input(shape=(200, 200, 3))

x = layers.Conv2D(16, 3, activation='relu')(input_layer)
x = layers.MaxPooling2D(2)(x)

x = layers.Conv2D(32, 3, activation='relu')(x)
x = layers.MaxPooling2D(2)(x)
```

```
x = layers.Conv2D(64, 3, activation='relu')(x)
x = layers.MaxPooling2D(2)(x)

x = layers.Flatten()(x)

x = layers.Dense(512, activation='relu')(x)

output_layer = layers.Dense(1, activation='sigmoid')(x)

model = Model(input_layer, output_layer)

model.summary()
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 200, 200, 3)]     0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 198, 198, 16)      448
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 99, 99, 16)        0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 97, 97, 32)        4640
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 48, 48, 32)        0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 46, 46, 64)        18496
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 23, 23, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 33856)             0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               17334784
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 513
=================================================================
Total params: 17,358,881
Trainable params: 17,358,881
Non-trainable params: 0
_________________________________________________________________
```

[ ]:
```
from tensorflow.keras.optimizers import RMSprop

model.compile(loss='binary_crossentropy',
              optimizer=RMSprop(learning_rate=0.001),
```

```
              metrics=['acc'])
```

[ ]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale=1./255,
                                   zoom_range = 0.2,
                                   horizontal_flip = True,
                                   vertical_flip = True)

val_datagen = ImageDataGenerator(rescale=1./255)

# Flow training images in batches of 20 using train_datagen generator
train_generator = train_datagen.flow_from_directory(
        train_dir,  # This is the source directory for training images
        target_size=(200, 200),
        batch_size=20,
        # Since we use binary_crossentropy loss, we need binary labels
        class_mode='binary')

# Flow validation images in batches of 20 using val_datagen generator
validation_generator = val_datagen.flow_from_directory(
        validation_dir,
        target_size=(200, 200),
        batch_size=20,
        class_mode='binary')
```

```
Found 16000 images belonging to 2 classes.
Found 2000 images belonging to 2 classes.
```

[ ]:
```
history = model.fit(
      train_generator,
      steps_per_epoch=800,  # 2000 images = batch_size * steps
      epochs=15,
      validation_data=validation_generator,
      validation_steps=100,  # 1000 images = batch_size * steps
      verbose=2)
```

```
Epoch 1/15
800/800 - 177s - loss: 0.6880 - acc: 0.5645 - val_loss: 0.6586 - val_acc: 0.5910
Epoch 2/15
800/800 - 174s - loss: 0.6152 - acc: 0.6749 - val_loss: 0.5942 - val_acc: 0.6505
Epoch 3/15
800/800 - 175s - loss: 0.5566 - acc: 0.7231 - val_loss: 0.5096 - val_acc: 0.7560
Epoch 4/15
800/800 - 175s - loss: 0.5332 - acc: 0.7395 - val_loss: 0.5261 - val_acc: 0.7380
Epoch 5/15
800/800 - 176s - loss: 0.5098 - acc: 0.7609 - val_loss: 0.4777 - val_acc: 0.7910
Epoch 6/15
```

```
800/800 - 176s - loss: 0.4880 - acc: 0.7708 - val_loss: 0.4657 - val_acc: 0.7800
Epoch 7/15
800/800 - 176s - loss: 0.4909 - acc: 0.7768 - val_loss: 0.4772 - val_acc: 0.8140
Epoch 8/15
800/800 - 175s - loss: 0.4652 - acc: 0.7887 - val_loss: 0.4105 - val_acc: 0.8370
Epoch 9/15
800/800 - 175s - loss: 0.4648 - acc: 0.7915 - val_loss: 0.4778 - val_acc: 0.7695
Epoch 10/15
800/800 - 176s - loss: 0.4730 - acc: 0.7854 - val_loss: 0.4527 - val_acc: 0.7835
Epoch 11/15
800/800 - 177s - loss: 0.4618 - acc: 0.7952 - val_loss: 0.4412 - val_acc: 0.7860
Epoch 12/15
800/800 - 177s - loss: 0.4626 - acc: 0.7966 - val_loss: 0.5354 - val_acc: 0.6915
Epoch 13/15
800/800 - 176s - loss: 0.4586 - acc: 0.7956 - val_loss: 0.4983 - val_acc: 0.7465
Epoch 14/15
800/800 - 176s - loss: 0.4698 - acc: 0.7962 - val_loss: 0.4762 - val_acc: 0.7845
Epoch 15/15
800/800 - 175s - loss: 0.4768 - acc: 0.7943 - val_loss: 0.4464 - val_acc: 0.8170
```

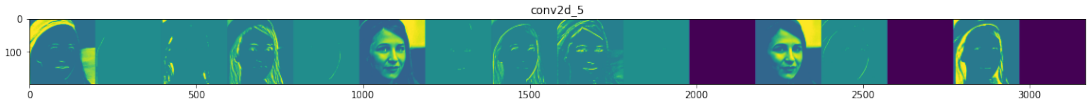
[ ]:
```
scores = model.evaluate(validation_generator, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

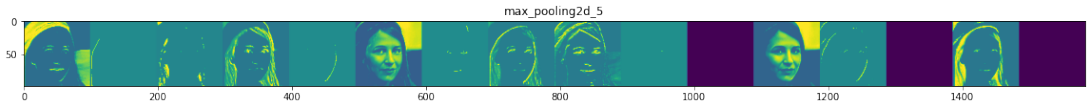
```
Accuracy: 81.70%
```

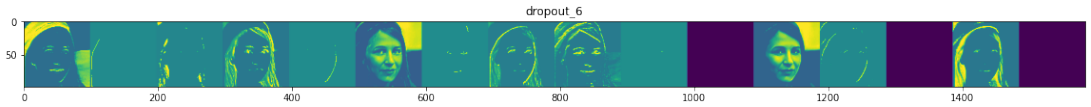
[ ]:
```
import numpy as np
import random
from tensorflow.keras.preprocessing.image import img_to_array, load_img

# define a new Model that will take an image as input, and will output
# intermediate representations for all layers in the previous model after
# the first.
successive_outputs = [layer.output for layer in model.layers[1:]]
visualization_model = Model(input_layer, successive_outputs)

# prepare a random input image of a FlickerFaces or StyleGAN from the training␣
 ↪set.
ff_img_files = [os.path.join(train_ff_dir, f) for f in train_ff_fnames]
sg_img_files = [os.path.join(train_sg_dir, f) for f in train_sg_fnames]
img_path = random.choice(ff_img_files + sg_img_files)

img = load_img(img_path, target_size=(200, 200))  # this is a PIL image
x = img_to_array(img)  # Numpy array with shape (150, 150, 3)
x = x.reshape((1,) + x.shape)  # Numpy array with shape (1, 150, 150, 3)

# Rescale by 1/255
x /= 255
```

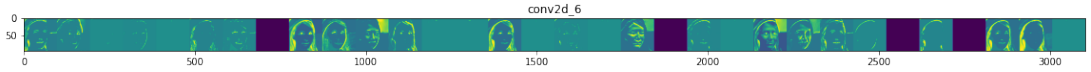
```
#  run our image through our network, thus obtaining all
# intermediate representations for this image.
successive_feature_maps = visualization_model.predict(x)

# These are the names of the layers, so can have them as part of our plot
layer_names = [layer.name for layer in model.layers[1:]]

# Now  display our representations
for layer_name, feature_map in zip(layer_names, successive_feature_maps):
  if len(feature_map.shape) == 4:
    # Just do this for the conv / maxpool layers, not the fully-connected layers
    n_features = feature_map.shape[-1]  # number of features in feature map
    # The feature map has shape (1, size, size, n_features)
    size = feature_map.shape[1]
    # We will tile our images in this matrix
    display_grid = np.zeros((size, size * n_features))
    for i in range(n_features):
      # Postprocess the feature to make it visually palatable
      x = feature_map[0, :, :, i]
      x -= x.mean()
      x /= x.std()
      x *= 64
      x += 128
      x = np.clip(x, 0, 255).astype('uint8')
      # We'll tile each filter into this big horizontal grid
      display_grid[:, i * size : (i + 1) * size] = x
    # Display the grid
    scale = 20. / n_features
    plt.figure(figsize=(scale * n_features, scale))
    plt.title(layer_name)
    plt.grid(False)
    plt.imshow(display_grid, aspect='auto', cmap='viridis')
```

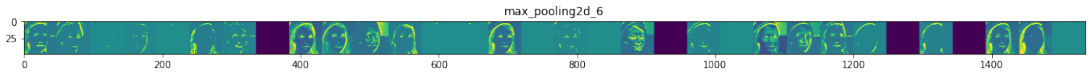
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:43: RuntimeWarning:
invalid value encountered in true_divide
```

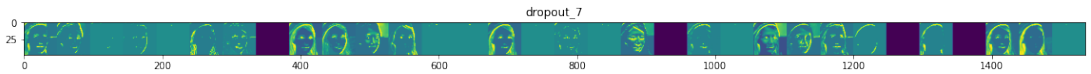


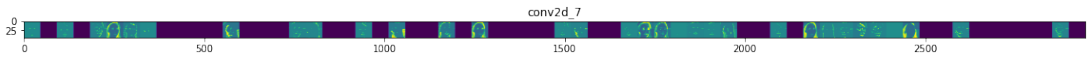


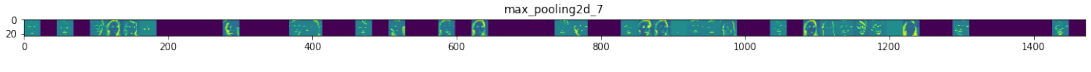

dropout_6
0
50
0
200
400
600
800
1000
1200
1400

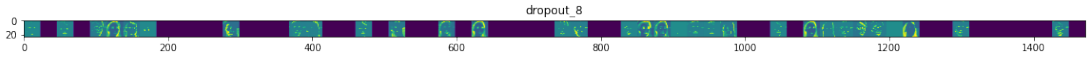

conv2d_6
0
50
0
500
1000
1500
2000
2500
3000

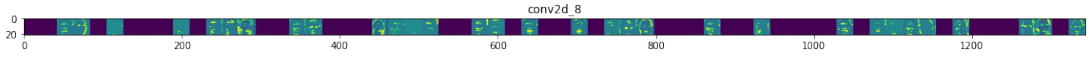

max_pooling2d_6
0
25
0
200
400
600
800
1000
1200
1400

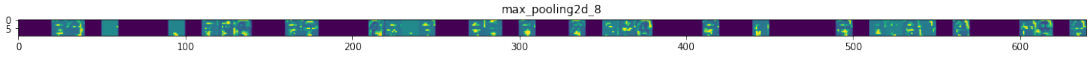

dropout_7
0
25
0
200
400
600
800
1000
1200
1400

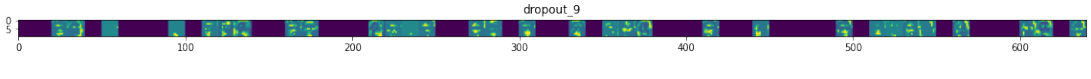

conv2d_7
0
25
0
500
1000
1500
2000
2500

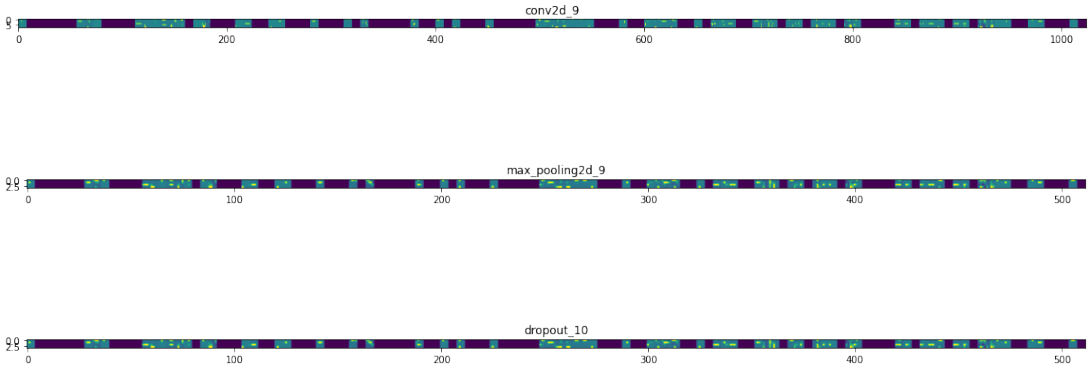

max_pooling2d_7
0
20
0
200
400
600
800
1000
1200
1400


dropout_8
0
20
0
200
400
600
800
1000
1200
1400


conv2d_8
0
20
0
200
400
600
800
1000
1200

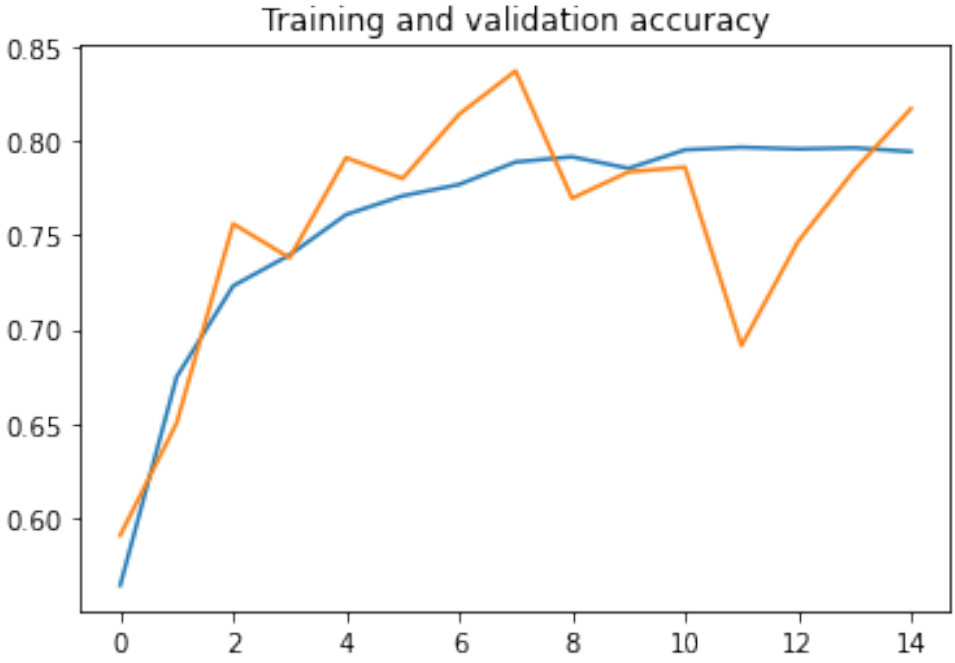

max_pooling2d_8
0
5
0
100
200
300
400
500
600


dropout_9
0
5
0
100
200
300
400
500
600



[ ]:
```
# Retrieve a list of accuracy results on training and validation data
# sets for each training epoch
acc = history.history['acc']
val_acc = history.history['val_acc']

# Retrieve a list of list results on training and validation data
# sets for each training epoch
loss = history.history['loss']
val_loss = history.history['val_loss']

# Get number of epochs
epochs = range(len(acc))

# Plot training and validation accuracy per epoch
plt.plot(epochs, acc)
plt.plot(epochs, val_acc)
plt.title('Training and validation accuracy')

plt.figure()

# Plot training and validation loss per epoch
plt.plot(epochs, loss)
plt.plot(epochs, val_loss)
plt.title('Training and validation loss')
```

[ ]:
```
Text(0.5, 1.0, 'Training and validation loss')
```



[ ]:
```
model.save('placeholderm2.h5')
```

```
[ ]: !zip -r /content/model_1 /content/model
```

```
  adding: content/tmp/ (stored 0%)
  adding: content/tmp/model_1/ (stored 0%)
  adding: content/tmp/model_1/saved_model.pb (deflated 90%)
  adding: content/tmp/model_1/variables/ (stored 0%)
  adding: content/tmp/model_1/variables/variables.index (deflated 68%)
  adding: content/tmp/model_1/variables/variables.data-00000-of-00001 (deflated
12%)
  adding: content/tmp/model_1/keras_metadata.pb (deflated 93%)
  adding: content/tmp/model_1/assets/ (stored 0%)
```

Appendix D

Jupyter Notebook: 2

[ ]:
```
import zipfile
from google.colab import drive

drive.mount('/content/drive/')

#poc_DATASET
zip_ref = zipfile.ZipFile("/content/drive/My Drive/sg_ff_filtered_red.zip", 'r')

#ISGI_20000_200gray_DATASET
# zip_ref = zipfile.ZipFile("/content/drive/My Drive/ISGI_dataset_200g.zip", 'r')

#ISGI_20000_200rgb_DATASET
#zip_ref = zipfile.ZipFile("/content/drive/My Drive/ISGI_dataset_200rgb.zip",␣
↪'r')

zip_ref.extractall("/tmp/")
zip_ref.close()
```

```
Mounted at /content/drive/
```

[ ]:
```

```

[ ]:
```
import os

base_dir = '/tmp/sg_ff_filtered_red'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')
```

```
# Directory with our training FlickerFaces pictures
train_ff_dir = os.path.join(train_dir, 'ff')

# Directory with our training StyleGAN pictures
train_sg_dir = os.path.join(train_dir, 'sg')

# Directory with our validation FlickerFaces pictures
validation_ff_dir = os.path.join(validation_dir, 'ff')

# Directory with our validation StyleGAN pictures
validation_sg_dir = os.path.join(validation_dir, 'sg')
```

[2]:
```
#imports
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.backend import clear_session
import tensorflow as tf
tf.test.gpu_device_name()
```

[2]:
```
'/device:GPU:0'
```

[ ]:
```
print('Training_FlickerFaces images total: \t', len(os.listdir(train_ff_dir)))
print('Training_StyleGAN images total: \t', len(os.listdir(train_sg_dir)))
print('Validation_FlickerFaces images total: \t', len(os.
 ↪listdir(validation_ff_dir)))
print('Validation_StyleGAN images total: \t', len(os.listdir(validation_sg_dir)))
```

```
Training_FlickerFaces images total:      1000
Training_StyleGAN images total:          1000
Validation_FlickerFaces images total:    500
Validation_StyleGAN images total:        500
```

[1]:
```
!pip install optuna
```

```
Requirement already satisfied: optuna in /usr/local/lib/python3.7/dist-packages
(2.10.0)
Requirement already satisfied: colorlog in /usr/local/lib/python3.7/dist-
packages (from optuna) (6.5.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages
(from optuna) (1.19.5)
Requirement already satisfied: cliff in /usr/local/lib/python3.7/dist-packages
(from optuna) (3.9.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages
(from optuna) (4.62.3)
Requirement already satisfied: cmaes>=0.8.2 in /usr/local/lib/python3.7/dist-
packages (from optuna) (0.8.2)
Requirement already satisfied: sqlalchemy>=1.1.0 in
/usr/local/lib/python3.7/dist-packages (from optuna) (1.4.25)
```

```
Requirement already satisfied: PyYAML in /usr/local/lib/python3.7/dist-packages
(from optuna) (3.13)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dist-
packages (from optuna) (21.0)
Requirement already satisfied: alembic in /usr/local/lib/python3.7/dist-packages
(from optuna) (1.7.4)
Requirement already satisfied: scipy!=1.4.0 in /usr/local/lib/python3.7/dist-
packages (from optuna) (1.4.1)
Requirement already satisfied: pyparsing>=2.0.2 in
/usr/local/lib/python3.7/dist-packages (from packaging>=20.0->optuna) (2.4.7)
Requirement already satisfied: greenlet!=0.4.17 in
/usr/local/lib/python3.7/dist-packages (from sqlalchemy>=1.1.0->optuna) (1.1.2)
Requirement already satisfied: importlib-metadata in
/usr/local/lib/python3.7/dist-packages (from sqlalchemy>=1.1.0->optuna) (4.8.1)
Requirement already satisfied: importlib-resources in
/usr/local/lib/python3.7/dist-packages (from alembic->optuna) (5.2.2)
Requirement already satisfied: Mako in /usr/local/lib/python3.7/dist-packages
(from alembic->optuna) (1.1.5)
Requirement already satisfied: autopage>=0.4.0 in /usr/local/lib/python3.7/dist-
packages (from cliff->optuna) (0.4.0)
Requirement already satisfied: cmd2>=1.0.0 in /usr/local/lib/python3.7/dist-
packages (from cliff->optuna) (2.2.0)
Requirement already satisfied: pbr!=2.1.0,>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from cliff->optuna) (5.6.0)
Requirement already satisfied: stevedore>=2.0.1 in
/usr/local/lib/python3.7/dist-packages (from cliff->optuna) (3.5.0)
Requirement already satisfied: PrettyTable>=0.7.2 in
/usr/local/lib/python3.7/dist-packages (from cliff->optuna) (2.2.1)
Requirement already satisfied: pyperclip>=1.6 in /usr/local/lib/python3.7/dist-
packages (from cmd2>=1.0.0->cliff->optuna) (1.8.2)
Requirement already satisfied: attrs>=16.3.0 in /usr/local/lib/python3.7/dist-
packages (from cmd2>=1.0.0->cliff->optuna) (21.2.0)
Requirement already satisfied: wcwidth>=0.1.7 in /usr/local/lib/python3.7/dist-
packages (from cmd2>=1.0.0->cliff->optuna) (0.2.5)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.7/dist-packages (from cmd2>=1.0.0->cliff->optuna)
(3.7.4.3)
Requirement already satisfied: colorama>=0.3.7 in /usr/local/lib/python3.7/dist-
packages (from cmd2>=1.0.0->cliff->optuna) (0.4.4)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-
packages (from importlib-metadata->sqlalchemy>=1.1.0->optuna) (3.6.0)
Requirement already satisfied: MarkupSafe>=0.9.2 in
/usr/local/lib/python3.7/dist-packages (from Mako->alembic->optuna) (2.0.1)
```

[3]:
```
import optuna
```

[ ]:
```
dropout_rate = [0] * 2

def create_model(trial):

    num_layers = trial.suggest_int("num_layers", 1, 7)
    activation = trial.suggest_categorical("activation", ["relu"])
    dropout_rate[0] = trial.suggest_uniform('dropout_rate'+str(0), 0.0, 0.5)
    dropout_rate[1] = trial.suggest_uniform('dropout_rate'+str(1), 0.0, 0.5)
    mid_units = int(trial.suggest_discrete_uniform("mid_units", 100, 300, 100))
    filters=trial.suggest_categorical("filters", [16, 32, 64, 128])
    kernel_size=trial.suggest_categorical("kernel_size", [3, 3])
    strides=trial.suggest_categorical("strides", [1, 2])

    classifier = Sequential()

    #step 1  - Convolution Layers

    classifier.add(
        Conv2D(
            filters=filters,
            kernel_size=kernel_size,
            strides=1,
            activation = activation,
            input_shape=(200, 200, 3),
        )
    )

    classifier.add(MaxPooling2D(pool_size=(2, 2)))
    for i in range(1, num_layers):
        classifier.add(
            Conv2D(
                filters=filters,
                kernel_size=kernel_size,
                strides=1,
                activation = activation,
            )
        )
    classifier.add(MaxPooling2D(pool_size=(2, 2)))
    classifier.add(Dropout(dropout_rate[0]))
    classifier.add(Flatten())
    classifier.add(Dense(units = mid_units, activation = activation))
    classifier.add(Dropout(dropout_rate[1]))
    classifier.add(Dense(units = 1, activation ='sigmoid'))

    return classifier
```

[ ]:
```
#image augumentation
from keras.preprocessing.image import ImageDataGenerator

#Data Preparation

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)

training_set = train_datagen.flow_from_directory(train_dir,
                                                 target_size = (200, 200),
                                                 batch_size = 10,
                                                 class_mode = 'binary')

test_set = test_datagen.flow_from_directory(validation_dir,
                                            target_size = (200, 200),
                                            batch_size = 10,
                                            class_mode = 'binary')
```

```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
```

[ ]:
```
training_set
```

[ ]:
```
<keras.preprocessing.image.DirectoryIterator at 0x7f4e91a45810>
```

[ ]:
```
def objective(trial):

    optimizer = trial.suggest_categorical("optimizer", ["sgd", "adam",␣
 ↪"rmsprop", "adadelta", "adagrad", "adamax"])

    classifier = create_model(trial)

    classifier.compile(optimizer = optimizer, loss = 'binary_crossentropy',␣
 ↪metrics = ['accuracy'])

    history = classifier.fit(training_set,
                         steps_per_epoch = 100, # num_samples // batch_size
                         epochs = 5, # entire iteration over dataset
                         validation_data = test_set,
                         validation_steps = 50) #https://keras.io/api/models/
 ↪model_training_apis/
```

```
    classifier.save('/drive/MyDrive/Models/trialmodel_' + str(history.
 ↪history['val_accuracy'][-1]) +".h5")


    return history.history["val_accuracy"][-1]
```

[ ]:
```
import pickle

study = optuna.create_study(direction="maximize", )

#studypik = pickle.load(open('study.pickle', 'rb'))
study.optimize(objective, n_trials = 10, timeout = 60 * 60 * 3,␣
 ↪show_progress_bar=True)
print(studypik.best_params)
print(studypik.best_value)
pickle.dump(studypik, open('study.pickle', 'wb'))
```

```
[I 2021-10-31 13:27:41,338] A new study created in memory with name:
no-name-57069c1e-1b4e-4142-9822-35a68cbcfece
/usr/local/lib/python3.7/dist-packages/optuna/progress_bar.py:47:
ExperimentalWarning:

Progress bar is experimental (supported from v1.2.0). The interface can change
in the future.


  0%|          | 0/10 [00:00<?, ?it/s]

100/100 [==============================] - 47s 171ms/step - loss: 0.7051 -
accuracy: 0.5170 - val_loss: 0.6786 - val_accuracy: 0.5620
[I 2021-10-31 13:28:29,917] Trial 0 finished with value:
0.5619999766349792 and parameters: {'optimizer': 'adagrad', 'num_layers': 1,
'activation': 'relu', 'dropout_rate0': 0.47587234574775245, 'dropout_rate1':
0.46183774326599186, 'mid_units': 300.0, 'filters': 64, 'kernel_size': 3,
'strides': 2}. Best is trial 0 with value: 0.5619999766349792.
100/100 [==============================] - 17s 159ms/step - loss: 0.7403 -
accuracy: 0.5790 - val_loss: 0.6311 - val_accuracy: 0.6700
[I 2021-10-31 13:28:51,470] Trial 1 finished with value:
0.6700000166893005 and parameters: {'optimizer': 'adam', 'num_layers': 2,
'activation': 'relu', 'dropout_rate0': 0.26769020451155284, 'dropout_rate1':
0.3386348492768438, 'mid_units': 100.0, 'filters': 32, 'kernel_size': 3,
'strides': 1}. Best is trial 1 with value: 0.6700000166893005.
100/100 [==============================] - 16s 156ms/step - loss: 0.7847 -
accuracy: 0.5600 - val_loss: 0.6496 - val_accuracy: 0.6620
[I 2021-10-31 13:29:08,209] Trial 2 finished with value:
0.6620000004768372 and parameters: {'optimizer': 'adamax', 'num_layers': 2,
'activation': 'relu', 'dropout_rate0': 0.362046754103899, 'dropout_rate1':
0.37466293295311437, 'mid_units': 100.0, 'filters': 16, 'kernel_size': 3,
'strides': 2}. Best is trial 1 with value: 0.6700000166893005.
```

```
100/100 [==============================] - 17s 159ms/step - loss: 0.6857 -
accuracy: 0.5740 - val_loss: 0.6632 - val_accuracy: 0.6320
[I 2021-10-31 13:29:25,547] Trial 3 finished with value:
0.6320000290870667 and parameters: {'optimizer': 'sgd', 'num_layers': 7,
'activation': 'relu', 'dropout_rate0': 0.08134882935920718, 'dropout_rate1':
0.41244931428641857, 'mid_units': 100.0, 'filters': 16, 'kernel_size': 3,
'strides': 1}. Best is trial 1 with value: 0.6700000166893005.
100/100 [==============================] - 17s 160ms/step - loss: 0.6980 -
accuracy: 0.5210 - val_loss: 0.6867 - val_accuracy: 0.5340
[I 2021-10-31 13:29:47,219] Trial 4 finished with value:
0.5339999794960022 and parameters: {'optimizer': 'adam', 'num_layers': 6,
'activation': 'relu', 'dropout_rate0': 0.2847882512202252, 'dropout_rate1':
0.03429756052223176, 'mid_units': 200.0, 'filters': 16, 'kernel_size': 3,
'strides': 1}. Best is trial 1 with value: 0.6700000166893005.
100/100 [==============================] - 17s 162ms/step - loss: 0.8187 -
accuracy: 0.5580 - val_loss: 0.6360 - val_accuracy: 0.6600
[I 2021-10-31 13:30:04,934] Trial 5 finished with value:
0.6600000262260437 and parameters: {'optimizer': 'adam', 'num_layers': 2,
'activation': 'relu', 'dropout_rate0': 0.11358935025708411, 'dropout_rate1':
0.3094802571565712, 'mid_units': 100.0, 'filters': 64, 'kernel_size': 3,
'strides': 1}. Best is trial 1 with value: 0.6700000166893005.
100/100 [==============================] - 21s 189ms/step - loss: 0.6956 -
accuracy: 0.4830 - val_loss: 0.6936 - val_accuracy: 0.4700
[I 2021-10-31 13:30:27,587] Trial 6 finished with value:
0.4699999988079071 and parameters: {'optimizer': 'adam', 'num_layers': 7,
'activation': 'relu', 'dropout_rate0': 0.19060102520992955, 'dropout_rate1':
0.4522466144468808, 'mid_units': 300.0, 'filters': 64, 'kernel_size': 3,
'strides': 2}. Best is trial 1 with value: 0.6700000166893005.
100/100 [==============================] - 18s 167ms/step - loss: 0.6928 -
accuracy: 0.5370 - val_loss: 0.6804 - val_accuracy: 0.6020
[I 2021-10-31 13:30:49,324] Trial 7 finished with value:
0.6019999980926514 and parameters: {'optimizer': 'adadelta', 'num_layers': 4,
'activation': 'relu', 'dropout_rate0': 0.22387331439595765, 'dropout_rate1':
0.014492194696432759, 'mid_units': 200.0, 'filters': 32, 'kernel_size': 3,
'strides': 2}. Best is trial 1 with value: 0.6700000166893005.
100/100 [==============================] - 18s 167ms/step - loss: 0.8435 -
accuracy: 0.5600 - val_loss: 0.6537 - val_accuracy: 0.6600
[I 2021-10-31 13:31:11,492] Trial 8 finished with value:
0.6600000262260437 and parameters: {'optimizer': 'rmsprop', 'num_layers': 5,
'activation': 'relu', 'dropout_rate0': 0.49238462917243947, 'dropout_rate1':
0.006749682377091726, 'mid_units': 200.0, 'filters': 32, 'kernel_size': 3,
'strides': 1}. Best is trial 1 with value: 0.6700000166893005.
100/100 [==============================] - 16s 158ms/step - loss: 0.6941 -
accuracy: 0.5090 - val_loss: 0.6923 - val_accuracy: 0.4940
[I 2021-10-31 13:31:28,309] Trial 9 finished with value:
0.49399998784065247 and parameters: {'optimizer': 'adadelta', 'num_layers': 2,
'activation': 'relu', 'dropout_rate0': 0.18028263462627542, 'dropout_rate1':
0.40114405301463785, 'mid_units': 100.0, 'filters': 32, 'kernel_size': 3,
```

'strides': 2}. Best is trial 1 with value: 0.6700000166893005.

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-21-f4463db52685> in <module>()
      5 #studypik = pickle.load(open('study.pickle', 'rb'))
      6 study.optimize(objective, n_trials = 10, timeout = 60 * 60 * 3,␣
 ↪show_progress_bar=True)
----> 7 print(studypik.best_params)
      8 print(studypik.best_value)
      9 pickle.dump(studypik, open('study.pickle', 'wb'))

NameError: name 'studypik' is not defined
```

[ ]:
```
study = optuna.create_study(direction="maximize", )
study.optimize(objective, n_trials = 10, timeout = 60 * 60 * 3,␣
 ↪show_progress_bar=True)
print(study.best_params)
print(study.best_value)
```

```
[I 2021-10-31 13:33:28,764] A new study created in memory with name:
no-name-4f7005ab-015a-4fe3-a772-92753c5144dc
/usr/local/lib/python3.7/dist-packages/optuna/progress_bar.py:47:
ExperimentalWarning:

Progress bar is experimental (supported from v1.2.0). The interface can change
in the future.


  0%|          | 0/10 [00:00<?, ?it/s]

Epoch 1/5
100/100 [==============================] - 19s 186ms/step - loss: 0.6906 -
accuracy: 0.5480 - val_loss: 0.6855 - val_accuracy: 0.6320
Epoch 2/5
100/100 [==============================] - 18s 183ms/step - loss: 0.6852 -
accuracy: 0.5540 - val_loss: 0.6794 - val_accuracy: 0.6180
Epoch 3/5
100/100 [==============================] - 18s 182ms/step - loss: 0.6731 -
accuracy: 0.6130 - val_loss: 0.6787 - val_accuracy: 0.5360
Epoch 4/5
100/100 [==============================] - 18s 183ms/step - loss: 0.6542 -
accuracy: 0.6030 - val_loss: 0.6498 - val_accuracy: 0.6100
Epoch 5/5
100/100 [==============================] - 18s 181ms/step - loss: 0.6419 -
accuracy: 0.6370 - val_loss: 0.6408 - val_accuracy: 0.6420
[I 2021-10-31 13:35:06,407] Trial 0 finished with value:
0.6420000195503235 and parameters: {'optimizer': 'adagrad', 'num_layers': 5,
```

```
'activation': 'relu', 'dropout_rate0': 0.15440207381298776, 'dropout_rate1':
0.3946011403479582, 'mid_units': 300.0, 'filters': 64, 'kernel_size': 3,
'strides': 2}. Best is trial 0 with value: 0.6420000195503235.
Epoch 1/5
100/100 [==============================] - 26s 223ms/step - loss: 0.6917 -
accuracy: 0.5360 - val_loss: 0.6913 - val_accuracy: 0.5040
Epoch 2/5
100/100 [==============================] - 22s 223ms/step - loss: 0.6783 -
accuracy: 0.5700 - val_loss: 0.6623 - val_accuracy: 0.6460
Epoch 3/5
100/100 [==============================] - 22s 222ms/step - loss: 0.6669 -
accuracy: 0.6200 - val_loss: 0.6478 - val_accuracy: 0.6420
Epoch 4/5
100/100 [==============================] - 22s 221ms/step - loss: 0.6435 -
accuracy: 0.6240 - val_loss: 0.6571 - val_accuracy: 0.6080
Epoch 5/5
100/100 [==============================] - 22s 220ms/step - loss: 0.6391 -
accuracy: 0.6400 - val_loss: 0.6108 - val_accuracy: 0.6800
[I 2021-10-31 13:38:15,804] Trial 1 finished with value:
0.6800000071525574 and parameters: {'optimizer': 'adagrad', 'num_layers': 4,
'activation': 'relu', 'dropout_rate0': 0.44497538687245153, 'dropout_rate1':
0.403020754907426, 'mid_units': 300.0, 'filters': 128, 'kernel_size': 3,
'strides': 2}. Best is trial 1 with value: 0.6800000071525574.
Epoch 1/5
100/100 [==============================] - 34s 311ms/step - loss: 0.6928 -
accuracy: 0.5190 - val_loss: 0.6922 - val_accuracy: 0.5120
Epoch 2/5
100/100 [==============================] - 31s 308ms/step - loss: 0.6913 -
accuracy: 0.5450 - val_loss: 0.6922 - val_accuracy: 0.4860
Epoch 3/5
100/100 [==============================] - 31s 308ms/step - loss: 0.6893 -
accuracy: 0.5700 - val_loss: 0.6886 - val_accuracy: 0.5800
Epoch 4/5
100/100 [==============================] - 32s 317ms/step - loss: 0.6833 -
accuracy: 0.5850 - val_loss: 0.6765 - val_accuracy: 0.6360
Epoch 5/5
100/100 [==============================] - 31s 308ms/step - loss: 0.6683 -
accuracy: 0.5910 - val_loss: 0.6603 - val_accuracy: 0.6320
[I 2021-10-31 13:41:13,413] Trial 2 finished with value:
0.6320000290870667 and parameters: {'optimizer': 'adagrad', 'num_layers': 7,
'activation': 'relu', 'dropout_rate0': 0.10574849589901586, 'dropout_rate1':
0.41902846877861305, 'mid_units': 300.0, 'filters': 128, 'kernel_size': 3,
'strides': 2}. Best is trial 1 with value: 0.6800000071525574.
Epoch 1/5
100/100 [==============================] - 16s 159ms/step - loss: 0.6974 -
accuracy: 0.4890 - val_loss: 0.6933 - val_accuracy: 0.5260
Epoch 2/5
100/100 [==============================] - 16s 161ms/step - loss: 0.6939 -
```

```
accuracy: 0.5240 - val_loss: 0.6920 - val_accuracy: 0.5540
Epoch 3/5
100/100 [==============================] - 16s 158ms/step - loss: 0.6959 -
accuracy: 0.5030 - val_loss: 0.6881 - val_accuracy: 0.5580
Epoch 4/5
100/100 [==============================] - 16s 157ms/step - loss: 0.6872 -
accuracy: 0.5560 - val_loss: 0.6870 - val_accuracy: 0.5680
Epoch 5/5
100/100 [==============================] - 16s 156ms/step - loss: 0.6877 -
accuracy: 0.5490 - val_loss: 0.6867 - val_accuracy: 0.5820
[I 2021-10-31 13:42:51,866] Trial 3 finished with value:
0.5820000171661377 and parameters: {'optimizer': 'adadelta', 'num_layers': 2,
'activation': 'relu', 'dropout_rate0': 0.3385236238961555, 'dropout_rate1':
0.3040985969239738, 'mid_units': 100.0, 'filters': 16, 'kernel_size': 3,
'strides': 1}. Best is trial 1 with value: 0.6800000071525574.
Epoch 1/5
100/100 [==============================] - 18s 168ms/step - loss: 0.7055 -
accuracy: 0.5030 - val_loss: 0.6708 - val_accuracy: 0.5160
Epoch 2/5
100/100 [==============================] - 16s 164ms/step - loss: 0.6135 -
accuracy: 0.6810 - val_loss: 0.5086 - val_accuracy: 0.7460
Epoch 3/5
100/100 [==============================] - 16s 163ms/step - loss: 0.5094 -
accuracy: 0.7620 - val_loss: 0.4488 - val_accuracy: 0.8080
Epoch 4/5
100/100 [==============================] - 16s 165ms/step - loss: 0.4797 -
accuracy: 0.7950 - val_loss: 0.4229 - val_accuracy: 0.8180
Epoch 5/5
100/100 [==============================] - 17s 165ms/step - loss: 0.4476 -
accuracy: 0.8120 - val_loss: 0.7007 - val_accuracy: 0.6040
[I 2021-10-31 13:44:16,156] Trial 4 finished with value:
0.6039999723434448 and parameters: {'optimizer': 'rmsprop', 'num_layers': 7,
'activation': 'relu', 'dropout_rate0': 0.49644395894413873, 'dropout_rate1':
0.1535162204490153, 'mid_units': 200.0, 'filters': 16, 'kernel_size': 3,
'strides': 1}. Best is trial 1 with value: 0.6800000071525574.
Epoch 1/5
100/100 [==============================] - 18s 174ms/step - loss: 0.6892 -
accuracy: 0.5620 - val_loss: 0.6748 - val_accuracy: 0.5600
Epoch 2/5
100/100 [==============================] - 17s 173ms/step - loss: 0.6716 -
accuracy: 0.6180 - val_loss: 0.6141 - val_accuracy: 0.6860
Epoch 3/5
100/100 [==============================] - 17s 173ms/step - loss: 0.6019 -
accuracy: 0.6880 - val_loss: 0.5594 - val_accuracy: 0.7180
Epoch 4/5
100/100 [==============================] - 17s 174ms/step - loss: 0.5672 -
accuracy: 0.7140 - val_loss: 0.5455 - val_accuracy: 0.7320
Epoch 5/5
```

```
100/100 [==============================] - 18s 176ms/step - loss: 0.5361 -
accuracy: 0.7350 - val_loss: 0.4644 - val_accuracy: 0.7800
[I 2021-10-31 13:45:56,903] Trial 5 finished with value:
0.7799999713897705 and parameters: {'optimizer': 'adamax', 'num_layers': 7,
'activation': 'relu', 'dropout_rate0': 0.1414120827493418, 'dropout_rate1':
0.4441663934981933, 'mid_units': 100.0, 'filters': 32, 'kernel_size': 3,
'strides': 2}. Best is trial 5 with value: 0.7799999713897705.
Epoch 1/5
100/100 [==============================] - 18s 172ms/step - loss: 0.6896 -
accuracy: 0.5500 - val_loss: 0.6764 - val_accuracy: 0.6560
Epoch 2/5
100/100 [==============================] - 17s 170ms/step - loss: 0.6705 -
accuracy: 0.5990 - val_loss: 0.6570 - val_accuracy: 0.6440
Epoch 3/5
100/100 [==============================] - 17s 166ms/step - loss: 0.6546 -
accuracy: 0.6200 - val_loss: 0.6586 - val_accuracy: 0.6000
Epoch 4/5
100/100 [==============================] - 17s 166ms/step - loss: 0.6275 -
accuracy: 0.6670 - val_loss: 0.5969 - val_accuracy: 0.6900
Epoch 5/5
100/100 [==============================] - 17s 167ms/step - loss: 0.6341 -
accuracy: 0.6250 - val_loss: 0.6041 - val_accuracy: 0.6820
[I 2021-10-31 13:47:37,215] Trial 6 finished with value:
0.6819999814033508 and parameters: {'optimizer': 'adagrad', 'num_layers': 4,
'activation': 'relu', 'dropout_rate0': 0.4475620563107447, 'dropout_rate1':
0.3007601379030714, 'mid_units': 300.0, 'filters': 32, 'kernel_size': 3,
'strides': 1}. Best is trial 5 with value: 0.7799999713897705.
Epoch 1/5
100/100 [==============================] - 19s 188ms/step - loss: 0.6909 -
accuracy: 0.5350 - val_loss: 0.6859 - val_accuracy: 0.5540
Epoch 2/5
100/100 [==============================] - 18s 184ms/step - loss: 0.6823 -
accuracy: 0.5710 - val_loss: 0.6666 - val_accuracy: 0.6460
Epoch 3/5
100/100 [==============================] - 19s 185ms/step - loss: 0.6710 -
accuracy: 0.5820 - val_loss: 0.6537 - val_accuracy: 0.6200
Epoch 4/5
100/100 [==============================] - 18s 181ms/step - loss: 0.6570 -
accuracy: 0.6200 - val_loss: 0.6536 - val_accuracy: 0.5960
Epoch 5/5
100/100 [==============================] - 18s 180ms/step - loss: 0.6510 -
accuracy: 0.6410 - val_loss: 0.6270 - val_accuracy: 0.6540
[I 2021-10-31 13:49:13,076] Trial 7 finished with value:
0.6539999842643738 and parameters: {'optimizer': 'adagrad', 'num_layers': 4,
'activation': 'relu', 'dropout_rate0': 0.2256719112217388, 'dropout_rate1':
0.45643380402886063, 'mid_units': 300.0, 'filters': 64, 'kernel_size': 3,
'strides': 1}. Best is trial 5 with value: 0.7799999713897705.
Epoch 1/5
```

```
100/100 [==============================] - 18s 176ms/step - loss: 0.6936 -
accuracy: 0.5080 - val_loss: 0.6952 - val_accuracy: 0.4860
Epoch 2/5
100/100 [==============================] - 18s 175ms/step - loss: 0.6927 -
accuracy: 0.5030 - val_loss: 0.6909 - val_accuracy: 0.5300
Epoch 3/5
100/100 [==============================] - 17s 173ms/step - loss: 0.6904 -
accuracy: 0.5370 - val_loss: 0.6917 - val_accuracy: 0.5180
Epoch 4/5
100/100 [==============================] - 17s 172ms/step - loss: 0.6918 -
accuracy: 0.5140 - val_loss: 0.6918 - val_accuracy: 0.4980
Epoch 5/5
100/100 [==============================] - 17s 171ms/step - loss: 0.6890 -
accuracy: 0.5540 - val_loss: 0.6913 - val_accuracy: 0.5200
[I 2021-10-31 13:50:53,707] Trial 8 finished with value:
0.5199999809265137 and parameters: {'optimizer': 'adadelta', 'num_layers': 7,
'activation': 'relu', 'dropout_rate0': 0.15795313983038067, 'dropout_rate1':
0.3708922576280311, 'mid_units': 200.0, 'filters': 32, 'kernel_size': 3,
'strides': 2}. Best is trial 5 with value: 0.7799999713897705.
Epoch 1/5
100/100 [==============================] - 19s 183ms/step - loss: 1.1202 -
accuracy: 0.5830 - val_loss: 0.5884 - val_accuracy: 0.7480
Epoch 2/5
100/100 [==============================] - 18s 177ms/step - loss: 0.5767 -
accuracy: 0.7030 - val_loss: 0.5959 - val_accuracy: 0.7120
Epoch 3/5
100/100 [==============================] - 18s 179ms/step - loss: 0.5101 -
accuracy: 0.7640 - val_loss: 0.5140 - val_accuracy: 0.8020
Epoch 4/5
100/100 [==============================] - 18s 178ms/step - loss: 0.5006 -
accuracy: 0.7680 - val_loss: 0.4608 - val_accuracy: 0.7780
Epoch 5/5
100/100 [==============================] - 18s 177ms/step - loss: 0.4943 -
accuracy: 0.7680 - val_loss: 0.5080 - val_accuracy: 0.7700
[I 2021-10-31 13:52:36,942] Trial 9 finished with value:
0.7699999809265137 and parameters: {'optimizer': 'adam', 'num_layers': 1,
'activation': 'relu', 'dropout_rate0': 0.13128267203628347, 'dropout_rate1':
0.360700887587538, 'mid_units': 200.0, 'filters': 128, 'kernel_size': 3,
'strides': 1}. Best is trial 5 with value: 0.7799999713897705.
{'optimizer': 'adamax', 'num_layers': 7, 'activation': 'relu', 'dropout_rate0':
0.1414120827493418, 'dropout_rate1': 0.4441663934981933, 'mid_units': 100.0,
'filters': 32, 'kernel_size': 3, 'strides': 2}
0.7799999713897705
```

[ ]:
```
print(study.best_params)
print(study.best_value)
```

[ ]:
```
fig = optuna.visualization.plot_optimization_history(study)
fig.show()
```

[ ]:
```
fig = optuna.visualization.plot_param_importances(study)
fig.show()
```

[ ]:
```
print(studypik.best_params)
print(studypik.best_value)
pickle.dump(studypik, open('study.pickle', 'wb'))
```

```
{'optimizer': 'adam', 'num_layers': 2, 'activation': 'linear', 'dropout_rate0':
0.19368366673208176, 'dropout_rate1': 0.3373004012393455, 'mid_units': 100.0,
'filters': 32, 'kernel_size': 3, 'strides': 2}
0.8974999785423279
```

[ ]:
```
print("Number of finished trials: {}".format(len(study.trials)))

print("Best trial:")
trial = study.best_trial

print("  Value: {}".format(trial.value))

print("  Params: ")
for key, value in trial.params.items():
  print("    {}: {}".format(key, value))
```

```
Number of finished trials: 5
Best trial:
  Value: 0.8725000023841858
  Params:
    optimizer: adam
    num_layers: 6
    activation: linear
    dropout_rate0: 0.3845052526638556
    dropout_rate1: 0.2421332530661529
    mid_units: 100.0
    filters: 16
    kernel_size: 3
    strides: 2
```

[ ]:
```
import pickle

studypik = pickle.load(open('study.pickle', 'rb'))
print(studypik.best_params)
print(studypik.best_value)
pickle.dump(studypik, open('study.pickle', 'wb'))
```

[4]:
```
!pip install pyyaml h5py
```

```
Requirement already satisfied: pyyaml in /usr/local/lib/python3.7/dist-packages
(3.13)
Requirement already satisfied: h5py in /usr/local/lib/python3.7/dist-packages
(3.1.0)
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/dist-
packages (from h5py) (1.19.5)
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-
packages (from h5py) (1.5.2)
```

[5]:
```
import os

import tensorflow as tf
from tensorflow import keras

print(tf.version.VERSION)
```

```
2.6.0
```

[6]:
```
new_model = tf.keras.models.load_model('/content/drive/MyDrive/optunam.h5')

# Check its architecture
new_model.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_36 (Conv2D)           (None, 198, 198, 64)      1792
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 99, 99, 64)        0
_________________________________________________________________
conv2d_37 (Conv2D)           (None, 97, 97, 64)        36928
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 95, 95, 64)        36928
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 93, 93, 64)        36928
_________________________________________________________________
conv2d_40 (Conv2D)           (None, 91, 91, 64)        36928
_________________________________________________________________
conv2d_41 (Conv2D)           (None, 89, 89, 64)        36928
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 44, 44, 64)        0
_________________________________________________________________
dropout_16 (Dropout)         (None, 44, 44, 64)        0
_________________________________________________________________
flatten_8 (Flatten)          (None, 123904)            0
_________________________________________________________________
dense_16 (Dense)             (None, 300)               37171500
_________________________________________________________________
```

```
dropout_17 (Dropout)         (None, 300)               0
_________________________________________________________________
dense_17 (Dense)             (None, 1)                 301
=================================================================
Total params: 37,358,233
Trainable params: 37,358,233
Non-trainable params: 0
_________________________________________________________________
```

[ ]:
```
import os


model = tf.keras.models.load_model("/content/trialmodel_0.9764999747276306.h5")
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
<ipython-input-7-7c7ae5ed1699> in <module>()
      1 import os
      2
----> 3 model = tf.keras.models.load_model("/content/trialmodel_0.976499974727630
 ↪h5")

/usr/local/lib/python3.7/dist-packages/keras/saving/save.py in␣
 ↪load_model(filepath, custom_objects, compile, options)
    199             (isinstance(filepath, h5py.File) or h5py.is_hdf5(filepath))):
    200           return hdf5_format.load_model_from_hdf5(filepath, custom_objects,
--> 201                                                   compile)
    202
    203         filepath = path_to_string(filepath)

/usr/local/lib/python3.7/dist-packages/keras/saving/hdf5_format.py in␣
 ↪load_model_from_hdf5(filepath, custom_objects, compile)
    165   opened_new_file = not isinstance(filepath, h5py.File)
    166   if opened_new_file:
--> 167     f = h5py.File(filepath, mode='r')
    168   else:
    169     f = filepath

/usr/local/lib/python3.7/dist-packages/h5py/_hl/files.py in __init__(self, name,␣
 ↪mode, driver, libver, userblock_size, swmr, rdcc_nslots, rdcc_nbytes, rdcc_w0,
 ↪track_order, fs_strategy, fs_persist, fs_threshold, **kwds)
    425                                fapl,␣
 ↪fcpl=make_fcpl(track_order=track_order, fs_strategy=fs_strategy,
    426                                fs_persist=fs_persist,␣
 ↪fs_threshold=fs_threshold),
--> 427                                swmr=swmr)
    428
```

```
    429             if isinstance(libver, tuple):

/usr/local/lib/python3.7/dist-packages/h5py/_hl/files.py in make_fid(name, mode,␣
 ↪userblock_size, fapl, fcpl, swmr)
    188         if swmr and swmr_support:
    189             flags |= h5f.ACC_SWMR_READ
--> 190         fid = h5f.open(name, flags, fapl=fapl)
    191     elif mode == 'r+':
    192         fid = h5f.open(name, h5f.ACC_RDWR, fapl=fapl)

h5py/_objects.pyx in h5py._objects.with_phil.wrapper()

h5py/_objects.pyx in h5py._objects.with_phil.wrapper()

h5py/h5f.pyx in h5py.h5f.open()

OSError: Unable to open file (truncated file: eof = 14680064, sblock->base_addr =␣
 ↪0, stored_eof = 448379760)
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: