

## Facial Expression Recognition with Machine Learning



## Oliver Dickens 1805914

---

## 1. Introduction

---

Facial expression recognition (FER) provides machines a way of detecting emotions through pattern analysis and artificial intelligence. The importance of facial expression recognition in key areas like CCTV security for shops and airports, especially in America with guns means it could make the difference of saving lives to detecting mental illnesses. Selling promotional items in shops to gauge reactions could also be a big use, it's not always easy to get verbal or written feedback so having facial emotion which could be very valuable.

Currently Google are exploring FER in their cars and would detect if someone who is in the car can be allowed to start the car or even drive it (Nagendhiran, 2019). FER could detect drowsiness in the face and alert the driver if they are away to fall asleep. If FER doesn't get banned in the auto-mobile industry it could go far.

Reading an article on the facial expression recognition (Neves, 2019) where the researchers use various pre-processing techniques on their images before passing them into the system. Face detection is a pre-processing technique which detects a face within an image and will place pixels at certain places along the face, if it can discover the facial features.

The researchers then used geometric transformations to correct head rotations in the images and then would crop around the face in the image. The images are then contorted and smoothed to remove noise. Data augmentation then mirrors, scales, and rotates the images as data sets. The data sets are small and are passed into the program as training data.

The aim was to replicate the work of these researchers but with a difference, instead of passing the data into a "histogram of oriented gradients" (Neves, 2019), the data will be Distance Normalized between the key facial features. This means the images will not require pre-processing and will become easier to implement as a way of measuring the similarity between two objects. The idea is by having the algorithm implemented in this manner the outputting of results will also become a simpler. All the expressions that this AI will train on are:

- Joy
- Fear
- Sadness
- Anger
- Disgust
- Surprise

These expressions or emotions are called universal expressions.

This program will be extracting these facial features from both the testing and training sets of data. This extracted data is then passed into the artificial Neural network to train and learn of the differences between the emotions. This model then needs to be trained and evaluated to evaluate how well it predicted the emotions using both micro and macro accuracies coupled with the precision and recall for each class. There is an option to evaluate a single image for any of the emotions within the directory.

---

## 2. Methodology

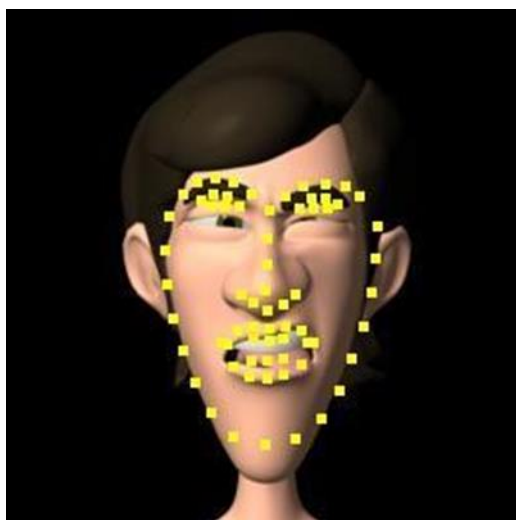
---

---

## 2.1 Gathering Data

Gathering data for training and testing the models didn't take long, Images were provided which and speeded up the process immensely. The images themselves were varying in quality and had to be separated to train the model. The MUG facial expression database (Delopoulos, 2010) was mostly used and the images it provided were of great quality and showed clear facial expressions. The Cohn-Kanade AUCoded Facial Expressions database also provided a good outcome of images (Kanade, 2000) and were good testing data. Additionally, the JAFFE images (Michael J. Lyons, 1998) were used but showed only female Asian people in black and white. Some of the images were taken out as they showed little to no emotion.

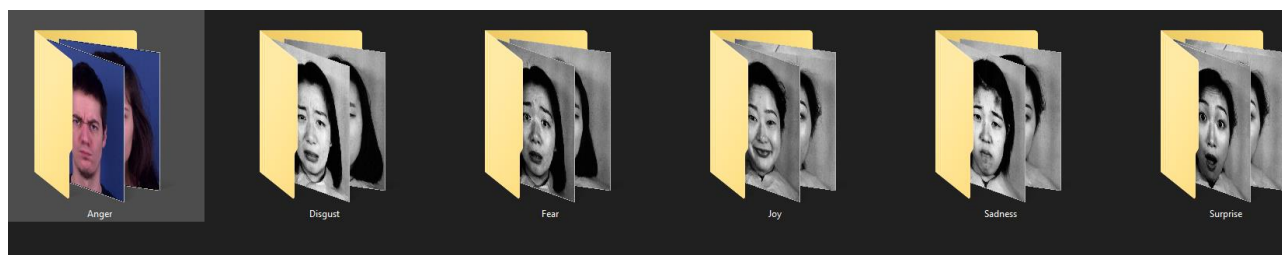
A dataset called Modeling stylized character expressions via deep learning data-set (DlibDotNet, n.d.) was considered however the plastic looking cartoon faces made it difficult to work with, here is an example.



(Grail.cs.washington.edu, n.d.)

With these plastic faces it quickly becomes difficult to mark out the eyes and lips and had to be scraped in the end.

All data was gathered and placed into separate folders, named after the emotion each image had. This is because it easier to feed the data into the program while also being easier to manually sort into each emotion folder for the user.



---

## 2.2 Feature Extraction

---

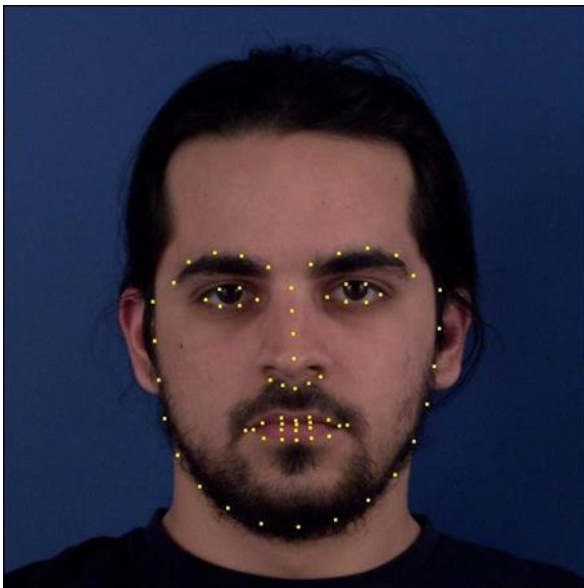
This is the extraction of the key facial features like eyebrows, eyes and lips from faces within the images.

The full extent of the key features used in this project are as follows:

- Left & right eyebrows,
- Left and right side of the lip,
- Lip width and height,
- Height and width of both the left and right eyes,
- Distance between the bottom lip and the nose,
- Distance between the left eye and the left side of the lips,
- Distance between the right eye and right side of the lips.

These feature distances will change depending on the facial expression and the individual person in the image. The program is then trained on the normalized distance between the features and learns what expressions corresponds to the different distances between features. One issue with this method when passing through the pixels from the image is the quality of the image, if the face in the image is not looking straight on then there could be less pixels in the areas needed to map and extract the pixels. Leading to the image not being registered as a human face and less consistency and poorer results.

Over the course of the project the images from DlibDotNet NuGet package by Takuya Takeuchi (DlibDotNet, n.d.) along with Visual Studios 2019 (Microsoft, 2019) were used to detect and mark out the key features area along the face. Below is an image and how the model reads it.



(Delopoulos, 2010)

---



(Kanade, 2000)

These images are contrastingly different to humans and one is noticeably in color whereas the other is in black and white, yet the model will recognize both as faces and as the images framing, scale, quality and looking straight on are consistent.

After extracting the feature vectors, they are stored within a excel.csv file along with the headers used to store the values.

```
2 references
public void CreateNewCSVFileToExtractTo(string fileName)
{
    // Header definition of the CSV file
    string header = "Label,LeftEyebrow,RightEyebrow,LeftLip,RightLip,LipHeight,LipWidth\n";

    // Create the CSV file and fill in the first line with the header
    File.WriteAllText(fileName, header);
}
```

The individual extracted directory data are then specified to be extracted from.

---

2 references

```
public void ExtractData(string directory, string extractTo)
{
    // Extract the surprise folder
    string surpriseDir = directory + "/Surprise";
    this.ExtractExpresionDirectory(surpriseDir, extractTo, "Surprise");

    // Extract the sadness folder
    string sadnessDir = directory + "/Sadness";
    this.ExtractExpresionDirectory(sadnessDir, extractTo, "Sadness");

    // Extract the fear folder
    string fearDir = directory + "/Fear";
    this.ExtractExpresionDirectory(fearDir, extractTo, "Fear");

    // Extract the anger folder
    string angerDir = directory + "/Anger";
    this.ExtractExpresionDirectory(angerDir, extractTo, "Anger");

    // Extract the disgust folder
    string disgustDir = directory + "/Disgust";
    this.ExtractExpresionDirectory(disgustDir, extractTo, "Disgust");

    // Extract the joy folder
    string joyDir = directory + "/Joy";
    this.ExtractExpresionDirectory(joyDir, extractTo, "Joy");
}
// https://medium.com/machinelearningadvantage/detect-facial-landmark-points-w
```

A loop happens which goes through the images individually and stores the directory and extracts the feature vector from each image.

```
// https://medium.com/machinelearningadvantage/detect-facial-landmark-points-with-c-and-dlib-in-only-50-lines-of-code-71ab59f8873f
2 references
public FacialData ExtractImageFeatures(string imageFile, string extractTo = "DontSave", string expression = "Default")
{
    // File-paths
    string inputFilePath = imageFile;

    // Facial features
    float leftEyebrow = 0f, rightEyebrow = 0f, leftLip = 0f, rightLip = 0f, lipHeight = 0f, lipWidth = 0f;

    // Change what the label is depending on if an expression was passed in or if the type is in the name of the file
    string label = "";
    if (expression == "Default")
    {
        label = GetExpressionFromImageName(imageFile);
    }
    else
    {
        label = expression;
    }
}
```

When detecting the face and facial features before extraction both the Dlib (DlibDotNet, n.d.) `Dlib.GetFrontalFaceDetector` and `ShapePredictor.Deserialize("shape_predictor_68_face_landmarks.dat")` are used.

```
// set up Dlib facedetectors and shapedetectors
using (var fd = Dlib.GetFrontalFaceDetector())
using (var sp = ShapePredictor.Deserialize("shape_predictor_68_face_landmarks.dat"))
{
    // load input image
    var img = Dlib.LoadImage<RgbPixel>(inputFilePath);

    // find all faces in the image
    var faces = fd.Operator(img);
    // for each face draw over the facial landmarks
    foreach (var face in faces)
    {
        // find the landmark points for this face
        var shape = sp.Detect(img, face);
    }
}
```

Here we can see the `Dlib.GetFrontalFaceDetector` setting up the object to detect faces and the operator function is called with an image directory to find the faces in the image. The `ShapePredictor` is then used to find the landmarks in the images that found faces in the `GetFrontalFaceDetector`. This is used to calculate the distance between features and is accomplished differently depending on the feature being calculated.

Below is an image showing how to get the distance between the

## 2.3 Training the Model

When training the model a plugin for Visual Studios (Microsoft, 2019) was used called `ML.Net` (Microsoft, n.d.). This lets you set up `MLContext` and `IDataView` objects. This allows the `Training Vector.csv` files to be extracted.

```
// Load data
IDataView dataView = mlContext.Data.LoadFromTextFile<FacialData>(directory, hasHeader: true, separatorChar: ',');

// Define the feature vector name and the label column name
```

Once the Vectors are loaded the pipeline is initiated to apply the data passed in. This is then used to train the model. Once the model is trained using this Vector data it is saved for testing.

```
// The image transforms the images into the model's expected format
var pipeline = mlContext.Transforms.Conversion
    .MapValueToKey(inputColumnName: "Label", outputColumnName: labelColumnName)
    .Append(mlContext.Transforms.Concatenate(featureVectorName, "LeftEyebrow", "RightEyebrow", "LeftLip", "RightLip", "LipHeight", "LipWidth"))
    .AppendCacheCheckpoint(mlContext)
    .Append(mlContext.MulticlassClassification.Trainers.SdcaMaximumEntropy(labelColumnName, featureVectorName))
    .Append(mlContext.Transforms.Conversion.MapKeyToValue("PredictedLabel"));

// Train model
model = pipeline.Fit(dataView);

// Save model
using (var fileStream = new FileStream("model.zip", FileMode.Create, FileAccess.Write, FileShare.Write))
{
    mlContext.Model.Save(model, dataView.Schema, fileStream);
}
```

## 2.4 Predicting Single Image

---

Predicting a single image comes from inputting an image into the system to get the desired expression from that image and loading the model in to read it.

```
2 references
private void LoadModel()
{
    DataViewSchema dataViewSchema = null;
    using (var fileStream = new FileStream("model.zip", FileMode.Open, FileAccess.Read))
    {
        model = mlContext.Model.Load(fileStream, out dataViewSchema);
    }
}
```

ML.Net (Microsoft, n.d.) is used to extract the feature Vector from the image and put into a class called FacialData. This predictor function then makes a prediction to figure out what emotion is being displayed and printing the result to the user.

```
1 reference
public void Predictor(string imageDirectory)
{
    // Week 8 Practicals
    // Loading in the model
    LoadModel();

    // Setup the predictor using the model from the training
    var predictor = mlContext.Model.CreatePredictionEngine<FacialData, DiffnetExpressionPrediction>(model);

    // Extract the features from the image and make a prediction
    FeatureExtraction featureExtraction = new FeatureExtraction();
    FacialData faceData = featureExtraction.ExtractImageFeatures(imageDirectory);
    var FacePrediction = predictor.Predict(faceData);

    // Output Values
    Console.WriteLine($"**** Prediction: {FacePrediction.Label } ****");
    Console.WriteLine($"**** Scores: {string.Join(" ", FacePrediction.Scores)} ****");
}

// Evaluating Performance
```

## 2.5 Evaluating the Model

To successfully evaluate the model and display the correct metrics the model is loaded in, then testDataView is used to load in the extended feature Vectors for the data that was extracted in 2.1 of this report. The test metrics are then defined which hold all the test metrics that have been evaluated from the model using the passed in test data.

```
2 references
private void LoadModel()
{
    DataViewSchema dataViewSchema = null;
    using (var fileStream = new FileStream("model.zip", FileMode.Open, FileAccess.Read))
    {
        model = mlContext.Model.Load(fileStream, out dataViewSchema);
    }
}
```



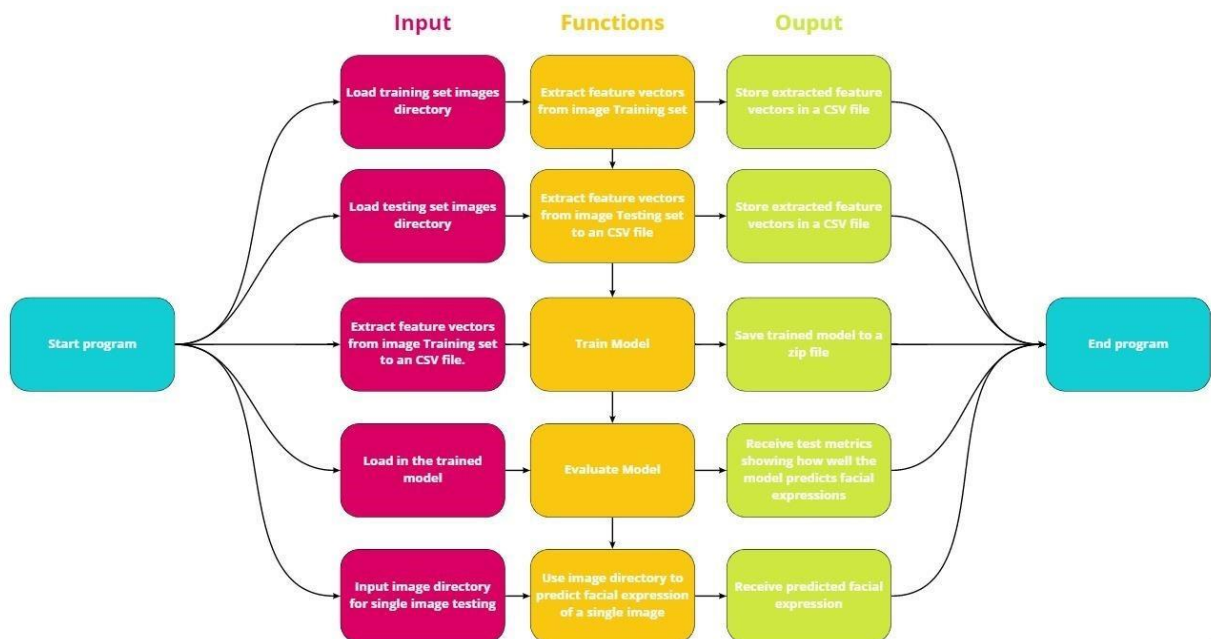
Using these metrics, we get back data printed out like this:

```
// Writing to the console
Console.WriteLine($"* Metrics for Multi-class Classification model - Test Data");
Console.WriteLine($"* MicroAccuracy:    {testMetrics.MicroAccuracy:0.###}");
Console.WriteLine($"* MacroAccuracy:    {testMetrics.MacroAccuracy:0.###}");
Console.WriteLine($"* LogLoss:          {testMetrics.LogLoss:#.###}");
Console.WriteLine($"* LogLossReduction: {testMetrics.LogLossReduction:#.###}");
```

When stepping through the evaluate function we see the precision and recall of each class, this can be immensely helpful when analyzing which class is performing lower then expected.

```
// Precision Per Class:
Console.WriteLine($"* ConfusionMatrixPrecision:");
System.Collections.Generic.IReadOnlyList<double> precisionList = testMetrics.ConfusionMatrix.PerClassPrecision;
for (int i = 0; i < precisionList.Count; i++)
{
    Console.WriteLine($"*    - {(DifferentExpressionType)i} : {precisionList[i]:#.###}");
}
// Recall per class
Console.WriteLine($"* ConfusionMatrixRecall:");
System.Collections.Generic.IReadOnlyList<double> recallList = testMetrics.ConfusionMatrix.PerClassRecall;
for (int i = 0; i < recallList.Count; i++)
{
    Console.WriteLine($"*    - {(DifferentExpressionType)i} : {recallList[i]:#.###}");
}
```

## 2.6 Program Layout



miro

---

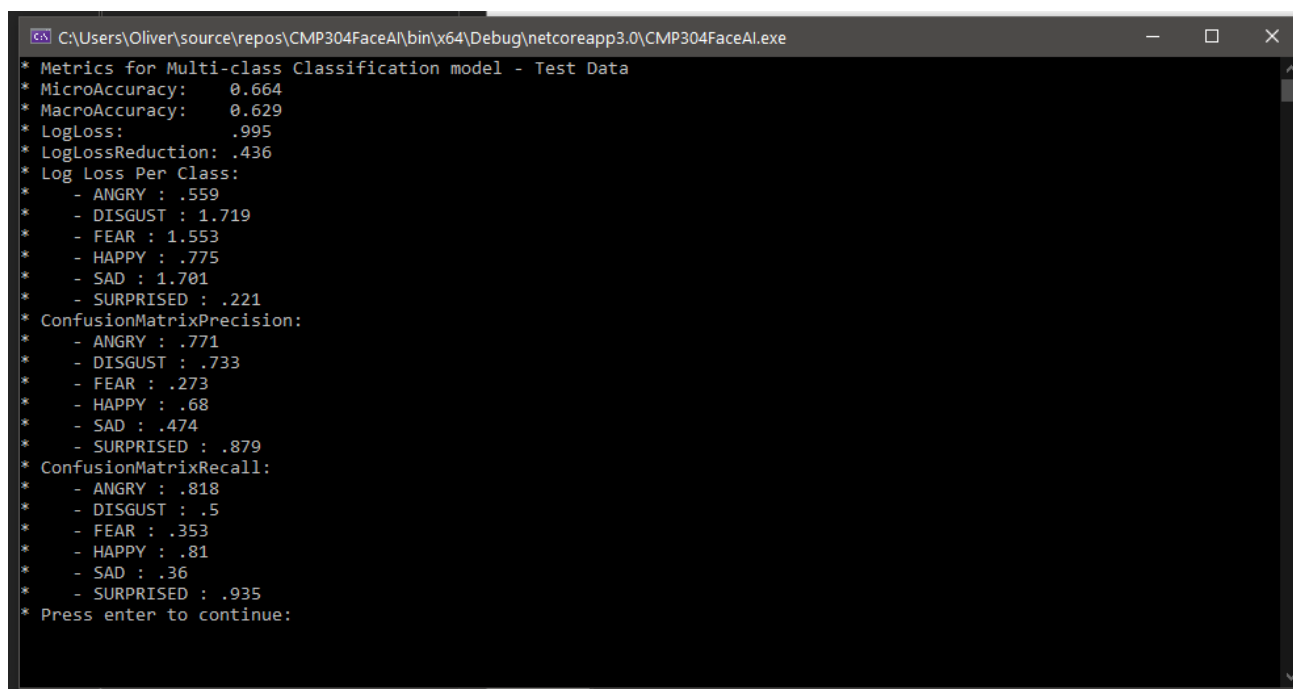
This program is structured so users can choose from a command menu what they want, and this is run in the way the graph is shown above. The user can choose to either extract the training data, extract the test data, evaluate the model and or pass a single image to receive a label for the emotion found in that image. Users can end the program after choosing option 5, after any of the options are chosen the results will be generated and saved.

---

### 3. Results

---

When running the program and extracted the feature vectors, the model was trained and then evaluated. Below is the outcome.



```
* Metrics for Multi-class Classification model - Test Data
* MicroAccuracy:    0.664
* MacroAccuracy:    0.629
* LogLoss:          .995
* LogLossReduction: .436
* Log Loss Per Class:
*   - ANGRY : .559
*   - DISGUST : 1.719
*   - FEAR : 1.553
*   - HAPPY : .775
*   - SAD : 1.701
*   - SURPRISED : .221
* ConfusionMatrixPrecision:
*   - ANGRY : .771
*   - DISGUST : .733
*   - FEAR : .273
*   - HAPPY : .68
*   - SAD : .474
*   - SURPRISED : .879
* ConfusionMatrixRecall:
*   - ANGRY : .818
*   - DISGUST : .5
*   - FEAR : .353
*   - HAPPY : .81
*   - SAD : .36
*   - SURPRISED : .935
* Press enter to continue:
```

#### 3.1 Micro-Accuracy

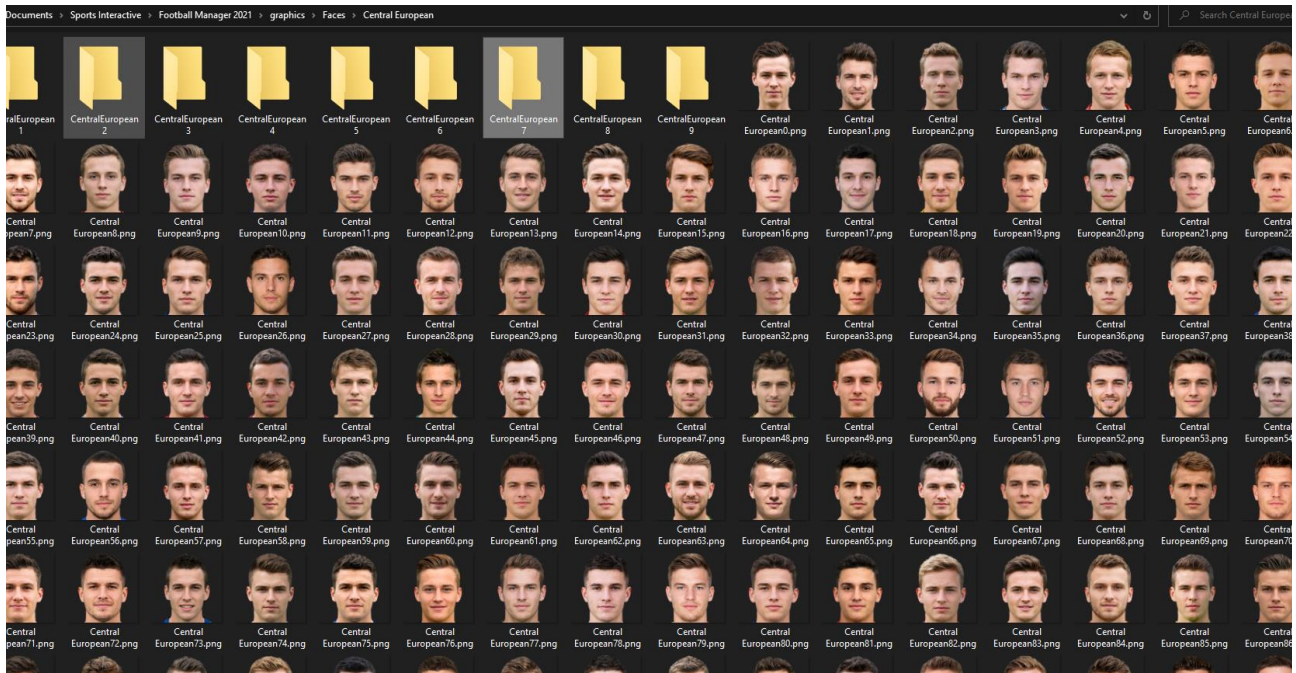
The micro accuracy above shows the result of 0.664 and tells us it is roughly 66% accurate after rounding up.

This has a higher than average chance on giving a correct prediction.

---

This is a good result but there is a lot more room for improvement. There is a good amount of test images to train with however most of the images themselves are of poor quality, the best images have been selected and salvaged for this program.

With access of nearly 9 gigs worth of football players face images, which have been artificially generated and aren't real. These images have been made through a machine learning program that mirrors faces and targets ethnicity and hair (Maradonna, 2021). The images are of great quality however, they all look relatively the same and if it was to be sorted through, the only benefit would be for the joy emotion as that is the only facial emotion that will benefit.



(Maradonna, 2021)

## 3.2 Macro-Accuracy

The macro accuracy of 0.629 comes out as 63%. This is low but the average prediction percentage per class was 63% accurate. This value is lower than the micro-accuracy and shows that some classes are performing better than others, this is still an accurate percentage.

To improve this percentage, adding more images to the testing and training data and balance out the number of images in each emotion. If there were 100 images in each emotion this would be a lot more accurate, however the downside would be the speed of the program and how long it takes to extract the data.

## 3.3 Log-Loss

The log loss came out at 0.995 and is a poor result. This means the probability of the program predicting and placing the image in the wrong class is quite high.

---

This is likely due to the images having similar facial features, for example fear and surprise both have open mouth and can cause the confusion.

### **3.4 Log-Loss Reduction**

The log loss reduction of 0.436 shows the models probability and is low. This means the models probability of predicting correctly is 43% better then guessing.

This is chalked up to the same issues addressed in 3.3 log loss.

### **3.5 Precision Per Class**

The precision per class results are as follows: anger at 77%, disgust at 73%, fear at 27%, joy at 68%, sad at 47% and surprised at 88%.

This was a good account of the accurate predictions and nearly all scored highly except fear and sad. For the two low scores, this was due to the amount of false positive being high and the similarity clash between the images for fear and sad. Being so similar it shows the limitations for the prediction model.

To improve the model, it should be made to pick up on more subtle changes when moving from image to image like hair and ethnicity. This could be a lengthy process training the model on more then emotions.

### **3.6 Recall Per Class**

The recall per class results are as follows: anger at 82%, disgust at 50%, fear at 35%, joy at 81%, sad at 36% and surprised at 94%.

I was taken back at how better the joy class in recall outperformed the precision class result. A 13% increase to 81% shows how many little false negatives it had. Surprised also scored highly and means it's rarely predicted in the wrong class at 94%. Fear shows an incredibly low 35% and this means it's been more than likely classified as other expressions.

The outcome could be improved by increasing the images within the data sets and for fear I would remove the more subtle looking fear expressions, this will make the prediction model more accurate during training.

---

## **4. Conclusion**

Results from the program initially were low as I only used the JAFFE (Michael J. Lyons, 1998) Database and soon discovered with more dataset images the model would improve. I added the MUG facial expression Database (Delopoulos, 2010) which improved the micro accuracy's. However, the results would vary from test to test, after clearing out some poor images from the datasets the micro accuracy was roughly 55%. After adding the DLlib (DlibDotNet, n.d.) cartoon faces the results were worse and the discovery was made that the faces are hardly

---

---

picked up by the AI. Passing through football images (Maradonna, 2021) helped the joy scores but sadly the images are mostly the same throughout emotion wise, however if ethnicity and hair would become a key feature for the AI that dataset would be invaluable. The result I have reported are based on both MUG (Delopoulos, 2010) and a sorted through set of Cohn-Kanade (Kanade, 2000) .

Overall, this program was a success to make a facial recognition program using machine learning and shows with simple steps a good outcome can be achieved with moderately good results. This version of the program isn't brilliant or perfect and would need a much larger dataset to output the outcomes of the researchers (Neves, 2019) however it is a solid foundation to go further.

---

## 5. References

---

### References

- Aifanti, N. P. (2010, April 21). *Multimedia Understanding Group*. Retrieved from The MUG Facial Expression Database: <https://mug.ee.auth.gr/fed/>
- Comprehensive database for facial expression analysis. (2002, August 06). *Comprehensive database for facial expression analysis*. Retrieved from IEEE Xplore: <https://ieeexplore.ieee.org/document/840611>
- Delopoulos, n. (2010, May 11). *The MUG facial expression database*. Retrieved from Research Gate: [https://www.researchgate.net/publication/224187946\\_The\\_MUG\\_facial\\_expression\\_database](https://www.researchgate.net/publication/224187946_The_MUG_facial_expression_database)
- DlibDotNet. (n.d.). *DlibDotNet*. Retrieved from Nuget.org: <https://www.nuget.org/packages/DlibDotNet/>
- Grail.cs.washington.edu. (n.d.). *Expressive Character Animation*. Retrieved from Expressive Character Animation: <http://grail.cs.washington.edu/projects/deepexpr/>
- Kanade, T. C. (2000, March). *Cohn-Kanade AU-Coded Facial Expression Database*. Retrieved from Fourth IEEE International Conference on Automatic Face and Gesture Recognition: <http://www.pitt.edu/~emotion/ck-spread.htm>
- Maradonna. (2021, Jan 11). *NewGAN Manager v1.2.0*. Retrieved from FMSCOUT: <https://www.fmscout.com/a-newgan-manager-tool.html>
- Michael J. Lyons, S. A. (1998, May). *JAFFE Coding Facial Expressions with Gabor Wavelets*. Retrieved from Research Gate:

[https://www.researchgate.net/publication/3745235\\_Coding\\_Facial\\_Expressions\\_with\\_Gabor\\_Wavelets](https://www.researchgate.net/publication/3745235_Coding_Facial_Expressions_with_Gabor_Wavelets)

Microsoft. (2019, April). *Visual Studio 2019*. Retrieved from Visual Studio 2019:  
<https://visualstudio.microsoft.com/vs/>

Microsoft. (n.d.). *ML.Net*. Retrieved from An open source and cross-platform machine learning framework:  
<https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet>

Nagendhiran, N. (2019, January). *Security and safety with facial recognition feature for next generation automobiles*. Retrieved from Research Gate:  
[https://www.researchgate.net/publication/330846841\\_Security\\_and\\_safety\\_with\\_facial\\_recognition\\_feature\\_for\\_next\\_generation\\_automobiles](https://www.researchgate.net/publication/330846841_Security_and_safety_with_facial_recognition_feature_for_next_generation_automobiles)

Neves, D. C. (2019, October 01). *Facial Expression Recognition Using Computer Vision: A Systematic Review*. Retrieved from MDPI: <https://www.mdpi.com/2076-3417/9/21/4678/htm>

Singh, A. (2018, April 27). *Facial Emotion Detection using AI: Use-Cases*. Retrieved from ParallelDots:  
<https://blog.paralleldots.com/product/facial-emotion-detection-using-ai/?0>