Computing Science and Mathematics
University of Stirling

Using Machine Learning to Identify Fake Images

Cameron Gray

2411977

Supervisor: Dr Deepayan Bhowmik

Submitted in partial fulfilment of the requirements for the degree of
B.Sc. in Computing Science

2018/19

Abstract

This project sets out to tackle the problem of fake images. Any image can be edited with ease and this creates a widespread problem of misinformation. There are many ways to doctor an image but one of the most common types of forgery is copy-move forgery. This is when one section of an image is copied and moved to another portion of the image. It is a common technique used to edit people and objects out of images. This kind of forgery is what this project focuses on.

There are many ways in which forgeries can be detected but part of the issue is that these techniques are very complex and time consuming. The goal of this project is to take one of these techniques and make it available to everyday people in an easy and timely manner so that anyone can get verification on the legitimacy of an image.

By following the paper "Framework for Image Forgery Detection and Classification Using Machine Learning" a Python program was developed to detect copy-move forgery. This Python program was then deployed via a website to make it easy to access and use. There was a large emphasis on finding the best model to do the classifying so a substantial amount of time was spent optimising the classifier to make sure that the end product was accurate.

The main objective of creating this tool that anyone can use was achieved. The classifier that was used was optimised well and managed to get an accuracy of around 90% when classifying images. The Python program was also optimised in the interest of time so that the user would not have to wait a long time for the results. This project provides a useful tool that anyone can use to help them have confidence in what they are looking at.

Attestation

I am aware of the university's current policy on plagiarism and confirm that this work is original and my own. Any diagrams, images, text or code that have been used throughout the project have been acknowledged and cited. All the work undertaken at the University and within the project timeline.

Signature: **Date:**

Contents

List of Figures

Chapter 1

Introduction

1.1 Background and Context

The ability to convincingly edit an image to suit a narrative is one available to everyone, from governments, to private companies, to newspapers, to everyday people. With the availability of powerful software such as Adobe Photoshop[1] and GNU Image Manipulation Program[2] anyone can change an image with surprising ease. When looking at images on social media, blogs and news sites there is increasing concern that what we are looking at is not real. Especially when it comes to news, people are sceptical of just about any image. People want to have confidence in what they are looking at and that is what this project is trying to help with.

1.2 Scope and Objectives

The primary objective of this project is to give a user a certain amount of verification when dealing with images that could have been tampered with. Due to the problems that the world now faces with fake news, the ability to have confidence in what you are looking at is invaluable.

Using the paper [11] the project will try to make available this technology to every day people. The algorithm that will be created to classify a given image will be deployed via a web page to make it easily accessible to everyone. It will not be perfect at telling if an image has been edited and there will be limitations to its abilities (e.g. depending on the training data for the machine learning aspect) but it can help add to the user's decision on whether or not what they are looking at is real.

The solution should be able to handle any image that is a common format (.jpeg, .png etc.) regardless of the dimensions. When an image is given to the program a confidence percentage should be returned to the user, telling them how likely it is that the image has been tampered with.

After the classification has been shown to the user their image will be displayed back to them

[1] Adobe Photoshop - https://www.adobe.com/products/photoshop.html

[2] GNU Image Manipulation Program - https://www.gimp.org/

alongside their image after the different stages of processing it goes through. This could help highlight tampering that the algorithm does not see. For example, one of the stages is contrast enhancement and comparing the original to the enhanced image could immediately make tampering obvious.

The program that will do the classification will be written in Python and will use libraries to identify if the image has been tampered with. The website UI (User Interface) should be lightweight and clean to remove ambiguity from the interaction. Navigation and data entry should be as clear and straightforward as possible. The site will then run the Python code server side and return the results to the user through PHP.

The end result should be a powerful, useful tool that anyone can access on their computer to give them confidence in what they are looking at and the ability to know whether or not to trust a source.

1.3 Use Case Scenario

The way that a user will be able to get verification for an image by using the software is as follows: firstly, the user connects to the web address for the project. The user will select the "Classification" link and then click on the button to upload an image. They will select the image they want to upload. Once the upload is completed the user will be presented with the classification (real or fake) along with how confident the classifier is in the classification. The program to produce the confidence score will be run on the sever side and then displayed to the user via the website. The user's image after the different stages of pre-processing will also be displayed so the user can look for tampering themselves. If the image uploaded is invalid then the user will be instructed that the upload has failed and then given the option to upload another image.

Once the user has uploaded an image and received a result they will be able to click back to the "Classification" page and do it again. When it comes to the UI it will be as simple and as straightforward as possible. The less ambiguity, the lower the chance of incorrect input and confusion. To achieve this the "Classification" page will have two buttons (one to upload and one to start the classification) and instructions. The other pages will follow suit and minimise the options for input. The goal is to make it easy for the user to get an idea if the image has been tampered with.

1.4 Achievements

The main achievement of this project is a fully functioning website which can classify a given image as either real or fake. The type of doctoring that can be identified is copy-move forgery where part of an image is copied and placed in another part of the image. A user can go to the site and get an idea if an image they are looking at is real or fake by uploading it. They can also look at their image through different filters which may highlight something that the algorithm misses.

The classifier that was used, a Multi-layer Perceptron, managed to reach a 90% accuracy with the

data set used and can accurately identify if part of an image has been copied and move to another part of the image. This includes if the part that has been copied has been rotated and scaled.

The final site that was used to host the algorithm has an easy and straightforward interface which makes getting the result simple for the user to get. The goal of the project to give the user confidence in what they are looking at has been achieved by providing an uncomplicated tool which provides a fast and accurate result.

1.5 Overview of Dissertation

The structure of this dissertation is as follows: there is a background of the problem, a state of the art analysis on the current works in the field, a work methodology, the results of the work and a conclusion. Each goes into depth on their particular area.

The background covers the issue of fake images and why they can be such a problem. There is a look at the history of image forgery and how it has developed over time. The chapter also looks at the political, financial and social implications of the creation and use of fake images around the globe.

The state of the art analysis looks at the different techniques that are out there to identify whether or not tampering has occurred in an image. The five categories of techniques used to identify forgeries are pixel based, geometric based, physical based, camera based and format based. It splits the different pieces of work into the kind of detection that is used and looks at the impact each paper had on the field.

The next chapter is the work methodology which is split up into the project requirements, the design and the implementation. It goes through what is needed from the project and then how a solution was designed and created with that in mind. From this chapter it should be easy to reproduce the work and improve upon it.

The penultimate chapter is the results from the project. It details how the most accurate model was found, the effects of each of the processing stages used on the images and the classifier that was used in the project. There is also a section on the final product i.e. the website which deploys the algorithm used.

Finally, there is the conclusion chapter which brings together the whole dissertation. It covers what was achieved and what went well with the project. There are also details on points of improvement that could have been made over the course of the project. Lastly, the conclusion covers the contribution that has been made by this piece of work and the future work to be done on this area.

Chapter 2

Background

2.1 Fake Images

2.1.1 What is a fake image?

A fake image is an image which has been altered to present something else. Up until recently people have, in general, been quite trusting of photos but with powerful software such as Adobe Photoshop and GNU Image Manipulation Program anyone with a computer can doctor a photo in a convincing manner. Fake images can spread false messages and can have quite a bit of influence over what people think. This chapter will also go into detail on examples of these doctored images and their political, social and economic implications.

Examples

Figure 2.1 shows two photos side by side of a model - one before the edit and one after. The most obvious difference between the two photos is that the legs, waist, wrist and face are all thinner. In both photos the model is attractive but the "after" photo is seen as an unrealistic bar to be set for the appearance of women. Images such as these contribute to people's dissatisfaction with their body - especially with women. They can also contribute to cases of eating disorders but they are not necessarily the main reason [12].

Figure 2.1: Model Doctored Photo [1]

Figure 2.2 is a comical edit and shows President Donald Trump as the Queen of England. As photo editing software is so readily available to the general public a common use for it is to create ridiculous images. This image is obviously fake but it does give an idea of how convincing an edited image can be.

Figure 2.2: Humorous Photo Edit [2]

The example in (*figure 2.3*) shows the before and after of the world leaders who were meeting in Paris after the Charlie Hebdo attacks in 2015. The meeting was to show strength with the grieving French people and was a positive display from the world leaders attending. However the other part of the figure shows the front page of an Israeli newspaper with all the female world leaders who attended edited out. This is because the paper did not want to show women in a position of power or leadership. The image is designed to deliberately discriminate against women and mislead the readers of the newspaper.

Figure 2.3: Edited World Leaders - 2015 [3]

2.1.2 Why do people create fake images?

There are multiple reasons why someone might doctor an image but essentially the main purpose is to present an alternative message than that of the original photo. Most of the time editing an image means that there is something that the editor does not like about the original. In *figure 2.1* the editor wants the model to be skinnier and in *figure 2.3* the editor does not want women to be seen in a position of leadership. In both cases the motivation behind changing the image is negative and presents a false narrative. The intent behind *figure 2.2* is primarily to make someone laugh but it could also be argued that the main intention behind it, like a lot of edited pictures of politicians, is to mock Donald Trump. However, it is probable that this is now part of the ridiculing process that all major politicians go through and is perhaps the more modern version of a slanderous cartoon in a newspaper. For the most part is seems that if an image is being doctored it is probably with malicious intent and is likely to have a negative message for whoever views it.

Another reason that someone might edit an image is in relation to a crime. They might edit themselves out of security footage or edit a photo to create some sort of alibi. The ease at which someone can change what a photo says calls into question any images that are used in evidence.

Doctored photos may also be used to stir up controversy before important events. For example, say a political nominee could edit a photo of the opposition in a way that would enrage voters and they release it the day of a vote, that could have disastrous political consequences. By the time that

word spreads that the image is fake the damage will have been done and people may have voted on misinformation.

Figure 2.4 is a good example of the political implications of doctoring an image. The edited image from 1950 depicts Senator Millard Tydings talking to the leader of the American Communist Party Earl Browder. This meeting never happened but the image was created to make voters think that the Senator sympathized with the communist party and the ideas of communism. Tydings then went on to lose the following election.

Finally, an additional example of why someone might edit an image is seen in *figure 2.5*. The pictures are from 2008 and the bottom one was released by the Iranian government to show a successful launch of four missiles. A closer look at the missile second from the right reveals that is in fact fake. Comparing the two missile trails of the middle missiles (circled in red) shows that they are the same with the only change being a black dot added to the third one and a the shape of the trail moved around a little. The more obvious, and more damning, edit is the plume at the bottom of the two missiles on the right (circled in blue). A quick comparison shows that one of them has been copy-pasted to the other. Looking at the right side of the plume for both the missiles, it is incredibly easy to spot the distinct pattern that both make proving that the edited job is fairly lazy. Releasing the top photo with a failed launch would make the Iranian military seem weak and their weapons program sub par so that is why they sent out the bottom photo. For a time it worked and it took a while before people caught on. The bottom photo was displayed in many major newspapers and by the time the fake was pointed out The LA Times, Chicago Tribune and The New York Times had all prominently displayed the fake photo. Once the edit is pointed out it seems ridiculous that large scale news operations would use it. However it seems that our initial instinct is to trust images and assume that they are real.

Figure 2.4: Senator Tydings talking to Earl Browder - 1950 [4]

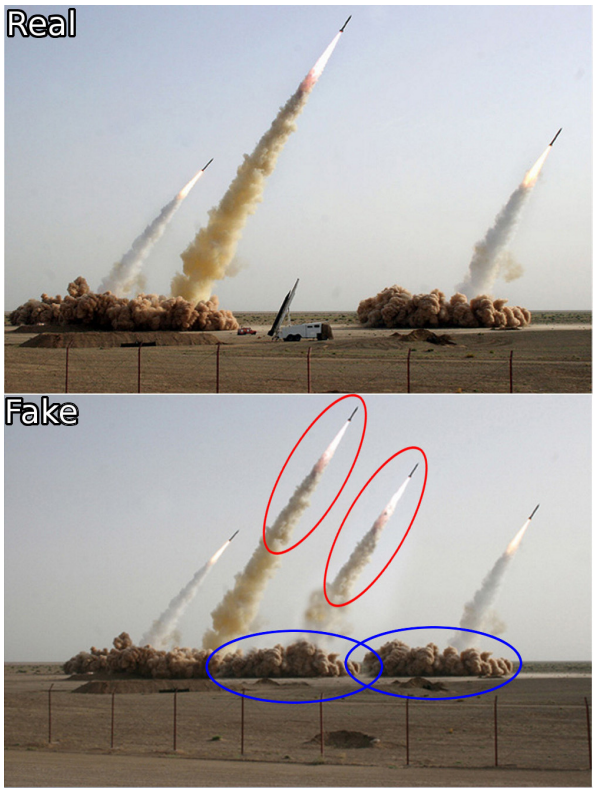

Figure 2.5: Edited Iranian Missile Photo - 2008 [5]

2.1.3 What is the implication of creating fake images?

This next sub section will cover the implication of fake images. Doctoring an image can have damaging effects on political, social and economic levels. The impact can range from minor to devastating.

Political

The most apparent political implication of creating fake images is to sway voters. A situation mentioned earlier was if a doctored image was released on the day of a vote then a whole group of voters could have switched their vote before finding out that the image was fake. Another possible example could be a politician edited into a hunting photo. Hunting for sport can be a sensitive topic as people care a lot about the treatment of animals [13]. This could change the way someone votes in an instant as if they saw an image of a politician next to a dead animal that they supposedly just killed.

Doctored images can also be used to assassinate someone's character and attack them personally. Again, *figure 2.2* is an example of this and is fairly tame in terms of what some people come up with. Politicians are ridiculed more than most and the ability to edit an image just adds to the arsenal of someone launching a personal attack. The goal of editing an image is to draw people away from the actual political issues. The danger of this is that if someone sees a politician constantly mocked then

they start to ignore the actual politics going on. The politician could be being perfectly reasonable but the attack on their character could detract from that.

It would be more difficult to fool as many people now but the history around editing images is rich. Both Stalin and Hitler would remove people from images where they featured together if they no longer wanted to be associated or no longer agreed with them. *Figure 2.6* and *2.7* show examples for both. By editing these photos it allowed these dictators to control the narrative that they wanted people to receive.

The edited photo in *figure 2.8* was used by the Canadian Prime Minister William Lyon Mackenzie King as an election poster. He is seen talking to Queen Elizabeth Bowes-Lyon (Queen Elizabeth II's mother). However the original image also included King George the VI. The reason that King George was removed was because it paints the PM in a more powerful light to have him not there. Part of the reason may have been because King George is taller than the PM. This is a similar situation to how current politicians try to get the left-side advantage in a handshake for media photos. While this alteration isn't on the same level as Stalin and Hitler it is once again trying to control the narrative.

Figure 2.6: Stalin airbrushes out an enemy - 1930 [4]

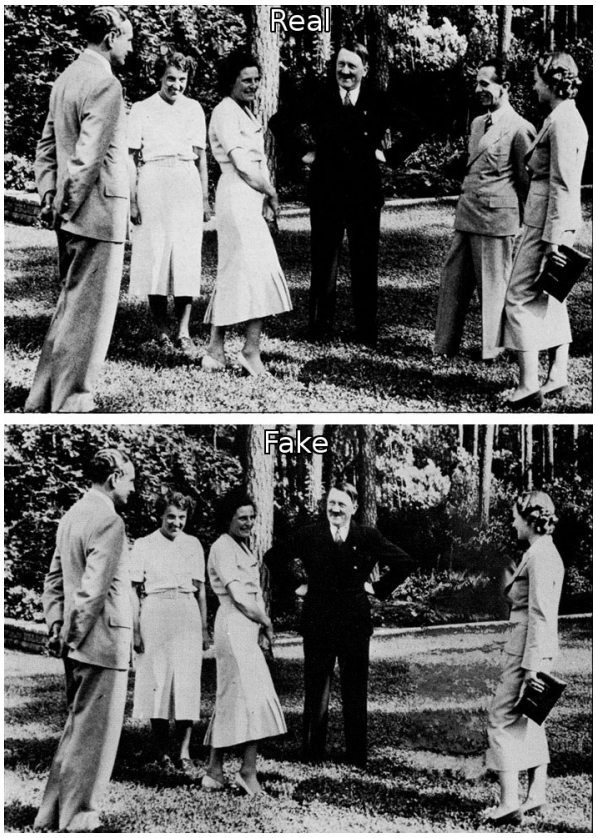

Figure 2.7: Hitler has Joseph Goebbels edited out - 1937 [4]

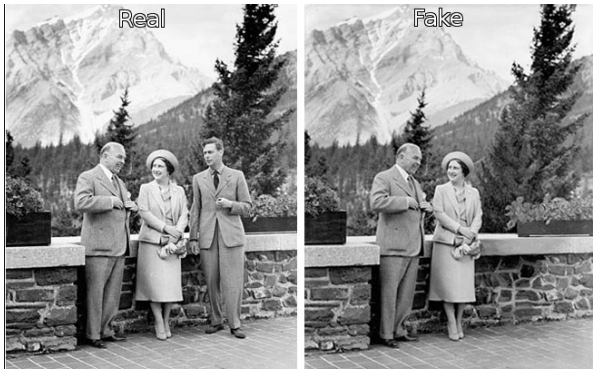

Figure 2.8: Canadian PM William Lyon Mackenzie King removes King George VI - 1939 [4]

Social

The main social implication that comes with editing images is that of body image. An edited model can create an unrealistic standard for what beauty should be. The problem is that this unobtainable standard is viewed as the goal. This leads to people starving themselves in hopes to reach what they see on-line and in magazines. People view the person on the cover as an image of health but to try

and be like that model they have to be unhealthy [12]. More people are beginning to understand that almost all images of models have been edited in some way. However images like these are everywhere and have a lot of negative impact as they reinforce what people think they need to be.

However, not all edits have negative connotations. An example of editing images for positive social effect are those of Walt Disney around Disney resorts. The primary target audience of Disney are children and children are easily susceptible to what they see. This has lead to all the pictures of Walt Disney around the resorts to be edited so that they don't show him smoking. See *figure 2.9* for two examples. At first glance nothing seems off about the photos but looking at Disney's right hand and it is clear that it should be holding a cigarette. Amongst all the trickery and lies that come with editing photos this is an example of good that it can do.

Figure 2.9: Images of Walt Disney with the cigarette taken out [6]

Economic

Like the political and social implications, the economic impact of doctoring a photo can be widespread. For example images in magazines such as *figure 2.1* can lead people, especially teens, to go to extreme lengths to meet what they see in the photos [12]. This, in turn, then has an economic impact in the UK by costing the NHS [14].

Similarly to how fake images can effect political campaigns due to people not checking if an image is real or not, fake images could cause people to riot without thinking or fact checking. *Figure 2.10* is an example of an image used to incite riots. The image is actually from a film called "Aurat Khilona Nahin" but the image was spread through social media in 2017 with captions claiming that groups from the area were attacking Hindu women. The result was then unnecessary destruction of property [7]. This particular image isn't doctored but it gives an idea of the power that a doctored image could have over mass groups of people and the possible economic effect.

Finally, an edited image could have an economic impact on a celebrity who uses a clearly doctored image to promote themselves or on social media. People don't like being lied to and if a celebrity is caught changing their images then it could negatively effect their public image. This could then lead to boycotts of what they are associated with and therefore have an economic impact.

Figure 2.10: Image used to incite Baduria riots - 2015 [7]

2.2 History of fake images

2.2.1 Timeline

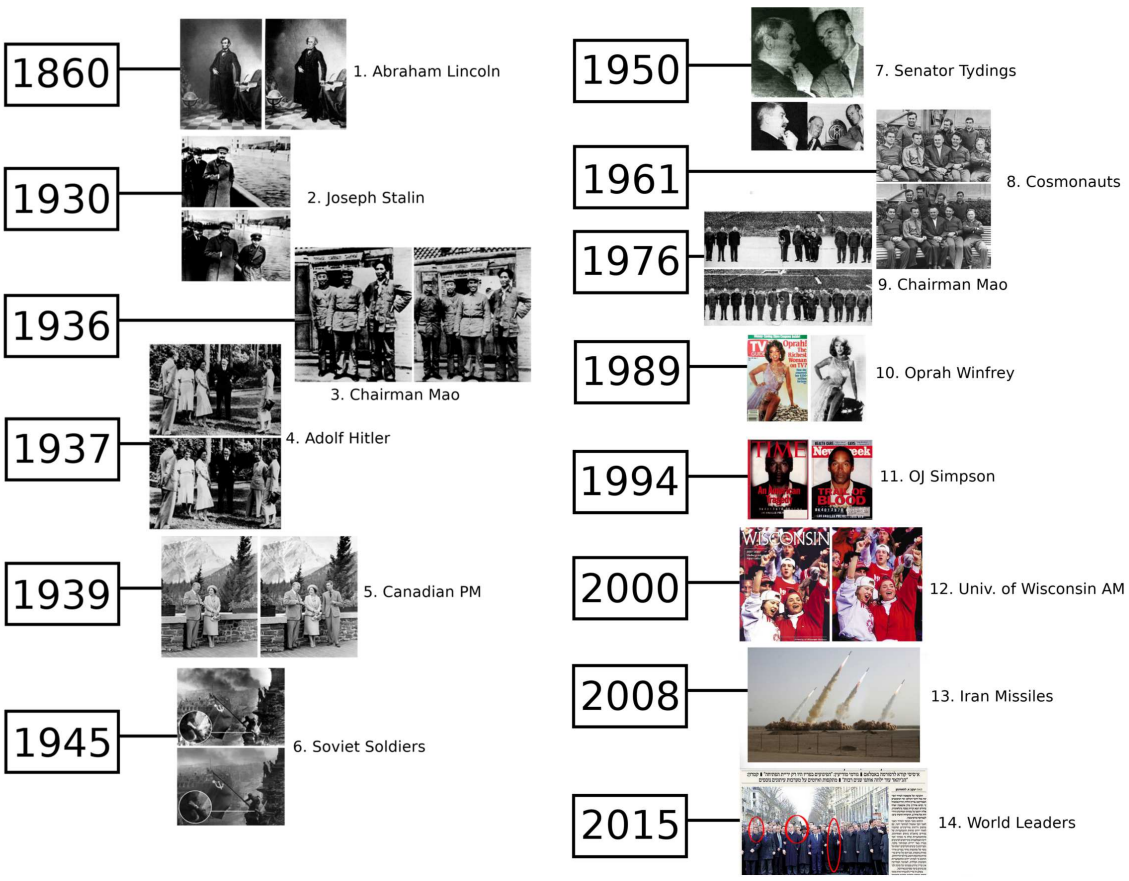

Figure 2.11: A timeline of examples of photo editing through history (images used: [4])

	Name	Year	Description
1.	Abraham Lincoln	1860	The first image is the famous portrait of Abraham Lincoln. The body used in the photo is actually that of a southern politician John Calhoun and Lincoln's head has been added on top of it.
2.	Joseph Stalin	1930	The second image is the same example used earlier (*figure 2.6*).
3.	Chairman Mao	1936	This image is an example of Chairman Mao editing out people similar to the Stalin and Hitler examples. The person edited out was Po Ku (senior leader in the Chinese Communist Party) who Mao no longer wanted to be associated with.
4.	Adolf Hitler	1937	This image is the same example of Hitler editing out Joseph Goebbels shown earlier (*figure 2.7*).
5.	Canadian PM	1939	Again, this is the example used previously of the Canadian PM editing out King George VI (*figure 2.8*).
6.	Soviet Soldiers	1945	This next example shows Soviet soldiers raising the Hammer and Sickle at the top of a German building in WWII. The edit here is on the soldiers wrist. The Russian magazine that published this image didn't want their readers to think that the soldiers had been looting so got rid of the watches on the wrist of the soldier. However the second watch on his wrist is thought to just be a compass. This edit gets rid of the negative interpretation that could happen.

7.	Senator Tydings	1950	This is the example discussed earlier with Senator Tydings *(figure 2.4)*.
8.	Cosmonauts	1961	The image here is of the Russian space team who completed the first orbit of Earth on April 12th, 1961. However due to drunken behaviour the cosmonaut Grigoriy Nelyubov was removed from the picture.
9.	Chairman Mao	1976	This is an image from a memorial service for Chairman Mao and the "Gang of Four" have been removed from the photograph.
10.	Oprah Winfrey	1989	The magazine TV Guide shows Oprah Winfrey on the cover. The head is obviously Oprah's but the body used in the edited image is actually that of Ann-Margaret.
11.	OJ Simpson	1994	This is the famous photo of OJ Simpson after his arrest on the cover of Time Magazine. Clearly, Time magazine have edited it to make it darker and more sinister when compared to the actual once displayed in News Week.
12.	Univ. of Wisconsin AM	2000	The University of Wisconsin at Madison wanted to portray the school as more diverse on the cover of their brochure. Their solution was to edit a black student into the photo amongst all the white students.
13.	Iran Missiles	2008	This image is the Iran missile photo discussed earlier (*figure 2.5*).
14.	World Leaders	2015	This is the world leader doctoring by the Iranian newspaper to remove all the female leaders mentioned previously (*figure 2.3*).

2.2.2 Map of doctored images

Below are maps visualising the occurrences of image manipulation. *Figures 2.12, 2.13* and *2.14* shows the data collected from sources [4] and [15]. Firstly, *figure 2.12* shows all the countries where at least one known case of photo tampering took place. For *figures 2.13* and *2.14* the more occurrences recorded the more red the country is. However the data is extremely weighted towards the USA so *figure 2.14* uses ranges to classify the data. Category one is zero to two occurrences, category two is three to four occurrences, category three is five to nine occurrences and category four is greater than ten occurrences. This gives a clearer picture than before.

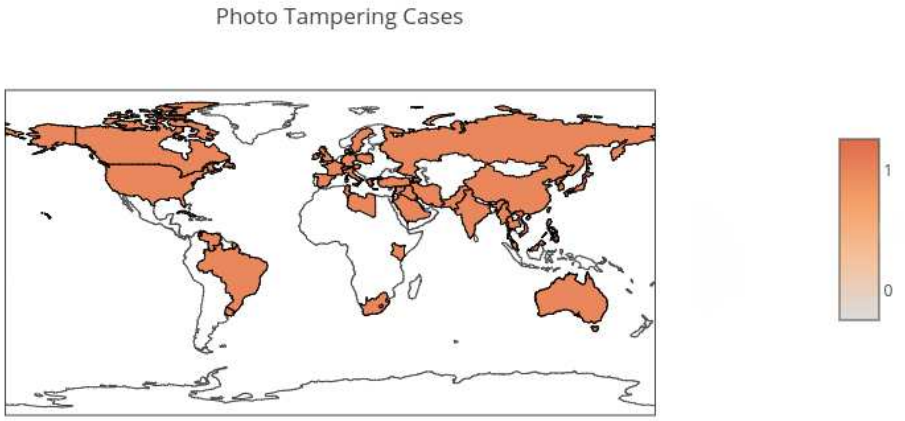

Figure 2.12: Map Data Cases - *created with Plotly [8]*

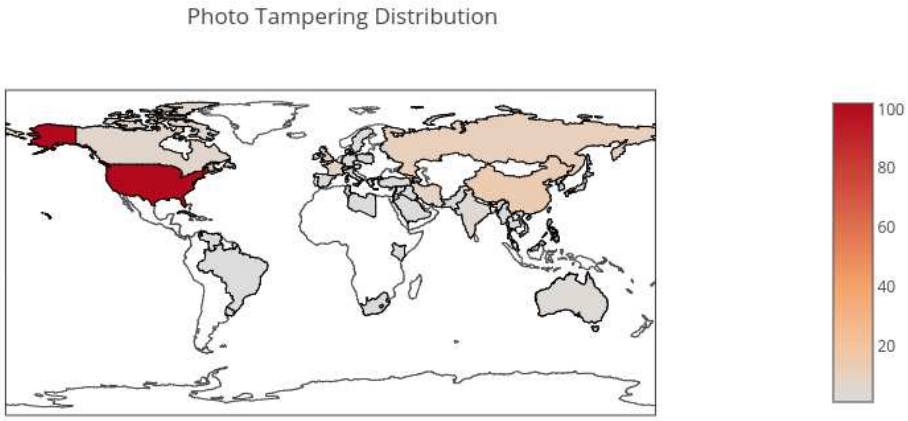

Figure 2.13: Raw Map Data - *created with Plotly [8]*

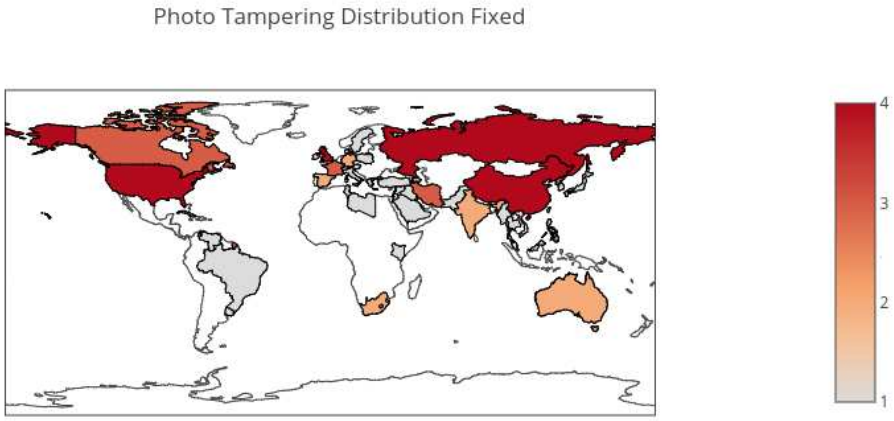

Figure 2.14: Corrected Map Data - *created with Plotly [8]*

Clearly the problem of doctoring images is one that occurs around the world and one which many people run into. While it seems to be the first world countries which have the higher concentration of tampered image cases it could just be because they have easier access to the technology. The more that the technology becomes available the more it will spread to other countries and the bigger the problem becomes.

Chapter 3

State-of-The-Art

3.1 Summary of Current Work

This summary is to cover current works within the field and to look at them with a critical eye. The papers have been split up into different categories within the review. These are based off of the kind of paper and the kind of technique (or techniques) that they discuss. The categories are: overview paper which cover the state of the field, detection using pixel based techniques, detection using geometric based techniques, detection using physical based techniques and detection using camera based techniques.

3.1.1 Overview Papers

In this first paper, "Image Forgery Detection", the author goes through the five categories of techniques used to identify fake images [16]. The categories of techniques covered are: pixel based detection, format based detection, camera based detection, physically based detection and geometric based detection (this state of the art analysis covers all the techniques except format based). For each of these the author goes through the different sub categories within the techniques. For example, in the pixel based category there are the sub categories of cloning, re-sampling, splicing and statistical edits. The paper is finished off by saying what the future could hold in terms of better methods for creating fake images and in turn better techniques for identification.

The overall picture presented by this paper gives the reader a good understanding of the current state of detecting fake images and it goes into quite a lot of detail on the mathematics behind the work. This broad understanding sets the stage for the reader to investigate and further understand more individual techniques.

Presented in "Digital image forgery detection using passive techniques: A survey" is a survey of the current techniques that are used to detect tampering when no previous information about the image is known [17]. This is similar to the first paper in that it is not presenting anything new on the topic

but reviewing the current situation in detecting fake images [16].

Active detection techniques use digital signatures and watermarks to detect tampering whereas passive techniques are based on tampering operations, inconsistencies and intrinsic regularities. One of the problems that they find with the current methods is that of automation. The results still need human interpretation. Current methods also have issues with identifying the area that the tampering took place if the edit was minor i.e. for the algorithms to find the area, a substantial change would have had to been made. There are also limits with camera identification and if the image is of lower quality.

Overall a paper such as this one is useful as it gives the reader a useful overview of the current situation in detecting doctored images. It also does well to highlight the issues with the current solutions and what needs to worked on.

The goal of "An evaluation of popular copy-move forgery detection approaches" is to determine what is the best current technique to detect copy-move forgery [18]. Again, this paper is a survey rather than one describing a specific technique. The authors describe the data set that they use to test the current techniques as "challenging" and "real-world". Part of the issue with data sets is they may contain edits that are really easy for a person to detect. The edits in the data set have to be realistic for the technology to be useful. Firstly, they describe in broad terms how the algorithms used to identify forgeries tend to work. A data set to use for testing had to be created as current benchmarks were not suitable for their needs. Each image added to the data set was edited in a realistic manner to test these algorithms in a more real world scenario.

The result of this study was a better way to test copy-move detection algorithms. This paper fills a gap in the current state of not having a robust way to test the current algorithms. It also helps to identify what algorithms will work the best in certain situations.

3.1.2 Pixel Based Techniques

Presented in "Image Splicing Detection via Camera Response Function Analysis" is a technique to identify images which have been doctored with some sort of splicing[19]. Splicing is when a section from another image is inserted into the image being edited. The paper focuses on looking at the blurs between the edges of objects in the image. Fake blurs are added to images to hide splicing and copy-move operations. Photos that are taken have natural blur and it is by comparing what the blur should be that the images are classified.

The paper goes on to describe training a neural network to use their technique to automatically identify fake images. Presented here is a useful, in-depth technique, that can be used to identify some edited images. The obvious limitation is that it only works with a couple of pixel based techniques but from the results it seems that this method is helpful when it comes to those specific techniques.

"A Statistical Prior for Photo Forensics: Object Removal" describes a way in which to identify the probability that a portion of an image has been removed [20]. They say that this paper will help to

improve the accuracy of other forensic techniques. The paper describes three ways in which an image can be manipulated: removal, addition and modification. In this case they tackle the removal category. The focus is on Adobe Photoshop's content aware feature which replaces an area with similar pixels to those around it but not exactly the same. This is essentially an automatic version of the clone tool in these software packages.

The results of this paper was that they managed to implement the technique but it took a lot of effort, depended on a vast number of factors and the implementation of the software in MATLAB was very inefficient (listed as taking between thirteen and fifty minutes on a 2.7 GHz processor). Work would need to be done on this technique for it to be of much use in this area.

The problem that "An efficient and robust method for detecting copy-move forgery" sets out to tackle is the copy-move detection [21]. Copy-move is when a part of the image is copied and pasted to another part of the same image. It's a common technique to remove someone or something from an image. The part of the image that is copied tends to be the background and it is pasted in front of the part that the editor wants removed.

The results of the paper were very successful. The algorithm that they used could efficiently and correctly detect copy-move forgery even if the tampering was rotated, scaled or highly compressed. This contribution is incredibly valuable due to how robust and successful the algorithm is. The only limitation that they set was the minimum area of tampering which was set at 32x32 pixels so doctoring within that size would not be identified. Compared to a few other methods this one seems to describe a much more helpful and practical way of identifying one of the most common types of forgery.

Finally, the paper "A Framework for Image Forgery Detection and Classification Using Machine Learning" describes a technique that they implemented to identify copy-move forgery [11]. The paper is similar to the previous but goes into more depth as to exactly how they did it and how they processed the images for machine learning.

The paper experimented with using both a Linear SVM and an ANN (Artificial Neural Network) as the machine learning component. While the results with the SVM were fair, they were much more successful with the ANN. The images used all go through the same process. First, their contrast is enhanced and then the pixels are de-noised. Following that the image is clustered and then features are extracted using a GLCM (Grey Level Co-occurrence Matrix). The framework laid out in this paper is what the detection part of this project is based on. In the paper they implement it in MATLAB but it was converted to Python for the purposes of this project.

3.1.3 Geometric Based Techniques

Due to the ease at which a sign or billboard can be edited convincingly, "Detecting photo manipulation on signs and billboards" uses geometric techniques to tell if photos have been doctored [22]. As the text is on a flat surface and, if the font can be identified, then it means that how the text on the sign/billboard should look can be worked out. However when tampering is involved the text will have

a certain amount of distortion. If this distortion can be found then the image has been edited and the sign/billboard is not real. They also identify a technique to detect tampering if the font is not known by trying to find the closest known font to the one in the image.

The results listed in the paper say that their technique has an overall success rate of 95%. With this high of a success rate it makes the geometric based technique presented a valuable asset when it comes to identifying fake images. Again, this paper is documenting a technique for a certain situation and it helps to contribute to the array of techniques that could be used to detect forgeries.

"Exposing photo manipulation with inconsistent reflections", by Hany Farid and James O'Brien, also discusses a geometric technique to use for identifying fake images [23]. The difference in this one is that they are looking at inconsistencies in the reflections in the image. In the physical world reflections follow certain rules and if the ones in the image do not line up then the image is a fake. While making an image look convincing at first glance can be easy, it can be incredibly difficult to make sure that all the reflections line up and follow the rules. When analysing the image certain assumptions have to be made but they specify that the technique makes "minimal assumptions about the scene geometry".

The paper describes how reflective geometry works and then goes on to show how they apply that to verifying images. When looking at images, they look at reflections on water, mirrors, windows or any other reflective surface. The contribution of this paper is similar to the last one as it adds another technique that could be used to identify doctored images. For example, if the previous technique couldn't find an inconsistency in the font maybe this one could find something wrong with a reflection.

3.1.4 Physical Based Techniques

"Exposing Photo Manipulation from Shading and Shadows" is written by the same authors as "Exposing photo manipulation with inconsistent reflections" and is very similar [24][23]. However the technique that is described is a physically based technique rather than geometric based. Rather than looking at reflections this paper looks at inconsistencies in the shading and shadows that are used in the image. Like reflections, light and shadows follow certain rules and if the shading in the image doesn't follow these rules then some kind of tampering has gone on. Once again the authors say that their analysis "makes minimal assumptions".

It is mentioned in the discussion that a talented editor could theoretically get around these methods by following how their techniques work but it would be incredibly time consuming and difficult to do. Even if they did manage to get around it then another technique described in another paper could be used to identify doctoring. This paper adds to the increasing number of ways in which tampering could be found.

3.1.5 Camera Based Techniques

Presented in "Detecting digital image forgeries using sensor pattern noise" is a technique to detect doctored images as long as the camera used to take the image is available or other images taken by the same camera are available [25]. The detection is based on camera pattern noise which is unique to each camera. At first this may seem to be quite limiting but this technique would be very useful when checking CCTV footage as, in this case, the camera model would be known. The experiment used six convincing fakes to test the algorithm. They used different cameras for the different images.

The result were marginally successful - they could reliably identify JPEG images with qualities as low as 70. As a contribution to the area overall it is not necessarily as useful as some of the other techniques presented. It does add to the area by presenting another technique to use for detection. However it does have the limiting factor of the camera type needing to be known or other photos from the same camera accessibly.

Finally, "Digital Image Forensics" describes how every time a camera takes a picture it unintentionally leaves a watermark through sensor noise [26]. This watermark survives with the image through filters and compression. As this is the case it means that it can be used for image forensics and, more specifically, detection of forgeries. It can also help with device identification and device linking which is also helpful for investigations.

The paper goes into detail about device identification and how the sensor affects the pixels. This could then be used in conjunction with the previous paper to identify if an image has been tampered with and is a useful addition to the field.

3.2 What gap will this project fill?

This project will try and make these techniques available in an efficient manner to anyone who wants to know if the image that they are looking at is fake. As has been discussed there are multiple ways to detect forgery but some of the techniques are slow, too complicated for the average person to carry out or limited in some way. By using machine learning and focusing on pixel based techniques I hope to give a user confidence in what they are looking at. The project will make use of current techniques and target images that are used in the spread of fake news giving the power to the user to verify the information that they are receiving.

Chapter 4

Work Methodology

4.1 Requirements

The objective of this project is to create a fast, reliable way for a user to check if an image they are looking at has been tampered with. There are four key goals of this project: speed, ease of access, ease of use and reliability.

The speed element is important as users do not want to be waiting a long time for a result. For example, if the user is reading a news story that may take them five minutes. If they want to verify an image in that story then they do not want to be waiting ten minutes for a result as that would be wasting time. Images can take a long time to process so there is a need in the project to optimise the time element for the user.

Next, the way that the software is deployed is important as it needs to be straightforward to access and use. The user should not have to have lots of technical skill in order to run the software and they also should not have to take out large amounts of time in order to learn how to use it or how it works. It does not matter how good the classifier is if the user cannot get a result.

Finally, the reliability of the classification is paramount. If the classification is wrong then that defeats the whole purpose of the project. The idea is to give the power back to the viewer of an image rather than the person tampering with it. A user may rely on this software to get an idea of the truth so the software has to be correct.

4.2 Design

The project is split up into two elements: the Python element and the web element. The Python element is the engine behind the interface. It does the work to identify if the image is fake or not. The web element is a way to present the results of the Python element so they are easy to understand but also easy to access. The main processing of the training images and creation of the classifier is done in advance so the user only has to wait for their image to be processed and classified. Once the

classifier has been saved then it can be loaded when the user uploads an image rather than having to create it again. *Figure 4.1* describes the overall design of the project.

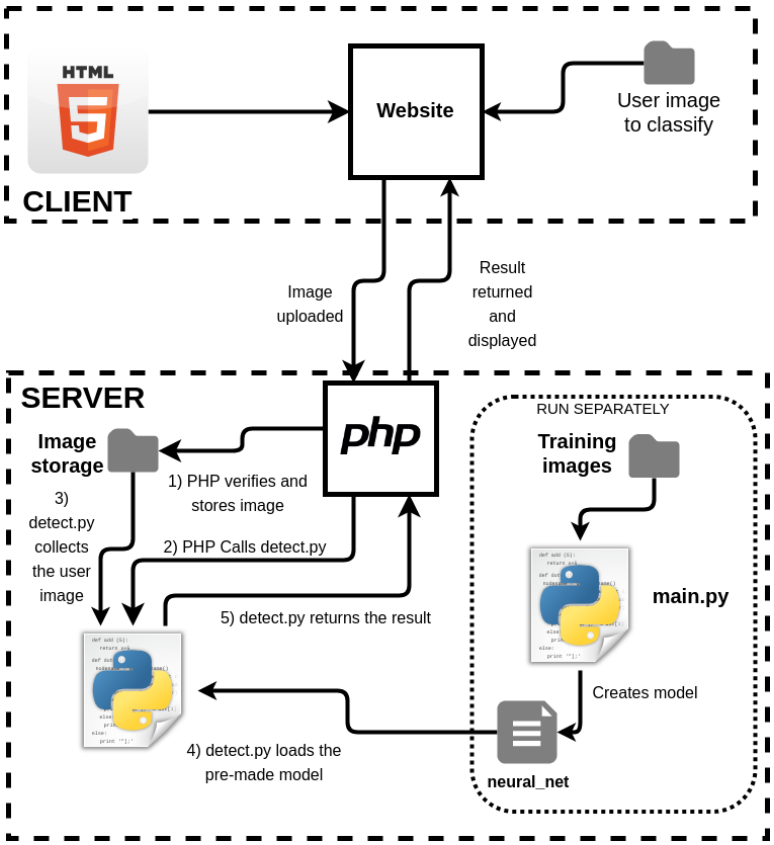

Figure 4.1: Project Structure - *created with draw.io [9]*

The optimisation of the classifier was done by creating a separate Python program to exhaustively create them based off the parameters that could be set and then test them all to see what the best combination was. This was to meet the requirement of reliable results for the user. Overall the web element is to meet the ease of access and use requirements and the Python element is to meet the speed and reliability requirements.

4.2.1 Python Element

Pre-processing

This section is to make some minor adjustments to the image. It emphasises the smaller details in the image that may be obscured by light or noise. The image goes through two stages: histogram equalisation and de-noising.

To create a histogram of an image the total number of pixels need to be plotted for each tonal value. Histogram equalisation is then making said graph more level. The end result is an image with

a higher contrast.

The second part of the pre-processing stage is to apply de-noising to the image. This gets rid of the rough parts of the image by changing pixel values to combat different types of noise. An image with a lot of noise will appear grainy and rough which may hide features of the image. Noise can be caused by many factor such as light disturbances in the image or distortion in the pixels.

Figure 4.2: Example of an image going through the pre-processing stage

Once the image has had these processes applied to it gets passed onto the segmentation phase. Below is the pseudocode for the pre-processing stage:

```
def preprocess(image)

    convert image list from 3D list to 2D list

    call external method to increase contrast (SKImage library)
    reshape image list to 3D list
    save contrast enhanced image for the user to view

    call external method to de-noise image (SKImage library)
    save the de-noised image for the user to view

    return image
```

Figure 4.3: Pre-processing Pseudocode

Segmentation

The process performed here is to split up the image into clusters using k-means clustering. K-means clustering is an unsupervised learning technique which produces k number of cluster centres (centroids) and the data points assigned to each one.

The first step in the algorithm is to generate the starting points for the centroids. This can either be done randomly or they can be chosen - in the case of this program it was done randomly. The second

step is to calculate the distance from each data point to each of the centroids. Distance is calculated using the Euclidean Distance formula:

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

Each data point is then assigned the centroid that is closest to it. Step three is to recalculate the centroids. This is achieved by calculating the mean distance of all the data points assigned to the centroid. Steps two and three are then repeated until a stopping condition is met e.g. data points no longer change centroid.

When k-means clustering is used in this project five clusters are produced (k is equal to five). Below is the image used in *figure 4.2* after going through the segmentation stage.

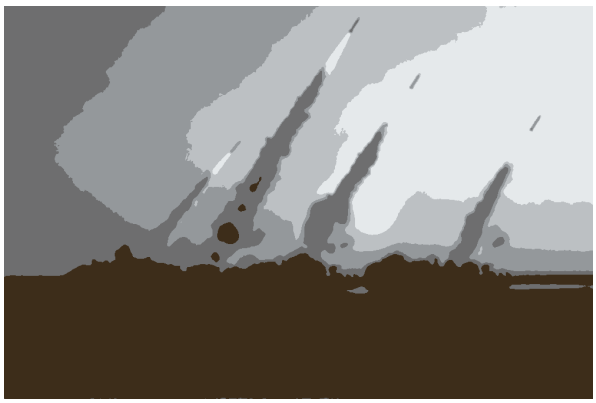

Figure 4.4: Image having gone through segmentation

Below is the pseudocode for the segmentation phase.

```
def segment(image)

    convert image list from 3D list to 2D list

    create K-means model where K=5 (SKLearn)
    fit image to model (SKLearn)
    reshape image using cluster centroids and labels
    save clustered image for the user to view

    return image
```

Figure 4.5: Segmentation Pseudocode

Feature Extraction

This is the final stage before training the model. To covert the image into a smaller number of data points, features were extracted with the GLCM (Grey Level Co-Occurrence Matrix). A GLCM is concerned with the texture and spacial relationship of the pixels.

Firstly, to create the GLCM the image needs to be converted to grey scale. What the GLCM actually calculates is "how often a pixel with grey - level (grey scale intensity or tone) value *i* occurs either horizontally, vertically, or diagonally to adjacent pixels with the value *j*." [27]

Once the GLCM is created there are different statistics that can be calculated. They are contrast, correlation, energy, dissimilarity, ASM and homogeneity. In the case of this project the ASM statistic is used.

The final extracted list provides fifty-four data points. These values then represent one image in the matrix that is used to train the classifier. Below is an example of those values produced from the previous figure (*4.4*).

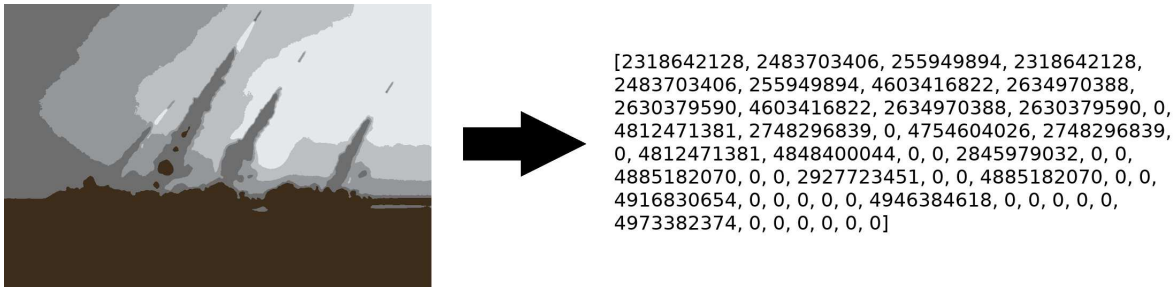

Figure 4.6: Feature extraction example

Below is the pseudocode for the feature extraction. Note i and j in the pseudocode are distances and angles respectively.

```
def feature extraction(image)

    convert image list from 3D to 2D
    multiply image list by 1000 to keep accuracy
    convert image list to an integer list

    for i, 1 to 10:
        for j, 0 to 6
            create GLCM (image, i,j): (SKLearn)
            add ASM feature to results (SKLearn)

    return results
```

Figure 4.7: Feature Extraction Pseudocode

Classifier

The initial design was to use a SVM (Support Vector Machine) as the model for the classifier however the same was done in the framework paper [11] and better results were obtained through a neural network. Due to this the classifier used was a MLP (Multi-layer Perceptron).

A MLP is a supervised learning technique which will be trained on images where it is known if they are fake or not. A MLP has at minimum three layers - an input layer, a hidden layer and an output layer. However the number of hidden layers is not limited to one and more can be added [10]. Below is a diagram of a MLP with one hidden layer.

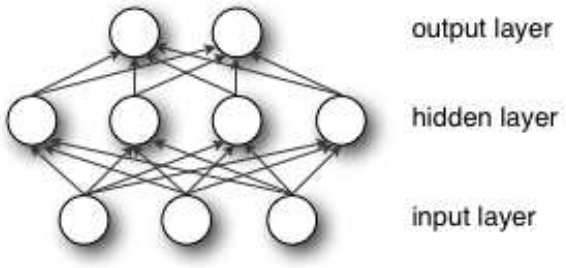

Figure 4.8: MLP with Single Hidden Layer [10]

The way that the MLP trains is by adjusting the weights between the nodes. By using a technique called backpropagation the neural network continually adjusts the weights between the nodes until the error cannot be minimised any more. This goal is called convergence [28]. There are no set best parameters (e.g. the number of hidden layers) in order to set up the MLP. The optimal parameters are found through creating multiple different versions of the MLP and testing them all.

Optimisation of Models

As the parameters to create the model are just passed in as strings or integers, lists were created with all the options for each of the different parameters. Multiple loops were then used to create all the different combinations. Then for each model the accuracy was calculated and the one with the highest was chosen.

When the models are initially setup they are given randomised weights between the nodes. These weights are changed during the training process but the initial randomised weights still have an impact on the final result. To get around this problem 1,000 models with the same parameters were created and the one with the highest accuracy was then selected.

Overall Pseudocode

Figure 4.9 shows the overall flow of the program used to detect copy-move forgery. This is the structure of the Python element that runs behind the website.

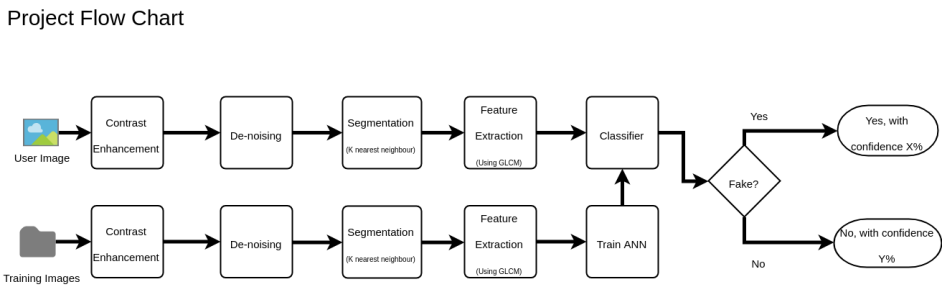

Figure 4.9: Project Flow Chart - *created with draw.io [9]*

Below is the pseudocode for the whole project. This is the algorithm that the project runs on. In the actual Python file there are a few extra details such as writing out training times to a results file and calculating how accurate the model is but this pseudocode is what is actually used to take an image and classify it as real or fake.

```
start
   for each file in training images directory
      if file name contains 'tamp'
         append 0 to training classification
         increment fake image counter
      else
         append 1 to training classification
         increment real image counter

   for each training image file
      open image
      apply pre-processing
      apply segmentation
      get results vector from GLCM
      append vector to training matrix

   set the model parameters
   fit the model to the training matrix and the classification vector

   load the user's image
   apply pre-processing
   apply segmentation
   get results vector from GLCM

   classify the user's image
   display the result to the user
end
```

Figure 4.10: Overall Pseudocode

4.2.2 Web Page Element

Overview

The web page element of this project is to create an easy to access tool to anyone who wants to verify an image. Rather than having a user download, install and setup software it is much more straightforward for them to it via the web. To verify an image a user just has to click on the "Classification" heading and upload an image. There is also information on how the technology works and about the project itself should the user want to find out. There is no limit to the amount of images that the user can upload and they can use the site as much as the want.

HTML, CSS and Javascript

The design of the website is based on a template that is provided by HTML5 UP[1]. They provide free to use templates and due to the time constraints on this project I decided to use a template rather than implement one from scratch.

Once I obtained the template I had to go through and change it to suit my needs. In the case of this project the website did not need to have a lot on it nor be very complex therefore editing the template mainly came down to removing elements from the original and editing text.

PHP

The PHP for the website needed to handle three tasks: the upload of the user's image to the server, passing the image onto the Python program and then returning the result produced by the program to the user.

The first task - handling the upload of the user's image - has a few requirements. As data is being passed from the user to the server, data validation and verification had to take place. This is because a user can be anyone and the data that they pass to the server could be anything. Firstly, the PHP had to check that the file being uploaded was in fact an image. If the file was not an image then the upload could be discarded. Next the file size is checked. This is to stop users uploading very large images as it takes up more space on the server but also slows down the image verification software. Following that the image format is checked to make sure that it is valid and that the software can handle it. Once all the checks are complete the image can then be moved in the file system so the Python element can access it and the user is notified that the upload has been successful (if any of the checks failed then the user is notified and asked to try again).

[1] https://html5up.net/massively

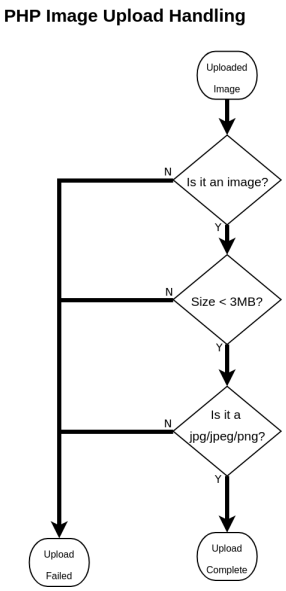

Figure 4.11: PHP Image Validation - *created with draw.io [9]*

Next the Python element is called using a system call. The PHP waits for the system call to finish before displaying the result returned from the Python. Below is the pseudocode for the PHP.

```
start
   set target directory
   store the uploaded file
   if upload successful
      if file is image
         upload is OK
      else
         upload is not OK

   if file exists
      upload unsuccessful

   if file size > 3MB
      upload unsuccessful

   if file not correct type
      upload unsuccessful

   if upload successful
      move file to storage
      call python program
      output result
end
```

Figure 4.12: PHP Pseudocode

4.3 Implementation

This next section will cover the implementation stage of the project. It will cover how the Python was implemented as well as the implementation of the web element of the project.

4.3.1 Python

Image Processing

Each image that is used goes through the same four step process before it can be used to train the classifier or be classified. This is to put all the images on a level playing field from the point of view of the machine. The four steps are as follows:

1. Load the image
2. Pre-processing
3. Segment the image using K-nearest-neighbour
4. Extract the feature using GLCM

Loading the image into Python was straightforward. By using the default OS library that comes with Python the program gets a list of all the images that are to be used for training. It goes through each item in the list and only loads the image once it is needed to be processed. This is done to save memory. The image itself is opened using the PIL library but that is not stored in a format for calculations so it needed to be converted using NumPy.

Now the image is ready for pre-processing. As mentioned before there are two stages to the pre-processing: the contrast enhancement (histogram equalisation) and the de-noising. The "equalize_hist" method in the SKImage library[2] is used to perform this task. Before running it through the method the image list had to be reshaped to a 2D list rather than a 3D one. This is not a permanent change however as once the method has been used the image can be returned to its original 3D state but with the contrast enhanced.

To de-noise the image it is run through SKImage's "de-noise_tv_bregman" method[3]. Once that is done it is returned from the pre-processing method and can be passed onto the segmentation phase. After both the contrast enhancement and de-noising the image is saved so that it can be viewed by the user in case it shows any other tampering that the algorithm missed.

In the same way that the image was reshaped before the histogram equalisation it needs to be reshaped again before it can be broken up into clusters. Using SKLearn's cluster library[4] the clustering is setup so that it is split into five clusters then the 2D image list is fitted to it. The image can then be

[2] http://scikit-image.org/docs/dev/api/skimage.exposure.html

[3] http://scikit-image.org/docs/dev/api/skimage.restoration.html

[4] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

reshaped using the returned cluster centroids and cluster labels to create an image similar to the style seen in *figure 4.4*. This image is then saved so that it can be displayed to the user. The clustered image is then returned from the method in order to be passed on for feature extraction.

Once again the image list is reshaped to a 2D image list so that it can be passed into the methods. To create the results list seen in *figure 4.6* two loops are used. Using the greycomatrix and greycoprops methods provided by SKImage[5] the loops select two points and then extract the feature from the GLCM based off of a distance and angle value. The two loops are used to select the different distances and angles. The returned vector then becomes a row in the matrix used to train the neural network.

Machine Learning and Parameter Optimisation

There were two different classifiers tested in this project. One was a Linear SVM and the other was a MLP Neural Network. Both were used but once it became clear that the neural network was superior then most of the testing was concentrated on getting the best out of that model. Both of the classifiers were imported from the SKLearn library.

When it comes to both the classifiers there are a lot of options when it comes to the parameters used. For example the SVM has parameters such as the C-value, loss settings and iterations. The MLP has ones such as activations, solvers, learning rates, hidden layers and hidden layer sizes. Some of the parameters have a lot of impact where as others can have a more minor impact on the final result. Instead of just picking semi-randomly and hoping for the best the solution a Python program was set up to exhaustively test multiple different combinations of the parameters. This was done by creating lists for each of the different parameters containing the different values that it could be set to. An example of this is the MLP parameter solvers can have the value lbfgs, adam and sgd. This is just represented as a list of strings and a loop is used to go through each one. Using multiple for loops the program creates all the variations and adds them to a list of classifiers. See *figure 4.13* for the code used to do this.

```
for i in range(0, len(activations)):
  for j in range(0, len(solvers)):
    for k in range(0, len(learning_rates)):
      for l in range(0, len(hidden_layer_sizes)):
         for n in range(0, len(max_iter)):
          neural_net = neural_network.MLPClassifier(activation=activations[i],
                                                    solver=solvers[j],
                                                    learning_rate=learning_rates[k],
                                                    hidden_layer_sizes=(hidden_layer_sizes[l]),
                                                    max_iter=max_iter[n])
          neural_net.fit(training_images, classification)
          networks.append([neural_net, neural_net_parameters])
```

Figure 4.13: Creating Multiple Neural Networks

Once the classifiers have been created then they all have to be tested. The program goes through

[5]http://scikit-image.org/docs/dev/api/skimage.feature.html

each one and gets the accuracy of the classifier. An early on problem was that the most accurate was one which was classifying all the images as the same classification i.e. all as real images. The overall accuracy would be high as there was sometimes more real than fake images in the test (or vice-versa). It could seem like a good result when but after further investigation actual model was of no use. Once this was noticed steps were added to calculate specific accuracy values for fake and real images. These would then be used to make sure that the classifier with the highest accuracy was actually making a useful model.

Data Sets

The two data sets that were used in this project were provided by the Image Communication Laboratory who make available multiple data sets for copy-move forgery detection [29]. The two that were used were the MICC-F600 and the MICC-F220. The MICC-F600 contains 440 original images, 160 fake images and 160 ground truth images. The ground truth images were not used in the project, just the real and fake images. The MICC-F220 is a smaller data set with 110 doctored images and 110 real images. The images in the MICC-F600 are larger and of higher quality than those in the smaller data set.

Both data sets contain images to train models on copy move forgery. Some of the forgeries are a bit more subtle but a lot of them are quite obvious at first glance. However the goal is to train the model what to look for and as long as that is accomplished then the data set is useful.

Initially in the project the MICC-F600 images were used. However due to the large size of the images the processing took a very long time. The initial solution was to crop the images so that they could be processed faster. The real images were just cropped over any region and the fake images were cropped over the doctored part of the image. This did speed up the run time but the model produced was not good. The decision was made to switch over to the MICC-F220 with the smaller images. If more powerful hardware was made available for the project then it would make more sense to use the MICC-F600 but due to the limitations of this project only the MICC-F220 could be used.

Libraries Used in Python element

Library	Used For	Link
Image (PIL)	Opening Images	*https://pillow.readthedocs.io/en/3.1.x/reference/Image.html*
svm (SKLearn)	Creating Linear SVM	*https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html*
cluster (SKLearn)	Apply KMeans Clustering	*https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html*
neural_network (SKLearn)	Creating ANN	*https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html*
greycomatrix (SKImage)	Creating GLCM	*http://scikit-image.org/docs/dev/api/skimage.feature.html*
greycoprops (SKImage)	Extracting GLCM features	*http://scikit-image.org/docs/dev/api/skimage.feature.html*
exposure (SKImage)	Increasing contrast	*http://scikit-image.org/docs/dev/api/skimage.exposure.html*
pickle	Saving variables	*https://docs.python.org/3/library/pickle.html*
os	System commands	*https://docs.python.org/3/library/os.html*
re	Regular expressions	*https://docs.python.org/3/library/re.html*
numpy	List management and calculation	*https://docs.scipy.org/doc/*

4.3.2 Web

HTML, CSS, Javascript

As mention previously, for the HTML, CSS and Javascript side a template was used. The template, which was created by HTML5 UP[6], was used mainly to save time but also because it is easier to decide on a design when you can see multiple options side by side. Once the template was acquired and setup, I needed to go through the HTML, CSS and Javascript to make it suit the needs of the project.

The template was originally for a news site but the overall style suited displaying the information about the project as well as the project itself. The template was also designed to handle a much larger site but it scaled down well for the purposes of this project. Once the client side was set up the next step was to implement the server side and the PHP so that the website could communicate with the

[6]https://html5up.net/massively

Python element.

PHP

The PHP had to handle three tasks: the upload of the image, calling the Python program to do the classification and then returning the result to the user. The code for the image upload was adapted from an example on the W3 Schools website[7]. Changes were made so that larger images could be uploaded, only .jpg and .png images were allowed and some of the output was changed. Once the image gets uploaded it is saved to a specific folder so that the Python element knows where it is. To call the Python program the PHP just executes a Python command via the system and then displays the text returned by the program.

The Python code also saves the user's image at each stage of the pre-processing so that they can be displayed to the user when the result is returned. This allows the user to see the different stages but also allows them to look for inconsistencies themselves. For example, the image with increased contrast may reveal something that the human eye can see but the algorithm misses.

[7] https://www.w3schools.com/php/php_file_upload.asp

Chapter 5

Results & Discussion

5.1 Introduction

This chapter details the results of the implementation. It covers the optimisation of the parameters and overall model used for classification. There will be details on both the neural network and SVM classifiers and how they compare. For both the classifiers time had to be taken to optimise the parameters used in creating the model. This chapter will also cover the effect of the pre-processing stage, the segmentation stage and the different features of the GLCM on the final accuracy. When testing and comparing the parameters of the neural network and the SVM the same pre-processing, segmentation and GLCM feature will be used for each one to allow for accurate results. Finally, the chapter will talk about the details of the classifier used in the final product and the final website produced.

5.2 Neural Network

This section will cover the results from the optimisation of both the combination of parameters used and the best network produced.

5.2.1 Parameter Optimisation

Parameters Used

Below are the different parameters that were changed in the optimisation:

Parameter	Options	Description
Activation	identity, tanh, relu, logistic	The activation function for each hidden layer
Solver	lbfgs, sgd, adam	The solver to optimise the weights within the neural network
Learning Rate	invscaling, adaptive, constant	Learning rate schedule for the weight updates
Hidden Layer Sizes	(number>0,)	Each number represents a hidden layer with X number of neurons
Max Iterations	number >0	Will iterate until convergence or the number set here

As discussed previously, the way that the best neural network was found was by setting up variables to hold the different options for each parameter and then iterating through all the combinations. For the hidden layer sizes and maximum iterations options a few sensible options were chosen. Different runs had to be conducted to test the hidden layer sizes as a hidden layer size cannot be zero. Three different runs were conducted: one hidden layer, two hidden layers and three hidden layers. Finally, some of the options did not suit the data set - for example the solver "sgd" is used for much larger data sets - so they were removed to save time.

To view the impact of each parameter for each of the different MLP (one hidden layer, two hidden layers, three hidden layers) the parameter will be graphed vs the overall accuracy. The graphs should reveal which parameter has the most impact on the overall accuracy as well as showing how many hidden layers to have.

One Hidden Layer Results

Parameters Used:

Parameters	**Options Used**
Activation	identity, tanh, logistic, relu
Solver	lbfgs, adam
Learning Rate	invscaling, adaptive, constant
Hidden Layer Size	28, 35, 50, 20, 54
Max Iterations	100, 200, 500, 750

The amount of neural networks that were created with one hidden layer was approximately 500.

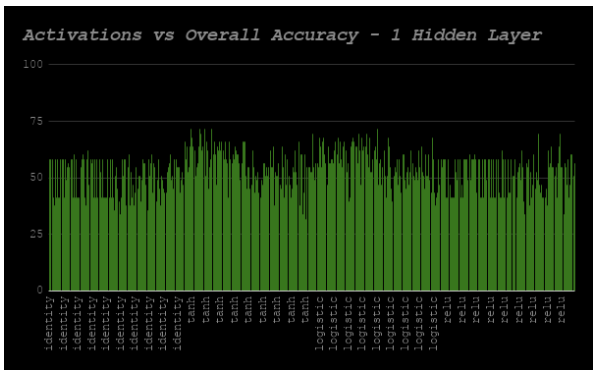

Figure 5.1: Activations vs Overall Accuracy (One Hidden Layer)

Firstly, the impact of the activations on the overall accuracy (*figure 5.1*). What can be taken from the graph is that there are two activations which produce a better performance than the other two. These are the tanh activation and the logistic activation. While the accuracy for all the networks is quite low there are two clear peaks above tanh and logistic.

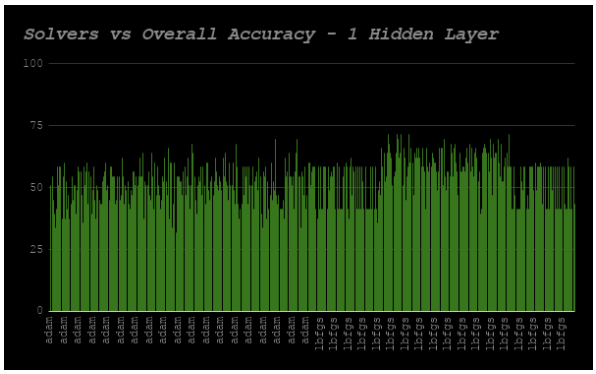

Figure 5.2: Solvers vs Overall Accuracy (One Hidden Layer)

The graph for the solvers (*figure 5.2*) is quite flat. The lbfgs solvers seems to have more varied results whereas the adam solver is more consistent. The networks with the best accuracy are within those that use the lbfgs solver. This is expected as the lbfgs solver is designed for smaller data sets which is what is being used here.

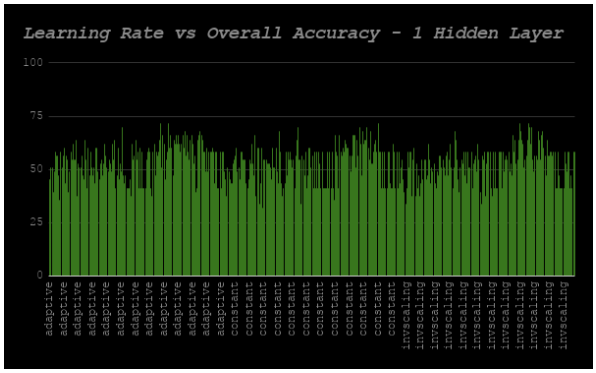

Figure 5.3: Learning Rates vs Overall Accuracy (One Hidden Layer)

The learning rates graph (*figure 5.3*) displays a cyclic pattern. Towards the end of each learning rate there is a spike in the graph. This means that the other variables are having more of an impact on the accuracy than the learning rate. Out of each one it seems that adaptive has the best overall accuracy but there is not a big difference.

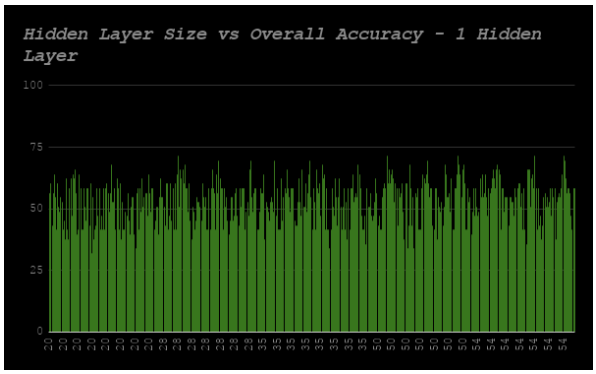

Figure 5.4: Hidden Layer Size vs Overall Accuracy (One Hidden Layer)

Next is the hidden layer size graph (*figure 5.4*). The graph is quite flat with very little patterns in the spikes. However, there does seem to be more spikes around the higher number of nodes in the hidden layer (50 or 54 nodes) when compared to the networks with lower numbers of nodes in the hidden layer (20 nodes).

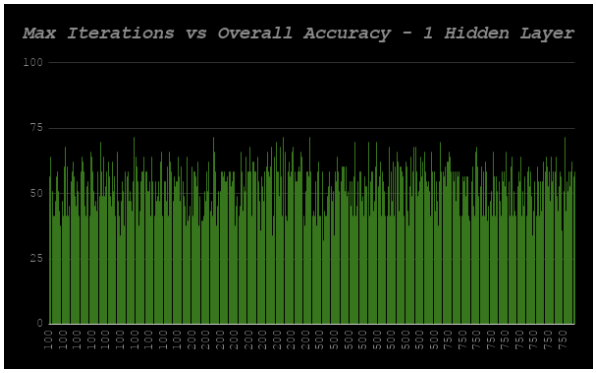

Figure 5.5: Max Iterations vs Overall Accuracy (One Hidden Layer)

What can be taken from the max iterations graph (*figure 5.5*) is that it does not have a huge impact on the overall accuracy. There is obviously some amount of variation but not enough to make a statement such as that having more or less iterations is better. As long as the model converges the maximum number of iterations set will not matter.

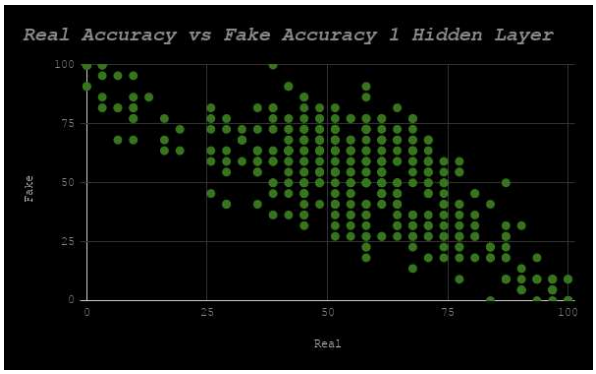

Figure 5.6: Real vs Fake Accuracy (One Hidden Layer)

Finally there is the scatter plot of the real vs fake accuracy of each model (*figure 5.6*). The best neural networks are those towards the top, right corner as those will be the most accurate for both real and fake classification. The models in the top left and bottom right are the most useless as they classify every image as always real or always fake. When looking at all the models with just one hidden layer the accuracy of them is not good enough. Those that classify around 75% for real classification accuracy tend to only just be over the 50% mark for fake classification and vice-versa. As neural networks with more hidden layers are tested the goal will be to move this scattering more towards the top right and to be able to produce a more accurate and useful network.

Two Hidden Layers Results

Parameters Used:

Parameters	**Options Used**
Activation	identity, tanh, logistic, relu
Solver	lbfgs, adam
Learning Rate	invscaling, adaptive, constant
Hidden Layer Size	28, 35, 50, 20, 54
Max Iterations	100, 200, 500, 750

The amount of neural networks that were created with two hidden layers was approximately 2,000.

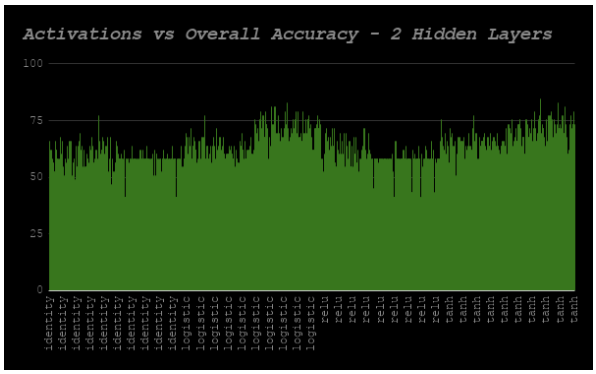

Figure 5.7: Activations vs Overall Accuracy (Two Hidden Layers)

The graph for the activations for neural networks with two hidden layers (*figure 5.7*) shows two major spikes. One towards the end of those that use the logistic classification and one towards the end of those that use the tanh classification. Clearly these two classification used in conjunction with some other variable combination produces the best result. The other point to note from the graph is that tanh has the most consistently good results compared to the other activations.

When compared to the neural networks with only one hidden layer (*figure 5.1*) there is definite improvement. All the results are consistently higher and there are now networks that are getting above 75% accuracy.

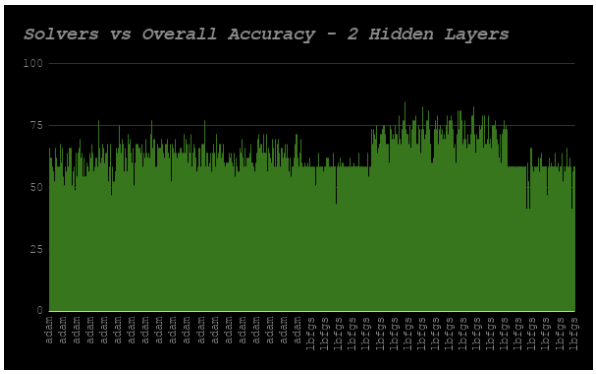

Figure 5.8: Solvers vs Overall Accuracy (Two Hidden Layers)

The networks which use the adam solver tend to get above 60% with some getting above 75% (*figure 5.8*). While the accuracy from the adam solver seems to be more level, the better results once again come from the lbfgs solver with a section of those networks getting over 75% accuracy. At the same time the worst results also come from the lbfgs solver. This could mean that when used in conjunction with another parameter (or parameters) the lbfgs solver is not as useful.

The shape of the graph is very similar to the one with networks when they only have one hidden layer (*figure 5.2*). This time the shape is much clearer due to having more data points and it also seems

to be amplified a little. This amplification could point towards the fact that having two hidden layers means better overall accuracy.

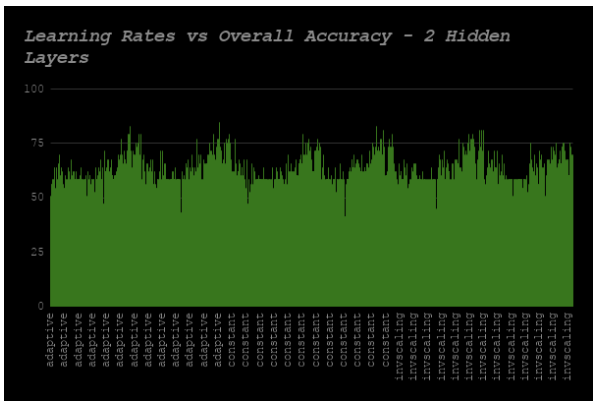

Figure 5.9: Learning Rates vs Overall Accuracy (Two Hidden Layers)

The learning rates graph (*figure 5.9*) is a similar shape to the previous one (*figure 5.3*) but as was the case with the solvers graph (*figure 5.8*) the shape is more obvious. Once again there are spikes towards the end of each learning rate. This is likely to mean that the learning rate is not having too much of an impact and it is the other parameters that make the difference.

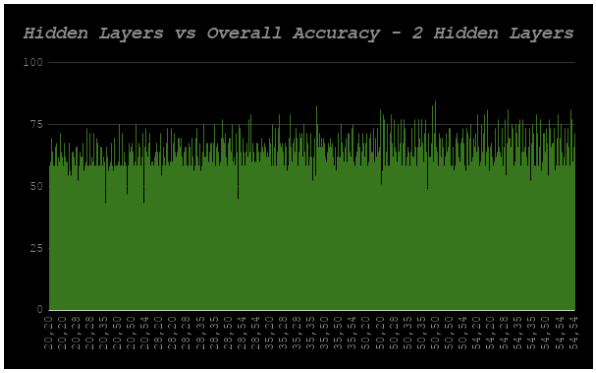

Figure 5.10: Hidden Layer Sizes vs Overall Accuracy (Two Hidden Layers)

The next graph (*figure 5.10*) is what combination of nodes for the hidden layer produces the best overall accuracy. Similar to the previous hidden layer size graph (*figure 5.4*) there is a very slight increase as the number of nodes get higher. Once the first hidden layer has at least 50 nodes the increase begins to flatten off.

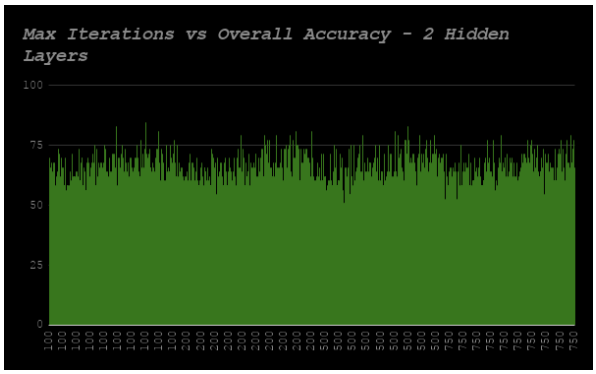

Figure 5.11: Max Iterations vs Overall Accuracy (Two Hidden Layers)

Similarly to the one hidden layer networks the maximum iterations graph for the networks with two hidden layers (*figure 5.11*) does not seem to have much of an impact on the overall accuracy of the network. As was mentioned previously the maximum iterations is the number of iterations that the network training has to meet if it does not converge. If the number of iterations needed to converge is less than the maximum iterations then the parameter will not be needed which is why there is not much of a pattern here.

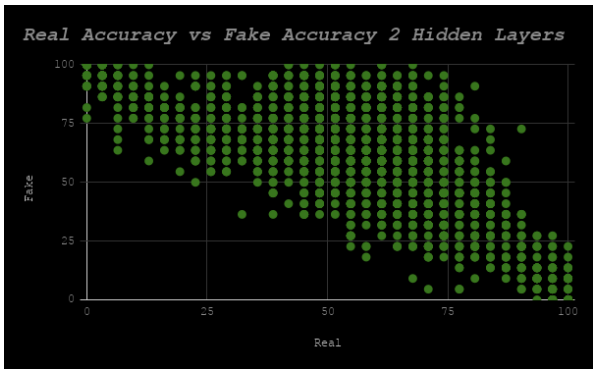

Figure 5.12: Real vs Fake Accuracy (Two Hidden Layers)

Finally, there is the real vs fake accuracy graph (*figure 5.12*). Once again there are networks which classify all the images as the same classification (those in the top left and bottom right corners). When it comes to networks with two hidden layers we are starting to see dots appear in the top right hand box. These are the best networks as the are getting both classifications correct and are therefore useful when it comes to detecting copy move forgery.

Three Hidden Layers Results

Parameters Used:

Parameters	Options Used
Activation	identity, tanh, logistic, relu
Solver	lbfgs, adam
Learning Rate	invscaling, adaptive, constant
Hidden Layer Size	28, 35, 50, 20, 54
Max Iterations	100, 200, 500, 750

The amount of networks created to exhaustively find the best parameters with three hidden layers was approximately 12,000.

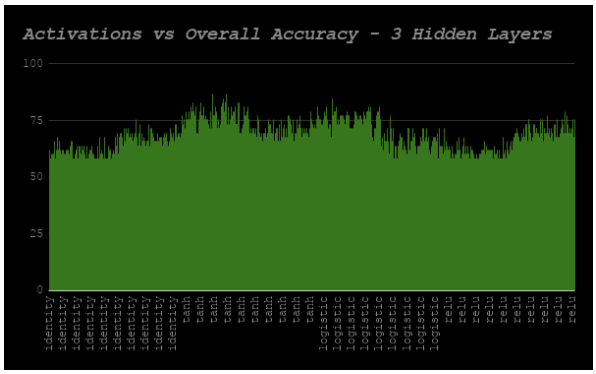

Figure 5.13: Activations vs Overall Accuracy (Three Hidden Layer)

For the activations (*figure 5.13*) the pattern is becoming clear. Once again the best results are obtained from either the logistic activation or the tanh activation. And, once again, the best results are those that use tanh. When compared to the networks with two hidden layers (*figure 5.7*) the results are slightly better. There was more improvement going from one to two hidden layers than going from two to three hidden layers. The other point to note is in those sections that appeared flat in the graph of the networks with two hidden layers are now more jagged. This means the accuracy is fluctuating a bit more than before in these combinations.

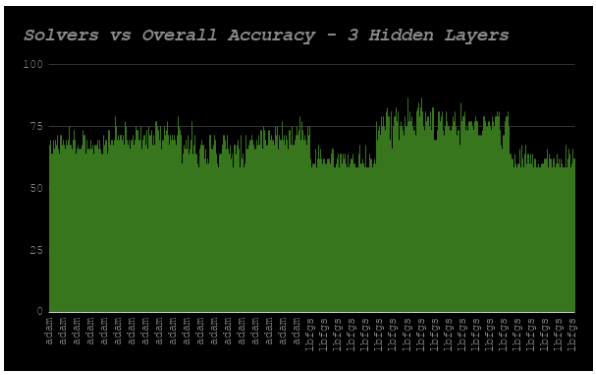

Figure 5.14: Solvers vs Overall Accuracy (Three Hidden Layers)

The shape of the solvers graph (*figure 5.14*) is the same as the previous one (*figure 5.8*). The adam results tend to produce more consistent overall accuracy whereas the lbfgs solver has a spike when it is used in combination with some other variable. Again, the lbfgs solver produces the best results.

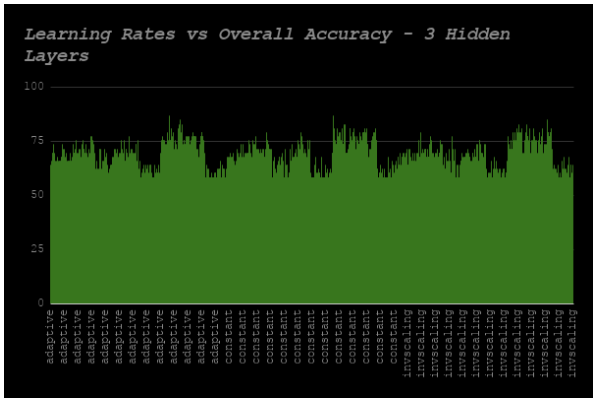

Figure 5.15: Learning Rates vs Overall Accuracy (Three Hidden Layers)

The learning rates graph (*figure 5.15*) displays the cycle seen before of a peak towards the end of each of the different learning rates. There also smaller wider peaks towards the start of each learning rates. As with the activations and solvers the graph shows the same shape as before just clearer due to the increase in data points. Again the conclusion is the same as it does not seem to matter too much what the learning rate is.

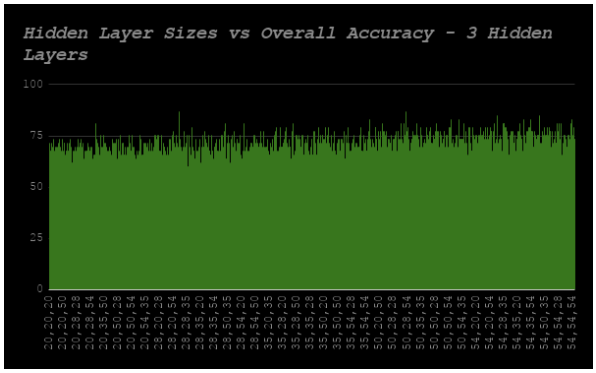

Figure 5.16: Hidden Layer Sizes vs Overall Accuracy (Three Hidden Layers)

The slight increase in the hidden layer sizes graphs talked about previously (*figure 5.4, 5.10*) is much more obvious in *figure 5.16*. Clearly having more nodes in the different layers does produce an increase in the accuracy. The extra layer however does not seem to have added much of an increase in the overall accuracy.

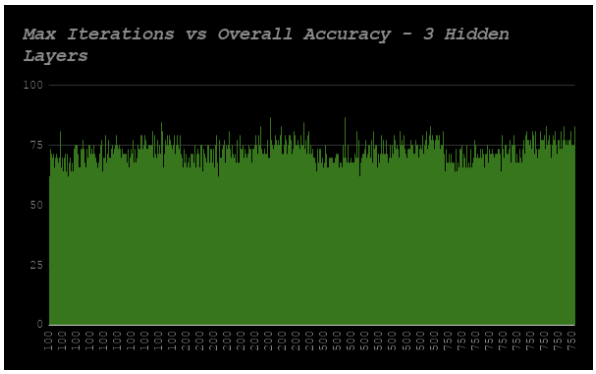

Figure 5.17: Max Iterations vs Overall Accuracy (Three Hidden Layers)

In the maximum iterations graph, unlike the previous cases, a cycle is beginning to appear. There is a rise towards the end of each maximum iterations. As was the case with the learning rates graph (*figure 5.15*) the cycle means that the other variables are the ones having the effect and not the maximum iterations. As long as the model converges then it does not matter what the maximum iterations is set to.

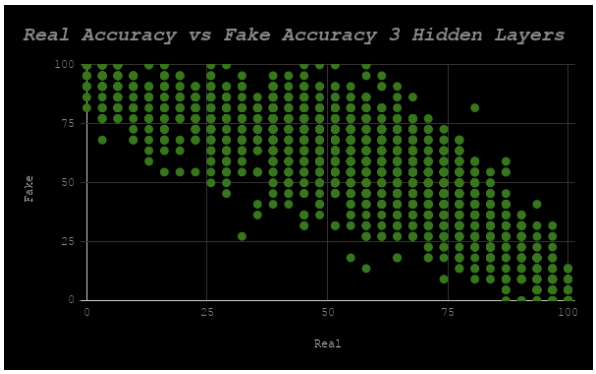

Figure 5.18: Real vs Fake Accuracy (Three Hidden Layers)

When looking at the graph of real vs fake for three hidden layers (*figure 5.18*) there is not much improvement compared to the previous one (*figure 5.12*). There are actually less networks appearing in the top right corner. The main jump came from going between one hidden layer and two but the improvement from going between two and three seems marginal. When looking at all three of the networks together (*figure 5.19*) the graph demonstrates that point. What is good about the network with three hidden layers in the top right is that it is more in the middle and does not lean towards classifying real or fake more often as the two hidden layer networks do. This makes for a more useful classifier.

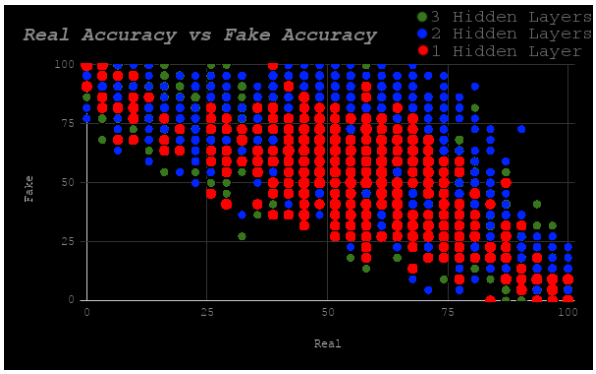

Figure 5.19: Real vs Fake Accuracy Overall

5.2.2 Optimisation of the best network

Once the best combination of variables has been found that network then needs to be optimised. When the network is created the weight between the nodes are randomised. This randomisation can help or hinder results. To get around this problem around 1000 networks were created with the same parameters and then the best one chosen from that group. This meant a jump from around 85% accuracy to 90% accuracy.

5.3 SVM

As mention earlier the SVM was the model used first. Once better results were obtained from neural networks the switch was made to use them instead. The SVM was optimised in the same way as the neural networks - by going through exhaustively and testing each combination of parameters. The parameters that were used in the combinations are shown below.

Parameters	**Options Used**
Penalties	l2
losses	hinge, squared hinge
Duels	True, False
C	1, 3, 5, 10, 20, 50, 100, 500
Fit Intercept	True, False
Max Iterations	1000, 3000, 5000, 7000, 10000, 20000

The table below shows the results of a few different tests with the SVM classifier. The accuracy of the SVMs is poor and when the same tests were run with the neural networks the results showed immediate improvement. Rather than spend time on optimising the SVMs the choice was taken to put all efforts into creating the best neural network. This does not completely rule out the use of SVMs

in this context as given the right kind of data set and set of parameters the SVM may perform just as well as the neural network. However due to time constraints I felt it was best to just focus on the one kind of classifier.

Run Information			Parameters					
Overall Accuracy	**Real Accuracy**	**Fake Accuracy**	**Penalty**	**Loss**	**Duel**	**C**	**Fit Intercept**	**Max Iter**
56%	-	-	l2	squared_hinge	T	3	T	3000
55%	56%	55%	l2	hinge	T	1	T	5000
58%	66%	50%	l2	squared_hinge	T	20	T	3000
59%	54%	64%	l2	hinge	T	5	F	1000

5.4 Effect of Image Processing Stages on Accuracy

This section is to cover the effect of each processing stage that the images go through on the overall accuracy, the fake accuracy and the real accuracy. The baseline that they will be compared to is the neural network that has the best results so far. This final neural network will then be compared to the current best one. Below are the details for the baseline network:

Overall Accuracy	**Real Accuracy**	**Fake Accuracy**	**Activation**	**Solver**
90.56%	87.09%	95.45%	tanh	lbfgs
Learning Rate	**Hidden Layers**	**Hidden Layer Sizes**	**Max Iterations**	**Extraction Type**
invscaling	3	50,35,54	100	ASM

5.4.1 Pre-processing

As mention previously, there are two steps at the pre-processing stage: contrast enhancement and de-noising. Three neural networks will be created to compare to the baseline. One without contrast enhancement, one without de-noising and one without either. Below are the results for these neural networks.

Name	**Overall Accuracy**	**Real Accuracy**	**Fake Accuracy**
Baseline	90.56%	87.09%	95.45%
No Contrast Enhancement	64.15%	61.29%	68.18%
No De-noising	73.58%	61.29%	90.9%
No Pre-processing	62.26%	64.51%	59.09%

The table shows that the more important stage in pre-processing is the contrast enhancement. The

overall accuracy from no contrast enhancement is much lower than the overall accuracy from no de-noising (64.15% vs 73.58%). This makes sense as the image goes through a much larger change when the contrast is enhanced than when it is smoothed over with de-noising.

The other interesting point to note is that when pre-processing is taken out the real accuracy is higher than the fake accuracy whereas when either or both pre-processing stages are used the fake accuracy is higher than the real accuracy.

5.4.2 Segmentation

The current neural network uses K-means clustering where k is equal to five. Two neural networks will be created for comparison: one where k is equal to three and one where k is equal to ten. Below are the results for these neural networks.

Name	Overall Accuracy	Real Accuracy	Fake Accuracy
Baseline	90.56%	87.09%	95.45%
$k = 3$	56.6%	38.7%	81.81%
$k = 10$	60.37%	54.83%	68.81%

The results for the different segmentation values are quite interesting. In both cases the models tend to lean towards classifying the models as fake. The k is equal to three model is only just above 50% accuracy making it the worst model out of the three. The k is equal to ten model has an accuracy of 60%. It's certainly not very useful but with some changes to the parameters and the right images it could be more useful.

5.4.3 GLCM Variable

Five features that can be extracted from the GLCM were tested to see how they effected the results. These are contrast, dissimilarity, homogeneity, ASM and energy. Each of these was tested with one, two and three hidden layered neural networks. The results are shown in the following table:

	Accuracy			*Parameters*			
Extraction Type	**Overall Accuracy**	**Real Accuracy**	**Fake Accuracy**	**Activation**	**Solver**	**Learning Rate**	**Hidden Layer Sizes**
Contrast	75%	80%	68%	tanh	lbfgs	adaptive	54
Contrast	79%	70%	90%	logistic	lbfgs	adaptive	54,20
Contrast	81%	74%	90%	logistic	lbfgs	invscaling	50,20,20
Dissimilarity	69%	61%	84%	tanh	lbfgs	invscaling	50
Dissimilarity	75%	61%	95%	logistic	lbfgs	constant	54,50
Dissimilarity	77%	74%	81%	tanh	lbfgs	adaptive	54,20,54
Homogeneity	71%	61%	86%	tanh	lbfgs	constant	50
Homogeneity	73%	80%	63%	tanh	lbfgs	adaptive	54,20
Homogeneity	77%	90%	59%	tanh	lbfgs	adaptive	54,35,54
ASM	75%	80%	68%	tanh	lbfgs	adaptive	54
ASM	83%	74%	95%	tanh	lbfgs	adaptive	50,50
ASM	84%	74%	100%	tanh	lbfgs	invscaling	50,35,54
Energy	73%	74%	72%	logistic	lbfgs	invscaling	50
Energy	79%	77%	81%	logistic	lbfgs	constant	35,54
Energy	83%	80%	86%	tanh	lbfgs	invscaling	50,35,35

The top three most useful features to use are (in descending order) ASM, contrast and energy. The one that was optimised and used in the final project was the ASM with three hidden layers. After going through optimisation the final accuracy came out to approximately 90%. Similar results may be obtained using the contrast and energy features however this was not tested in this project.

Clearly the dissimilarity and homogeneity features are not useful when compared to the others, but they are still good results when compared to the SVM's accuracy. The data set that was used with all of these feature extraction types was the MICC 220 so different results may be obtained depending on both the size and the content of the data sets. This was not tested in this project but could be future work for the field. Another point of note is that the different features may perform better or worse depending on the type of forgery that is trying to be detected. ASM is the best feature in this case when it comes to copy move forgery but other features may be better at detecting other kinds of forgery.

5.5 Classifier Used

Below is the same table as earlier which displays the details of the Multi-layer Perceptron that is used in the final project.

Overall Accuracy	Real Accuracy	Fake Accuracy	Activation	Solver
90.56%	87.09%	95.45%	tanh	lbfgs
Learning Rate	**Hidden Layers**	**Hidden Layer Sizes**	**Max Iterations**	**Extraction Type**
invscaling	3	50,35,54	100	ASM

The accuracy of the final model is weighted towards classifying the images as fake rather than real. This was not intentional but is actually better than if it was weighted towards classifying as real because it is better to keep people skeptical due to the massive amounts of misinformation.

Figure 5.20 displays the ROC curve for the model that was used in the project. A ROC curve compares the true positive rate to the false positive rate. In the case of this project a true positive is a case where the model classifies a fake image as fake and a real image as real. A false positive is a case where the model classifies a real image as fake or vice versa. The curve for this project is the right kind of shape with a sharp increase and then leveling out. The closer the blue curve is to the red dashed line the less useful the model is and if it is below the red line then it is not useful at all.

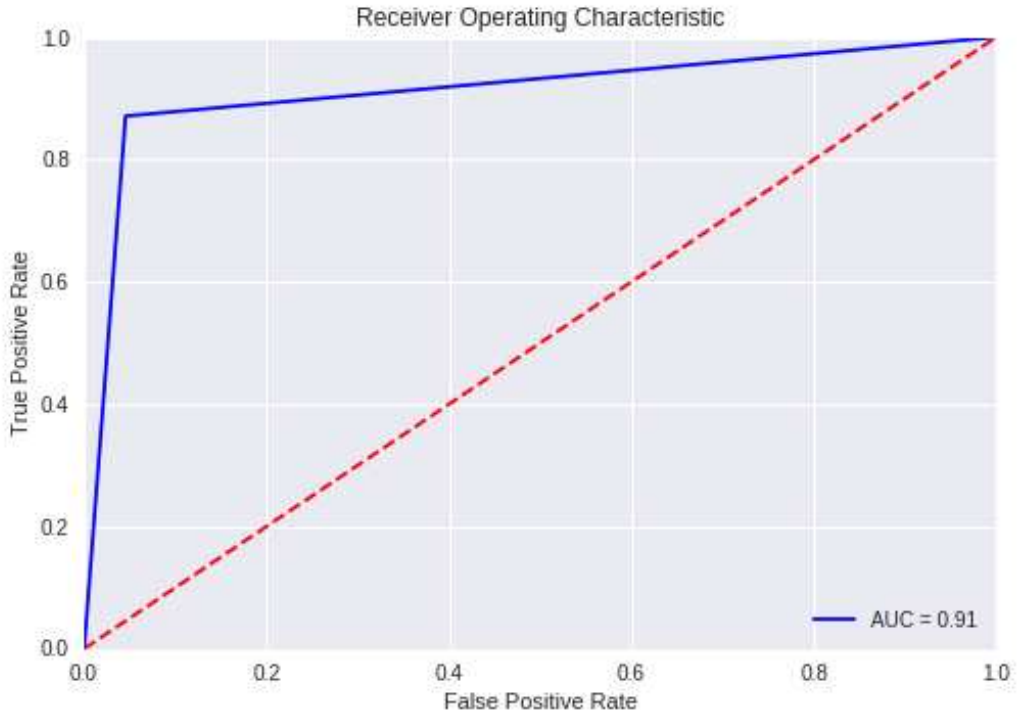

Figure 5.20: ROC Curve for the model used

The table below is the confusion matrix for the model. A confusion matrix breaks down how the model is actually classifying. The more accurate the model the higher the numbers along the main

diagonal. In the case of this model the matrix is showing that the model has a slight leaning towards classifying images as fake. However, as mentioned before, this is not necessarily bad. Overall both the confusion matrix and the ROC curve show an accurate and useful model.

Confusion Matrix		
	Predicted Real	**Predicted Fake**
Actual Real	21	4
Actual Fake	1	27

5.6 Final Product

The final product of this project is an easy to use website that classifies an image. The website is made up of four main pages: a homepage, a classification page, a page describing how it works and a page describing the project. *Figures 5.21, 5.22, 5.23* are the screenshots of the homepage, how it works page and about the project page. All three are simple and to the point. The background image looks strange but this is just the nature of the screenshot. In reality the background image scrolls with the user.

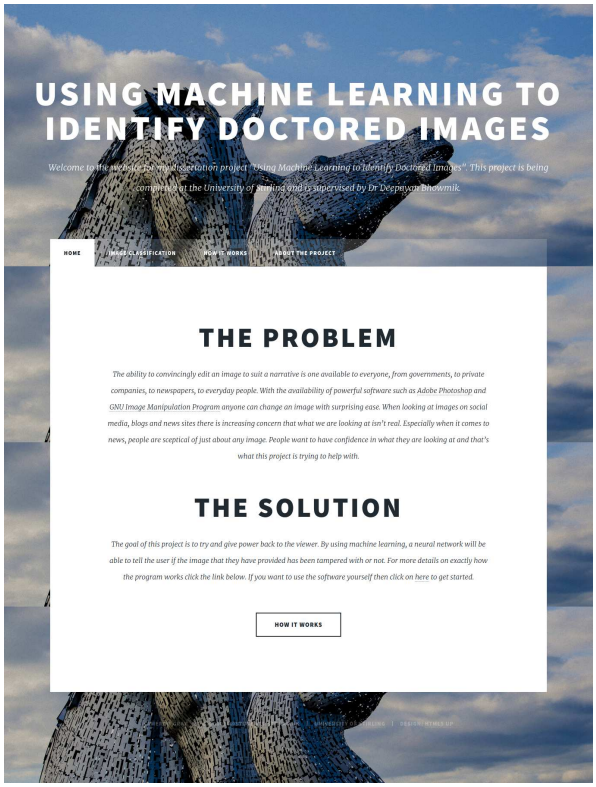

Figure 5.21: Home page

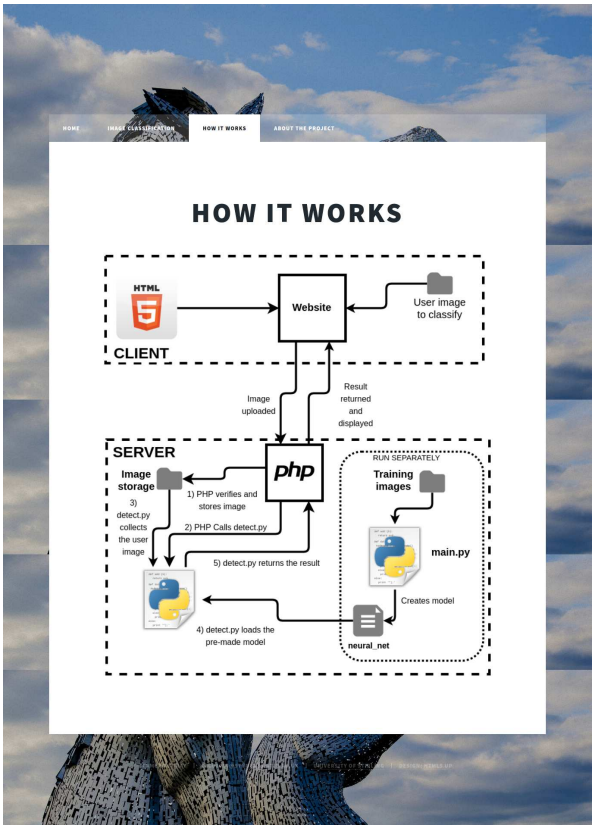

Figure 5.22: How it works page

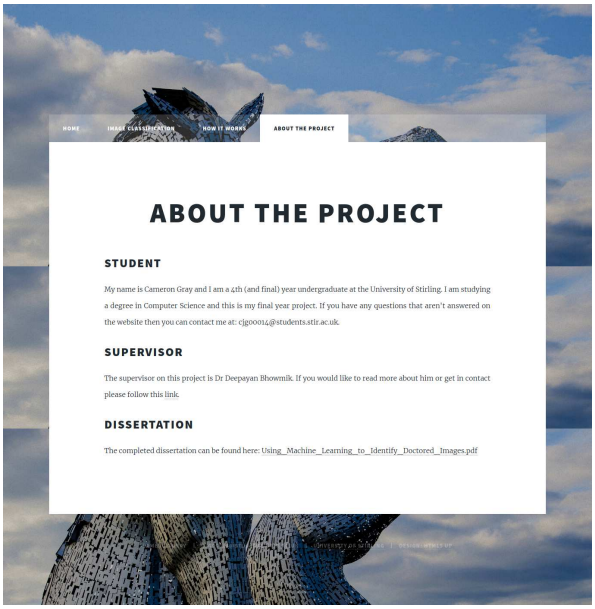

Figure 5.23: About the project page

The next two figures (*5.24, 5.25*) show the classification page before and after uploading an image to classify. Again, the interface is simple and straightforward to meet the requirements of the project. There is only one button to upload an image and one to classify. The less input from the user the less there is to go wrong with the website. It also reduces the confusion from the user's side too.

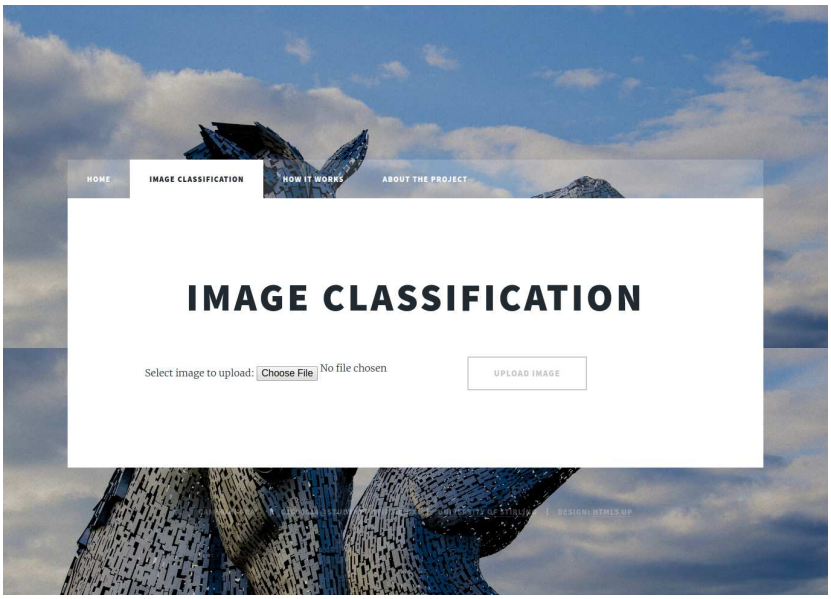

Figure 5.24: Classification before uploaded file

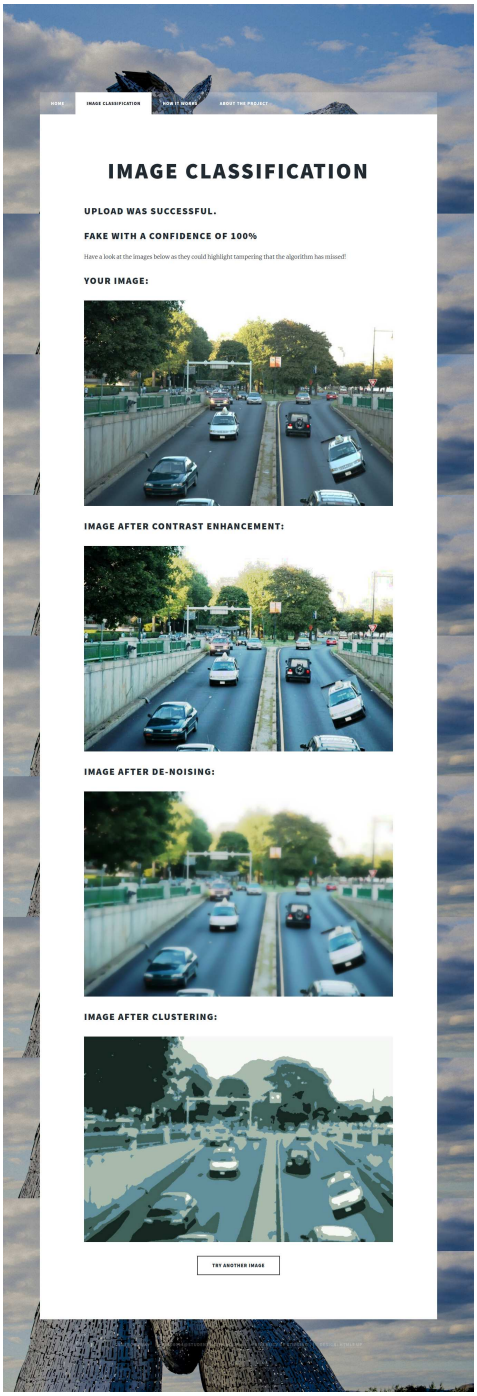

Figure 5.25: Classification after upload

Once the user has had their classification and has viewed the different stages of their image they have an option to upload another image. Again the amount of input has been limited to make it as straightforward as possible.

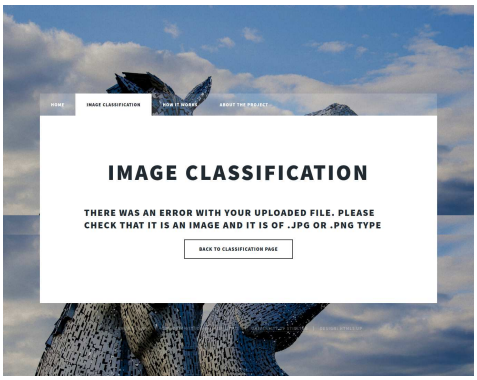

Figure 5.26: Error with the file upload

Finally, *figure 5.26* is an example of a user uploading a file that is not an image. The output from the website in unambiguous and makes it easy for the user to try again while at the same time telling them that something went wrong.

Overall the UI meets the requirement of being easy and straightforward to use. The design of the website is clean and is not overloaded with information. It makes it really simple for the user to get the information that they want without them having to do lots of work.

Chapter 6

Conclusion

6.1 Evaluation

This project set out to give power back to the person viewing an image. The large amount of fake images and people creating images to suit their own narrative means that everyday people do not know if what they are looking at is genuine. Everything from social media posts to large scale news operation can and have been susceptible to doctored images. This project is a step in the direction of making technologies used to identify fake images available to anyone.

The idea was to give the user a certain amount of verification with images they are looking at. The classification is not 100% accurate but it can give the user some sort of idea if the image has been tampered with. It also displays the submitted image after it has gone through the different stages of processing. This lets the user have a look for doctoring themselves and to look for any other inconsistencies.

The software is also able to handle both .jpg and .png images which are two of the most common image formats. If a user has an image not in these formats then it is reasonably straightforward to convert these images so that the program can handle them. The program can also handle any image in the correct formats regardless of dimensions. As long as it is below three megabytes in size the image can go through the detection process. This limitation is to save on storage space and with a larger server it could be increased or removed.

When the user submits an image they are returned both the classification and a confidence percentage. The percentage is how confident the model is in its classification. It is necessary to display as an image closer to 50% confidence might not be fake or might not be real. As mentioned previously, alongside the result is the user's image after going through the different stages of processing. With all this information the user can make a decision themselves without just being told if their image is fake or not.

The overall accuracy of the model used was approximately 90%. The paper that it was based on managed to achieve 96% so there is some work to be done there [11]. However the results that they

achieved were pretty closely replicated.

The user interface is unambiguous and easy to use. The website is not very large and gets to the point so that the user can get to what they want. The image upload is very straightforward and makes it easy for the user to get a result. As time was a key requirement the only waiting that the user has to do is waiting for their image to be processed. The time taken to load the previously created model and then classify the image is minimal. The larger the user's image the longer that the user will have to wait. However the more powerful the server that the software is hosted on the faster they will be able to get a result.

This project has managed to give the user a certain amount of confidence in what they are looking at and has made a useful technology available to everyday people.

6.2 Future Work

There are some limitations to this project. First of all, this project only handles copy-move forgery. While this is a very common kind of forgery there is a range of different kinds of forgeries and ways to change an image. The website also only handles one image at a time. If a user has a whole range of images they want to know the nature of then uploading them one by one could be very tedious.

The main future development of this project would be to train the network on other kinds of forgery. For example this project would not be able to detect if part of an image has been stretched or spliced. There are ways to identify these kind of forgeries and it would mean adapting the data set so that the neural network can identify these changes too. The more kinds of forgery that the software can handle the more powerful a tool this becomes.

Other features could also be added to the website. For example an option to upload and get the results for multiple images would be useful. As the technology is there, it would just be a case of running each image through the process then displaying all the results together. Another feature that could be of use is one where the user would enter in a website and the program could inform the user about all the different images on that page. This would be particularly useful when looking at news articles and would save the user having to download images first only to have to upload them again. Finally, it would be useful to the user if the software told them what part of the image could have been tampered with. This addition could be quite difficult to implement due to the black box nature of machine learning.

If more powerful hardware was available then the results of the project could be improved by using larger data sets. One of the limitations was that the initial data set used was too large and meant that training took a very long time. It would be interesting to compare the results of different data sets or even to combine them into one large set. Different data sets could also open up identifying other kinds of forgeries.

The contribution of this project to the field is bringing the technology to everyday people. This

is a growing field and there are many ways to identify all the different kinds of forgeries. However a lot of these technologies take a lot of research and powerful hardware in order to use them. There are also constraints with a lot of the identification techniques when it comes to the amount of time that is needed to perform them. People want to know if what they are looking at is real and this project makes one of these techniques available.

This project matters as the amount of fake news and fake image is a growing problem that is only going to get worse. Image doctoring has been around as long as images themselves and the technology used to edit them is only getting more powerful. This project helps to give the power back to the person viewing the image rather than whoever is editing the image.

References

[1] "Pintrest." *https://www.pinterest.co.uk/stealhollywood/models-celebs-without-photoshop/*. Last Accessed: 2018-10-13.

[2] "Boredpanda." *https://www.boredpanda.com/funny-donald-trump-queen-elizabeth-photohop-trumpqueen/*. Last Accessed: 2018-10-13.

[3] "Telegraph." *https://www.telegraph.co.uk/women/womens-politics/11342250/Charlie-Hebdo-Women-Photoshopped-from-Paris-rally-picture.html*. Last Accessed: 2018-10-13.

[4] "Photo tampering throughout history." *http://pth.izitru.com/*. Last Accessed: 2018-10-13.

[5] "Nytimes." *https://thelede.blogs.nytimes.com/2008/07/10/in-an-iranian-image-a-missile-too-many/*. Last Accessed: 2018-10-13.

[6] "Petapixel." *https://petapixel.com/2016/10/12/disneyland-used-photoshop-cigarettes-portraits-walt-disney/*. Last Accessed: 2018-10-13.

[7] "Alt news." *https://www.altnews.in/vicious-cycle-fake-images-basirhat-riots/*. Last Accessed: 2018-10-13.

[8] "Plotly." *https://plot.ly/*. Last Accessed: 2018-10-13.

[9] "Drawio." *https://www.draw.io/*. Last Accessed: 2019-02-28.

[10] "Deeplearning - mlp." *http://deeplearning.net/tutorial/mlp.html*. Last Accessed: 2019-02-21.

[11] S. Ranjan, P. Garhwal, A. Bhan, M. Arora, and A. Mehra, "Framework for image forgery detection and classification using machine learning," in *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*, pp. 1–9, IEEE, 2018.

[12] J. Polivy and C. P. Herman, "Causes of eating disorders," *Annual review of psychology*, vol. 53, no. 1, pp. 187–213, 2002.

[13] J. Tawse, "Consumer attitudes towards farm animals and their welfare: a pig production case study," *Bioscience Horizons*, vol. 3, no. 2, pp. 156–165, 2010.

[14] "Independant." *https://www.independent.co.uk/life-style/health-and-families/health-news/revealed-15bn-hidden-cost-of-eating-disorders-10062009.html*. Last Accessed: 2018-10-17.

[15] "Time." *http://time.com/3778075/doctored-photos-the-art-of-the-altered-image/*. Last Accessed: 2018-10-13.

[16] H. Farid, "Image forgery detection," *IEEE Signal processing magazine*, vol. 26, no. 2, pp. 16–25, 2009.

[17] G. K. Birajdar and V. H. Mankar, "Digital image forgery detection using passive techniques: A survey," *Digital Investigation*, vol. 10, no. 3, pp. 226–245, 2013.

[18] V. Christlein, C. Riess, J. Jordan, C. Riess, and E. Angelopoulou, "An evaluation of popular copy-move forgery detection approaches," *arXiv preprint arXiv:1208.3665*, 2012.

[19] C. Chen, S. McCloskey, and J. Yu, "Image splicing detection via camera response function analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5087–5096, 2017.

[20] W. Fan and H. Farid, "A statistical prior for photo forensics: Object removal,"

[21] S. Bayram, H. T. Sencar, and N. Memon, "An efficient and robust method for detecting copy-move forgery," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pp. 1053–1056, IEEE, 2009.

[22] V. Conotter, G. Boato, and H. Farid, "Detecting photo manipulation on signs and billboards," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pp. 1741–1744, IEEE, 2010.

[23] J. F. O'Brien and H. Farid, "Exposing photo manipulation with inconsistent reflections.," *ACM Trans. Graph.*, vol. 31, no. 1, pp. 4–1, 2012.

[24] E. Kee, J. F. O'brien, and H. Farid, "Exposing photo manipulation from shading and shadows.," *ACM Trans. Graph.*, vol. 33, no. 5, pp. 165–1, 2014.

[25] J. Lukáš, J. Fridrich, and M. Goljan, "Detecting digital image forgeries using sensor pattern noise," in *Security, Steganography, and Watermarking of Multimedia Contents VIII*, vol. 6072, p. 60720Y, International Society for Optics and Photonics, 2006.

[26] J. Fridrich, "Digital image forensics," *IEEE Signal Processing Magazine*, vol. 26, no. 2, 2009.

[27] “Psu glcm.” *http://web.pdx.edu/~jduh/courses/Archive/geog481w07/Students/Hayes_GreyScaleCoOccurrenceMatrix.pdf*. Last Accessed: 2019-02-20.

[28] “Skymind.” *https://skymind.ai/wiki/multilayer-perceptron*. Last Accessed: 2019-02-21.

[29] “Copy-move forgery detection data sets.” *http://lci.micc.unifi.it/labd/2015/01/copy-move-forgery-detection-and-localization/*. Last Accessed: 2019-03-26.