Math 156 Project Report

Aayushi Choudhary, Jared Patel, Jessie Zhou

Github repo: https://github.com/jessiejzhou/math_156_final_project
Extra Credit: https://www.youtube.com/watch?v=wBkObOrwdQM

Abstract

The purpose of this project is to develop a computational framework for detecting and interpreting emotional nuances in human facial expressions. Using a convolution neural network (CNN), the model will analyze various facial expressions to classify emotions into 7 categories: happy, sad, anger, surprise, disgust, fear, and neutral. The dataset contains 35,685 labeled images of diverse faces varying in age, gender, and ethnicity. The project leverages CNN's ability to capture spatial patterns and the model is optimized through hyper parameter tuning such as batch size, learning rate, etc. to achieve a robust model. The model has the potential to be applied in many areas such as virtual therapy, personalized learning environments, and security systems to enhance emotional understanding in different settings.

1 Introduction

Human emotions are deeply complex and are expressed through different methods such as voice, text, and facial expressions. They play a vital role in understanding and connecting with the people around us, offering deep insights. Emotions have the power to not only enhance communication but also reveal critical details in social and psychological contexts, making the detection and interpretation of emotions an important area of study.

However, the ability to discern emotions through facial expressions are not without its challenges. Facial expressions are influenced by numerous factors such as individual differences, cultural nuances, and environmental conditions. Such complexities make emotion detection a demanding task for computational models.

The goal of this project is to be able to address these challenges by developing a convolutional neural network (CNN) to detect and classify emotions based on facial expressions. CNN's are good at extracting spatial hierarchies and identifying patterns in images which enables them to recognize subtle features associated with facial emotions. To ensure model robustness, we will implement hyperparameter tuning such as adjustments to batch size, learning rate, epochs, etc.

Unfortunately, challenges remain, including imbalanced datasets, high computational demand, and difficulty models across diverse populations and cultural contexts. Models also frequently struggle with capturing the nuances of human emotions in real world settings with noise, light variations, and occlusions. We will leverage CNN's pattern recognition capabilities to create a model with potential to be applied in practical settings while remaining mindful of its potential limitation and ethical considerations

2 Background

Neural networks are a subset of machine learning and comprise of node layers, containing an input layer, one or more hidden layers, and an output layer. Each node connects to another with its own associated weights and thresholds. Data is only sent to the next layer of network if the output of a single node is above a specified threshold.

Convolutional Neural Networks are primarily used for classification and image processing. It leverages principles from linear algebra (matrix multiplication, to identify patterns within images. It takes an image in as input, assigns learnable weights and biases to aspects/objects in the image, and differentiates the images from one another. CNNs reduce images into forms that are easier to process without losing critical features. On the downside, they are computationally demanding and require GPUs to train the models.

CNNs have 3 main types of layers: Convolutional layer, Pooling layer, and Fully-connected (FC) layer.
1. Convolution Layer: for feature extraction. It does this by applying a convolution function and then an activation function on the output of that. A kernel function (a.k.a filter) is used to extract features. A small matrix of numbers (called kernel or filter) is passed over the image and transformed based on the values from the filter. The input image is denoted by f and the kernel by h, with m, n indexing the position of the output after convolution and j,k indexing the elements of the kernel:

$$G_{m,n} = (f * h)_{[m,n]} = \sum_j \sum_k h_{[j,k]} * f_{[m-j,n-k]}$$

After placing the filter over a selected pixel, each value from the kernel is multiplied in pairs with the corresponding values from the image. Everything is then summed up and the results are put in the right place on our output feature map.

Sometimes images are padded with an additional border to solve the problem of the image shrinking. If so, then p = (f-1)/2

Forward Propagation: we calculate an intermediate value Z which is obtained from the convolution of the input data from previous layer with W tensor (contains filters), and then we add bias b. Next, we apply a non-linear activation function g to Z.

$$Z^{[l]} = W^{[l]} * A^{[l-1]} + b^{[l]} \tag{1}$$

$$A^{[l]} = g^{[l]}(Z^{[l]}) \tag{2}$$

Backward Propagation: We calculate the intermediate value dZ[l] by applying a derivative of our activation function to the input tensor

$$dZ^{[l]} = dA^{[l]}(g'(Z^{[l]})) \tag{3}$$

Now for the backward propagation of the convolution itself, a matrix operation called full convolution is used. The filter is W, dZm,n is a scalar of the partial derivative above.

$$dA+ = \sum_{m=0}^{n_h} \sum_{n=0}^{n_w} W * dZ_{[m,n]} \tag{4}$$

2. Pooling Layer: (a.k.a downsampling) conducts dimensionality reduction by reducing number of parameters in the input to speed up calculations. The kernel applies an aggregation function to values within the receptive field. The 2 main types of pooling are: Max pooling and Average pooling. In max pooling (which we are using in our model), the filter moves across the image, selecting pixels with the max values to send to output array. Pooling Backpropagation: We do not need to update any parameters here, just distribute gradients. A mask is created that remembers the position of the values used in the first phase, which we later utilize to transfer gradients so that it does not affect elements in the forward pass.

3. Fully-connected layer: each node in the output layer is directly connected to a node in the previous layer. Here, the task of classification based on features extracted through the previous layers and their filters is performed. In the FC layer we leveraged a soft max activation function to classify the inputs, producing probability from 0 to 1.

3 Dataset

The dataset we are using is available on Kaggle. There are 35,685, 48x48 pixel images that are already divided into training and test sets using an 80-20 split. The split is also proportional so that 20% of each emotion is included in the test set. It includes faces varying in age, gender, and ethnicity to encompass the natural diversity we have in the real world.

As shown in Table 1, the proportion of images for each emotion is uneven. For example, the class happy has 7,215 images while disgusted has only 436 images.

Table 1: Value Counts for each Emotion

Label	Count
Happy	7215
Neutral	4965
Sad	4830
Fearful	4097
Angry	3995
Surprised	3171
Disgusted	436

A large difference in the data counts can lead to bias in the model. The model can learn more about dominant classes at the expense of underrepresented classes. With less data, the model is not able to learn as many patterns for underrepresented classes. Because of this, we decided to remove disgusted from the data set. The rest of the emotions in the model have sufficient sample size so we can preserve them. To decrease bias in the model that may be caused by uneven class sizes, we will implement class weights. An alternative to implementing class weights is to use identical counts of each class, however this will decrease the amount of data available to train our model, so we opted against it.

To clean our data, we iterated through the images first removing unreadable or missing images. Next, we converted all the images to grayscale to decrease the complexity of the model and prevent color from adding noise to the model. Finally, we scaled the images to 48x48 pixels.

4 Model

We decided to build our CNN using Tensorflow's Sequential class, as it allows for a simple feed-forward stack of layers. Being that our model only has one input, this is a natural choice. Since we are limited by computing resources, we decided to make a simple architecture consisting of the input, 2 convolutional blocks, a fully connected layer, and the output. Each block has a 2D convolutional layer, MaxPooling, and then applies Dropout to reduce overfitting.

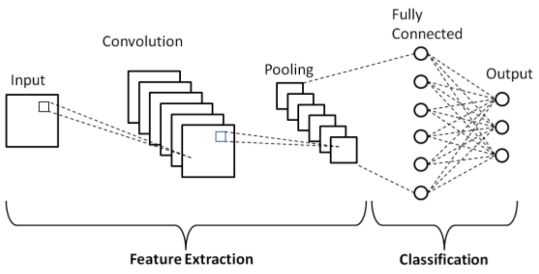

Figure 1: CNN Architecture

To determine the optimal hyper-parameters, we performed k-fold cross validation and used the best performing values to fit further models. Once again as we are limited by computing resources, we could only fit a certain number of hyperparameters with a limited number of epochs (maximum 100). The hyperparameters we decided on are the number of filters, kernel size, number of dense neurons, and dropout rate as they are crucial choices in defining the training process of our CNN. Following the cross-validation, we experimented with L2 regularization to try and reduce overfitting. We also included batch normalization in our model to mitigate the risk of extreme weight updates and to improve generalization.

From the cross-validation, we determined that higher values for all hyperparameters except number f dense neurons is the most optimal. Less dense neurons protects from overfit. The kernel size of (5,5) had the highest accuracy as it is more global than a smaller kernel size. However, we went with (3,3) as it performed comparably and takes less computational power. It is also the standard choice as it balances efficiency and feature extraction capability. A larger dropout rate helps with prevent overfitting.

As this is a multiclass classification task, categorical cross-entropy loss was the best choice for the loss function, and softmax the best for output activation function. In order to use these, one hot-encoding is necessary. The Adam optimizer was also ideal as it utilizes adaptive learning rates for each parameter. Accuracy is used as our main evaluation metric.

5 Results

After choosing the best model from our cross validation, we fit a final model with added regularization. Our final model gave us a test accuracy of 44.40%. Taking a look at training loss vs validation loss, we notice that the graphs follow the same pattern and have reasonably close gaps. The same can be said about the training accuracy vs. validation accuracy. Thus, we were able to prevent overfitting to a certain degree.

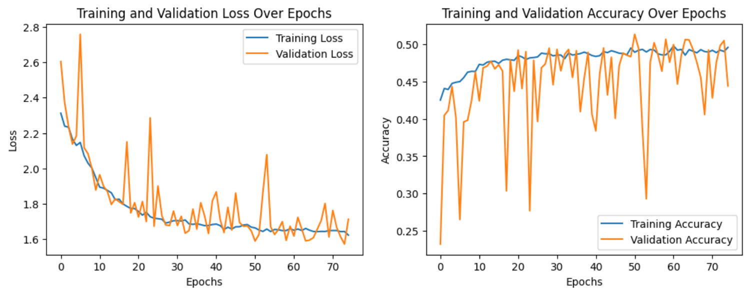

Figure 2: Loss and Accuracy Graphs

However, looking at the values our model accuracy is low with a peak around 0.5, and we see from our confusion matrix the high number of misclassifications.

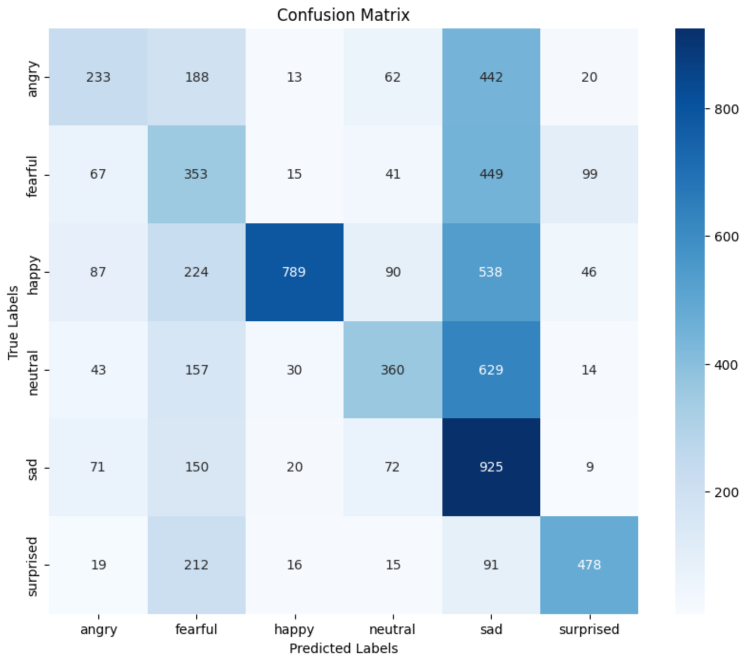

Figure 3: Confusion Matrix

In addition, our loss values are all above a value of 1.0. Thus, there are many areas of potential improvement: further cross-validation and data preprocessing. With a second cross validation, we could test different training parameters like number of epochs, batch size, and number of folds. Another potential improvement we can make to the model is to transform the images. Rotating, flipping, and cropping images in the dataset can change pixel values while still maintaining the information of the image. This additional level of information can help the model learn patterns better in the images. Furthermore, introducing more applicable data to the model can improve its efficacy. The introduction of new data is another way a model can learn more patterns from the images and become more generalizable. While introducing more data to the model will make it more computationally expensive, it would help improve the accuracy of our model.

6 Conclusion

The aim of this project was to develop a convolutional neural network (CNN) for emotion recognition from facial expressions. The model was trained to classify emotion into 7 categories: happy, sad, anger, surprise, disgust, fear, and neutral. However, we ended up dropping disgust as an emotion due to a lack of data. We implemented hyperparameter tuning, data preprocessing, class weights, etc. to improve model performance. Cross-validation and regularization techniques such as dropout and batch normalization helped optimize our model and reduce overfitting. Despite challenges like dataset bias and computational limitations, we were able to achieve a test accuracy score of 44.40%. Additional steps to improve this model would be to increase dataset diversity, obtain more computational power, and to explore deeper CNNs. Nevertheless, the model has potential application in many fields like virtual therapy, personalized learning, security systems, advertisements, etc.

7 Author Contributions

Aayushi: Worked on model building, hyperparameters, data processing. Wrote ’abstract’, ’introduction’, and ’background’ sections.

Jared: Worked on model building, cross-validating, and hyperparameters, data processing. Wrote ’dataset’ section.

Jessie: Worked on model building, cross-validation, and model results. Wrote the ”Model” and ”Results” sections.

8 Important Notes:

With more time and computational power we could have tried difference image transformations (rotating, flipping, cropping), incorporating more relevant data, and further cross-validation and hyper parameter testing.

References

Agrawal, A., Mittal, N. Using CNN for facial expression recognition: a study of the effects of kernel size and number of filters on accuracy. Vis Comput 36, 405–412 (2020). https://doi.org/10.1007/s00371-019-01630-9

Ibm. “What Are Convolutional Neural Networks?” IBM, 17 Oct. 2024, www.ibm.com/topics/convolutional-neural-networks.

Moore, Susan. ”13 Surprising Uses for Emotion AI Technology.” 13 Surprising Uses for Emotion AI Technology, 18 Sept. 2018, www.gartner.com/smarterwithgartner/13-suprising-uses-for-emotion-ai-technology.

Nur Alia Syahirah Badrulhisham and Nur Nabilah Abu Mangshor 2021 J. Phys.: Conf. Ser. 1962 012040

Skalski, Piotr. “Gentle Dive into Math behind Convolutional Neural Networks.” Medium, Towards Data Science, 14 Apr. 2019, towardsdatascience.com/gentle-dive-into-math-behind-convolutional-neural-networks-79a07dd44cf9.

Tariq, Farina. “Breaking down the Mathematics behind CNN Models: A Comprehensive Guide.” Medium, Medium, 2 May 2023, medium.com/@beingfarina/breaking-down-the-mathematics-behind-cnn-models-a-comprehensive-guide-1853aa6b011e.