### DS 4400: Machine Learning and Data Mining I

Spring 2024

Project Title: How Are You Feeling?: A Facial Image Classification Task

Team Members: Eden Rozenblit, Mackinley Morgan, Ivy Hsiao

# **Problem Description**

We wanted to develop a supervised learning model to detect and categorize emotions in images. This involved analyzing images of faces displaying varied expressions to determine the emotional state of individuals at the time of the photo. Since we are trying to predict what category each photo falls into, our problem regards classification, and not regression.

We believe that such a classifier may be useful for understanding whether there are patterns in how humans convey emotions with facial expressions. This may be relevant for collecting emotional feedback from a person or group and having confidence that responses are unbiased.

Studying facial processing through computational models may also give us insight into the neural mechanisms behind difficulties with emotion processing as seen in neurodevelopmental disorders such as autism spectrum disorder and ADHD.

# **Dataset**

The dataset used was [The Facial Expression Recognition 2013 (FER-2013) Dataset](https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data), prepared by Pierre-Luc Carrier and Aaron Courville. This dataset was created for research purposes and was uploaded to Kaggle as part of a competition.

The dataset contains 35,685 records of 48x48 pixel grayscale images of faces. The images have already been centered and cropped. Additionally, the data has already been split into training and testing sets, represented by the “Usage” column value. There are 28709 instances of training data, 3589 instances of PrivateTest data, and 3589 instances of PublicTest data. We split the data so that the PrivateTest data and PublicTest data were combined into a general test set. This means that there are 28709 instances of training data and 7178 instances of test data in our implementation.

Otherwise, the data was represented with one “pixels” column (feature variable/s) and one “emotions” column (target variable). The "pixels" column represents the image’s pixels as a string in row major order, with each pixel value separated by a space. We converted this string into a vector with 2304 (48 x 48) numerical values. This meant that each data point had 2304 individual features, with each feature being a pixel value. Values were then converted to floats and normalized (divided by 255) so that each value lies between 0 and 1.

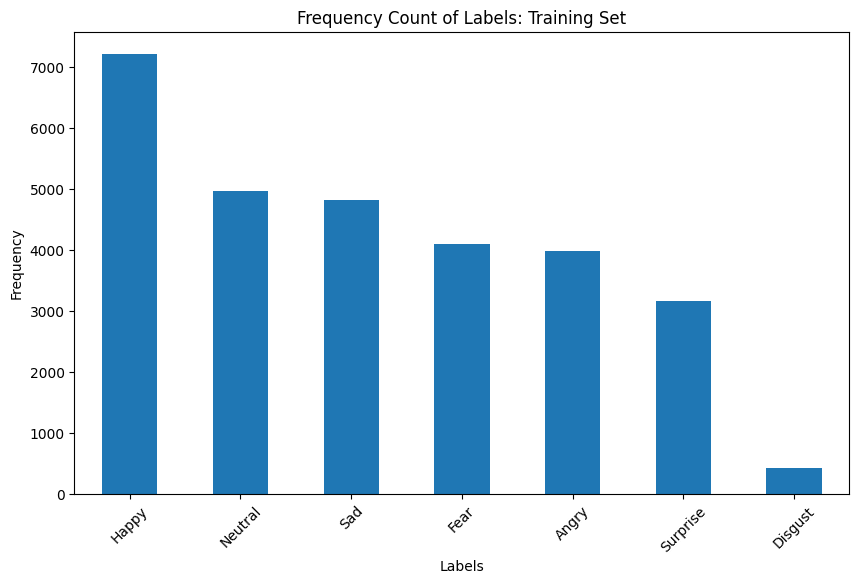
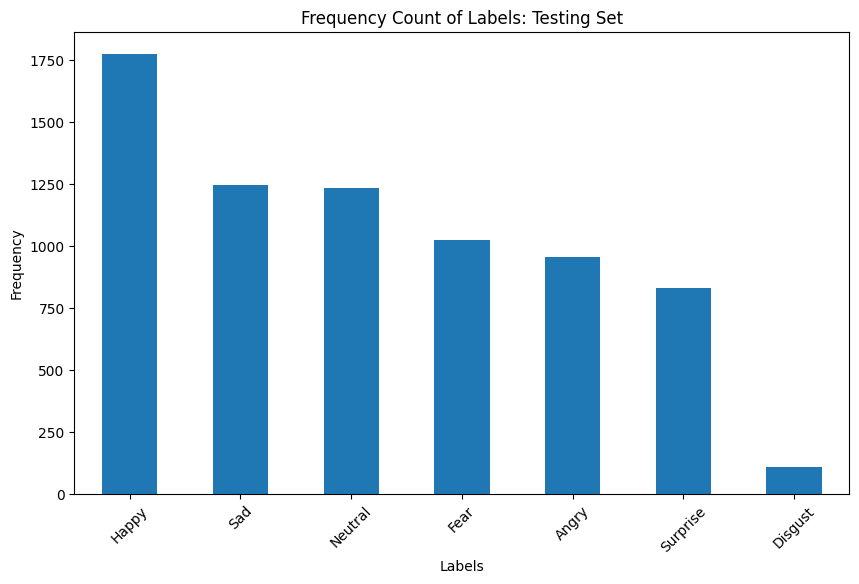
The target variable, “emotion”, represents the facial expression being conveyed in the image (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). Labels were encoded to be represented as binary categories.

# **Exploratory Data Analysis**

To understand the data further, we visualized a random selection of 25 samples. Looking at the images produced, you can see that the dataset contains a variety of images. Faces are depicted in people of diverse ages in many different contexts and spatial orientations. In our opinion, many of the images are hard to classify. Some of the expressions in the photos seem ambiguous and up to individual interpretation.



Next, we displayed the frequency counts for each label. For both the testing set and the training set, Happy images were the most representative label type. Neutral, Sad, Fear, Angry and Surprise were all represented somewhat equally. Finally, images labeled as Disgust were underrepresented. Because we wanted to see how our models performed for each of the 7 standard emotions, we opted to leave the dataset as it was.

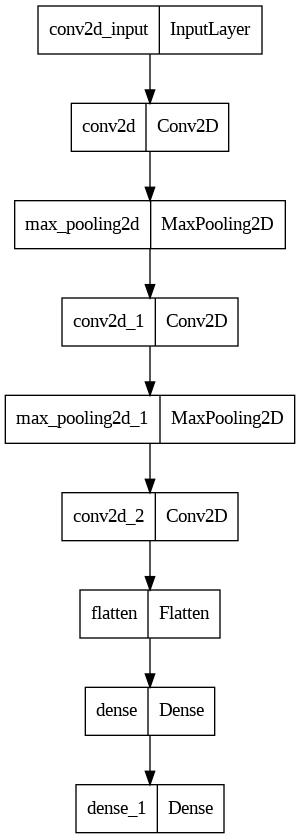
**

# **Approach and Methodology**

**K-Nearest Neighbors**

We chose to train K-Nearest Neighbors models to evaluate our data because it’s a simple model that is reliable for classification tasks. Cross-Validation was performed to tune the number of neighbors. We tested k values between 1 and 10. Cross validation scores scored very poorly for all models of the n\_neighbors values tested, but a value of 3 was ultimately selected as the hyper parameter. The KNN model was the quickest in terms of training time.

**Logistic Regression**

Another model we used was logistic regression, fit for multiclass classification. The argument ‘multinomial’ passed to the model specifies that the logistic regression model was multiclass and utilized softmax activation. We chose to use multinomial logistic regression with softmax because it is primed for classification tasks. Cross Validation was used to tune hyperparameter max\_iter, selecting the value 1000 out of a list of candidates [50, 500, 1000, 10000]. The logistic regression took a bit longer than the KNN to train.

**Feed Forward Neural Network**

We chose to use a Feed Forward Neural Network because it is a complex model, so it would serve as the foil to our KNN on the basis of complexity. We thought it would be interesting to run models on different ends of the spectrum of complexity to see how performance varied. The FFNN included three dense layers. The first two layers had 128 nodes, and used a relu activation function. The final output layer had 7 output nodes and used a softmax activation function because of the multiclass labels. In total, the model had 312455 parameters. The model was fit with 12 epochs, a batch size of 64, and a learning rate of 0.0001. Cross entropy loss and accuracy were used to evaluate progress.

**Convolutional Neural Network**

Lastly, we chose to use a CNN because they are particularly well suited for image classification. This is because CNNs learn hierarchical representations of features, in this case our pixel values. This was the model that we hoped would be the most successful. The CNN included alternating convolutional and max pooling layers, leading to a total of three convolutional layers and two max pooling layers. These were followed by two dense layers, and a final output layer. This was modeled after the LeNet model shown in class, with a few alterations. Hyperparameter tuning was mostly done through trial and error, as well as comparing with source code from other models across the internet. Ultimately, each convolutional layer had a kernel size of 3 and a stride of 1, with a relu activation function. Filters increased from 32 for the first convolutional layer to 62 for the next two. Each pooling layer had a size of 2x2 and a stride of 2. The final output layer used a softmax activation function because we are doing multiclass classification. In total, the model had 318407 parameters. The CNN was fit with 12 epochs, a batch size of 64, and a learning rate of 0.0001. Cross entropy loss and accuracy were used to evaluate progress. Each epoch took around 70 seconds to train, leading to quite a high training time.

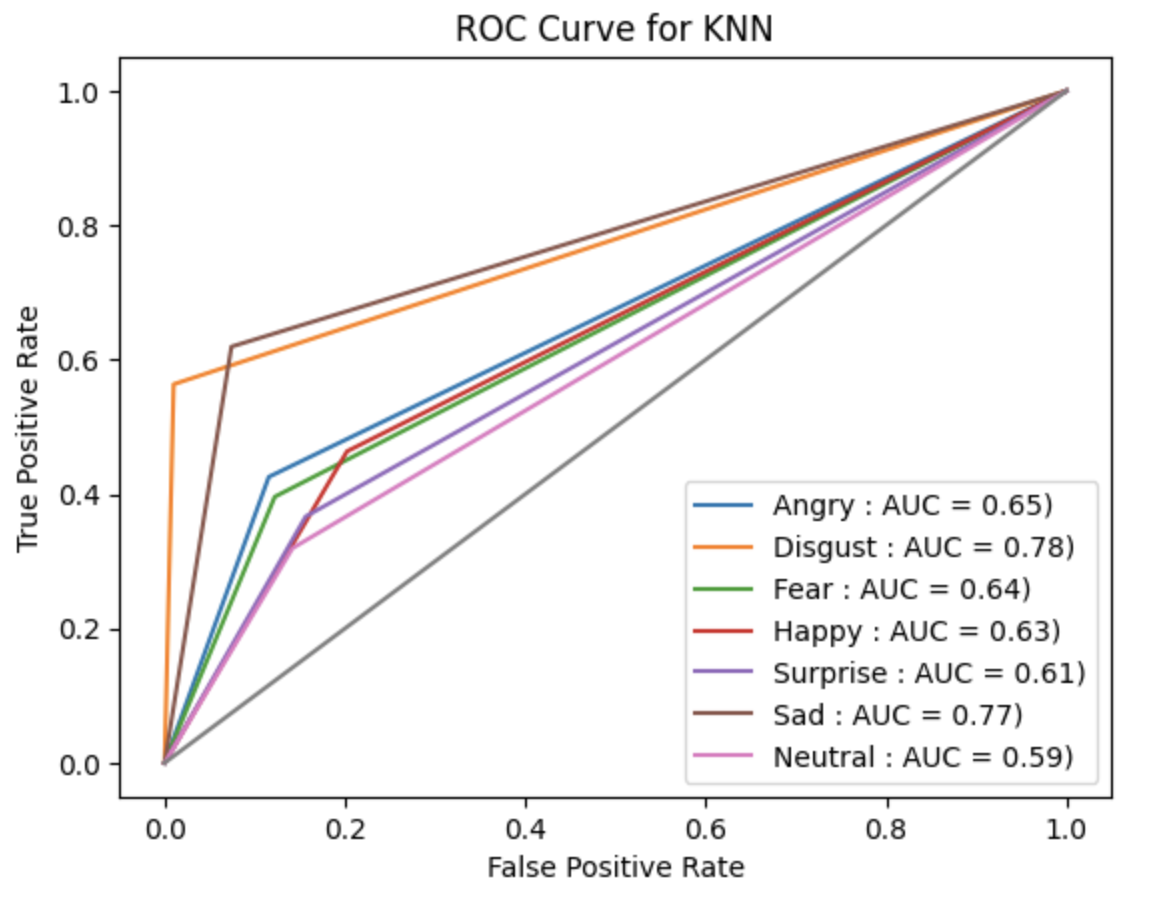
# **Discussion and Result Interpretation**

**Individual Model Metrics**

To evaluate our KNN, Logistic Regression, FFNN, and CNN models, we employed a variety of metrics on each model based on its specific characteristics:

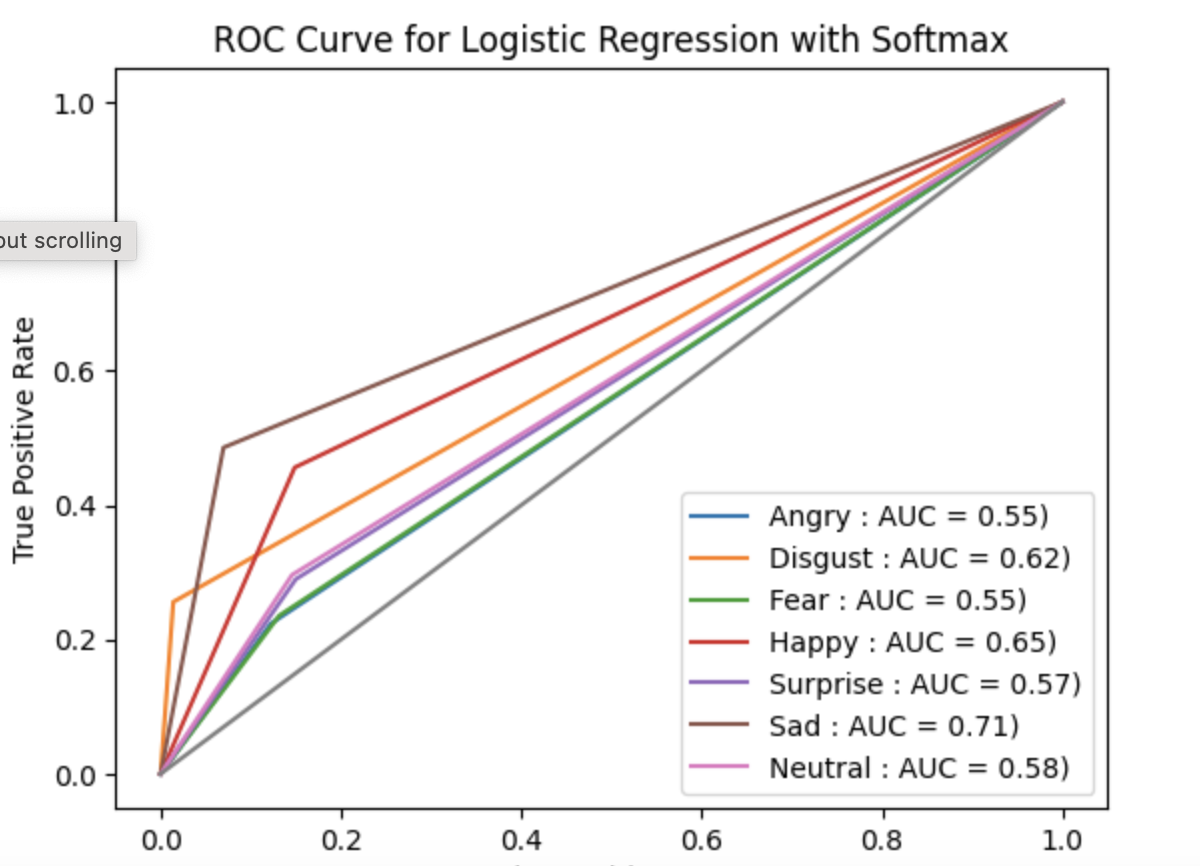
* **KNN & Logistic Regression**: For these models, we focused on metrics like Precision, Recall, and F1 Score, as well as ROC curve and AUC. These metrics help in understanding the trade-offs between correctly predicting positives and the overall rate of correct predictions across classes.
* **FFNN & CNN**: Given their complexity, we opted to use just two metrics to evaluate for the neural network models. CE loss and accuracy scores were used during training to monitor convergence and model fit. Additionally, we used a confusion matrix to visualize the overall accuracy of our models, particularly between each class label.

**Comparative Metric: Accuracy**

While we used different metrics to evaluate each model during the development stages, we used accuracy as the primary criterion for comparing these models. Accuracy provides a straightforward and universally understood measure of overall effectiveness for each model.

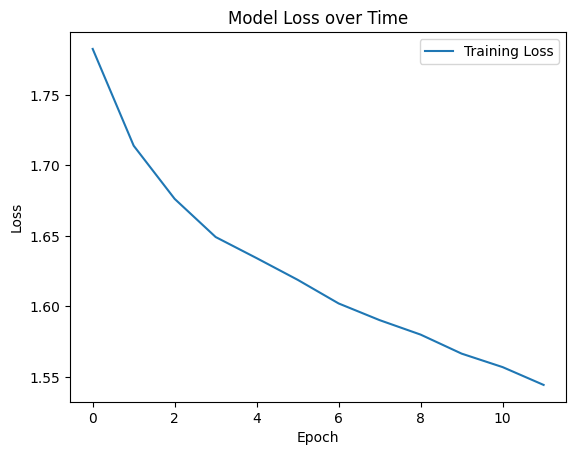
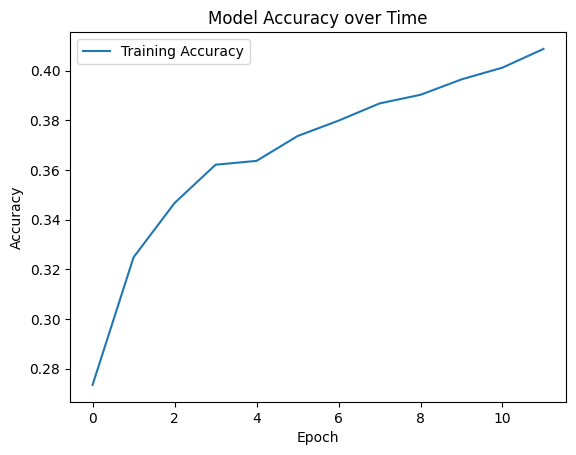
**K-Nearest Neighbors (KNN)**

The ROC Curve analysis for the KNN classifier indicates a nuanced performance across different emotions, with the best discrimination for 'Disgust' and 'Sad', while 'Neutral' is challenging to classify for the model. However, the overall accuracy is at 0.2712, suggesting that while it may distinguish certain emotions, it generally performs poorly in correctly identifying the true class labels across the entire dataset.

**Logistic Regression**

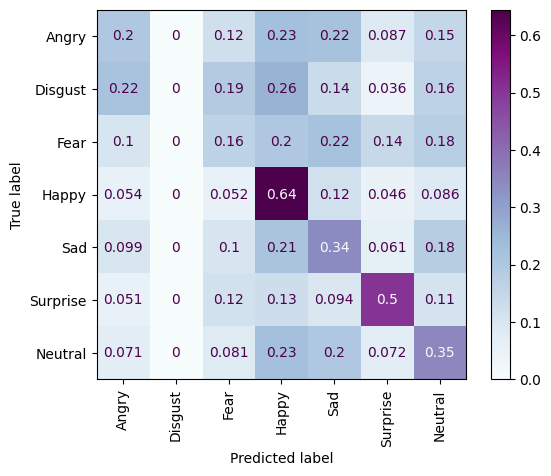
The ROC curve for Logistic Regression with a Softmax shows that the 'Sad' category shows the highest AUC at 0.71, suggesting the model is relatively good at identifying 'Sad' expressions. 'Disgust' and 'Happy' emotions also show fair performance with AUCs of 0.62 and 0.65, respectively. However, the 'Angry' and 'Fear' categories at 0.55 only have AUCs of 0.55, and the 'Neutral' emotion's AUC at 0.58 and 'Surprise' at 0.57 suggest limited ability in distinguishing these emotions. The overall accuracy of the model is approximately 0.3566, and the average cross-validation (CV) score is around 0.3451. These figures indicate that Logistic Regression is moderately effective at classifying certain emotions, its performance varies across categories.

**Feed Forward Neural Network (FFNN)**

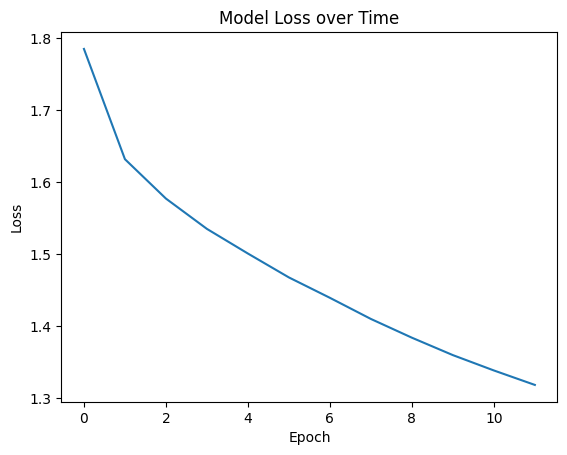
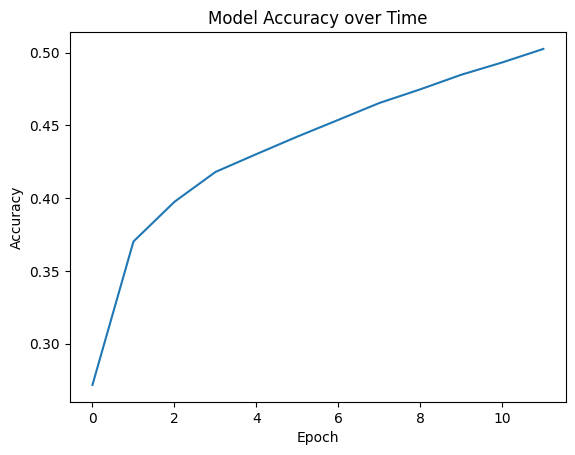


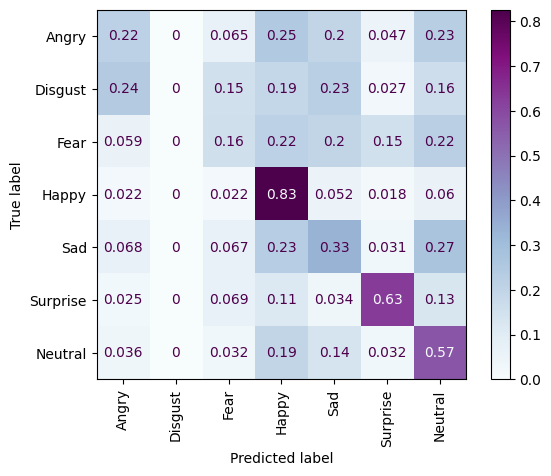
The training log for the Feed-Forward Neural Network (FFNN) over 12 epochs indicates a positive learning trend, with accuracy improving from 0.2734 to 0.4087 and loss decreasing from 1.7826 to 1.5443. This progression suggests that the FFNN's weights were adjusted to enhance its accuracy over time.

The final training and test results for the FFNN show that the model achieved a training accuracy of 0.419 and a test accuracy of 0.386. This indicates that the model was able to learn from the training data and generalize to some extent to new, unseen data.

The confusion matrix shows that the model is proficient in recognizing 'Happy' expressions, correctly classifying them 64% of the time, but it struggles to distinguish 'Sad', often confusing it with ‘Happy’ and ‘Neutral’. 'Neutral' and 'Surprise' emotions are recognized with moderate accuracy, 35% and 50% respectively. However, 'Neutral' is frequently misclassified as 'Happy' or 'Sad'. The model has significant difficulty with 'Fear', which is more often confused with other labels. 'Angry' and 'Disgust' emotions are poorly identified, along with a complete misrecognition of 'Disgust'. This is likely because of the relative low number of ‘Disgust’ instances in the training set.

**Convolutional Neural Network (CNN)**



The training log for CNN over 12 epochs indicates a positive learning trend, with accuracy improving from 0.2716 to 0.5026 and loss decreasing from 1.7846 to 1.3179. The final training and test results for the CNN show that the model achieved a training accuracy of 0.503 and a test accuracy of 0.484.

From the confusion matrix for CNN, we can see that the model is highly effective at identifying 'Happy' expressions, with a high accuracy of 83%. 'Surprise' and 'Neutral' emotions are also well-recognized with accuracies of 63% and 57% respectively. However, 'Angry' and 'Disgust' are still challenging for the model, although there is a slight improvement in 'Angry' recognition compared to 'Disgust'. Misclassifications among 'Fear', 'Sad', and 'Neutral' suggest the model still struggles with distinctions between these expressions.

**Performance Comparison Between Models**

The models showed varied performance levels. The CNN had the highest test accuracy of all the models at 0.484, indicating a stronger overall ability to learn and generalize from the data. The FFNN followed with a testing accuracy of 0.386. Logistic Regression and KNN had lower overall accuracies, with KNN at 0.27 and Logistic Regression at 0.36. There were clear patterns indicating that the more complex models, particularly the CNN, performed better at classifying emotions.

**Analysis of Performance**

In both the training and testing sets, some emotions, such as 'Happy', are heavily represented with high frequencies, while others, like 'Disgust', have relatively few instances. This disparity means that the models would have had much more data to learn from for certain emotions compared to others, which may have led to a bias towards the more frequently occurring labels.

In the confusion matrices for both FFNN and CNN, we observed a high accuracy in predicting 'Happy' expressions. This aligns with the class distribution in the dataset, where 'Happy' examples are abundant. Due to this, the models may be overfitting to 'Happy' expressions, leading to a high true positive rate for this emotion. 'Disgust' had the least instances in the dataset, and it was the least recognized emotion in the confusion matrices. This indicates that our models had insufficient examples of 'Disgust' to learn from, resulting in a lack of generalization for this class.

For the emotions with a moderate amount of examples like 'Sad', 'Surprise', and 'Neutral', the models' performance was varied. In some cases, such as 'Sad' for the FFNN and 'Surprise' for the CNN, the models performed relatively well. This may be because these models captured features that are well-represented within the data for these emotions. However, even these classes faced some misclassification into ‘Happy’, likely because the model was biased toward ‘Happy’.

Interestingly, the KNN model performed best on the ‘Disgust’ emotion, where the neural networks were unable to identify this emotion. This shows that the neural networks may have been biased towards the more commonly seen emotions, while the KNN model could account for labels with fewer observances. This may be because KNN creates a decision boundary based on few closely related instances, allowing for a more individual consideration of labels.

**Future Work**

Looking at our results, we believe there are a number of ways to improve model performance. For one, instead of the current train-test split, we may decide to switch to a train-validation-test split to allow for more adaptations on our models before using the test set.

Addressing the class imbalance in our dataset will be important, especially since the overrepresentation of certain emotions like 'Happy' and the underrepresentation of others such as 'Disgust' may have biased our model's performance. Techniques such as resampling could help provide a more balanced training approach. Also, given the model’s poor performance on the 'Disgust' labeled images, we are considering whether to remove these instances or to employ boosting methods like AdaBoost to improve their classification. Finally, although our convolutional neural network performed the best, we believe it can be improved upon. We would like to continue testing out different model architectures and hyperparameters to see which ones work well for our data. Unfortunately, we were somewhat limited by computational and timing restrictions, so we would like to spend more time on training and tuning the model in the future.

# **Conclusion**

Our project aimed to develop a supervised learning model to detect and categorize emotions in facial images. We trained models on the Facial Expression Recognition 2013 (FER-2013) Dataset using four different models: K-Nearest Neighbors (KNN), Logistic Regression, Feedforward Neural Network (FFNN), and Convolutional Neural Network (CNN). Ultimately, the CNN was the highest performing model, though test accuracy was still low at 0.48. We believe that many steps can be taken in the future to improve our model’s performance to a certain degree, although much of the error may be a result of the ambiguous nature of the classification task as well as the quality of the data.

**Team member contribution**

We called over spring break to decide on our proposal and how to go forward. We decided to have Eden do the EDA, then split responsibilities on the models. Mack trained the K-Nearest Numbers and Logistic Regression models, Eden trained the Convolutional Neural Network and Ivy trained the Feedforward Neural Network. We made sure to contact each other about model performance and bugs. All visualizations created were also done by the member that the corresponding model was assigned to. For our project video, we recorded together on zoom. Throughout the process we met to discuss progress and help where needed.

# **References**

<https://www.kaggle.com/code/ahmedmahmoud16/facial-expression-recognition-with-logistic>

<https://github.com/aamini/introtodeeplearning/blob/master/lab2/Part1_MNIST.ipynb>

<https://www.kaggle.com/code/drcapa/facial-expression-eda-cnn>

<https://www.kaggle.com/code/mohammed94/facial-emotion-detection-cnn>

**Code** [DSproject.ipynb](https://colab.research.google.com/drive/12qNOwFJyQj8iIRwNgK7gx_znIwayJwNC?usp=sharing)

**Video** [Recording.mov](https://drive.google.com/file/d/1ub9t6JkO03uFRb96lxo6JiPQnVY10xg0/view?usp=drive_link)