Deep Learning Project – Face Recognition

# Introduction:

There is much to be gained from using face recognition technology to predict emotions, but the whole idea is ‘problematic’ at best. How unrelated such images are to actual emotions, I encourage all to read “How Emotions Are Made” by the psychologist Lisa Feldman Barrett. While the science is split about how emotions are expressed across individuals and culture, there are worthwhile reasons to continue to predict emotions, such as the detection of mental health difficulties making early intervention possible.

There are many less laudable reasons – e.g., unsanctioned gathering of emotional responses to products or online content - that are ethically dubious. But an open and frank discussion about what the software is doing and how the data is being used, would do much to assuage fears among the public. In a very real sense, the misuse of this software would be a form of unsanctioned data collection, and of easily misunderstood data at that.

### Background:

The face recognition dataset is made up of a training and testing set of pictures of different people ‘emoting’. The pictures are labelled as one of six different emotion types: anger, fear, happiness, neutral, sadness, surprise. The training set has **28,273** images and the test set has **7067** images. There images are not equally distributed across the six emotions; ‘happy’ has the most with **7215** while ‘surprise’, as the least, has **3171**. This may well make ‘happiness’ the easiest for the model to recognise, and this assumption is borne out by the evidence.

The object of the assignment is to train a CNN model on the dataset and to use this trained model to recognise the facial emotions of a subject when they are captured by the computer camera. A further task is to consider how this technology could be utilised in different contexts.

### Research Questions:

There are a range of accuracy scores online from different models but, overall, they range from about **60%** up to **80%**. This may not seem like a lot but given the complex range of cues and the subjective nature of the categorisation, even a score of 60% is impressive. My goal was to research how different models fared with the same dataset, and which one was the most accurate in identifying ‘emotions’.

As the model architecture for the CNN would look roughly the same from student to student, I put more of my focus into how this technology could be applied to the video capture technology. There is a lot that could be done, but I focused on two main areas: mental health and gaming. I even tried to write some simple code that might illustrate my ideas, but in this I was only moderately successful.

# EDA (Exploratory Data Analysis):

A collage of different people's faces

Description automatically generatedThe data is divided into training and testing files under folders divided into the six emotions named above. I downloaded the full folder as I was aware that I might be working away from the net on occasion and that it was better to have the full dataset to hand. I used ‘os’ to access the training folder and to print out 6 images of each emotion as well as its label. I then printed out the total number of pictures in each folder of the test data along with the labels.

I used the ‘.shape’ function to print out the number and dimensions of the pictures in both the training and testing folders. Here, one could see that the total associated with each emotion was uneven. It was also interesting to note that the video model was based on a face recognition database of 7 emotions while the database, as it stands now, is made up of only 6.

# Theory: Method and Modelling:

I found a function that would load images into train and test files of any chosen length, and then created small datasets that I could use to train the model quickly. The function also exported arrays and transformed them into categorical values. Using io from skimage I could then separate the pictures from their labels, and so create y\_train and y\_test sets.

I used ImageDataGenerator to preprocess the images. I did, however, investigate what, why, and how it did it, and availed myself of the ‘data’ augmentation’ possibilities the function allows. By flipping, shifting, and zooming images it’s possible to add extra layers of variation to the model and so reduce over-fitting.

A screenshot of a computer program

Description automatically generatedI researched online the kinds of CNN models that were most popular for identifying images, and most of them were variations on the basic model printed on p 461 of Geron’s book.

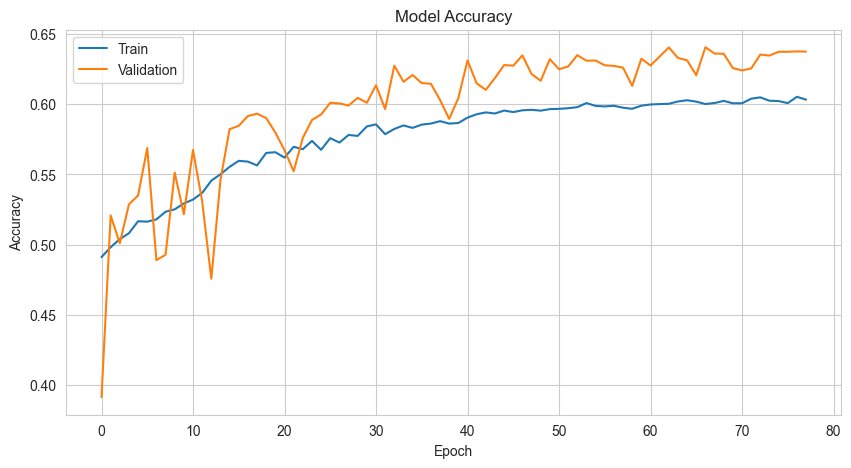
Given the range of possibilities it made sense to me to use keras tuners to identify the most accurate hyperparameters possible. I built a function with four Conv2d layers, a flatten layer, and two Dense layers, and added tuning options into Dropout and the learning method – where there was a choice of two learning methods and a range of values for each. I applied ‘early stopping’ and a save checkpoint, as well as a learning rate plateau (that I learned about online 😊).

The model reached about 63% validation accuracy before the early stopping kicked in. This was easily high enough to run the capture software to a relatively high degree, especially for more distinctive emotions such as ‘happiness’.

# Results & Analysis:

Due to the many problems I encountered (or created for myself), I ended up running three different models of the data. I have outlined each in the ‘problems’ section at the end of the document.

### Results:

Even though I missed the fact that the input shape on my second model was incorrect, it was interesting to see how the training process worked. My first model (with the hypertuner) took some time to run through the many various parameters until it found the optimal one, but once that stage was reached the training process speed was relatively fast. The second model (with different input shape values) took a very long time to process – each epoch took about 16 minutes or so – but the accuracy increased gradually and steadily, as can be seen in the graph. In the end, it was time restrictions which led me to stop the modelling, it could yet be improved with more training.

A screenshot of a computer screen

Description automatically generatedWhat was clear to me throughout the process was that no matter how small the changes to the parameters were, so long as the initial structure was close to the optimal CNN range of layers and cell count, the model would achieve roughly the same level of accuracy.

That accuracy, though, is not consistent across all the emotions, as can be seen in the confusion matrix of the results (from model 2). The model was best at predicting ‘happiness’, with **1586** correct predictions and false negatives for all the other emotions in the single or double digits.

This is because, as this model was trained against the full dataset and ‘happy’ had the highest number of examples, it had a higher sample against which to refine its weights. At the same time, ‘happiness’ is the most visually distinctive of the all the emotions, and this makes it easy for the model to identify it (on a side note, the same goes for humans).

### Analysis:

The different models worked equally well on recognising different emotions through video capture – model 2 had the highest validation accuracy with **64.8**. The main difference between them was the time they took to train. The measured use of Dropout and Batch Normalisation kept the models from over-fitting and helped improve accuracy over time. Some emotions were easier to discern than others (‘happy’), but the results were always interesting and would be of great interest to psychologists; for example, how ‘fear’ and ‘anger’ were the two emotions that were most frequently confused with each other.

Interpretation of the results have applications beyond the technical sphere and could be expanded to include several other disciplines (maybe even the humanities 😊).

# Conclusion:

CNN models are incredibly powerful tools for analysing pictures and predicting labels on new data. They manage this in a relatively short period of time, although they do need a lot of input data to reach higher accuracies (which is same for all machine learning models). Data Augmentation reduced overfitting and allowed the models to increase in accuracy over time for both training and validation datasets. The ability to save models and to use them at another time with different datasets means that the process of strengthening models can take place over an extended period. This also allows for the importation of pretrained models which can powerfully enhance the performance of your software.

# Applying the Model:

In approaching this project, I wanted to place most of my focus on how to apply the model in creative ways. I determined that each CNN model would look roughly the same and would achieve scores within the range I mentioned in my introduction. Therefore, the best way to individualise the project would be to devise possible real-world applications.

### Mental Health Check:

My initial feeling was that this software could be an early warning signal for mental health problems, but only if combined with other pattern recognition software such as voice recognition (to check for pitch, amplitude and pauses), and body recognition models. In this way multiple inputs could be combined to lead to more confident mental health predictions.

Aside from differing cultural displays of emotion, doubts about the scientific validity of emotional archetypes due to recent neurological research, and the subjective choice of the categorisation, there are definite benefits to programs such as these. Any early indication of incipient mental health problems would allow individuals or carers to immediately address problems while they are in their infancy.

I attempted to adapt the video capture program so that a text-to-speech message would identify the emotion and prompt the framed person to talk about how they were feeling (the python file is included in my submission). I ran into difficulties with having two programs running at the same time with both tyring to use the video software simultaneously. But I think the idea has merit and potential.

### Drinking Game:

My second idea was more successful (but now wholly). Imagine a group of friends hanging out and deciding to play a drinking game. In this game, the person looks into the camera and is told a series of jokes. They goal is to keep a straight face and not laugh, because as soon as they laugh or even smile, the counter starts to go up. If they rack up more than 30 points during the game they lose – and, of course, have to down a shot.

I wanted to use the ‘text to speech’ to read out jokes from a list, but I had to create two functions and have them run simultaneously. In the end I couldn’t get it to work, but you can get a taste of what I had planned to do from the attached program (just link it to a successful face recognition CNN). It should add points every time you smile and then inform you at the end whether you went above the limit or not.

### Breathalyzer:

My last idea is a counterpoint to the drinking game. If a CNN was trained on thousands of faces in various stages of inebriation with attached labels of their actual alcohol quotient when the picture was taken, the camera software could be used to inform drivers whether they are above the legal limit to drive or not. In Sweden that limit is so low that any alcohol intake at all is too much. Unfortunately, that is not the case in a lot of other countries.

That is what I came up with, but there are many other possibilities to the how this technology could be ethically applied to improve the lives of individuals and the broader society.

# Summation:

### Problems:

I had a few mini disasters during this process; not enough to derail me, but they drained time and energy that I wanted to spend developing products and applications.

The saved h5 file of my first model got over-written. This was completely my fault, I decided to run the model again but forgot to load the original model and the ‘save model’ function over-wrote what was there, so I was back to 19% accuracy again.

I took the opportunity to try out another model that I’d found online, to see if it would work more quickly and more accurately. But I forgot to adjust the input shape and ended up with something that wouldn’t work with the video capture software. I tried to find a solution but was unsuccessful and the attempt cost me half a day.

Due to the time deficit, the third model I ran was based on the YouTuber’s one. I needed to quickly produce a functioning model that I could test my application ideas on. Training this model on a smaller dataset greatly speeded up the learning process and I quickly reached an accuracy that would apply well to the capture software. However, it was nowhere close to as accurate as my first model.

### Grade:

I’m going to be very non ‘prestigelös’ and say that I deserve a **VG**. Even though I had many issues with my modelling and didn’t get the accuracy I was hoping for, or get a chance to use a pretrained model (which was my plan) to see if it performed better on the data; I took the time to apply a tuner, and spent so much time running different models that I feel I have a ‘deep’ (pun intended) understanding of how CNN’s work.

I also tried to come up with creative ways to apply the model. I made a ‘valiant’ effort to write programs that would realise those ideas, and while they didn’t quite come off, I should extra earn points for the effort made.

### Tips:

* Better understanding the architecture of your model, and how you can achieve optimal results.
* Understand how the ‘save’ and ‘load model’ functions work so you don’t overwrite a day’s work.
* Do more research into how images are processed for modelling. It will help with hyperparameter choice, even if that choice is a range of values being entered into a tuner.
* Use a pretrained model to determine how much better it can perform on the same data.