CO3519 Assignment 2 – Facial Emotion Recognition using Advanced AI

Author : James Birkenhead | G20983016

## ***Introduction***

Facial Emotion Recognition (FER) encompasses computer vision tasks which are aimed at identifying various human emotions depicted via a face and then categorizing them into various emotional categories. The process of detecting emotions normally revolves around analysing facial landmark positions such as nose, eyebrows and mouth. Changes to these positions can be analysed and then categorised into emotions.

FER is deemed to be an important and advantageous application of Artificial Intelligence technologies, since much communication is non-verbal. The exact amount is contested, however some studies suggest up to 60-80% of communication is non-verbal, providing a significant area for potential use and deployment. FER also has numerous applications across various sectors from education, to autopilot systems, to neuroscience.

The previous paper on this topic discussed the implementation of an FER algorithm using Machine Learning via Histogram of Oriented Gradients feature extraction and Random Forest Emotion Classification. This was somewhat powerful; however, it also had some limitations.

* Performance was mixed across various emotions. Whereas the algorithm performed well on some emotions such as happiness and surprise, the algorithm struggled with classifying anger and sadness.
* Due to the simplicity of the model. underfitting was a problem. This meant that certain patterns in the data couldn’t be captured, and performance on the testing data for some emotions was comparatively poor.
* While some of the observed discrepancies could’ve potentially been due to biases in the training datasets, there was nonetheless deemed to be areas for improvement which could be built on by using a more complex algorithm.

To try and improve the performance of FER, this paper will discuss the implementation of an Advanced AI algorithm based on Convolutional Neural Networks (CNNs). These networks are specialized deep learning algorithms designed for tasks that involve object recognition. CNNs contain several key components…

* **Convolutional Layers** apply a sliding window function to the matrix of pixels. The sliding function is called a filter, and several filters can be applied.
* **Rectified Linear Unit Activation Functions** are applied after each convolution operation, aiding the network to learn non-linear relationships between image features. This makes the network stronger at identifying different patterns.
* **Pooling layers** pull the most significant features from a convoluted matrix by applying aggregation operations which reduce the dimension of the matrix. This in turn reduces the amount of memory when training the network
* **Fully connected layers** are the last layer of the CNN and their inputs correspond to the flattened one dimensional matrix generated by the last pooling layer.

An example architecture of a CNN can be found below…

A diagram of a diagram of a block diagram

AI-generated content may be incorrect.

The potential CNN architectures that will be discussed include VGG and ResNet.

## ***Literature Review***

Visual Geometry Group (VGG) is a standard deep CNN. VGG-16 is an example of such network and consists of 16 layers. This model has been found to achieve a test data accuracy of 92.77% with the ImageNet dataset. Images of size 224x224x3 are passed through various layers. The convolution layer has a 3x3 filter size while the max pooling layer has a filter size of 2x2. VGG-19 consists of three more convolutional layers than VGG-16 which should allow it to capture more complex image features.

ResNet50 is another CNN consisting of 50 convolution layers, making it a deeper network than VGG-16 and VGG-19. There are four main sections, the convolutional layers which extract features from the input image, an identity block and convolutional block which process and transform the features extracted, and then the fully connected layers which are used to make the final classification. In addition, unlike VGG, ResNet-50 uses residual blocks which allows the network the skip certain connections which enables better performance.

Mascarenhas and Agarwal (2021) carried out feature extraction on a dataset of 6000 images which could be classified into 5 different categories; shoes, beauty, jewellery, watches and bags. Once the features were extracted, they were fed into the various models to classify the images. All of the images needed to be compressed since they were of too great a size to be inputted into the convolutional layers. Classification was tested using each algorithm with different numbers of epochs. The study found that ResNet50 exhibited the highest accuracy followed by VGG-19 and VGG-16. They also found that epoch values above 20 induced overfitting and values below 15 induced underfitting, indicating that an optimal range of epoch values is between 15 and 20. This gives some indication about the benefits, however a potential limitation of this study was that it was specifically covering the classification of images of objects into certain categories, not facial recognition. Therefore, further research was required to understand whether this was a consensus or an outlier.

The conclusion that ResNet50 produces higher accuracy than VGG appears to be a consensus across the literature. Truong Van Nguyen and Tuan Duc Chu (2023) also compared VGG and ResNet50 however this was specifically for facial recognition. The two algorithms were implemented to measure the accuracy of each one and compare them. The learning rate was set to the same value of 0.0001 for both algorithms and the models were trained for 80 epochs on 2900 face picture from 52 people. This makes this paper more relevant to this use case than Their study also found that the accuracy of ResNet-50 was higher than VGG-16, with values of 95.6% for ResNet and 94% for VGG-16. Both algorithms were more accurate than SVN at 88% highlighting the benefit of deep learning. However, while this study does cover the application of the algorithms for facial recognition, it only covers targeting a face in an image rather than classifying emotions.

Dhankhar (2019) on the other hand compares the algorithms for specifically classifying emotions. This study involved developing CNN models to classify emotions using VGG and ResNet 50, leveraging the Kaggle Facial Expression Recognition Challenge and Karolinska Directed Emotional Faces (KDEF) datasets. The images from the two datasets were classified into six separate emotions – anger, disgust, fear, happiness, sadness, surprise and neutral. The study once again backed up the consensus that ResNet is advantageous compared to VGG, since on both the KDEF dataset, and the KDEF dataset with transfer learning from the Kaggle models, ResNet-50 produced a higher accuracy score, but also a higher precision and recall score. This made the consensus that ResNet-50 would be advantageous over VGG even more clear, since it covered the specific use case which the model to be developed will be deployed for.

In conclusion, this literature review has indicated that ResNet-50 will be the appropriate CNN to use for this task. The literature has widely indicated that ResNet-50 produces higher levels of accuracy compared to more traditional CNNs such as VGG, both for general classification but also for facial recognition and facial emotion classification. Therefore, this will inform the development of a CNN model using ResNet-50.

## ***Datasets***

**CHANGE TO AVOID SELF PLAGIARISM**

The JAFFE dataset is a set of 213 images from 10 Japanese female subjects with various facial expressions. This dataset was created for non-commercial scientific research. Each of the ten subjects did seven facial expressions to correspond to the following emotions…

* Happiness
* Sadness
* Surprise
* Anger
* Disgust
* Fear
* Neutral

A sample of these images can be found in Appendix A.

The images are already in grayscale format and clearly show a face. Converting images to grayscale is important for this study since some detection algorithms like HOG are defined based on grayscale values, and generally using grayscale also reduces complexity which in turn reduces the demands on memory and processing power. There is also no evidence of significant background noise in the sample images that would need to be filtered out. A link to the dataset can be found here <https://paperswithcode.com/dataset/jaffe>. Before being inputted into the model, the various images were organised into their respective emotions and grouped together into folders.

The Cohn-Kanade Dataset (CK+) is a dataset that was created in 2000 for the purpose of promoting research into automatically detecting facial expressions. There are 593 video sequences from 123 different subjects ranging from 18 to 50 years old with various genders and heritage. This is advantageous compared to the JAFFE dataset since it only focuses on Japanese female subjects, meaning the model can be trained on a more diverse range of subjects. Once again, there is no evidence of significant background noise in the sample images that would need to be filtered out. A link to the dataset can be found here <https://www.kaggle.com/datasets/shuvoalok/ck-dataset?resource=download>.

Preprocessing of the images was required…

* **Image resizing :**  The images needed to be resized to 224x224 pixels. This is because ResNet expects input images to be of that size, so not resizing the images could’ve caused errors.
* **Tensor Conversion** : This was required to convert the images to PyTorch tensors. This is done to enable the images to be used more efficiently in a deep learning pipeline.
* **Normalization** : The images were normalized using the mean and standard deviation of the ImageNet dataset. This ensured that the input data was aligned with the expectations of the model, making the process of training the model more efficient.

## ***Model Development***

A diagram of the proposed model can be found in Appendix B. The stages are as follows…

**Input grayscale image.**

Images are input from the dataset. Each of these images are grayscale which reduces complexity which in turn reduces the demands on memory and processing power.

**Resize images to 224x224**

The images are then be resized to 224x224 pixels. This is because the ResNet50 CNN expects inputted images to be of this size, so inputting images which aren’t that size has the potential to cause errors.

**Convert images to Tensor and Normalize**

In Python, PyTorch Tensors are multi-dimensional arrays with a uniform type, similar to multi-dimensional arrays in libraries such as NumPy but provide greater flexibility due to their dynamic dimensions and are more optimized for machine learning tasks (Harshan Kumar and Porchezhian, 2023). Therefore, the Tensor type is used for the deep learning pipeline. Following this, the pixel values of the image are normalized using the mean and standard deviation values of the ImageNet dataset which ensures that the input data is aligned with the expectations of ResNet50. This makes the training of the model more stable and more efficient.

**Load ResNet50**

The next step is to load in the ResNet50 model which has been pre-trained on ImageNet, making this a form of transfer learning which reduces the amount of time and resources spent training the model.

**Train Model**

In line with the ResNet50 architecture provided in Appendix C, the images will pass through several main parts…

* The initial convolution layers will apply filters to the inputted images to detect patterns, edges and textures in the faces. This will be followed by batch normalization, where the activation of a layer is normalized within a “mini batch” which should speed up training and provide regularization to aid with preventing overfitting. Following this, the ReLU activation function will be applied to induce non-linearity.
* The convolution blocks consisting of multiple convolution layers followed by batch normalization and activation functions will facilitate extraction of high-level-features.
* A set of residual blocks will allow for shortcuts or the skipping of connections, which will allow the model to potentially skip layers if required. This will aid with smoother flow of information through the model and mitigate the vanishing gradient problemwhich would’ve caused the gradients being used to update the network to become incredibly small or even vanish completely, which would hinder the training process.
* A set of fully connected layers will make predictions of the emotion based on the features that have been extracted.

## ***Model Evaluation***

The model was trained and tested using the two datasets outlined in the Datasets section. These datasets contained various images for training the model and also images for testing the model.

The model was implemented using python via the following libraries…

* Cv2
* NumPy
* pyTorch
* torchvision
* matplotlib
* sklearn.metrics
* os

The model was trained on six separate emotions…

* Anger
* Fear
* Happiness
* Neutral
* Sadness
* Surprise

The first round of training was carried out using a combination of the JAFFE dataset and the CK+ dataset.

Following the training of the model, the model was tested using **GET NUMBER** images which had been categorised into the same emotions used for training. Following this, several measures were used to evaluate the performance of the model…

* Accuracy measured the percentage of correct classifications.
* Loss measured the difference between the predictions that were made and the actual classification.
* An accuracy and loss plot are also rendered both for the testing data and the training data across the various epochs. This shows how the level of accuracy and loss varied across epochs.
* A confusion matrix was produced. This was a table with several combinations of predicted and actual values, with the cells showing the number of correctly classified emotions and incorrectly classified emotions. This provided a more in-depth visualization of the accuracy of the model. The confusion matrix was used to deduce the number of true positives, true negatives, false positives and false negatives.
* Precision measured the proportion of true positive predictions compared to all the positive predictions
* Recall measured the proportion of actual positive predictions.
* F1-Score combined the precision and recall scores to give a general indication of the performance of the model.

The various results from this run can be found in Appendix D.

From this data, several conclusions can be drawn…

* **The model progressively became more accurate on the training data as the number of epochs increased :** The model’s accuracy on the training data initially was poor at 44%, however this increased over time and after 10 epochs, the training accuracy had increased to 90%.
* **On testing data, the model followed a similar trajectory however it was much slower and less smooth** : The model’s accuracy on the test data on the first run was just 31%, however it did increase over time albeit with much variation. On epoch 5, the accuracy was 61% however on epoch 7 the accuracy was much lower at 35%. However, the final test accuracy on epoch 10 was 57% which was still an increase over the initial accuracy. However, this is still 33% lower than the training data, which could indicate overfitting.
* **The model performed best on classifying Happiness, Neutral and Surprise** : The model performed best on these emotions with F1 Scores of 79%, 65% and 70% respectively.
* **However, 00the model performed very poorly on others** : On Sadness and Fear, F1 Scores of 30% and 0 were respectively logged. Fear was the most notable with no true positive classifications. This could’ve been due to a small amount of fear samples in the testing dataset. **CHECK**
* **The deep-learning model generally performed better compared to the previously defined Machine Learning Model** : As expected, the model defined in this report generally performed better across all of the emotions barring an outlier of fear than the traditional Machine Learning model discussed in the previous reports going off the F1 Score. Given the two models were trained on the same data, this emphasises the advantage of deep learning models.

This indicates that the model generally performed well and also better than the traditional Machine Learning model previously defined, however there were still some glaring issues that needed to be addressed, notably the gaps in performance on training and testing data and the poor performance on classifying sadness and fear. Several factors could potentially explain this…

* **Overfitting** : This was less expected with ResNet50 since it is designed to reduce overfitting, however it’s still a potential problem on a CNN, especially with smaller datasets.
* **Dataset is too small :** Because of the higher complexity of a CNN model compared to a traditional machine learning model, particularly ResNet50, having a small dataset increases the risk of overfitting. This model was trained on the same dataset which induced underfitting on the traditional model, **(VERIFY)** which could point to the dataset being a limitation.
* **Imbalance in the training dataset :** The poor performance on some emotions could be explained by there being more images in the training dataset for some classes compared to others.

The first route that I tried was to replace the dataset with a larger one. I replaced the dataset with a subset of the FER dataset.

## ***Conclusion***

## ***Appendix***

Appendix A:

Appendix B : Diagram of implementation



Appendix C : ResNet50 Model

A diagram of a graph

AI-generated content may be incorrect.

Appendix D : First run results

A diagram of a confusion matrix

AI-generated content may be incorrect.

A graph of a line and a line

AI-generated content may be incorrect.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | TP | FP | TN | FN | Accuracy | Precision | Recall | F1 |
| Anger | 13 | 13 | 127 | 5 | 87.5% | 50% | 72% | 59% |
| Fear | 0 | 0 | 153 | 7 | 95.4% | 0 | 0 | 0 |
| Happiness | 20 | 1 | 132 | 10 | 93.38% | 95% | 67% | 79% |
| Neutral | 23 | 12 | 118 | 13 | 85.29% | 66% | 64% | 65% |
| Sadness | 3 | 0 | 160 | 14 | 92.65% | 100% | 18% | 30% |
| Surprise | 23 | 8 | 134 | 12 | 89.71% | 74% | 66% | 70% |

## ***References***

Dhankhar, P. (2019). ResNet-50 and VGG-16 for recognizing Facial Emotions. *International Journal of Innovations in Engineering and Technology*, [online] 13(4). doi:https://doi.org/10.21172/ijiet.134.18.

European Data Protection Supervisor (2021). *Facial Emotion Recognition*. [online] Available at: <https://www.edps.europa.eu/system/files/2021-05/21-05-26_techdispatch-facial-emotion-recognition_ref_en.pdf>.

Harshan Kumar, K. and Porchezhian, P. (2023). *Understanding the Difference Between Arrays and Tensors in TensorFlow*. [online] Medium. Available at: https://medium.com/@mysterious\_obscure/understanding-the-difference-between-arrays-and-tensors-in-tensorflow-175a9555b70 [Accessed 27 Mar. 2025].

Mascarenhas, S. and Agarwal, M. (2021). *A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification*. [online] IEEE Xplore. doi:https://doi.org/10.1109/CENTCON52345.2021.9687944.

Truong Van Nguyen and Tuan Duc Chu (2023). Comparative study on the performance of face recognition algorithms. *Eureka, Physics and Engineering./Eureka, Physics and Engineering*, (4), pp.120–132. doi:https://doi.org/10.21303/2461-4262.2023.002831.