Plymouth University

School of Computing, Electronics and Mathematics

PRCO304
Final Stage Computing Project
2018/2019

BSc (Hons) Computer Science

Conor Jeffery Worthington

10528570

Artificial Intelligence to remodel facial emotions of an image

Acknowledgements:

Before I begin this paper I would like to thank Dr Thomas Wennekers for his constant guidance as both my project supervisor as well as my personal tutor throughout the difficult process of completing this project.

Abstract:

This report examines the approach of using machine learning techniques to aid in the analysis, processing and altering of images using filters for the explicit intent of altering emotions displayed on subjects. Investigations are carried out to examine relevant datasets, adequate neural network architectures and the resultant performance observed. Similar investigations are also carried out for the purpose of the design of the dataset to train the 'Fisherface' gender classification tool as well as to explore the best performing algorithms to implement for various other challenges such as angle estimation and image blending.

The report outlines the fundamental requirements of the application and subsequently aligns up how requirements were developed, with which technology and overall project management structure.

The report also goes into depth about conscious design choices made in both software and visually. The graphical aspect is examined with respect to normal design principles and user feedback from case studies and the changes which resulted from important feedback gathered. The software review is approached from reviewing good object-oriented choices in the design of the program and various choices which had to be made throughout development.

This report also analyses the legal, ethical and moral responsibilities that are relevant to this program – in particular the legal challenges of analysing as well as producing datasets for the purpose of training machine learning algorithms.

The report further continues with a look into a viable future approach in the form of generative adversarial networks – why they couldn't be used due to constraints on VRAM as well as training time and the challenge required in designing them as well as the theoretical gains exhibited in papers which utilised GANs for image emotion manipulation.

The report summarily ends with an evaluation and post mortem of the project which finds that whilst most algorithm selection is great and the functionality required is as desired improvements could still be made to improve network performance as well as some fundamental changes to increase performance in the future such as using face alignment algorithms or allowing for the ideal emotional faces to be dynamically changed through a file which the program reads in.

Word count: 10231

Table of contents:

1 Introduction to problem

Each and every year mobile phone users upload 24 billion photos of themselves to google photos alone (Sarah Cascone, 2016, Artnet, “24 Billion photos prove our selfie obsession is out of control”). At a minimum these photos include at least once face and it stands to reason that a large number of these photos can not be of perfect quality from either an error in the person taking the photo to a person conveying the wrong emotion within said photo.
One application which makes use of this obsession to take many photos of ourselves and share them with friends is the social media platform snapchat. Snapchat has over 100 million active daily users with over 10 billion video views per day with a large percentage of them involving what snapchat coins ‘filters’ (Dara Fontein, 2016, hootsuite,”The Top snapchat statistics you need to know for your business”). These ‘filters’ are automated image improvements that account for the face and make subtle adjustments – they can be used to make a face more expressive – which is useful if the face is exhibiting an emotion which you don’t wish to be conveyed in your photo.

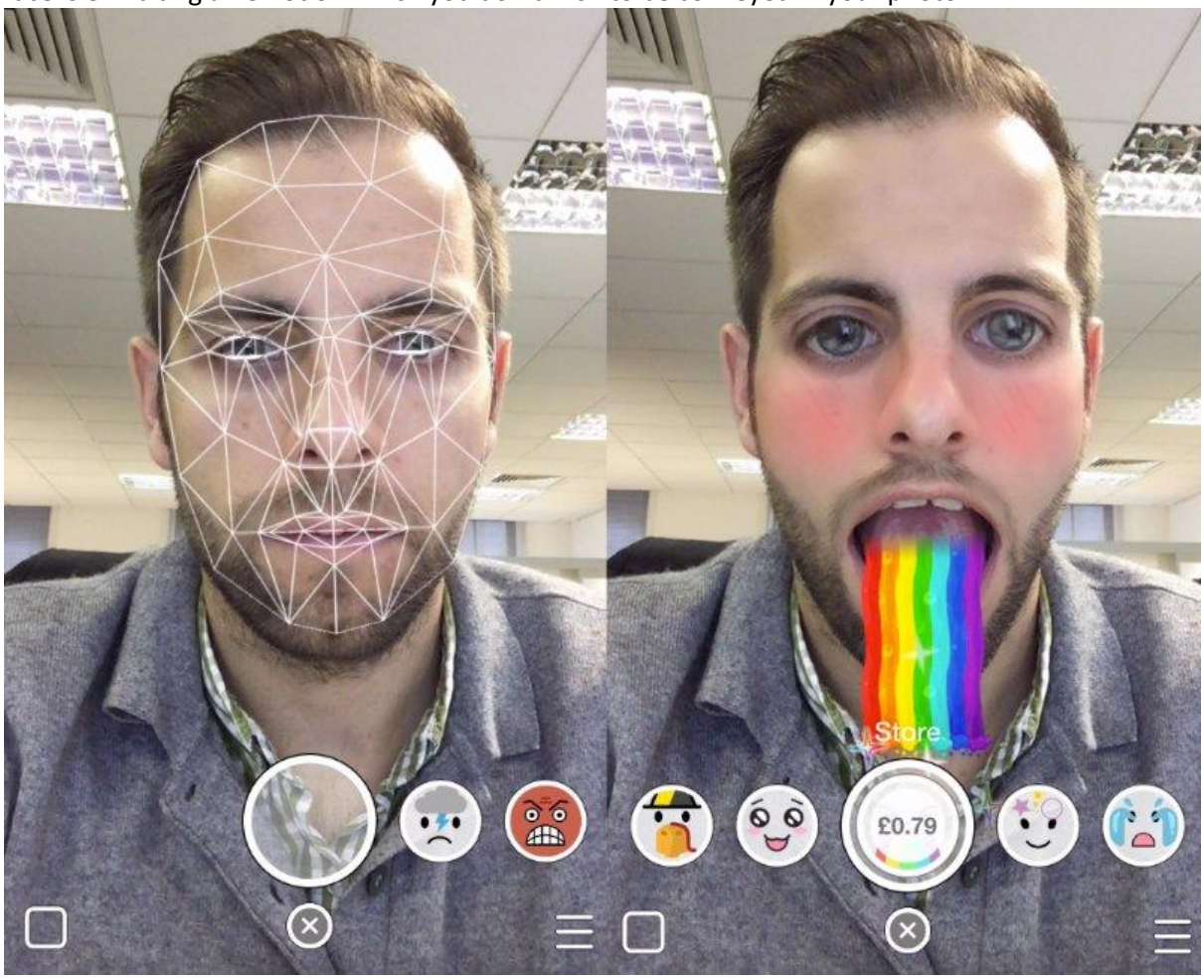


Figure 1 – an example of snapchat ‘filters’ being applied (Kayla O’Shea, 2017,Medium,”How snapchat filters work”)
This notion of snapchat filters being used for emotional transformation purposes is noted in one article which states that “we want to be funny, express emotions and show our attitude to things. Face filters allow doing that better than words” (Banuba, 2019, Medium, “5 types of Face filters like Snapchat that engages users best”). As such there is a clear market for filters which can alter faces (and by extent their emotions) but currently snapchat does not currently possess anything which is

photorealistic – currently filters are just comical or can not yet provide substantial enough change to fix an image. As of right now that would require a vastly more complex applications such as photoshop which requires expertise and a large amount of time.

It must be stated that to many the 'performance' (social media likes etc) is very important – in this social media age one study found a strong correlation between performance of a 'selfie' (i.e a photo of ones self) and increases in stress displaying in coping behaviours (Pengxiang Li et al,2018, Telematics and Informatic, "Likes" as KPI: An examination of teenage girls' perspective on peer feedback on Instagram and its influence on coping response). As such having the ability to photo realistically transform an image is something very attractive to younger audiences who are heavily influenced in how they portray themselves in photos by social pressures.

Another study found that when portraying themselves in 'selfies' users tended to bias the left cheek 41% more (Annukka K. Lindell,2017,frontiers in psychology, Consistently Showing Your Best Side? Intra-individual Consistency in #Selfie Pose Orientation). The same paper goes on to cite numerous other papers which discuss how the left cheek is more emotionally expressive than any other head orientation and discusses how this could infer a need for emotional expression in selfies and how we wish to convey emotion when taking them. As such there is a clear need for a good quality photo which will garner more positive attention on social media to have a clear emotional expression.

Overall this drive to try and improve the selfie using machine learning is currently being pressed – in one example a network is trained to identify the optimal crop as well as the projected quality of the image out of 100 (Andrej Karpathy, 2018, GitHub, "What a deep neural network thinks about your selfie"). Whilst this example is ultimately very brief it shows there is a methodology towards predicting and moving towards higher quality images which will garner more positive feedback.

As we can see that emotional state can influence the positive feedback of an image and how users ultimately try to portray emotion in images – as well as the importance of positive feedback it's important to garner an understanding of emotional states – in one paper a leading researcher in the field Paul Ekman states that we should 'Consider emotion to be discrete states' (Paul Ekman, 1993, American Psychologist, "Facial expression and emotion") whilst this paper was very early on – it was the first to state that emotional states could be universal and non verbal with all cultures being able to fundamentally understand some discrete emotional expressions on faces. In a recent study to gauge current scientific consensus amongst psychological researchers it was found that 80% of respondents to the journal were in support of universal emotional signals which could be interpreted across cultures (Paul Ekman,2016,Sage Journals, "What scientists who study emotion agree about"). As such it stands to reason that this is now a fairly normal view to hold on emotion and it could allow for both classification of emotional states as well as transformations into new emotional states. Fundamentally these emotional states are described as the Ekman categories which are anger, disgust, fear, happiness, sadness, surprise (management Mania, "Six basic emotions"). This doesn't include a neutral state which is worth considering as a face at rest is a genuine state albeit not necessarily emotional.

One early paper which attempts to deal with the problem of emotional classification by decomposing the face into various emotional regions and subsequently utilising optic flow to analyse the motion of said regions when expressing an emotion (Gianluca Donato et al, 1999, IEEE Transactions on pattern analysis and machine intelligence, VOL 21, "Classifying facial actions")**.** This data was used to predict

the probable emotional state given the changes observed, this method saw results which whilst not very good were above average and certainly good considering the time. Fundamentally this research indicates that various components of the face or regions are key to expressing certain regions – if they were not this research could never have found any correlation and gives rise to the notion that should a transformation occur on said regions it could result in a change in perceived emotional state.

As such from this we can see that there is fundamentally a demand for users who wish to modify their images prior to sharing them with peers as the perceived increase in emotional expressiveness can increase the 'likeability' of an image. As such by aiming to provide better transformations, which are subtler and more realistic a user's image may look more genuine and garner even better reactions from peers – one focus of this should certainly be to focus on targeting Ekman's categories. By targeting these distinct emotional states, we can take pre-existing emotional states for key regions of emotional display on a face and seek to apply them to images as improvements – in theory this should be able to perform a similar purpose to snapchat 'filters' in that we are adding an emotion to an image but do this in a simplistic and far more realistic manner.

2 Project management:

2.1 Agile

Over the course of the project Agile principles were applied to assist in the production of the software. Agile focuses on iteratively building up functionality – focussing on building working components rather than a section of the software at once. As Agile does not require a distinct project manager it requires as a developer the continued use of project planning and making use of regular testing to ensure that functionality is hit whilst working through sprints. As this is a solo project this is rather fitting as I will be responsible for my managing my own time so any other development techniques such as waterfall which assume a dedicated project manager may not be as viable.
The selection of Agile is also supported by professional findings in one paper how Agile principles when applied "can be used to increase throughput and reduce lead times for customers as well as to provide a visual means of tracking progress and the status of requirements "(Des Greer, 2011, Wiley Online Library, "Agile Software Development").
As such focussing on an Agile methodology and quickly focussing time on development rather than a lengthy setup stage will aid in maximising developmental time which should resolve most user stories if not all.

2.2 Scrum

A framework I subsequently chose to make use of throughout my implementation of this project was Scrum. The scrum framework was chosen as it is the most professionally utilised Agile framework (Margaert Rouse, 2011, tech target, "Agile project management")and I have experience from working with it on prior projects.
Some principles of Scrum were also assessed by being implemented and evaluated alongside CMMI in a series of software development findings – overall the paper goes on to state that the companies found that 72% of the suggested practices from the mixed scrum model were beneficial towards software development (K. Lukasiewicz et al, 2012, European Systems and Software Process improvement and Innovation, Improving agility and discipline of software development with the Scrum and CMMI).
One of the key steps of scrum which was implemented was the utilisation of user stories – by creating a series of user requirements and thinking through what a user would desire was immensely useful. This then allowed for the production of far more accurate requirements which could subsequently be classified by development priority which benefitted the development process greatly.

2.3 Prince2

One of the further project management methodologies implemented was Prince2. Prince2 is a management schema that intends to drive emphasis on dividing up the project into clearly defined stages. By dividing the project up into clearly defined stages and subsequently trying to build functionality incrementally as in line with Agile any short falls or areas which are dragging are identified and work can be appropriately scheduled and dependencies managed.
Furthermore shortfalls are identified more easily as the deviations from the plan are noticed and development time can be dynamically re-allocated as dependencies require. This is done at stage boundaries where the progress towards specific stories is reviewed and subsequent planning is done to ensure development stays on track and that dependencies aren't missing in pursuit of requirements. It also allows for similar requirements to be timetabled so they're developed together to potentially alleviate problems throughout the development process. This was helpful during the neural network training portion as once one network was implemented and trained it was trivial to re-apply the same techniques to the other network massively accelerating development. Throughout this project there were 3 project boundaries and subsequently 3 reports which were written.

2.4 Technology choice

One of the first tasks in managing this project was the choice of technologies with which I would carry out this task. As such I compiled a very in-depth review of the various technologies which would allow me to build a viable solution. In the end after reviewing all of my available options for my development language I opted for python on version v3.6 as this would allow me the most compatibility with libraries, the easiest installation of libraries and a very good language for machine learning as well as computer vision without risking the language becoming depreciated anytime soon. As for IDE I selected visual studio on my main workstation as it allowed me to quickly make use of PyPi to setup my necessary environments, I had it installed already and I knew from previous projects I could visualise my GPU utilisation clearly whilst training my neural networks. It must be said that as I develop on multiple systems and I predominantly bar training do work on my laptop the vast majority of development was done on PyCharm due to the superior terminal support and the fact the Visual studio has no stable release on OS X with full functionality.
As for the machine learning framework I originally chose TensorFlow as I have wanted to learn it for a very long time, however as I was aware of how scheduling was working within a couple weeks of development I swapped TensorFlow out to be my back end and develop my networks using Keras. This was a pragmatic decision aimed at quickly building as good a product as possible, had this decision not been made a lot of functionality could now be missing and the benefits of using TensorFlow were ultimately not capable of being utilised by a novice such as myself – things like the TensorBoard debugging really were not beneficial.
For the GUI I originally chose Tkinter as it is by default packaged with Python, whilst I did briefly become sick of Tkinter and how slow it is to build a UI in especially a well formatted UI at which point I experimented with Wx. I subsequently found that Wx was not any more beneficial than Tkinter in terms of beauty and would ultimately make installation and turning the code into a single application vastly more challenging as well as producing bugs with IntelliSense on visual studio. As such I reverted to Tkinter and accepted its flaws.

3 Project objectives:

3.1 User stories:

As the project has no definitive user, requirements had to be elicited by background research on social media users as well as from personal experience and expectations I would hold as I myself am an avid user of social media who does not have the capability to use complex photo editing software. In keeping with agile principles these were first considered as user stories.

3.1.1 User story

The user stories as said were identified by building up reasonable assumptions about expectations from this system based off a middle ground between applications such as snapchat and photoshop. This produced a set off expectations on how a user would expect the application to perform as well as the features a user would expect. This culminated in a series of both functional and non-functional considerations to be generated which ultimately assisted in generating the requirements for building the software.

3.1.2 Machine learning story

The machine learning story was created to ensure that reasonable assumptions were being made to ensure a functional environment for the agent and that any requirements that could be ascertained to ensure the best performance were taken into consideration. Ultimately the exercise did not contribute much to final requirements in either functional or non-functional requirements but did however prove a valued exercise to ensure proper data set selection and evaluating the neural networks performance.

3.2 Requirements

With the user stories ascertained it seemed both pragmatic and in keeping with good agile practices to produce a list of both functional and non-functional requirements. These requirements were then subsequently organised using the MOSCOW system to organise importance. This system sub organises the requirements both functional and non-functional into must, should, would and could. This was key in planning as it allowed for the development of an essential minimum viable product first before subsequent improvements were made.

4.0 Software Design
4.1 User interface design:
As this product has the fundamental aim of intending to make complicated facial transformations into a simplistic process which normal social media users could make use of it is of great importance the user interface is not just well designed but easy to use. Some of the best guidelines come from Ben Schneiderman in his book 'Designing the user interface' where he outlines eight rules to design a user interface that is both effective and easy to use. These rules were subsequently kept in mind whilst designing the user interface and proved an effective guideline to follow.

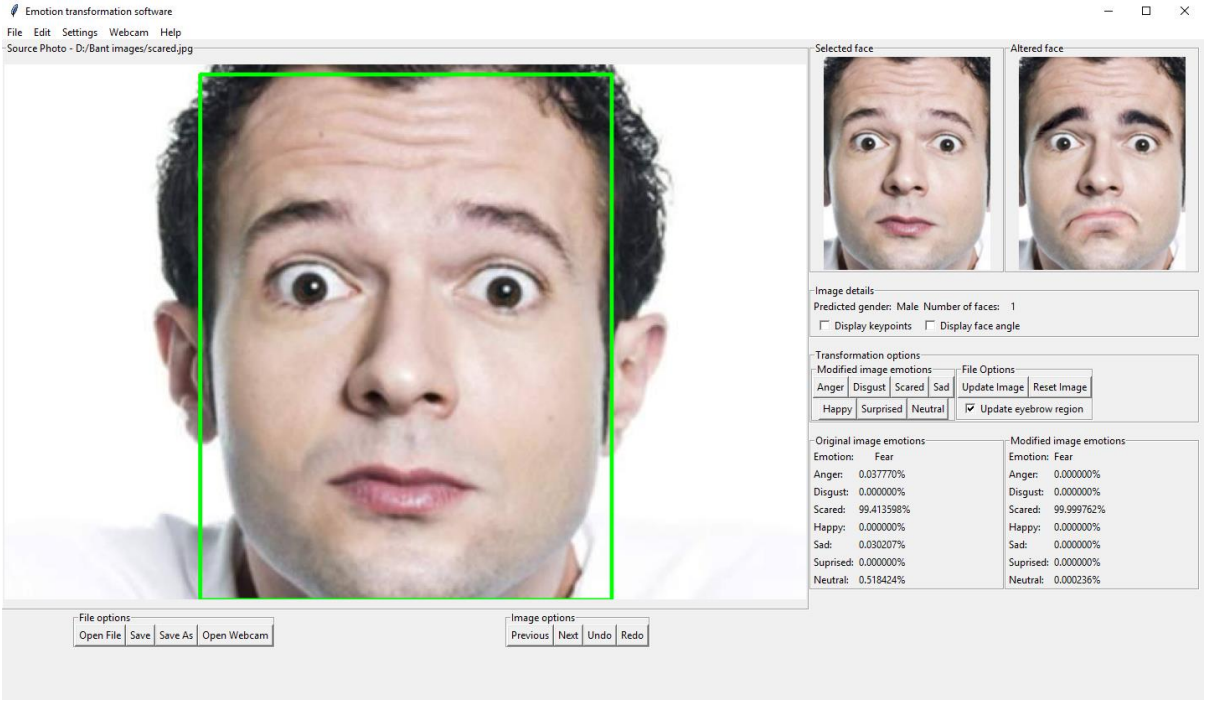


Figure 2 – an example of the GUI in use

The first of the golden rules is to strive for consistency, this was applied in a multitude of ways. The first was to keep to design guidelines, this enabled a familiar toolbar to designed – as most users natively from Microsoft software packages would recognise the use of 'File', 'Edit' and 'Settings' they would be instantly familiar that within File there would be further options for both saving and opening new files. Another design feature which keeps in line with the first feature was the selection of shortcut keys – by electing to have the shortcuts remap based on the operating system users will already be familiar with commands such as CMD-S to save on Mac and Control-S to save on windows. This familiarity should once more be consistent with the rest of their computing experience and be easy to pick up.
One final design choice pertaining to consistency is the retention of the user interfaces layout throughout rescaling and the retention of the proportions of the images that are loaded. This is quite key as when a user is using a widescreen display with a wide aspect ratio the images loaded should not feel stretched but instead retain the aspect ratio they originally had so the modifications can be seen as they will ultimately appear. Furthermore the layout has to retain its consistency whilst the window is resized and the image retains its aspect ratio which is done using padding to avoid any buttons or information moving around when the user least expects it.
The second of Schneiderman's golden rules is to enable frequent users to use shortcuts – this is something that has already been touched on in that these shortcuts are designed to be OS specific. There are currently shortcuts for all file operations, undo, redo, reset altered image and open webcam. Whilst there are more buttons which could be automated to have shortcuts – options such as the user

guide most likely won't need a shortcut as it's for novices and implementing the shortcut could frustrate users who press it due it opening up the browser tab.
The third of Schneiderman's golden rules is to offer informative feedback. This is done through the software by every action having a visual queue – either through a popup message to alert the user to an error by using a plain text explanation of what's been done wrong or by modifying an image or region to display what the user has requested. As a result of this the user is perpetually fed-back the changes they implement in the software – through either asking it to now display key points or displaying the new face they just loaded in.
The fourth of Schneiderman's golden rules is to design dialogue to yield closure. This means when a user completes an action – something will happen – this is managed very well in my system with every single option either invoking a graphical queue immediately or where impossible such as the draw eyebrows flag it invokes a message to alert the user that when they next modify an image the eye brows will be drawn as the user specified.
The fifth of Schneiderman's rules is to offer simple error handling. Whilst the software is designed to minimise errors – when errors do occur the software uses a default python error message and alerts in plaintext what the error is and a potential solution. One example of this is the webcam – when attempting to open the window if no webcam is inserted or detected by OpenCV an error message will alert the user to this and suggest they insert one or reseat the cable. Another example is when either taking a photo or loading one should a face not be detected the user will be alerted and instructed to either take a new one or load a new one.
The sixth of Schneiderman's rules is to permit the easy reversal of actions. This is handled in a variety of ways in this program. The first is via a conventional undo and redo button combo. This is done so that when a change is committed to the main image if when given the context of how the face looks in its full surroundings its not liked it can be removed – but also re-added should the user feel it was actually beneficial. This is also further supported with the reset image button – whilst the altered face should reset with every emotion and it won't commit any changes until told to – some users find it comforting to be able to reset the face to the before face to allow for more comparison or before they carry on with adding new transformations.
The seventh of Schneiderman's golden rules is to support the internal locus of control. This is defined as gaining the trust from users that the application will behave as they expect. All in all this is done quite well by having every single available action instantly giving a response without delay- even if it's an error the user will be instantly alerted and if something changes the screen will instantly change to display the new information without removing any information that will be key to going forward.
The final one of Schneiderman's golden rules is to reduce the short-term memory load. This is done in a variety of ways, the first and foremost is the usage of before and after when performing transformations. Whilst modifying a face you can see a before – the face as you loaded it, altered – the face as you have changed it to be and the main image which will display the original face in the context of an image.
During the process after a sufficient enough UI was designed and implemented the decision was made to get peer feedback to attempt to further improve the UI and see what flaws average users had. As such I contacted 5 of my peers and had them attempt to complete a task sheet that I considered a typical but heavy user flow through the program. This thankfully did bring up some very glaring flaws in the UI.
The first was in my initial design both the main image had an undo option and the altered image had a redo option as opposed to a reset image option. As a result of this when users had to undo their changes to the main image as the cursor was already on the upper left after committing changes every single user incorrectly used the wrong undo button. This was very confusing for them as there were no changes currently committed to the altered face and as such an alert came up saying there was nothing

to undo. As a result this was resolved by changing the undo redo combo of the altered image to a simple reset button to alleviate any confusion.
Another criticism of the UI that was levelled occasionally was the Ui felt very sparse and that the toolbar felt underutilised. This was a very fair issue as in the initial design I hadn't made use of 'labelframes' and as such the buttons would space out as the images would expand leaving large empty gaps with nothing between buttons that should be grouped together. By implementing label frames and subsequently grouping items the UI feels far 'busier' alongside the tool bar which now has far more functionality added.
One final critique of the UI which was subsequently fixed was that when a user loads a file or commits a save as the file name is not displayed. As by default there is a save option the user at times struggles to know if they're saving over the original file or the file at the newly specified destination. To help resolve this issue there is now a permeant file name for the image that's currently loaded in the main image label frame.

4.2 Software design:

As python is a language which supports object oriented programming it made sense to utilise this when developing my software. As such the first of my considerations was making sure my code was in line with the principles outlined in "Design principles and Design patterns" (Robert C Martin, 2000, Design Principles and Design patterns) as this is the paper in which the idea of 'SOLID' code originated. Solid code means code which keeps to the following principles:

Single responsibility
Open/Closed principle
Liskov substitution
Interface segregation
Dependency inversion

As my program follows a very simple design pattern in the sense that it has a singular main application class which handles commands from the GUI as well as having two classes below it which are solely to build neural network objects and provide predictions. As such a large number of the designs for good OO code are not relevant as there is no inheritance present.
The code does however show a good respect for the single responsibility concept as the main class is to simply handle GUI events and subsequently make use of the neural network classes to build projections before modifying and updating the GUI. In the codes initial design, the program initially had a separate graphics class that was far more OO but unfortunately as Tkinter has no resize event which can be overloaded as such this required a timer thread to be invoked. Unfortunately Tkinter does not support multiple threads and this lead to periodic seg-faults which lead to the discovery that the timing thread was responsible (Official TCI Documentation, TCI-WIKI). As such the graphics code was separated into a global invocation in main as it must be managed using a polling system inside the main class. This did however give the advantage that the same polling system can be used to manage the popup window for the webcam allowing for video capture timing to be easily managed without invoking another timer. This ultimately results in some fairly clean code, although the graphics class could do with some cleaning up by incorporating it into the handling class and moving out a lot of the processing, this just seemed very unreadable.
The code does also follow the Open closed principle as whilst you can invoke the classifier classes you can not simply modify them or their models inside. As such this aspect of the code is in accordance with good OO practices and should be very easy for any developer to maintain. Furthermore, as the code is so separate and not modifiable should a change ever be needed to be made to any of the models only the source file and the actual model definition on creation needs to be changed, everything else should be dynamically handled. This will allow the code to get better as classification challenges become easier over time as new techniques are developed.

5 Algorithms used:

5.1 Face recognition:

The first algorithm which was made use of is called MTCNN which was first defined in the paper "Joint Face Detection and Alignment using Multi-task Cascaded convolutional networks" (Kaipeng Zhang et al, "Joint Face detection and Alignment using multi-task cascaded convolutional networks"). This algorithm is used for the purpose of detecting faces within the input image and building up arrays of the boxes which bound around them. The network works by detecting a face using a large bound before regressing the box inwards as it detects features within the face to form as idealistic a crop as possible.

Ultimately this approach was chosen to be pursued because when compared to approaches such as Haar cascades it is consistently superior. This is seen in one paper which produces findings that Haar cascades detect roughly 82.6% of faces where as the MTCNN library which has been implemented in the project has a detection rate of 89.85% (Tanin Phongpandecha, 2018, datawow, Face detection: Haar Cascade versus MTCNN). Another finding in the paper is that the Haar cascades algorithm detected almost 950 extra faces where as MTCNN detected about 400 extra faces.

All in all this was the optimal choice of a face detector as it's both more accurate than a conventional Haar cascade detection method and is more precise.

5.2 Gender classification:

The second algorithm implemented was a classifier for gender. One of the easiest to implement and fundamentally most effective ways to provide a binary classification of faces was to make use of Fisherface classification. Fisherface is an algorithm which takes in a training set and finds a subspace which represents most of the data variance. As such the Fisherface attempts to build up a heatmap of sorts of components of the input data to find similarities and build up averages which ultimately can be compared to an input face and return a confidence scoring. In the case of gender classification by having two classifications and choosing the classification with the value closest to 0 (i.e the most confident value) a gender can be estimated.

Ultimately this implementation was not the best implementation performance wise – something such as a neural network could perform better however this would have required a far more vast dataset than the one I built furthermore it seemed overly intensive compared to a Fisherface approach as this method is called for every face that's detected when an image is loaded – if too much computation is added this could make the program feel slow and unresponsive. Therefore when considering other implementations have a peak accuracy in the mid 90% range (Tian-Xiang Wu et al, 2012, Neural Computing and applications, Multi view gender classification using symmetry of facial images) even with occlusions on the face or noise it seems reasonable that this technique is sufficiently accurate at providing an estimate at the faces gender to decide which features to subsequently blend.

5.3 Face angle estimation:

The third key algorithm which was implemented was the choice of the solve point and perspective algorithm. This algorithm was implemented to draw a two dimensional representation of where the subject is facing. This is important as it can provide a more accurate sense of how accurate the key point calibration is – as assuming the face be similar to the 3d model I found online the estimation of angle should be relatively accurate. As such if the angle is massively off it is a good indication that the key point model is going to perform very poorly, conversely a good fit is indicative of a face that can be transformed very accurately.

The overall model and tricks used to gain estimations of the focal length of the lens (as this code is not camera specific this is necessary) are from a tutorial (Satya Mallick, 2016, Learn OpenCV, Head pose estimation using OpenCV and DLIB) as I did not have the sufficient means to accurately gain a 3d model estimate for a face I know to be average.

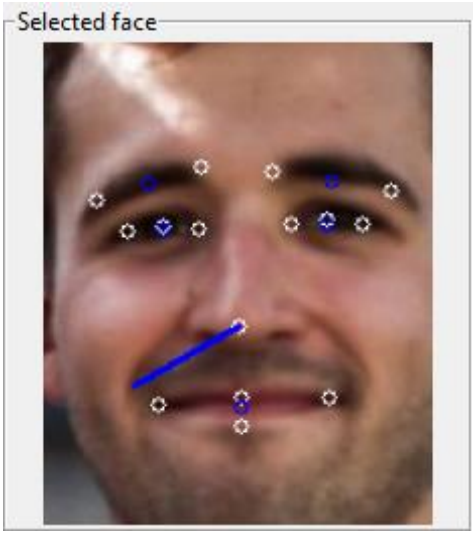


Figure 3 – an example of a successful face angle estimation being produced from good key point data

Overall this method is probably the best way to provide more visual information to the user (through the form of the angle estimation) whilst also providing a good estimate of how well the key points fit. Overall the only other way to resolve the faces angle without training an estimator would be to use a technique such as homography which requires multiple images and a very highly tuned model of the camera intrinsic which is simply not feasible for this project as users will only want to upload a singular image and modify in a simple manner.

5.4 Poisson blending:

To achieve the actual transformation a blending algorithm is utilised named Poisson blending. This is currently implemented via the OpenCV library as after sifting through documentation I found it being utilised under the name ‘seamlessClone’ which is the same in all but name. The algorithm works by using a mask on a source – this mask marks a region for extraction – which is subsequently remove from the source and the features are extracted. This is done by utilising a field of vectors which identifies a domain of values which are to be extracted and a domain of values which can be disregarded / have the opacity increased on dramatically to blend.

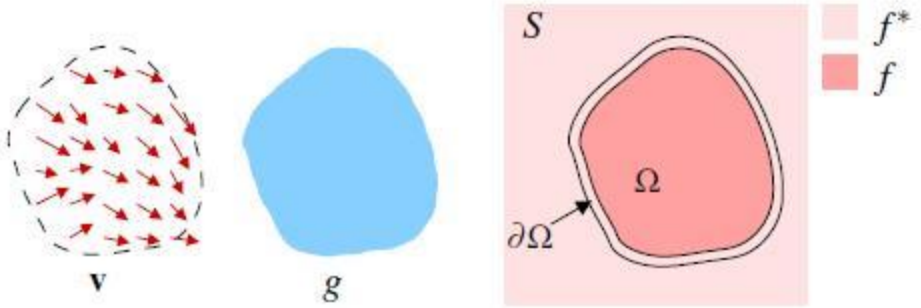


Figure 4 – The Poisson blending algorithm visualised with the mask and feature extraction being shown (Eric Yuan, 2013, Eric-yuan.me, Poisson Blending)

Overall in OpenCV there are two implementations of Poisson blending one is called mixed clone and the other is called normal clone. Whilst mixed cloning in theory should provide the best transformations as it attempts to more aggressively control the blending it ultimately to aggressively retains features from the original image – this leads to artefacts such as part of a mouth existing behind a now smiling one. Overall this is less than ideal so a normal clone is used with a slightly oversized mask for each region that is being transformed so as to ensure the initial region is reset – this is achieved in part by scaling the initial image and by extension its mask to match the new one.

5.5 Convolutional networks:

To achieve both the estimation of the key points and the estimation of the current emotional state a series of convolutional neural networks were utilised. Convolutional networks work like any other deep neural network but with the addition of convolutional layers which assess clusters of pixels and subsequently build up weights based off of the regions using edge detectors. These models are currently the best models for assessing images effectively and relating it to be abstract classifications such as gender or mapping regions.

This approach has been specifically utilised and examined many times before for key point classification, one such paper found that by taking a slightly altered approach and training networks to separately identify regions they got optimal results – this paper found a range in error from an RMSE of 6.87 to 0.37 (Naimish Agarwal et al, 2017, arXiv, “Facial Key points detection using deep Convolutional neural network”). Fundamentally this network still struggled with rotating faces and wasn’t able to deal with a face tilting too far.

The approach of estimating the emotional state of an input image has also been utilised in many papers – in one such paper an accuracy was found to be of about 71% for outputting the correct emotion (Octavio Arriaga, 2017, arVix, Real-time Convolutional Neural networks for emotion and gender classification). It should be noted that these emotional classifiers including the one implemented in this project tend to use SoftMax output and as such return a confusion matrix showing an increase in certainty as well as accuracy over time. An alternate approach this convolutional method as outline in another paper is to make use of the Fisherface classification system but for each of the Ekman categories. This subsequently results in significantly worse performance, circa 57% (James Pao, 2014, Stanford, Emotion Detection through facial feature recognition), which is supporting evidence that the choice to make use of a convolutional network in this project was a good one.

6.0 Neural networks:

6.1 Key points detection network

The first of my neural networks to build was the key point detection network. This network works by taking an input of a cropped face and summarily outputting points for various locations on the face.
It was natural that this network was convolutional the main choice was the selection of a dataset. The first dataset I found was named 'i.bug300w' – this dataset had the distinctly good property of having 32 marked data points per image and fairly high resolution images. Unfortunately this dataset open further inspection was designed to be incredibly challenging and not ideal for training this network on. For one there was a varied amount of faces per image which made parsing the data harder, secondly the colour scheme was varied per image and thirdly the images weren't all distinctly aligned.
This subsequently led to me discovering the dataset which did get used which is the Kaggle facial challenge dataset, this dataset contains around 3000 faces. Unfortunately some of these faces are not fully labelled as such some code had to be written to exclude these from the training – with this complete there were still 2100 faces with 15 labelled data points. These images were uniformly 96x96 and were very much all aligned with some minor alterations here and there to assist in training.
To further assist in training and to make sure I could evaluate performance I used the inbuilt 'test_traint_split' method from scikit learn which enabled me to quickly build a test dataset which I could use to evaluate the performance of the model on unseen data.
The size of this dataset was also perfect, as my GPU 'only' has 6Gb of VRAM of which only 5 is available at any given time this network does manage to occupy almost all of it but doesn't cause a VRAM bottleneck and as such maintains 100% GPU utilisation allowing the epochs to all be sub one second.
Another trick which I used to optimise the data was to ensure that data was frequently randomly organised in the process of training – by making use of this shuffle function that's built into keras I could help ensure that my network was not overfitting to my specific dataset and the real world performance would still be similar to the training set.

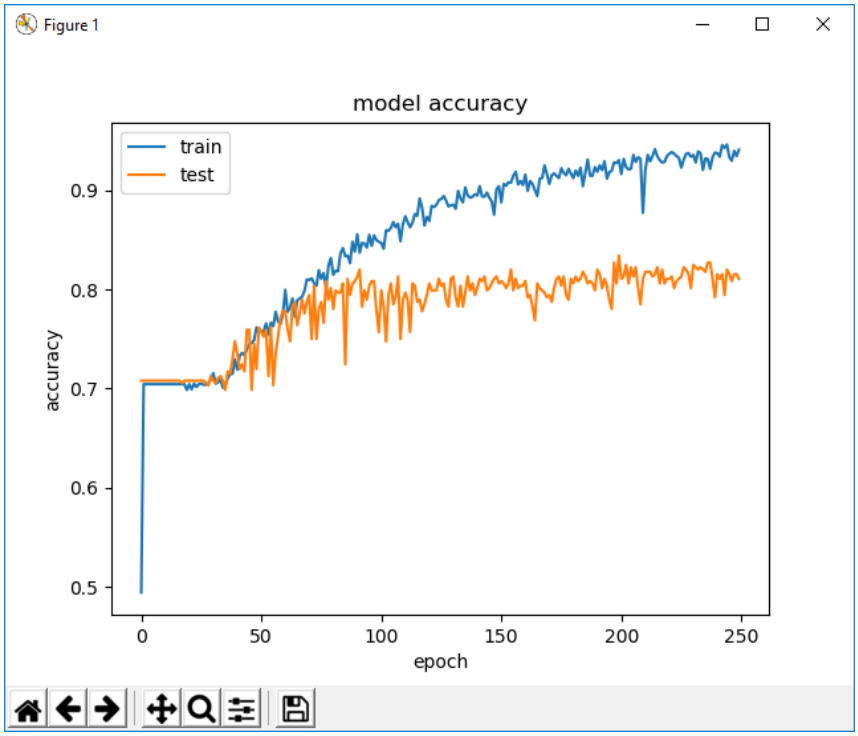


Figure 5 – Training graph for Key points neural network with regards to accuracy versus epochs

Overall we can see this was a success with a terminal training accuracy of about 95% and with a terminal test set accuracy performing at about 80%. This means the network is making fairly accurate estimates of the co-ordinates and has successfully learned.

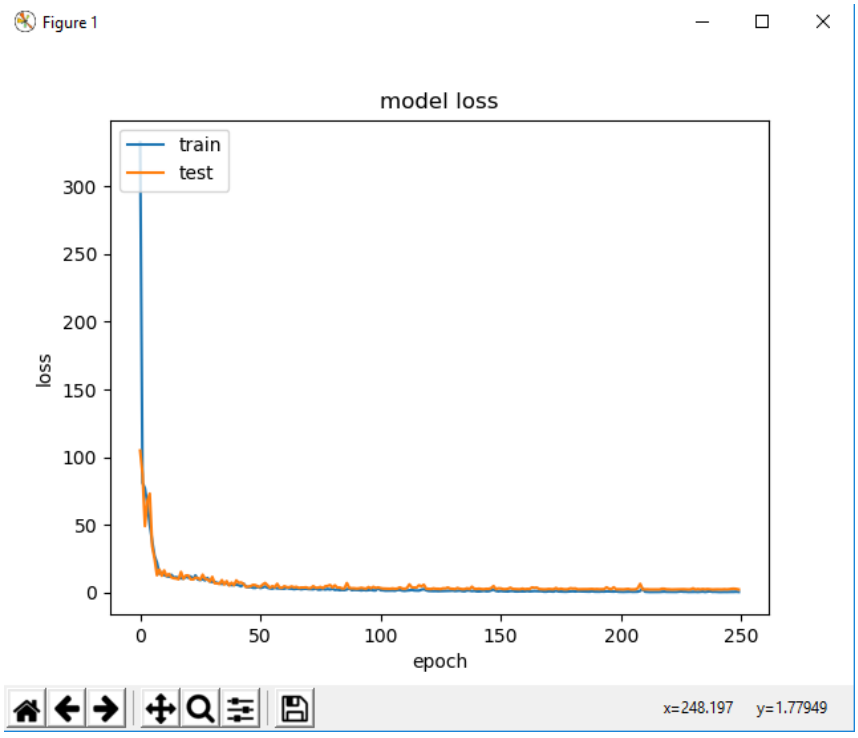


Figure 6 – Training graph for key points neural network with regards to loss versus epochs

When we look at the loss we can see the network is performing exceptionally well after about 50 epochs and is not overfitting as both the training dataset and the test dataset are getting exceptionally low scores with an error of about 1.8 as mean squared error. For reference the paper previously

mentioned scored an RMSE ranging from between 6.87 and 0.37 (Naimish Agarwal et al, 2017, arXiv, “Facial Key points detection using deep Convolutional neural network”)which whilst it was on a different dataset is a vastly worst performance than the one exhibited by this network.

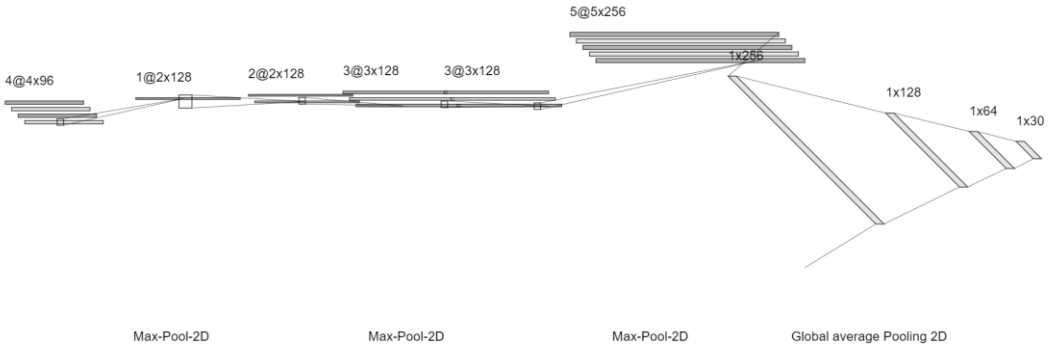


Figure 7 – Network structure diagram for the key point detection network

Overall this model seems very complex but it really isn’t. The model is a series of convolutional layers of different sizes. Several of the convolutional layers are doubled up to try and encourage increased edge detection performance prior to any pooling operations. One of the most important features if the global average pooling. This causes a performance uplift of over 25% with regards to accuracy – it takes a prior consideration of all convolutional operations prior and subsequently targets the maximal areas that have been found on the image across all edge detectors of all sizes. This subsequently feeds into a fully connected network that is 256 into a 128, into a 64 and finally into a 30x1. This terminal point is subsequently treated as a 15x2 array which as such corresponds to the dataset which has 15 points each with an x and y co-ordinate.

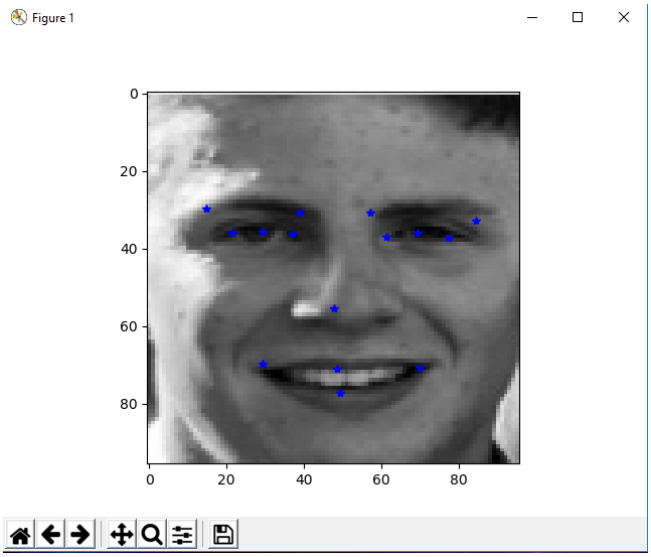


Figure 8 – Output of key point network estimations on input face

Overall the network is then subsequently tested after training on an image from my phone. As we can see the network has learned very well and has subsequently plotted fairly accurately where we expect the eyebrows, eyes, nose and mouth to be.

6.2 Emotional Classifier

The second of my machine learning algorithms is my emotional classifier. To train my emotional classifier I went in search of data which corresponded to the Ekman categories. One such success which I found was a dataset hosted on GitHub by a professor which contained approximately 500 images of celebrities which are grey scaled and aren't facing head on consistently. Whilst this dataset did pose the advantage that it had the images saved as images and wouldn't require a manual conversion to 'numPy' arrays reducing code complexity when parsing the data it unfortunately did not detract from its negatives.

To begin with this dataset is simply too small, 500 images of which aren't very evenly distributed and of faces which would be difficult to process as certain regions at times will be specifically occluded from view.

As such I subsequently found a second dataset which is far better, the new dataset seems to have a relatively even distribution of all 7 of the Ekman categories with a sum total of 35000 images. One of the main issues with this dataset is that as it is exceptionally large it tends to over encumber my 6Gb VRAM GPU as quickly it reaches 100% memory usage and the subsequent GPU utilisation drops to between 50% and 65%. This then leads to 16 second epochs but thankfully this network requires relatively few epochs (5) to build a successful network.

This dataset is subsequently split into a more manageable 28000 images for training data and a testing set of about 7000 random images – this should lead to about 1000 images of testing data for each emotion – which should provide a relatively accurate indicator of performance.

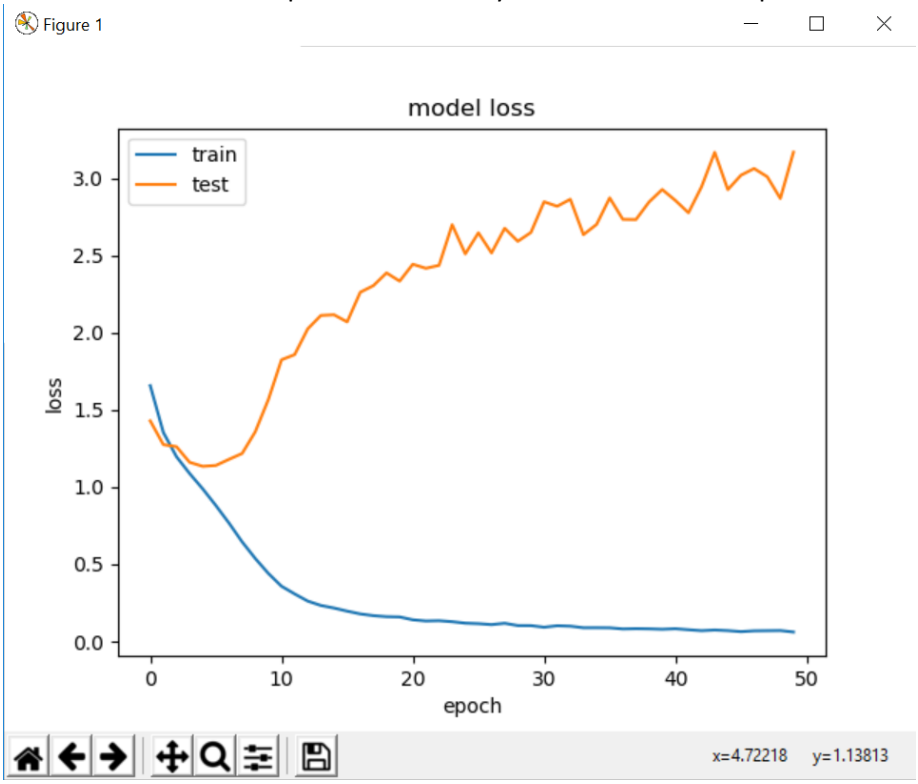


Figure 9 – Training graph for emotional classification network showing loss versus epochs

Overall with such a large dataset and only 7 possible outputs it's important that I considered overfitting – whilst there are methods in place such as shuffling of the datasets to try and prevent the training dataset being overfitted we can see that around 5 epochs that network start to produce larger losses for the test data set which is indicative of the data set being overfit. Thankfully the dataset is saved with the lowest loss recorded of the test set – so after epoch 5 the model is no longer being updated.

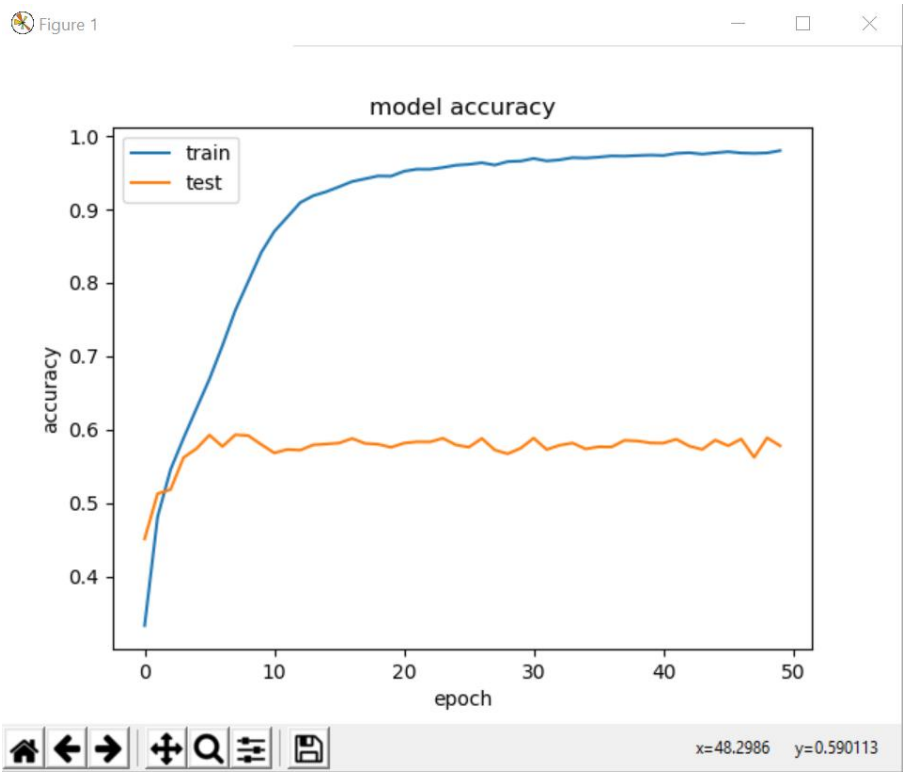


Figure 10 – Training graph for emotional classification network showing accuracy versus epochs

The model when looked at with regards to accuracy can subsequently also be seen overfitting after about epoch 7. We know this as whilst training performance continues to increase greatly until eventually going parabolic at about 95% the test data set remains at about a peak of 60% accuracy. Overall though a real world classification rate of about 60% is very good considering many humans would struggle to achieve perfect accuracy.

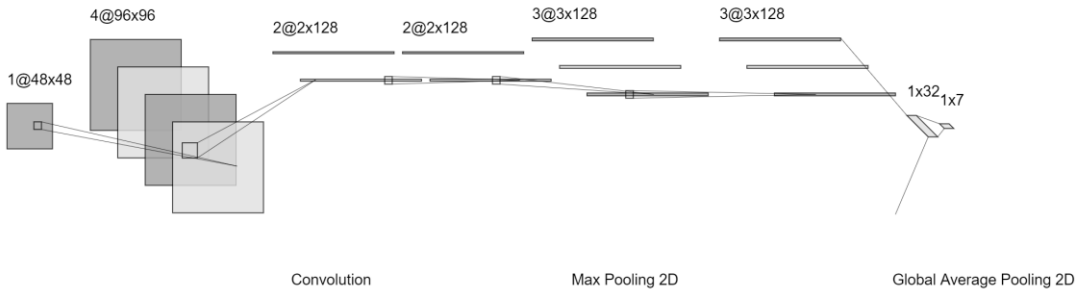


Figure 11 – Emotional classification network architecture

Overall the network architecture is a simplification on the existing architecture from the key point classifier. By retaining the basic structure of the existing network we already have utilised the doubling up of edge detectors per max pooling and the retention of the global max pooling feeding into the dense network. Like the other network this model makes use of an Adams optimiser but unlike the previous network the output is SoftMax meaning it outputs a confusion matrix of the confidence it has of the input being that image. One other difference is unlike the key point classifier the network makes use of categorical cross entropy which dynamically increases loss as the correct prediction confidence diminishes.

7.0 Fisherface:

The Fisherface algorithm trains by making use of datasets which are already categorised, unlike previous examples such as the neural networks the dataset does not have to be a vast one and can infact be under 100 images. This as such presented an opportunity to produce my own dataset to train the Fisherface algorithm (also because a good one could not be found on the internet).

As Fisherface attempts to build up a model of how a face looks I had to make a series of conscious decisions to try and build a good dataset. First and foremost when selecting males I tried to keep facial hair at a minimum- by doing this I helped ensure that the Fisherface wouldn't inherently assume all men had facial hair and lose confidence in any man without any.

Secondly I tried to go for faces which were very distinctly male or female, by not using faces which looked even slightly androgynous it keeps the dataset more stereotypical which is easier for the Fisherface to pick up traditionally masculine and feminine traits from.

The third choice was to make sure that every image had little to no angle. This means making sure that images are face on with a minimal amount of tilt – this is done so that once MTCNN has boxed the face and optimised the alignment the faces should all have their distinct features such as eyes, noses, mouths etc lined up on top of each other. This is important as the Fisherface algorithm is comparing regions and subsequently trying to build up a picture of the average man and female.

Overall my network performed amicably, using a test data of a series of images from recent images on my phone I was able to build a test dataset with 10 female faces and 10 male faces (some photos contain multiple faces). By running these faces through my Fisherface code I was able to evaluate the performance based on either a correct assessment or an incorrect assessment. Overall the network correctly predicted 8/10 of the males and 5/10 of the females. This results in an average of a 65% accuracy which I think is very amicable and above that for which we'd expect the classifier to perform at by randomly guessing.

8.0 Legal, Social, Ethical

Throughout this project as many datasets have been analysed it has been of great importance that a legal proceeding is followed throughout development. As such with respect to both of my neural net data sets I have ensured I have followed the terms of the licencing agreement very clearly.

For the first dataset the key points dataset as this is a Kaggle challenge dataset it is entirely free to use and train models off for educational purposes. As such this project falls well within the remit and is completely fine to be utilised to its full extent throughout this project.

As for the second dataset 'fer2013' this once more was a Kaggle challenge dataset. As I fully comply with the terms of service of Kaggle and my account was approved I am fully allowed to train a network for the purposes of education.

Another legal consideration that must be considered is the dataset on which I trained the Fisherface network – this was a mish mash of celebrity faces which had been publicly distributed online with no licence associated. As such as I don't stand to profit from this I can use these images, whilst they're not public domain there is no obligation for me legally to purchase exclusive rights to them.

The final legal concern of this project is the rights to distribute the compiled application. As I don't stand to profit I am covered under licence for the use of OpenCV, Keras and all of my other key libraries utilised. I also was covered under licence to develop and subsequently distribute the compiled application on mac as the compiler which converts the pythonic C code to bytecode is licenced in such a way that any developer can do so. Unfortunately a copyright issue fundamentally blocks this on windows – as on windows there are a series of necessary DLL's which enable the C compilation to take place which are distributed through the full version of visual studio by Microsoft. Unfortunately these DLLs are not present in my current installation as I don't hold a fully licenced version of visual studio – this blocks the compilation of my application as an executable file as legally if the DLL's were included I

would have no legal right to provide distribution and could be legally culpable to pay damages to Microsoft.
The application ultimately does have some social and ethical questions – as this software is designed to provide fun filters which can improve the emotion of an image it could lead to sensitivity amongst users. As noted earlier in this paper positive reinforcement is exceptionally important with social media, as such whilst it would be very possible to build a model which could evaluate how 'likeable' an image is or make subtle improvements to beautify an image it would not be ethical nor good for the users mental health to encourage such developments. As such the product stays very much clear of these moral hazards and instead encourages a fun transformation to a new emotional state of any input face.

9.0 Overall Face transformation performance:

Overall the application functions fairly well, for example if we consider say a transformation image for a typical woman using the angry filter.

Figure 12 – Emotional transformation of female sad to angry

In this example we see a very successful filter being applied and performing a transformation to the source image. We see the eye brows tilting downwards and the muscle texture above tightening and we see the lips successfully being brought closer together. The only minor critiques of this transformation are that the lips are ever so slightly off centre when compared to the cleft pallet and the left most eyebrow is not quite blending as naturally with the hair as the source photo does. The skin also is blending exceptionally well as its matching the tone whilst retaining the detail of the muscle stretching, although the consistency in colour blending is not consistent as the lips are noticeably redder giving the illusion of lipstick.

Figure 13 – Emotional transformation of male fearful to sad

In this second example an initial image is loaded which displays fear and it is transformed utilising the male sadness filter. This image provides more of an insight into the follies of this project, ultimately whilst the transformation does look realistic and the after image does most certainly look comically sad the resultant face has distinctly different eyebrows as the source image has small thin eye brows and the image from which the emotion has been transferred has distinctly large and bushy eyebrows. Another issue is the wide open eyes – this ultimately causes the classification algorithm to classify this as fearful still as the eyes and crinkle lines across the forehead are tell-tale signs of fear which the network is picking up on. Ultimately because not every emotive region on the face can reasonably be altered whilst retaining the original images appearance this leads to images not being as transformative as would be ideal.

Overall the transformation tend to be successful when given ideal situations but ultimately do still have fundamental shortcomings and can sometimes miss the target area and not correctly blend out the old area leaving behind an old eyebrow for example. One solution to this shortcoming was to simply provide the option to disable eyebrow blending so that should an image be sufficiently change by a change to the mouth area but the eyebrows aren't being removed properly this transformation is possible – this also increases the amount of control the user has over the transformation process which is always a positive.

10.0 Future Alternative approach:
10.1 GAN:

One approach which fundamentally inspired the undertaking of this project is the use of a generative adversarial network. A generative adversarial network is first defined in the paper “generative adversarial nets” (Ian J goodfellow et al, 2014, arVix, Generative adversarial nets), GANS fundamentally are two networks which compete against each other whilst learning in the process.

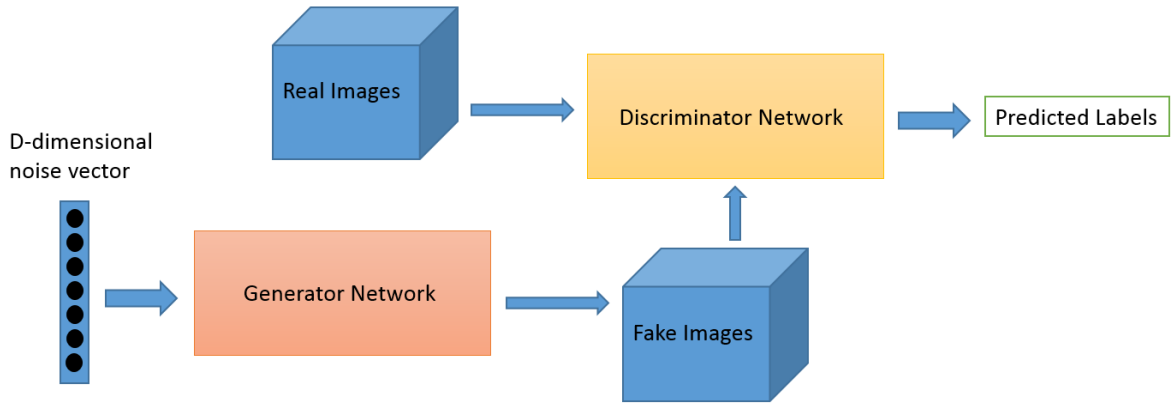


Figure 14 – Diagram of GAN (Owen Carey, 2018, towards data science, Generative Adversarial networks GANS) A beginner’s guide)

The first of these networks is a generator network which when given a specific class of data attempts to build up a feature set that it would expect within a given classification rather than classifying an input. The second network involved in a GAN is a discriminatory network which attempts to classify its input and map specific features to a given set of labels.

In the overall architecture of a GAN the GAN begins by attempting to generate new data instances whilst the discriminator attempts to discern what the real data is when compared to a given dataset. One example could be a popular character dataset called MNIST – a GAN might have to learn to build new handwritten characters based off of this dataset. The generator network would attempt to build better and better images by seeing what is classified as real or fake based off what the probability of it being real is assigned to it by the discriminatory network. Overall the discriminatory network and the generator should not be trained simultaneously as one will eventually begin to outperform the other significantly preventing further training and causing the network to fail.

As such to train a GAN a ground truth dataset is required but typically these are relatively small datasets when compared to what convolutional datasets are trained upon.

Figure 15 – GAN emotional transformations shown (Valentin Vielzeuf et al,2018, The many moods of emotion)

It must be noted that this project had it been implemented utilising a GAN would not be the first application for of GANs for emotional transformation – one instance which is distinctly different albeit more creative is to utilise GANs to emulate emotional painting styles for most of the Ekman categories(David Alvazerz-Melis et al, 2017, 31st Conference on neural Information Processing systems, The emotional GAN: Priming adversarial generation of art with emotion).

Another way a GAN could have been implemented in this paper rather than entirely restyling the face is seen in another paper as it makes use of a GAN to perform more intelligent feature blending (Huikai Wu et al, 2017, arXiv, GP-GAN Towards realistic High-Resolution Image blending). This would be useful for this project as it could replace the current mask and Poisson blend system in place to perform the emotional transformation.

Finally there are a plethora of papers (Yuchi Haung et al,CVPR, DyadGAN: Generating facial expressions in Dyadic interactions)(Xiaoqing Wang et al, 2018, Computation Intelligence and Neuroscience, Unsupervised domain adaptation for facial expression recognition using generative adversarial networks) which implement GANs entirely to perform the entire transformative process on a face to move between Ekman categories. All in all these models are vastly more accurate than my model for facial transformation as they take the input face in as an input when considering the final emotional appearance and subsequently remodel the noise as a new emotion. This is just simply vastly more complex and has a series of drawbacks but ultimately results in what can only be described as a superior transformation.

As GAN's are very new and currently experimental there are still many issues which encumber their development (Jonathon Hui, 2018, Medium, GAN – Why is it so hard to train generative adversarial networks!). One of the most significant is a phenomenon known as mode collapse. Mode collapse is when the discriminator over trains briefly and begins to train to find the optimal image to fool that specific iteration of the discriminator. This can lead to the model moving towards a point that is completely unrelated to the image noise giving the program some noise or colour loss in the case of a partial collapse or creating complete nonsense data in the case of a total collapse. Furthermore as this model is fundamentally based on hyperparameters and the relationship between the discriminator and generator eventually converging on an optimal output the network requires a large amount of experimentation without a lot of documentation or training yet available. These challenges make the network technical very difficult to implement as well as particularly hard to train as such it may not have been a wise choice for such a crucial and time dependant project despite yielding potentially the best results.

One of the main reasons why a GAN was not viable for this project was the high computational cost of training, in one paper the training of a GAN is cited as taking 96 hours using an optimal dataset and many tricks to reduce the computational time (Tero Karras et al, 2018,arXiv Progressive growing of GANs for improved quality stability and variation). This was done on hardware roughly 10x more powerful than my own (assuming linear scaling) by utilising eight very expensive GPUS, I meanwhile have 1 very used GPU for training which whilst it works for this particular dataset it could take weeks to train which is simply not feasible. Assuming this approach was taken on current hardware it would allow a single attempt at training that if failed would leave the product utterly devoid of functionality.

One partial solution to this problem is that new Nvidia hardware can perform mixed precision training. This allows for FP16 computation as opposed to FP32 computation theoretically halving the amount of memory required to store datasets. This in theory allows for the training of larger models or training with larger mini batches which yields vastly greater performance – the official documentation states a speed up of between 1.7X and 3.3X (Nvidia official documentation, 2019, Deep learning SDK documentation). Still this would require a new GPU to make us of which is prohibitively expensive, otherwise unneeded and still wouldn't provide enough acceleration the networks to train timely on a single GPU system still placing the project in peril of not being finished had this approach been taken.

Overall had this approach not been so risky from a project management perspective and so resource intensive it would have been a great technique that could have yielded better results but as it fundamentally would replace almost all of the other image processing code it would have been too risky to develop it alone and a waste of time if not impossible to develop it concurrently whilst developing the currently implemented system.

11.0 Post mortem:

Overall this project can be viewed as a success in that a functioning product has been produced which hits every single requirement. In that sense this project was a project management success and fundamentally the development was good too, ultimately though I feel a few changes and further development could produce a vastly superior product.

Fundamentally one of the biggest flaws is the lack of support for facial misalignment, this causes a major issue as the key point network struggles greatly which throws off the facial angle estimation, it throws off the replacement of the regions and it throws off gender estimation as the training data is aligned for the fisher face equation. As such if I were to develop this again I would make use of something such as Haar cascades to search the image of the face and detect the eyes, then subsequently use the different in y axis to determine orientation and then feed this newly aligned face into the image processing algorithms I developed as this should generate a rather large improvement in performance over this misaligned images. Another thing which I would change is if this were a real project DLIB which is a C++ library which can be brought into python is vastly superior to anything currently produced by convolutional networks and it outputs 32 points across of a face. Whilst this would have removed a lot of the challenge from the project as processing faces to form key points was a large challenge – this would also increase performance greatly and allow for more regions to be created which could be subsequently transformed emotionally.

Another area where this project ultimately could be improved is that currently the models are 2d and static by having a more accurate estimation of the facial angle using the existing solve PNP model alongside a better key point estimation 3d models of facial expressions could be built and the angle which most closely matches overlaid. This will prevent some of the horrible 'artifacting' and the outright misses that occur when a rotated face has the lips misaligned and the transformation targets off centre. I think with those improvements in mind it would from there be difficult to add anything else to the system to improve its performance towards the current specification without a fundamental change in approach. As stated earlier in this paper GANs are better and as time progresses this challenge of emotional remodelling will consistently be handled by them as training becomes faster and better documented as they are simply superior.

As for things that I wish could be done – a psychological evaluation on how people perceive the transformations and to see if they truly do have as positive a perceived affect as snapchat 'filters' have with users would be interesting as it would allow for the project to evaluated from another angle. Unfortunately I lack a basis in social sciences and the scope of the project does not allow for time to spent garnering either external support to perform a study or doing a lesser one myself to gauge typical reactions to the filters and if they do result in a perceived emotional difference or an improvement to the input image where the original emotion was not idealistic.

Overall a working product has successfully been delivered that is complicated in both the research that it required for implementation and in the implantation itself. Whilst a lot more time than I am proud of went into just figuring out how to perform some of these processes it has ultimately paid off and I gained a large amount of knowledge from completing this project as well as developed my skills in management immensely.

Appendix:
References:

1 (Sarah Cascone, 2016, artnet “24 Billion photos prove our selfie obsession is out of control”)
https://news.artnet.com/art-world/24-billion-selfies-uploaded-to-google-in-a-year-508718
2 (Dara Fontein, 2016, hootsuite,”The Top snapchat statistics you need to know for your business”)
blog.hootsuite.com/snapchat-statistics-for-business/
3 (Kayla O’Shea, 2017,Medium,”How snapchat filters work”)
https://medium.com/@kaylaoshea/how-snapchat-filters-work-499a4a1a7c5
4 (Banuba, 2019, Medium, “5 types of Face filters like Snapchat that engages users best”)
https://medium.com/@banuba/5-types-of-face-filters-like-snapchat-that-engage-user-bcbffcaf1cce
5 (Pengxiang Li et al,2018, Telematics and Informatic, “Likes” as KPI: An examination of teenage girls’ perspective on peer feedback on Instagram and its influence on coping response)

6 (Annukka K. Lindell,2017,frontiers in psychology, Consistently Showing Your Best Side? Intra-individual Consistency in #Selfie Pose Orientation)

https://www-ncbi-nlm-nih-gov.plymouth.idm.oclc.org/pmc/articles/PMC5318447/
7) https://www.diyphotography.net/the-pixel-3s-ai-automatically-takes-your-selfie-when-it-thinks-it-will-look-best/

8) (Andrej Karpathy, 2018, GitHub,”What a deep neural network thinks about your selfie”)

http://karpathy.github.io/2015/10/25/selfie/

9) (Paul Ekman, 1993, American Psychologist, “Facial expression and emotion”)

https://search-proquest-com.plymouth.idm.oclc.org/docview/614321272/fulltextPDF/13513383F04C4D6FPQ/1?accountid=14711

10) (Paul Ekman,2016,Sage Journals, “What scientists who study emotion agree about”)

https://journals-sagepub-com.plymouth.idm.oclc.org/doi/full/10.1177/1745691615596992

11 (management Mania, “Six basic emotions”)

https://managementmania.com/en/six-basic-emotions

12 (Gianluca Donato et al, 1999, IEEE Transactions on pattern analaysis and machine intelligence, VOL 21, “Classifying facial actions”)

https://ieeexplore-ieee-org.plymouth.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=799905

13 (Des Greer, 2011, Wiley Online Library, “Agile Software Development”)

https://onlinelibrary-wiley-com.plymouth.idm.oclc.org/doi/epdf/10.1002/spe.1100

14 (Margaert Rouse, 2011, tech target, “Agile project management”)

https://searchcio.techtarget.com/definition/Agile-project-management

15 (K. Lukasiewicz et al, 2012, European Systems and Software Process improvement and Innovation, Improving agility and discipline of software development with the Scrum and CMMI)

https://ieeexplore-ieee-org.plymouth.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=6334806

16) (Official TCI Documentation, TCI-WIKI)

https://wiki.tcl-lang.org/page/Tcl+event+loop

17 (Robert C Martin, 2000, Design Principles and Design patterns)

https://fi.ort.edu.uy/innovaportal/file/2032/1/design_principles.pdf

18) (Kaipeng Zhang et al, “Joint Face detection and Alignment using multi-task cascaded convolutional networks”)

https://arxiv.org/ftp/arxiv/papers/1604/1604.02878.pdf

19) (Tanin Phongpandecha, 2018, datawow, Face detection: Haar Cascade versus MTCNN)

https://blog.datawow.io/face-detection-haar-cascade-vs-mtcnn-13af4aa180e6

20) https://arxiv.org/pdf/1804.01753.pdf

21) (Tian-Xiang Wu et al, 2012, Neural Computing and applications, Multi view gender classification using symmetry of facial images) **https://link-springer-com.plymouth.idm.oclc.org/article/10.1007/s00521-011-0647-x**

22) (Satya Mallick, 2016, Learn OpenCV, Head pose estimation using OpenCV and DLIB)

https://www.learnopencv.com/tag/solvepnp/

23) **(Eric Yuan, 2013, Eric-yuan.me, Poisson Blending)**

http://eric-yuan.me/poisson-blending/

24) **(Naimish Agarwal et al, 2017, arXiv, “Facial Key points detection using deep Convolutional neural network”).**

https://arxiv.org/pdf/1710.00977.pdf

25) https://www.studocu.com/en/document/stanford-university/convolutional-neural-networks-for-visual-recognition/other/facial-keypoints-detection-using-neural-network/751825/view

26) (Octavio Arriaga, 2017, arVix, Real-time Convolutional Neural networks for emotion and gender classification)

https://arxiv.org/pdf/1710.07557.pdf

27) (James Pao, 2014, Stanford, Emotion Detection through facial feature recognition)

https://web.stanford.edu/class/ee368/Project_Autumn_1617/Reports/report_pao.pdf

28 – Original dataset for emotional classifier https://github.com/muxspace/facial_expressions

29 (Ian J goodfellow et al, 2014, arVix, Generative adversarial nets)

https://arxiv.org/pdf/1406.2661.pdf

30) **(Owen Carey, 2018, towards data science, Generative Adversarial networks GANS) A beginner's guide)**

https://towardsdatascience.com/generative-adversarial-networks-gans-a-beginners-guide-5b38eceece24

31(Valentin Vielzeuf et al,2018, The many moods of emotion)
https://www.researchgate.net/figure/The-7-emotion-classes-generated-with-the-three-different-approaches-discreteGAN-first_fig3_328651470

32 (David Alvazerz-Melis et al, 2017, 31st Conference on neural Information Processing systems, The emotional GAN: Priming adversarial generation of art with emotion)

https://nips2017creativity.github.io/doc/The_Emotional_GAN.pdf

33 (Huikai Wu et al, 2017, arXiv, GP-GAN Towards realistic High-Resolution Image blending).

https://arxiv.org/pdf/1703.07195.pdf

34 Yuchi Haung et al,CVPR, DyadGAN: Generating facial expressions in Dyadic interactions

http://openaccess.thecvf.com/content_cvpr_2017_workshops/w41/papers/Khan_DyadGAN_Generating_Facial_CVPR_2017_paper.pdf

35 (Xiaoqing Wang et al, 2018, Computation Intelligence and Neuroscience, Unsupervised domain adaptation for facial expression recognition using generative adversarial networks)

https://www.hindawi.com/journals/cin/2018/7208794/

36(Jonathon Hui, 2018, Medium, GAN – Why is it so hard to train generative adversarial networks!).

https://medium.com/@jonathan_hui/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b

37(Tero Karras et al, 2018,arXiv Progressive growing of GANs for improved quality stability and variation).

https://arxiv.org/pdf/1710.10196.pdf

38 (Nvidia official documentation, 2019, Deep learning SDK documentation)

https://docs.nvidia.com/deeplearning/sdk/mixed-precision-training/index.html
39 Data set for key points - https://www.kaggle.com/c/facial-keypoints-detection
40 – Data set for emotions - https://www.kaggle.com/c/facial-keypoints-detector/data

1.0 Installation dependencies: -

Hardware requirements:
>= 600MB of available space on your hard drive
At least a Core 2 Duo
4gb of ram
Ideally a GPU with CUDNN compliant drivers installed
1200x700 Resolution minimum screen
A webcam

To run the default compiled application on OS X:
No additional requirements for software – should run with default runtime parameters in OS X

To run the main application within a python environment on any OS:
Python v3.6.6
Numpy >= 1.16.1
Pillow >= 5.4.1
Mtcnn >= 0.08
OpenCV Contrib >= 4.1.0.25 (Must be contrib version – may require CMake or GCC to compile depending on specific configuration of system)

If using a GPU for acceleration:
Ensure CUDA support with CUDNN 9.0 v7.5 or higher currently installed
Tensorflow-gpu >=1.12.0

To train a gender detection network within a python environment on any OS:
Python v3.6.6
OpenCV Contrib >= 4.1.0.25
Numpy >=1.16.1
Mtcnn >= 0.08

To train key points or emotion detection network within a python environment on any OS:
Python v3.6.6
OpenCV Contrib >= 4.1.0.25
Numpy >=1.16.1
Scikit-learn>=0.20.3
Pandas>=0.24.1
Matplotlib >=3.0.2

2.0 Installation Guide:

2.1 Mac compilation Guide:

To compile the file simply target the folder with main in using terminal and run the command
Python3 setup,py py2app -A
This builds the application and places it within the directory in a folder called dist

2.2 Mac with compiled binary:

Select either DMG or ZIP of downloaded file
Extract the folder
Open ‘Dist’ folder or use search and search for ‘Emotion transformation suite’
Copy .app file from ‘Dist’ to wherever wanted
Application can now be ran from where pleased

2.3 Any OS in a python environment:

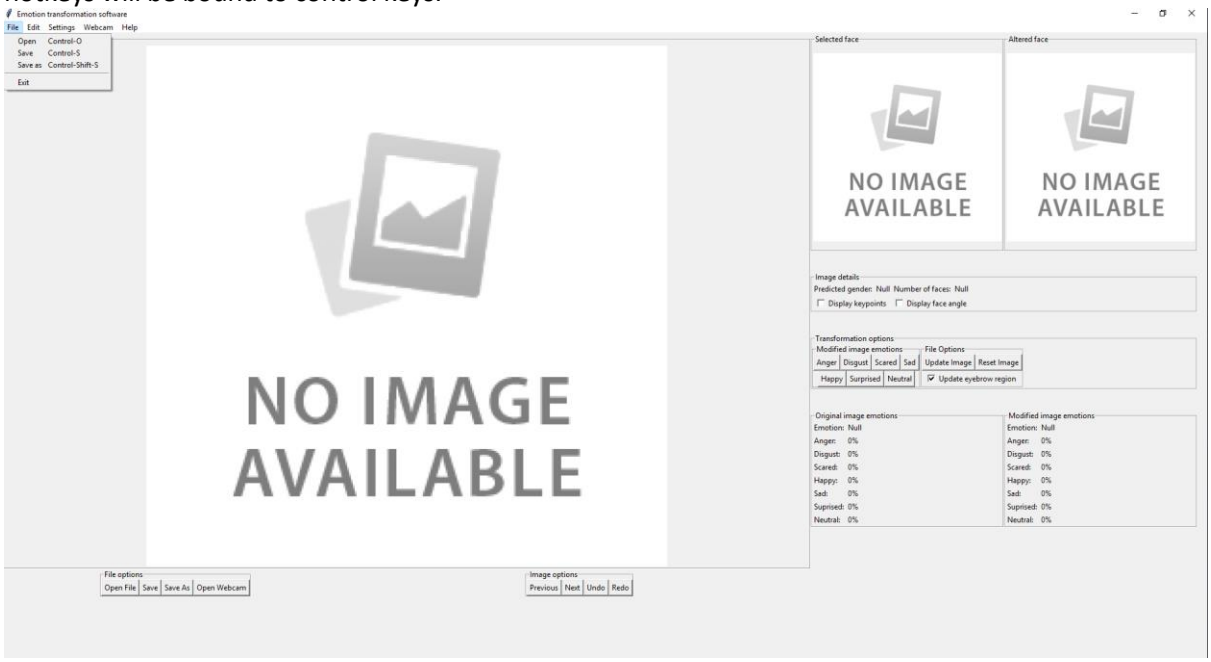
Select packages from requirements documentation
Use either PyPI or PIP to install or update to the minimum spec within your selected environment
Using either an IDE or Terminal execute the default python run command for main.py – example below in windows

```
PS C:\Users\Conor\source\repos\Emotion Transformer\Emotion Transformer> py -3.6 -m main.py
```

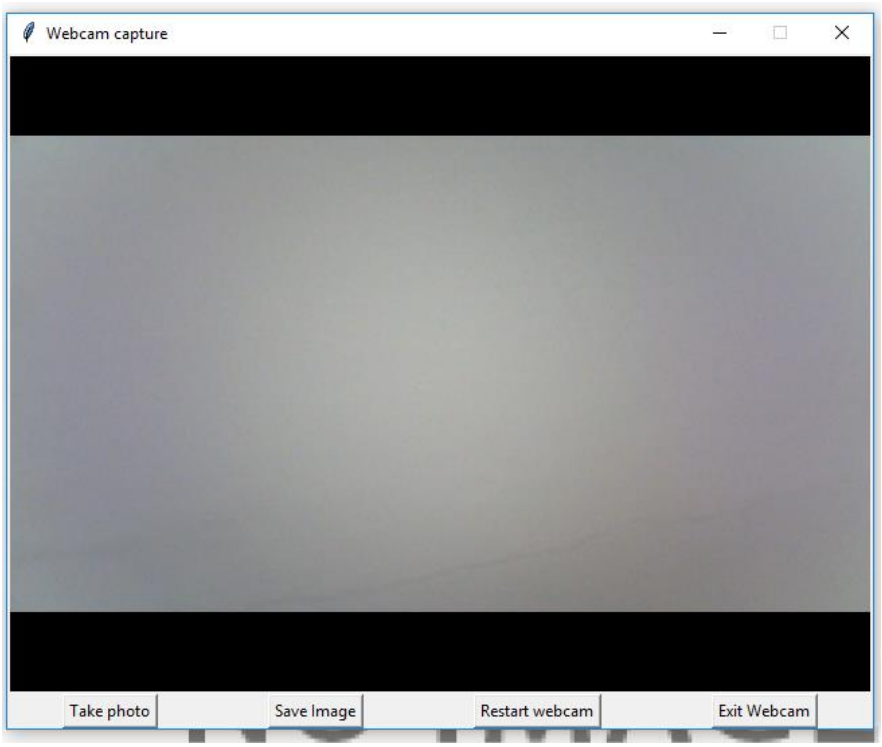
Application is now running and can be ran using said command when needed

3.0 User Guide: -

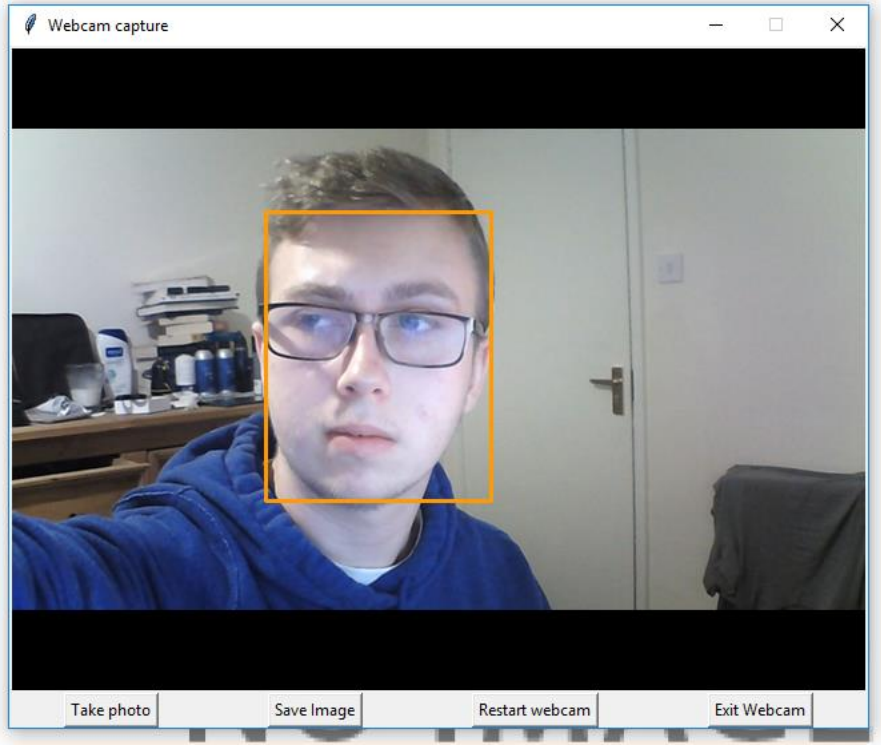
For additional user assistance – a user guide is available in the help section of the application as well – this links to the following video:
https://www.youtube.com/watch?v=4inEahRSWQ4&feature=youtu.be&fbclid=IwAR2IQeGlcjCIoXCAzsUgzic2hU6rfG4KH1TOiQoF89Tqcky7AVzcEw5LCyQ
After the application has been installed first begin the application by either clicking the icon or running the main.py file in your python environment.
At first you should be greeted with a screen alerting you that no image is currently available – during setup also anticipate that the webcam light may flash on and off – this is merely to ensure that a webcam is installed.
The screen should also have an assortment of OS specific hotkeys. If you are used to OS-X in line with development convention the hotkeys will be bound to command, if you a linux variant or windows the hotkeys will be bound to control keys.



From this screen to begin with the application we will want to either capture an image using an installed webcam or open an image from a file.
To capture an image from the webcam you can press the hotkey (Control OR Command) + W or press open webcam under the settings window or Open Webcam in the bottom left corner.
This should result in a webcam window being opened in a popup – if it does not then an error message should appear – please ensure your webcam is connected and try again.

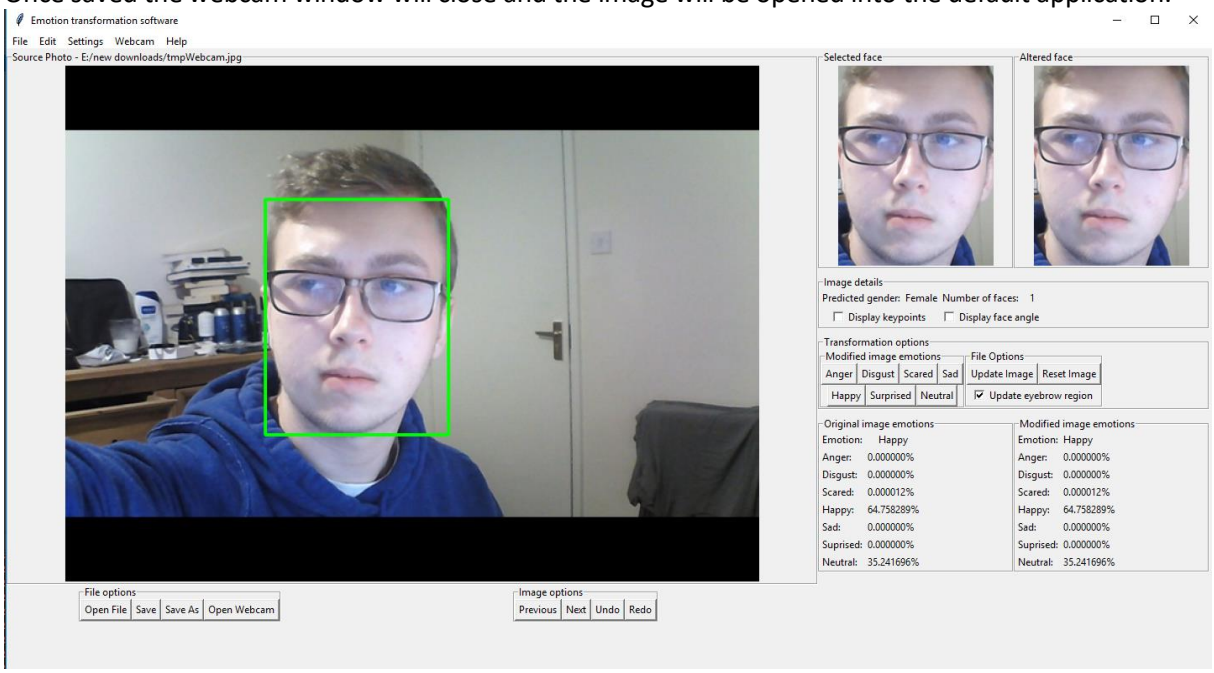


The resultant window will have a live updating stream of your webcam whilst opened, to take a photo simply press the take photo button. This will pause the stream and capture the last image when the button was pressed. If this image contains any faces a box will be drawn around them and you can proceed if not an error message will occur and the stream will continue until you attempt to take a photo again.

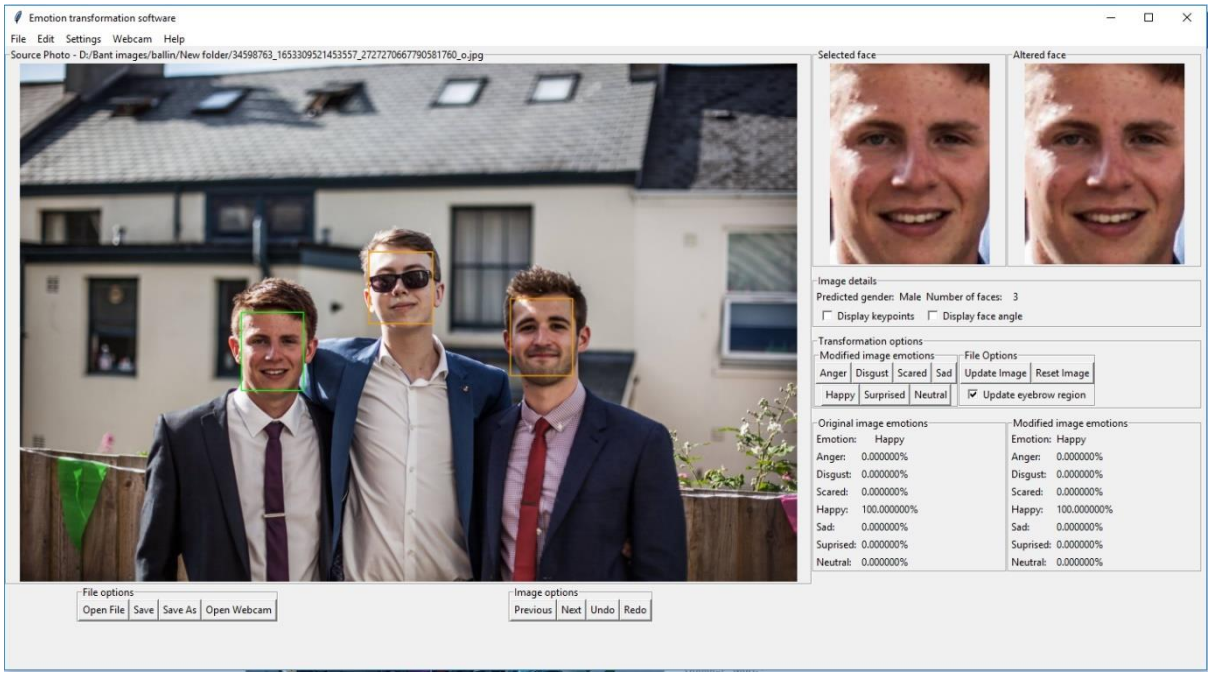


If you are unhappy with this image please press the restart button this restart the capture of the webcam and allow for a new photo to be taken. If you want to exit the webcam and no longer wish to

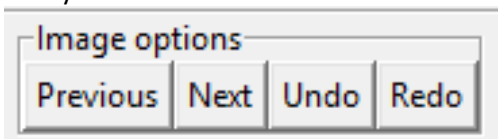
capture an image simply press exit and the popup window will close – alternatively you can close the popup window normally to the same effect.
If you wish to proceed with this image then press ‘Save image’ this will load your operating systems default filing system and ask you where to save your image and what to save it.
Select your file name and file save location then press okay – if you do not provide a file name then this can not proceed but the file location will by default be the last accessed area by the program.
Once saved the webcam window will close and the image will be opened into the default application.

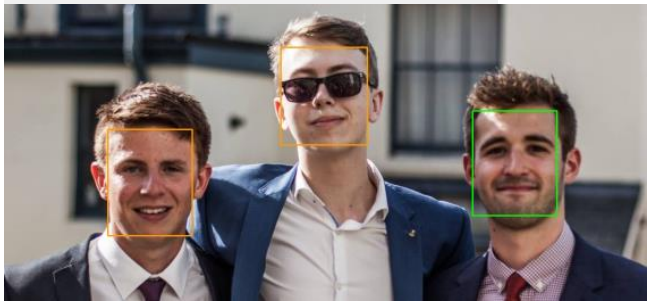


Conversely if you wish to simply load an image you can press either (Cmd or Control) + O or File and Open File or simply from the bottom left in file options press Open File. This will load up the file dialog specific for your operating system that will allow for you to browse to your selected image and open it.
Once open it will open to the default application providing estimates of Gender, number of faces, current emotional state and a series of colour coded highlight boxes around the found faces.

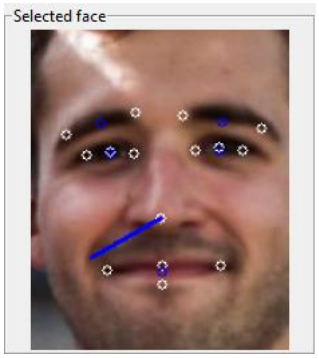


To select a new face there are buttons below the image in the image options section. You can iterate through to either the previous or next face, this will cause an update in emotional state, reset the altered face to default, update the selected face and redraw the selected face box to green around the newly selected face.

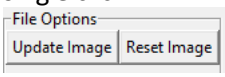




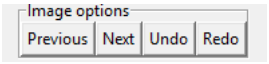
With the newly selected face we can see even further details if we so wish, by using the check boxes in image details or in settings (changes in one carry through to the other) we can enable either displaying the keypoints our neural network has drawn or the face angle estimate to be drawn on as a 2d line.



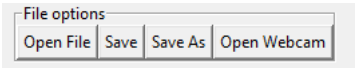
To transform the emotional state simply press any of the modified image emotions buttons, on press the altered face will reset to the selected face and then be transformed into the new emotion with a single click.



If the change is not as desired the altered face can be changed back to the selected face by pressing the reset image button – this will reset the altered face to be the original of the selected face and update the modified image emotions back to their original state. If you press next or previous then return to the face this will also have the same effect.

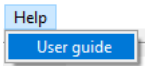


To commit these changes to main simply press the update image button – this will update the main screen with the new altered face and update the selected face to now reflect the new current emotion. If an undesired change has been made to the main image i.e an emotion was committed that in hindsight you don't like – this can be undone by pressing undo and subsequently redone by pressing redo.



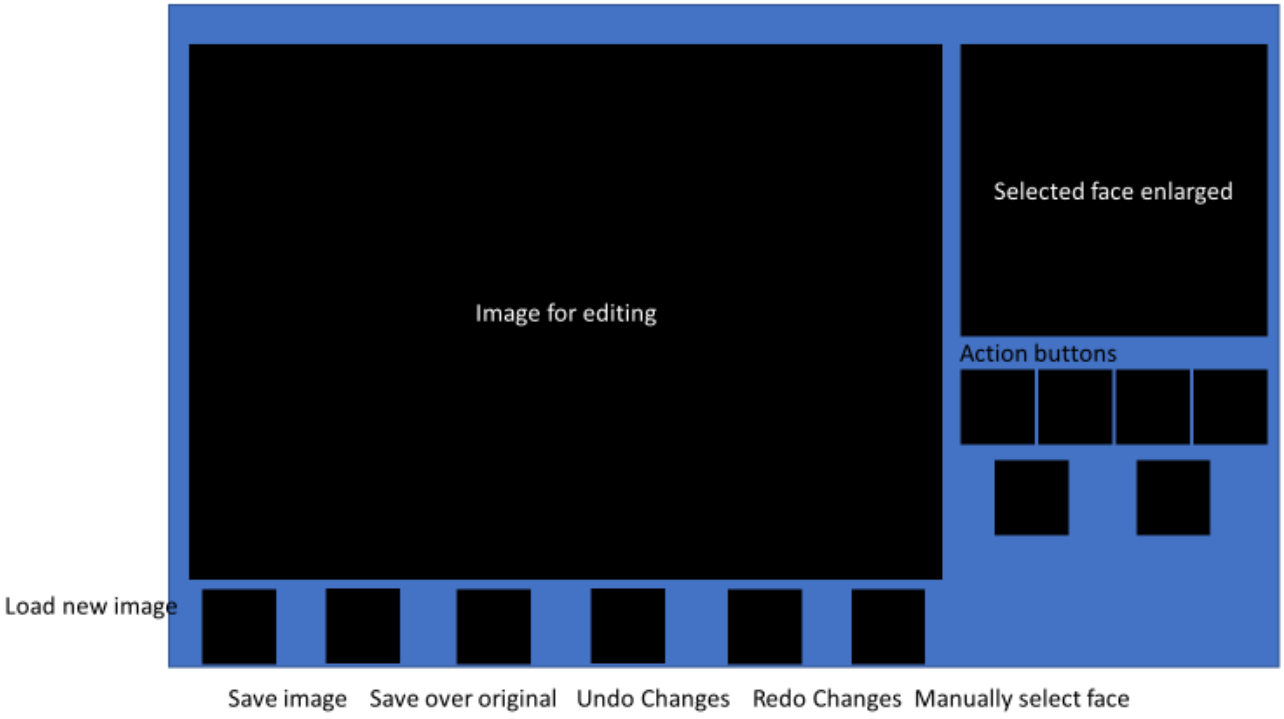
When an image has been successfully modified the image can be saved – using either the toolbar, the hot key (Command-S or Control-S) or pressing the button at the bottom left of the page. This will save the image over the default and in its current file location.

To Save as simple use the hot key (Command-Shift-S or Control-Shift-S) or press the button at the bottom left of the page or use the toolbar. This will open an OS specific dialogue which will allow the user to modify the file name and save location. This will allow file overwriting but it will warn the user before doing so. Upon completion the source photo header which dictates the file location will update to show the new location.



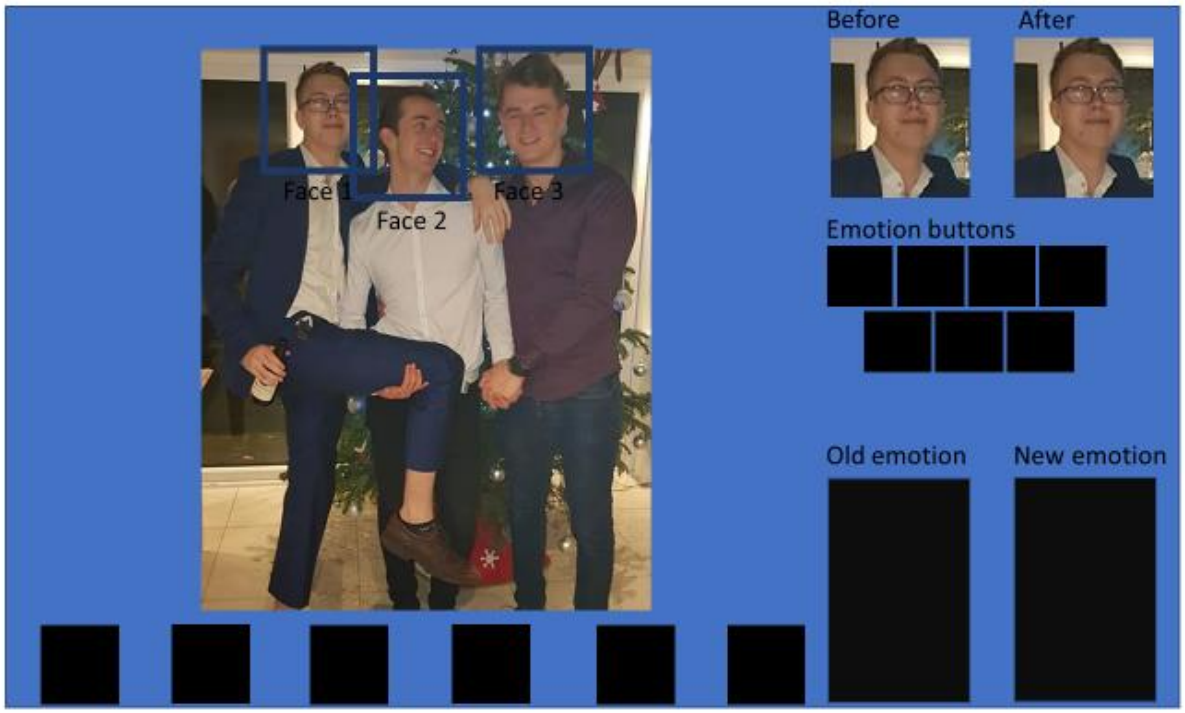
If at any point a user becomes confused whilst operating the software under the help section in the toolbar there is a button which when clicked will open up a video user guide in their default browser.

4.0 Designs:
4.1 Initial designs:

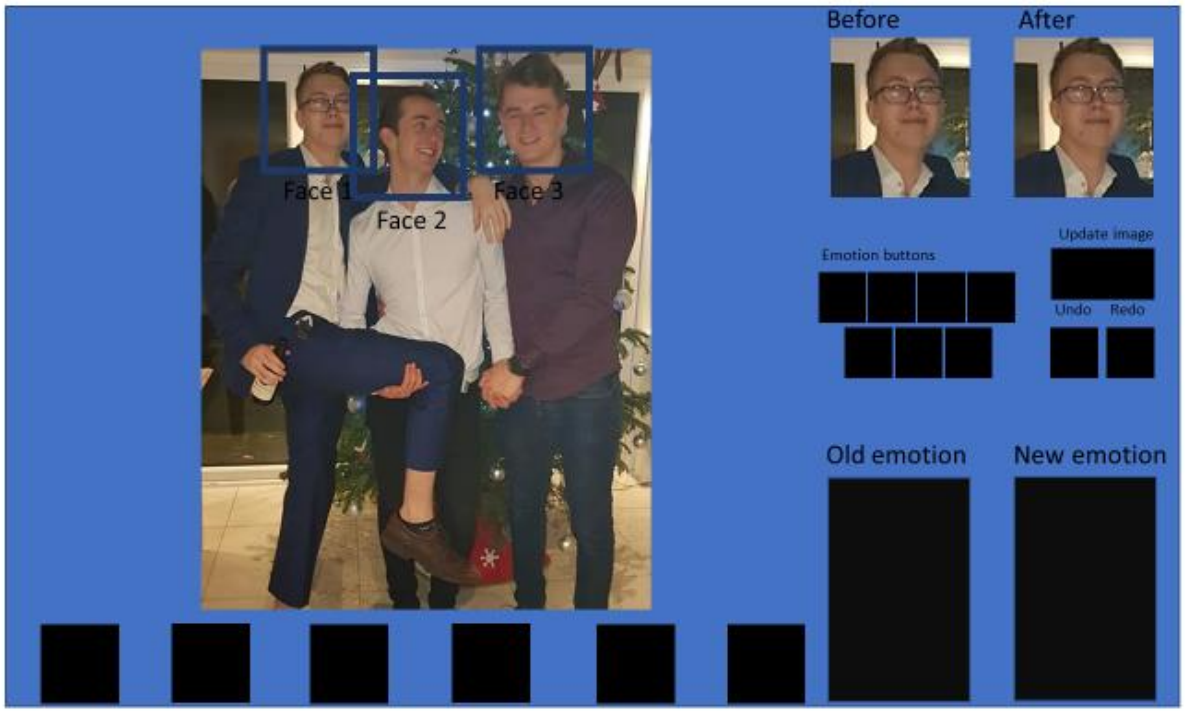



Face 1
Face 2
Face 3
Action buttons

4.2 Updated design:



4.3 Further updated design:

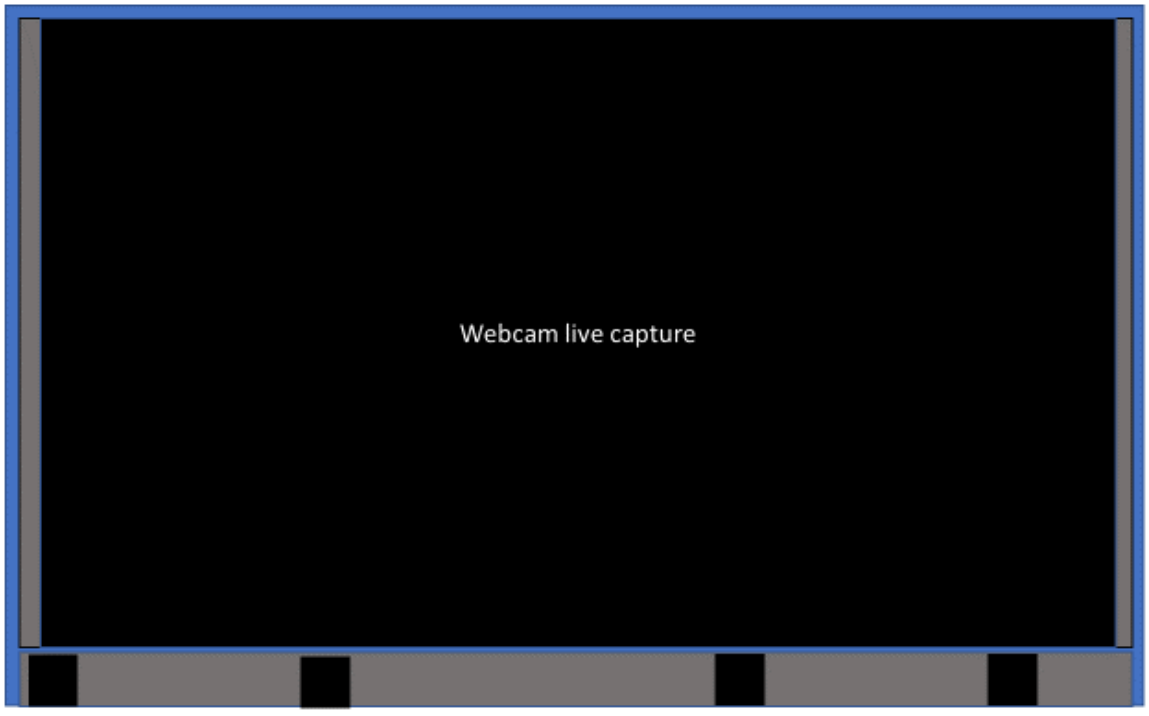


4.4 Post user feedback final design:

Open file, Save, Save as, Open webcam

Previous face, Next face, Undo,Redo

Popup window



Take photo, Save image, Restart webcam, Exit webcam

5.0 User stories: -

End user:

As an end user I want to be able to capture my own images
As an end user I want to be able to save these webcam images where I want
As an end user I want to be able to load these images instantly into the emotion software
As an end user I want to be able to load any image from my computer
As an end user I want to be able to see all the faces detected in an image
As an end user I want to be able to see the predicted gender of all the faces in an image
As an end user I want to be able to see the number of faces in an image
As an end user I want to be able to see an estimate of the current emotion of each face
As an end user I want to be able to see which areas are going to be edited
As an end user I want to be able to see which face I'm currently editing
As an end user I want to be able to transform the face into as many emotions as possible
As an end user I want to be able to easily get help when needed
As an end user I want to be able to resize the application to my needs
As an end user I want to be able to undo my mistakes easily
As an end user I want to be able to see if the transformation is going to be good or not
As an end user I want to be able to see the predicted emotion of my output face
As an end user I want to be able to compare the before and after of a face transformation

Learning agent (Key-point recogniser etc):

A learning agent will need consistent data sets
A learning agent will need split test and validation data sets
A learning agent will need inputs which are consistent with the standard outlined in the training set
A learning agent will need support for GPU acceleration
A learning agent will need to have minimal loss and high accuracy
A learning agent will need to be capable of performing on real time for classification tasks

6.0 Project initiation document:

Facial emotion swapper

Project Initiation Document

Name: Conor Worthington
Student number: 10528570

Contents:

1) Often images do not turn out as they are intended, people can have convey the wrong emotion on their faces and ruin the overall tone of a photo. This could be noticed long after the opportunity to retake a photo has passed. As such the ability to easily identify faces in photos and modify facial expressions tastefully to an ideal state would be beneficial to many users particularly as proliferation of cameras in the form of smartphones has allowed anyone to be a photographer
2) As currently there is no easy to use face modification software this puts users off and simply leaves most users assuming they must manually modify the photo using an application such as photoshop or gimp which requires extensive training and expertise particularly to perform such a complex operation as tastefully changing a facial expression

2.1) Buisness objectives:

To allow for instant face recognition
To allow for face scanning
To provide a simple user interface
To allow the selection of idealised emotions
To allow the training of a network for a new facial expression
To output a new image matching the targeted expression

3) To identify various stratergies for identifying faces
To identify effective stratergies for training neural networks
To find the optimal stratergy for classifying people
To find the optimal end transformation process using a neural network
4) Initial Scope

To allow for instant face recognition
To allow for face scanning
To provide a simple user interface
To allow the selection of idealised emotions
To allow the training of a network for a new facial expression
To output a new image matching the targeted expressio

5) Method of approach

My system will be fundamentally an agile development focussing on producing an initial face detection and picture input system with a clean gui then an emotion transformation algorithm and a picture distortion algorithm.
Possible technologies:
Python:
Python is a very easy to develop in language with a syntax similar to pseudo code and with a vast amount of documentation.
TensorFlow:
TensorFlow is currently the best documented neural network library and also boasts an incredible efficiency which will need to be utilised to allow for picture processing in decent time.
OpenCV:

OpenCV allows for rapid development of facial detection algorithms which can be used to identify faces in the input images.

6) Initial project plan

7. Project plan			
Stage	**Expected Start Date**	**Expected Completion Date**	**Products/Deliverables/Outcomes**
1. Initiation		14th December	PID
2. Investigation and outline requirements	29 Jan	7 Feb	Outline requirements; Evaluation of possible technologies; Analyse similar artificial intelligence papers
3. Initial high level design	6 Feb	14 Feb	Design documents
4. Increment1	14 Feb	21 Mar	Initial GUI and image input
5. Increment2	21 Mar	4th April	Face recognition and emotion training
6. Increment3	4th April	18th April	Face transformation and Output
7. System and user acceptance testing	18 Apr	23 Apr	Test results, final system; user training
8. Assemble & complete final report[1]	Mon 23 Apr	Fri 4 May	PRCO304 Report

6.1) Control plan

I will be making use of a Prince2 based control plan, using different aspects of the methodology – these include:

Highlight reports

Exception Reports

Regular progress reviews

Risk management

7) Initial Risk list

The key identified risks within the project include:

Hardware failure:

A separate system in the form of my laptop is available as an alternate development environment as well as the CUDA lab in uni.

Schedule Run over:

Highlight reports as well as regular progress reviews will serve to mitigate the risk of schedule run over

Issue with development technologies:

The use of online development aids and research papers should help to alleviate any issues during the production of the software.

File corruption:

All software developed on my desktop has a raid backup providing parity for drive failure as well as the option for an off site backup

[1] You should aim to write-up as you go along, so this final stage is assembly and final tidying up.

7.0 Highlight reports:

PRCO304: Highlight Report
Name: Conor Jeffery Worthington – 10528570
Date: 6/2/2019
Review of work undertaken Successfully identified – installed and tested technologies which will be utilised (CUDNN, CUDA 9.0, Tensorflow, Tensorflow-GPU, Tkinter, OpenCV, Python 3.6, Visual studio) Logged development issues – Anaconda SSL communication issue and PIP incompatibility / issues with TensorFlow in python 3.7 Designed user interface / planned out user interactions Identified various paths to perform facial transformation Began building user interface
Plan of work for the next week Implement user interface Prototype user interface with friends / fellow students Implement facial recognition Have facial selection working
Date(s) of supervisory meeting(s) since last Highlight – 30/1/19
Brief notes from supervisory meeting(s) since last Highlight Briefly discussed overall structure of module and expectations as well as how to deal with any issues with GAN implementation and alternate implementations

PRCO304: Highlight Report
Name: Conor Jeffery Worthington 10528570
Date: 13/02/19
Review of work undertaken GUI partially implemented - design revised Moved from HAAR classifier to DNN library called MTCNN for greater accuracy and improved facial recognition performance Implemented facial recognition
Plan of work for the next week *Finish off GUI* *Have subpictures generated* *Start on emotional classification / building face masks*
Date(s) of supervisory meeting(s) since last Highlight 7/2/14
Brief notes from supervisory meeting(s) since last Highlight

PRCO304: Highlight Report
Name: Conor Jeffery Worthington – 10528570
Date: 21/2/2019
Review of work undertaken Moved from Tkinter to Wx Used glade to build rebuild UI Prepared datasets for neural networks to be produced
Plan of work for the next week Build facial mask classifier Begin work on emotion classification
Date(s) of supervisory meeting(s) since last Highlight – 14/2/19
Brief notes from supervisory meeting(s) since last Highlight Slightly behind progress – intended to begin building Conv net and training for facial classification but due to deadline and sickness fell behind a few days

PRCO304: Highlight Report
Name: Conor Worthington
Date: 27/02/2019
Review of work undertaken Moved from pure tensorflow to KERAS with a tensorflow backend – this was done to aid simplicity as the networks were very hard to develop and additional benefits from pure tensorflow were going unused Performed tests to ensure KERAS is still interfacing with GPU Created a convolutional network in KERAS and trained to produce estimates of 15 facial points
Plan of work for the next week *Fix GUI bugs – two very severe functional bugs have crept up as implementation has begun of new features* *Polish up facial point network* *Train emotional recogniser*
Date(s) of supervisory meeting(s) since last Highlight
Brief notes from supervisory meeting(s) since last Highlight

PRCO304: Highlight Report
Name: Conor Worthington
Date: 6/3/2019
Review of work undertaken Updated and changed model to perform better – now performing face mapping at 85% accuracy Trained initial emotional recogniser Gained a lot more insight and learned a lot about optimisation – found sources for write up of conv net and decisions made in choices
Plan of work for the next week *Fix gui – was supposed to this week but didn't have time* *Implement NN saving and loading into GUI* *Increase emotional recogniser performance*
Date(s) of supervisory meeting(s) since last Highlight 28/2/2019
Brief notes from supervisory meeting(s) since last Highlight

PRCO304: Highlight Report
Name: Conor Worthington
Date: 13/3/2018
Review of work undertaken GUI bugs finally fixed – now fully functional – capable of correctly creating inputs for networks etc Implemented face detection and mask building in GUI – imported model Vastly improved mask building performance using GlobalAveragePooling Performed lots more research into implementation of transformation – viable paths identified
Plan of work for the next week *Improve emotion detection* *Implement emotion detection into GUI* *Begin implementing face transformation*
Date(s) of supervisory meeting(s) since last Highlight 7/3/2018
Brief notes from supervisory meeting(s) since last Highlight

PRCO304: Highlight Report
Name: Conor Worthington
Date: 21/3/2019
Review of work undertaken Further interface bugs fixed – crashing on improperly limited faces Sought peer opinion on UI Made UI tweaks Improved aspect scaling Found seamless cloning in OpenCV replacing need of updating repository PyPOI from python 2 to 3 and associated bugs Improved neural net performance
Plan of work for the next week *Keep improving emotional network performance* *Begin implementing masks for various emotions* *Implement saving over a given region*
Date(s) of supervisory meeting(s) since last Highlight 14/03/2019
Brief notes from supervisory meeting(s) since last Highlight *No substantial issues logged – progress last week was good, discussion was brief and to the point about new improvements as well as how to best write up*

PRCO304: Highlight Report
Name: Conor Worthington
Date: 28/3/2019
Review of work undertaken Made further UI tweaks to allow for dynamic scaling Implemented seamless cloning algorithm whilst accounting for pixel density to produce optimal transformations Looked into fisherface gender classification as optional extra algorithm Added several key button functions i.e update main, save as
Plan of work for the next week *Finish tweaking emotional classifier* *Improve as much as possible the seamless cloning – current results are very poor using the dynamic keypoints made by AI despite relative accuracy of points*
Date(s) of supervisory meeting(s) since last Highlight 21/03/2019
Brief notes from supervisory meeting(s) since last Highlight Last week I felt my project was seeming a touch lacklustre as such I've sought to rush a lot this week and attempt to produce more to allow for increasing a bit of scope – i.e adding webcam inputs via a modal and gender classification to allow for more accurate selection of seamless cloning

8.0 Testing:-

Modify an image

Step:	Action:	Expected outcome:	Actual result:	Result type:
1	Open application	Application opens successfully and GUI begins	As expected	Pass
2	Resize application	Application resizes retaining aspect ratio of images – when shrank it does not crop any items	As expected	Pass
3	Open trial.jpg	Main image is updated, file name is updated to point to trial.jpg and information about number of faces and emotions etc are updated	As expected	Pass
4	Check display keypoints	When ticked the keypoints are displayed on the before image	As expected	Pass
5	Check display face angle	When ticked the expected angle in 3d should be drawn as a 2d line on the face from the nose	As expected	Pass
6	Iterate through to second face	Before and after image are updated to the new face – main image gets a new outline to indicate it's the selected face	As expected	Pass
7	Change to disgust	Expect the after image to now display a disgust emotion and an update to the emotional classifier	As expected	Pass

8	Update image	Main image should now reflect these changes	As expected	Pass
9	Undo this change	Main image reverts to unedited state	As expected	Pass
10	Redo this change	Main image once again gains disgust modification to selected face	As expected	Pass
11	Go to the third face	Before and After sub images are updated to display newly selected face	As expected	Pass
12	Change emotion to Anger	After image now has an angry expression – emotional classifier has been updated	As expected	Pass
13	Save as newImage to desktop	Desktop should now have a new image called 'newImage' without any boxes drawn on and new expressions	As expected	Pass
14	Open image on desktop	Image should be clear and the original size of original image edited	As expected	Pass
15	Modify first face to happy	First face should now express a happy emotion and emotional classifier should be udpated	As expected	Pass
16	Save image	Should update image on desktop to have happy face over first face	As expected	Pass

Capture and modify a webcam image

Step:	Action:	Expected outcome:	Actual result:	Result type:
1	Open webcam	New popup window should open with webcam capture being displayed	As expected	Pass
2	Exit webcam	Popup window should close and regular application can be used as expected	As expected	Pass
3	Re-open webcam	Popup window will open once more	As expected	Pass
4	Take photo	Capture should stop, new image is displayed and faces detected are drawn	As expected	Pass
5	Restart webcam	Capture restarts and is drawn in webcam window	As expected	Pass
6	Take photo with no face visible	Error message stating that no faces were found, camera begins capturing images again	As expected	Pass
7	Take photo with face visible	Capture stops, new image displayed and faces detected are drawn	As expected	Pass
8	Save image	Image is saved to location, the popup window for webcam closes and the main window subsequently loads the image as if it had been selected	As expected	Pass

Generic tests

Step:	Action:	Expected outcome:	Actual result:	Result type:
1	Try load an image without a face	An error message is displayed stating no faces were found	As expected	Pass
2	Test an image with two people of different genders in it	Each face will have a different, correct, gender assigned when selected	As expected	Pass
3	Resize and see if aspect ratio of all images is retained	Aspect ratio should be retained by altered padding and size	As expected	Pass
4	Test hotkeys on windows	Control keys are bound to save, open etc – should execute commands	As expected	Pass
5	Test hotkeys on mac	Command keys are bound to save, open etc – should execute commands	As expected	Pass
6	Try compile application on mac to see if setup.py works	A build and dist folder should be subsequently built with a runnable compiled bytecode application in dist	As expected	Pass
7	Check all emotions on female face	All emotion buttons should perform a distinct transformation with retention of feminine qualities	As expected	Pass
8	Check all emotions on male face	All emotion buttons should perform a distinct transformation with retention of masculine qualities	As expected	Pass
9	See if keypoint	See if the	As expected	Pass

	prediction is correct	keypoint prediction draws roughly were it's expected to draw (i.e left eye where the left eye is)		
10	See if display angle is correct	See if the 3d estimation of the face is roughly accurate and drawn where you expect the face to be facing	As expected	Pass
11	Attempt to close application whilst working on project	A prompt should appear asking the user if they want to exit – if they confirm it should fully close – if they cancel it should remain open	As expected	Pass
12	Check if predicted emotions works on load	When loading an image the emotional state should be predicted in the before and after area of the application	As expected	Pass
13	Check if predicted emotions works when modifying a face	On alteration of a face i.e being committed to the main image the before emotion should be updated as it's reclassified	As expected	Pass
14	Attempt to open user guide	Browser should open and begin playing video user guide	Pointed to wrong video Updated video link – now works	Fail Pass
15	Clear changes to modified face	When the clear button is pressed any filters on the after image are removed as it reverts to the before image	As expected	Pass

9.0 User tests:-

9.1 Sheet user given

Please complete the following tasks

1	Open the application
2	Resize application to your desired size
3	Open trial.jpg
4	Check display keypoints
5	Check display face angle
6	Iterate through to second face
7	Change to an emotion of your choosing
8	Update image
9	Undo this change
10	Redo this change
11	Go to the third face
12	Change emotion to an emotion of your choosing
13	Save as newImage to desktop
14	Open image on desktop
15	Modify first face to happy
16	Save image

Did at any point you struggle with completing the tasks? If so which tasks?

If you struggled or confused what do you think confused you?

Do you think at any point the application was missing information which would have helped you?

Did the UI represent the transformation clearly to you?

Are you okay with having your results anonymously collected and notes taken by Conor Worthington? You can withdraw at any point by emailing conor.worthington@students.plymouth.ac.uk

9.2 Results from user testing:

Testing was done individually on my laptop with peers who I messaged in advance. Testing was done in a reasonably busy room with me observing and making notes on my phone. Individual feedback was summarily collated into the following brief report on the user interface:
Users often got confused on step 9 – all five users incorrectly pressed the wrong undo button – this is very confusing as it resets the altered sub image and ultimately very much frustrates the user. Whilst they often found the subsequent correct reset – it was not natural.
Users were occasionally confused as to how the next face system works – but it wasn't very confusing – it was mentioned as a minor issue 2/5 times –
Users commented that large parts of the UI at times felt blank as the current spacing schema between buttons is based upon how large the images are scaled to. This was mentioned in 2/5 tests and was made with particular reference to the space below the main image.
Users felt the toolbar was very sparse with only a save, open and save as button at the top – this was brought up in 3/5 tests.
One further criticism levelled against the UI was the file name not being displayed anywhere making it unclear if save as has worked. This criticism was levelled against the project 2/5 times.
Overall the users were able to quickly navigate the project and quickly gain an understanding of how it works and quickly implement new facial emotions – as designed.

10.0 Development tool choice: -

Language:

C++

C++ Has the advantage of being the fastest language for execution of code and it's a very old language thus it's very documented. I am also very experienced with C++ having completed several projects using it very recently. However C++ has the enormous disadvantage of not being a managed memory environment – this could lead to memory leaks and other issues.

One further advantage is how easily C++ interfaces with native CUDA code and could if any algorithms necessitate invoke parallel computation to solve it – however this is unlikely for this project.

The largest downside to C++ is that there are very few well maintained libraries readily available – libraries are also hard to install and the development time is very slow. This could overall make the scope impossible to complete in C++ within the time frame.

Java

Java is an OO language that I am semi experienced in – It has the advantage of being a managed memory environment and as such is fairly secure – as well as having fairly good support for quick development practices.

However, Java lacks a large amount of support in the form of libraries for both computer vision and neural network development. Furthermore java has only fairly recently gotten dedicated CUDA support allowing for GPU acceleration during training – it would be unwise to assume the same stability or performance with relation to more mature implementations.

Python

Whilst I have the least experience in python having only played around with it when younger as a scripting language it is undoubtedly the easiest language with the shortest development time to produce an equivalent piece of code.

Python unfortunately does have the largest performance handicap as it naturally is only an interpreted language which can make use of C code when called to. This does still allow it to retain the hardware speed of native C code when invoking a CUDA kernel which poses numerous advantages when training a neural network.

Furthermore python has the largest amount of library support and is the easiest to maintain with packages such as PIP and PyPI allowing for rapid deployment of new libraries into the application.

It's even further supported by professionals who generally tend to develop in python - https://www.datasciencecentral.com/profiles/blogs/which-programming-language-is-considered-to-be-best-for-machine

Version:

Python v2.7

Python v2.7 has the advantage that as it's a derivation of Python 2.X it will have potentially the greatest amount of support and more guidance will be available on how to fix any issues with python 2.x code as it's been available for longer. The issue however with python 2.7 is that as it's a branch of Python 2.X it is currently depreciated code and will no longer be maintained in the very imminent future – as a result of this it seems very imprudent to develop a project which will not be maintainable should anyone else wish to pursue modifying my code in the future.

Python v3.7

Python V3.7 has the advantage that it's a branch of python 3 and as such is going to be maintained for an exceptionally long time – it also has the most recent libraries written for it such as PILLOW instead of PIL. Unfortunately, as v3.7 is the very latest release a large amount of libraries either do not officially support v3.7 or simply will not be able to be installed to v3.7 as a result it risks necessary libraries not being able to run which would be a major development issue.

Python v3.6
Python v3.6 as such seems to rectify a large number of the previous problems by both being old enough that it has sufficient existing development support but being a branch of python V3.X it will be maintained with support for a very long time to come should future development be warranted – allowing the code to be maintained by other developers.

IDE:
Visual studio
Visual studio has a few advantages – first and foremost it's currently installed on my workstation windows computer although not the full version. Secondly visual studio has a gui interface for making use of PyPI to manage packages which are installed in a given environment. Unfortunately though visual studio does not allow for the easy establishment of virtual environments to manage application specific libraries – for example for OpenCV there are multiple variants with the same name that cannot be concurrently installed – it would be nice to manage which one is being used rather than forcing an install each time I want to use it within that environment. Another disadvantage of using visual studio is that the environment it sets up by default is locked for external modification – this means you can not invoke the windows terminal to begin making use of libraries such as PIP to alter the python environment. The third major disadvantage of visual studio is it takes up an obscene amount of storage with what is fundamentally bloat as a large amount of features will not be used on this project.

PyCharm
PyCharm has a number of advantages – as a large amount of my development time will be spent on OS X the cross platform similarities between windows and mac will allow me to develop seamlessly on both platforms. Furthermore it allows for virtual environments to be set up easily to manage the packages necessary to run the application. Finally it has an additional benefit of having a built in terminal to easily allow for quick execution of terminal commands for python code such as build commands or pip install commands. One disadvantage however is there are very few visual elements and as such it will require more of a learning curve to learn commands than the visuals of visual studio.

IDLE
Idle is the default python editor included upon install of the python environment. Idle unfortunately lacks a vast amount of features and ultimately amounts to a text editor rather than an IDE. Whilst it can edit code and provide grammatical mistakes for syntax it has no inbuilt terminal or graphical interface and ultimately was only designed for writing brief scripts.

Neural networks:
Theano
Theano is a purely pythonic neural network and as such is very easy to interface with and can be modified extensively. Unfortunately Theano has the negatives of having an extraordinarily high build time when compared to other networks and also being the worst documented framework partly due to its low level construction bringing additional complication.
Tensorflow
Tensorflow was designed by google to replace Theano. It offers the advantage of scabalibility for more powerful computing and has a large amount of documentation and pre built models to learn from. Tensorflow provides a slight abstraction when compared to Theano but is still a very complicated framework considering I only have experience building multi layer perceptrons up until this point.

Keras
Keras offers the most simple solution to my need for neural networks by providing the most documentation and the most abstraction. Whilst you lose an aspect of control and you lose tensorboard debugging etc you gain the ability to have backend networks in either tensorflow or Theano which can be built and trained very rapidly.

Computer vision algorithms:
SciKit-image
Sci kit learn ultimately falls down a lot in computer vision tasks when compared to its competition, it benefits from having a bit more support for some advanced data processing tools such as support vector machines but that's pretty much the only advantage.
OpenCV
OpenCV ultimately is the industry default it's a C++ based library which has more features than SciKit image and a variety of pre trained classification libraries. Furthermore I have experience with OpenCV in its C++ form and am going to continue to grow this during my other module alongside this project.
https://stackshare.io/stackups/firesize-vs-opencv-vs-scikit-image

11.0 Requirements

Requirement:	Functional or Non functional	Description:	Importance:
Load images from file	Functional	Allow users to load images they already have saved on their computer	Must have
Save changes	Functional	Allow for modifications to be committed to a file	Must have
Detect a face	Functional	Detect faces and display the detection to a user	Must have
Be able to process multiple faces in an image	Functional	Find multiple faces and allow for them to be moved between and independently processed	Should have
Be able to classify the current emotion of a face	Functional	Detect the current emotional state of the selected face and display it as an Ekman category	Must have
Be able to classify the current gender of a face	Functional	Detect if the face has mostly masculine or feminine traits	Could have
Be able to assess the angle of a given face	Functional	Estimate the current posture of the face so that it can assessed if the transformation is going to be successful or not	Could have
Produce confusion matrix of Ekman categories	Functional	Allow the user to see a probability distribution of the emotions that the selected face is displaying	Should have
Allow for transformation to any Ekman category	Functional	There should be as many emotional transformations as possible as such there should be at least one for every Ekman category emotion.	Must have
Undo changes on the go	Functional	Provide users with a sense of control by allowing to undo mistakes instead of restarting the editing process	Should have

Have a helpful guide section for new users	Functional	Users must be able to seek guidance as to how best use the app from within	Must have
Allow for webcam to capture images	Functional	Ensure the webcam can capture and save pictures the user takes	Must have
Provide insight into neural network transformations	Functional	Provide feedback and display keypoints which are used in the transformation process	Must have
Support non GPU accelerated hardware	Functional	Ensure that non GPU accelerated systems can make use of the neural nets	Must have
Compare transformed face with original face	Functional	Ensure the user can always see a before and an after to their decisions to inform them	Should have
Have a simple to use GUI	Non- Functional	Ensure the GUI follows simple design principles and is minimalistic	Must have
Have a responsive GUI	Non- Functional	Ensure that every action has an instant response	Should have
Have transformations look realistic	Non- Functional	Ensure the transformations look like human beings expressing emotions rather than cartoonish creations	Should have
Retain image quality and aspect ratio	Functional	Ensure that the image does not get downgraded or the aspect ratio ruined	Must have

12.0 Sprint plan

	Week 1	Week 2	Week 3	Week 4	Boundary	Week 5	Week 6	Week 7	Week 8	Boundary	Week 9	Week 10	Week 11	Week 12	Boundary
Load images from file		x													
Save changes			x												
Detect a face			x												
Be able to process multiple faces in an image				x											
Be able to classify the current emotion of a face						x	x								
Be able to classify the current gender of a face												x			
Be able to assess the angle of a given face													x		
Produce confusion matrix of Ekman categories							x								
Allow for transformation to any Ekman category								x							
Undo changes on the go															
Have a helpful guide section for new users														x	
Allow for webcam to capture images													x		
Provide insight into neural network transformations									x						
Support non GPU accelerated hardware									x						
Compare transformed face with original face											x				
Have a simple to use GUI	x														
Have a responsive GUI	x														
Have transformations look realistic														x	
Retain image quality and aspect ratio											x				

13.0 Prince 2 boundary reports:

13.1 Boundary 1 – Week 4

The initial progress throughout this project has been very solid. All the designs have been produced for the UI and functionality / requirements has been ascertained. Alongside this there was rapid progress in week one in which I managed to finish a very complicated install chain and subsequently test to ensure that my system is working correctly i.e that TensorFlow is successfully detecting my GPU and utilising it correctly.

Beyond this the designs were implemented in a very brief form in Tkinter, whilst there was brief experimentation with WX and a couple of other UIs ultimately, they were overcomplicating the process for very little gain over the default Tkinter.

Subsequently time was set aside to test these designs on a few friends, the results overall were positive however it has become clear that a multitude of changes must be made with regards to the undo functionality and how its graphically presented.

13.2 Boundary 2 – Week 8

At this point the project is progressing very well the two neural networks are as of now trained successfully with datasets located for them. This is now allowing for the neural nets to undergo fine turning – admittedly the development of the networks has taken a lot longer than allocated originally – these networks seem to persistently be increasing in performance but also taking up more development time that could be spent implementing other functionality.

As a result of this the transformations into any other category is not currently functional whatsoever – whilst I have found a logical way to implement this using OpenCV I have not yet got enough pre-requisite code to implement this. As such this will have to become a pressing issue moving forward.

13.3 Boundary 3 – Week 12

At this point development has essentially halted. All of the requirements have been hit, admittedly this third section of development has been vastly more time intensive and been very stressful. I feel that had more time been spent in section two building additional functionality so much wouldn't have rolled over into this third of the project. This as such has currently left me with a very devoid report, as such the remaining time is going into producing the report and documenting the code as best as possible with only occasional bug fixes from here on out.

14.0 Project Logo: