Experimental CNN-RNN Approaches for Emotion Prediction via Facial Features

Maggie Zhang, Josh Talla, Brandon Sun

1. Introduction

Live emotion detection holds great importance in today's world. When employing machine learning methods for real-time detection, accurate emotion classification plays an integral role in mitigating situations relating to surveillance and security.

2. Objective

Using various neural network models and activation functions, we built our final model to identify emotions based on extracted facial features. The Facial Expressions Vision Dataset, which contains 13,000 images of facial expressions across eight emotion labels, serves as the basis for our training and validation. Our complete supervised learning model consists of a convolutional neural network (CNN) for pattern recognition and classification. When considering models, we additionally experimented with Long Short Term Memory (LSTM) units within recurrent neural networks (RNN). In the following sections, we provide further support and explanations for our model's development and final performance.

3. Analysis and Processing of Dataset

The initial unprocessed dataset contains raw images represented as colored pixel matrices. To eliminate potential issues from variations in image size and color, we preprocessed the data by removing images with inconsistent dimensions and converting all resulting images to grayscale (FIG. 3.1). The processed images and their corresponding labels were then stored in an array for further analysis.

FIG. 3.1: Processed images are of the same size (50x50). Grayscale is applied onto each pixel.

The resulting image dataset was highly imbalanced, with each emotion label represented by between 12 and 6,000 images (FIG. 3.2). This imbalance can potentially affect the model's performance by overfitting on data from sparser labels. To avoid that possibility, we experimented with training and cross-validation results from two modified dataset distributions.

In the first approach (FIG. 3.3), we standardized the original dataset to reach 1,500 images across all labels. This involved randomly duplicating images for underrepresented labels and randomly reducing images for overrepresented labels. While not pictured, we also experimented with duplicating data within each label to exactly match the size of the largest label ('NEUTRAL', 6,717 images).

In the second approach (FIG. 3.4), we balanced the original dataset by randomly duplicating images in labels with less than 5,000 original images until each underrepresented label reached a range of 5,300 to 6,700 images. We tuned this range based on validation results per experimental dataset. This approach introduces slight variations in the modified dataset, which is a more realistic representation of real-world distributions.

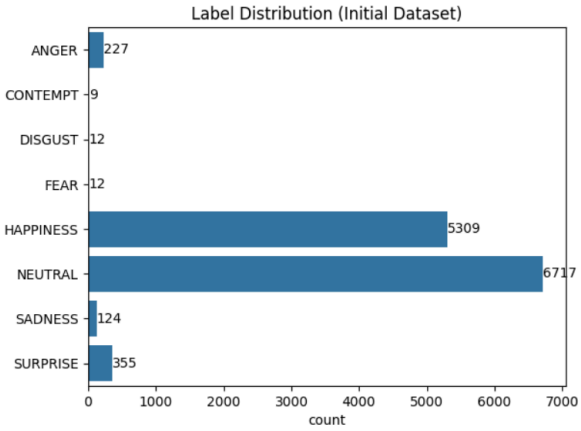

FIG. 3.2: Processed image dataset before distribution modification.

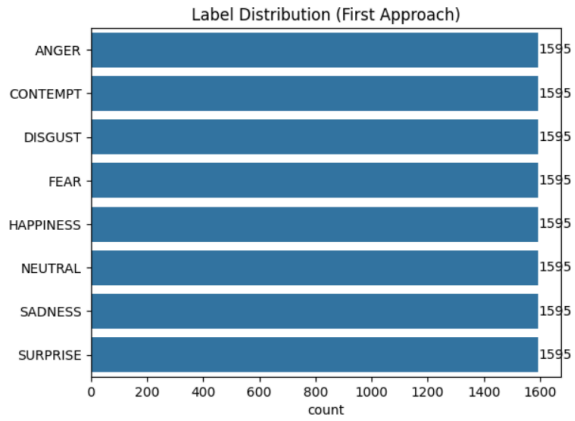

FIG. 3.3: First Distribution Approach.

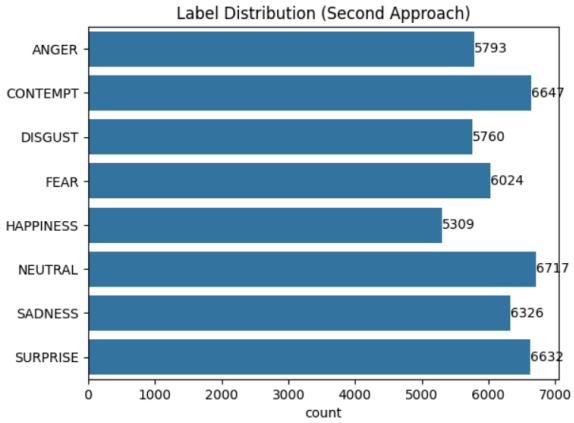

FIG. 3.4: Second Distribution Approach.

We initially hypothesized that creating an exact uniform distribution of data across labels would result in the highest accuracy after training and validation. However, accuracy scores from such approaches did not produce the results we anticipated. In fact, datasets with some variation in size across labels performed better than uniform distributions, even if those uniform distributions contained greater amounts of data to train and validate on. These results are likely due to aggressive overfitting within the uniform distributions. This occurs when data from classes that were originally large in size are unnecessarily duplicated.

While the number of images per emotion label in the second approach is not exactly uniform, this approach allows the modified dataset to closely reflect most labels from the original dataset. We are able to minimize the possibility of underfitting or overfitting when cutting or increasing label sizes solely for the sake of uniformity.

We decided to move forward with the second approach (FIG. 3.4) as the basis for our learner, which would allow us to train and validate the final model on a larger range of data (compared to original values) without compromising every label. This decision is further supported by accuracy and loss statistics from our learner in the sections below.

4. Neural Network Learning Model

Our classification process requires multiple iterations through a learner to accurately capture complex, non-linear edges and features of faces across different angles. As patterns within the data are recognized by a learner, predictions per epoch should gradually improve. Overfitting can be detected and avoided when an imbalance between training accuracy and validation accuracy is identified. For these reasons, neural network models are especially ideal. Applying loss optimization on predictions from previous epochs is especially valuable for improving pattern recognition within each layer of a network. On our data, we experimented with two types of neural network models: convolutional neural networks (CNN) and recurrent neural networks (RNN).

4.1 Convolutional Neural Network Learner

In particular, convolutional layers within a CNN are highly suitable for identifying facial structure patterns and features of importance in image classification. Each neuron in a layer focuses on a specific feature within an image. The connections between the positions of features and layers eventually assemble an entire face. Adjustments of weights on important features, like eyebrows, eyes, and the mouth, gradually improve pattern recognition within specific emotions.

4.1.a. Convolutional Layers

Our model contains three passes through convolutional layers. Each layer contains 32, 64, and 32 filters, respectively. Filter choices help identify facial features while still regulating overfitting on the training data.

For the first layer, low-level features are extracted across 32 filters based on edges and shadows in each image. Within our data processing, applying grayscale transformation to each image assisted with creating uniform shadows for the convolutional layers. A moderate number of filters sufficiently detects basic patterns while controlling computational complexity. When 64 layers were experimentally applied to the first layer, the resulting classifications were slow to compute and noticeably overfit to the training data.

The second layer builds upon the low-level features identified in the first layer. This creates the basis of facial recognition, including the eyes, nose, and contour lines. As the most complex layer, we opted for 64 layers to differentiate between features among all eight emotions.

The complexity from the second layer allows for compression within our final layer, where we reduced filters to 32. Feature maps from the previous layers are further distinguished before compression, so the most noticeable features are kept for further analysis in future activation and pooling layers. Figures 4.1.1 to 4.1.6 demonstrate the visual gradients of the final convolutional layer of our CNN as a feature heatmap. With red for high activation and blue for low activation, each figure specifically indicates which areas most strongly activate the model and, subsequently, are most important towards our final predictions. Not all feature maps are entirely perfect (FIG. 4.1.3), but the inclusion of layers in the following sections assist with mitigating misattributions.

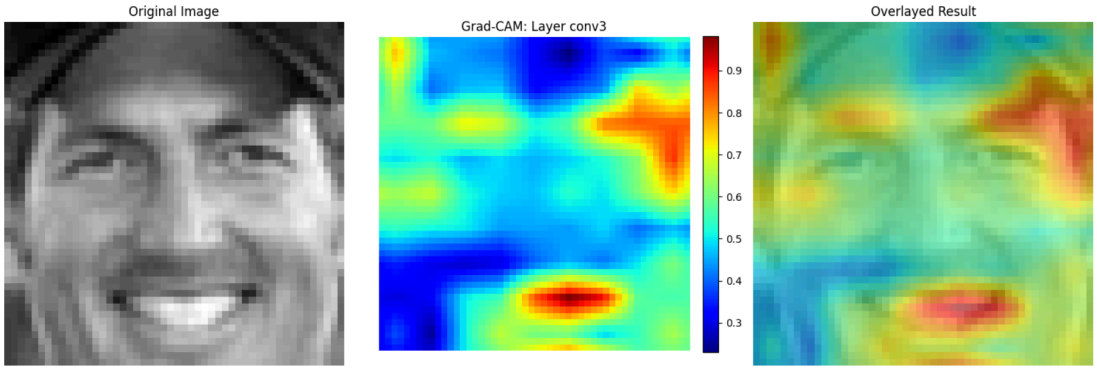

FIG. 4.1.1 [HAPPINESS]: The final layer highlights the mouth and teeth as of highest importance. Intuitively, open-mouthed expressions are common for ‘happy’ facial expressions.

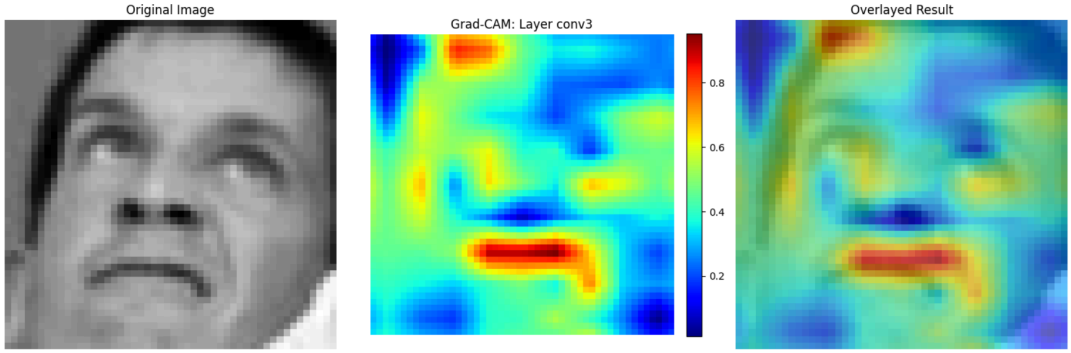

FIG. 4.1.2 [CONTEMPT]: The final layer highlights the downcurve of the mouth as of highest importance for final predictions.

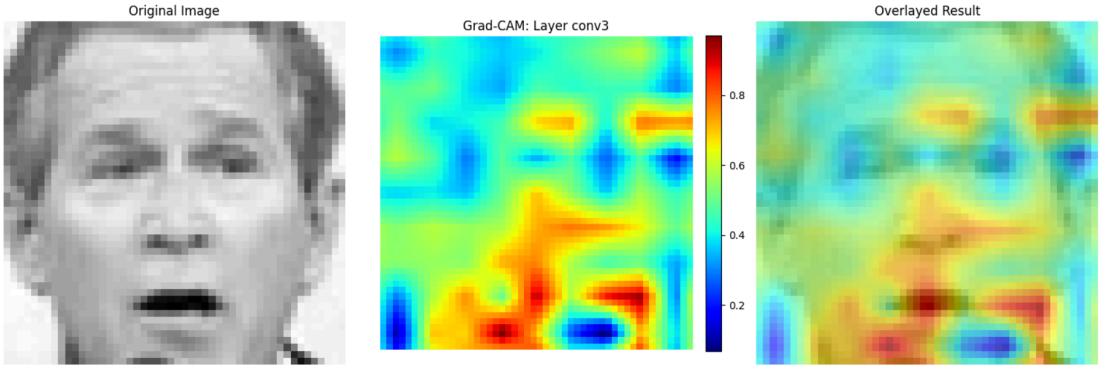

FIG. 4.1.3 [SURPRISE]: The final layer highlights multiple areas centralized around the mouth. It ignores whitespace around the corners of the image and ignores the eyes. This is an example of misattribution. Intuitively, eyes are crucial features.

4.1.b. Pooling Layer Dimension Reduction and Batch Normalization Data Compression

Following each of the first two convolutional layers, pooling layers are added to our CNN to reduce dimensions within the produced feature maps of each layer. Each 2x2 block in the feature maps is condensed into its maximum pixel value. This process greatly increases speed within the model while reducing overfitting on pixel noise within various features.

Introducing batch normalization after each convolutional layer and before each pooling layer further improved our CNN's final performance. Differences in pixel brightness have large impacts on the model performance. Nodes may confuse differences in image brightness as differences between facial features, subsequently adjusting the weights incorrectly. Smoothing the image landscape with normalization stabilizes the data and speeds up convergence between layers.

The inclusion of batch normalization decreased epoch computation speeds by roughly 20 seconds. Accuracy across the first three epochs improved as well, with an increase of 0.30 in validation accuracy for the initial epoch (FIG. 4.1.4, 4.1.5).

```
Epoch 1/3
1131/1131 ━━━━━━━━━━━━━━━━━━━━ 82s 71ms/step - accuracy: 0.2078 - loss: 2.4127 - val_accuracy: 0.6548 - val_loss: 0.8287
Epoch 2/3
1131/1131 ━━━━━━━━━━━━━━━━━━━━ 85s 76ms/step - accuracy: 0.7072 - loss: 0.6999 - val_accuracy: 0.8382 - val_loss: 0.4119
Epoch 3/3
1131/1131 ━━━━━━━━━━━━━━━━━━━━ 88s 77ms/step - accuracy: 0.8783 - loss: 0.3166 - val_accuracy: 0.8982 - val_loss: 0.2550
283/283 ━━━━━━━━━━━━━━━━━━━━ 8s 29ms/step - accuracy: 0.8955 - loss: 0.2636
```

FIG. 4.1.4: CNN model final predictions without batch normalization. Each epoch takes an average of 85 seconds to compute, with first epoch validation accuracy of 0.6548 and third epoch validation accuracy of 0.8982.

Epoch	Time	Step Time	Accuracy	Loss	Val Accuracy	Val Loss
1	63s	51ms/step	0.8058	2.0858	0.9333	0.1885
2	60s	50ms/step	0.9611	0.1103	0.9517	0.1391
3	59s	49ms/step	0.9733	0.0708	0.9672	0.0958

FIG. 4.1.5: CNN model final predictions with batch normalization. Each epoch takes an average of 60.7 seconds to compute, with first epoch validation accuracy of 0.9333 and third epoch validation accuracy of 0.9672.

4.1.c. LeakyReLU Activation

Experimenting between various activation functions resulted in large differences between various methods. Initially, our convolutional layers solely relied on the ReLU activation parameter for each node. After the addition of specific LeakyReLU layers, our validation accuracy increased from 0.91 to 0.95. LeakyReLU activation converts negative inputs to a

smaller value instead of simplifying outputs to zero. This difference prevents dying nodes and maintains information that passes through the model. We can afford this complexity when paired alongside pooling and batch normalization layers. In contrast, the inclusion of LeakyReLU would noticeably increase overfitting if either simplification layers were removed.

4.2 Recurrent Neural Network Learner

When initially drafting our model, we experimented with the implementation of an RNN to add a layer of complexity after the CNN. Initial research into RNN functionality, specifically Long Short Term Memory (LSTM) units, revealed that inputs undergo multiple interations through the model. LSTM units pass a hidden state that adds connections from previous inputs and maintains patterns over multiple inputs. We hypothesized that this process, which mirrors gradient descent via backpropagation, could further improve the models accuracy.

We noticed a decrease in performance when examining differences between our CNN versus CNN-RNN hybrid. Comparing actual versus predicted outputs further suggested that the RNN approach drew incorrect correlations between inputs of different classes. Ultimately, we realized that images within this dataset are separate and randomized. Since the RNN holds memory from previous inputs, labels from previous inputs directly affect future classifications.

An RNN learner would be more appropriate for emotion prediction trained on image frames of one continuous video. However, data in our dataset is not connected. Ultimately, we removed RNN computations from our final model.

4.3 Drop-Out Regularization and Optimization

For the final layer, we implemented dropout regularization to reduce overfitting and improve generalization. During training, it deactivates a fraction of neurons at random to enforce diversity and avoid data memorization. We utilized a dropout rate of 0.20 to balance regularization and sufficient network capacity, which we finalized after multiple iterations of validation accuracy comparison.

We initially tested the Adam optimizer to compile our final predictions. While powerful for computer vision, research revealed that Adam optimization can plateau at lower accuracies during validation. After testing multiple iterations, we finalized RMSprop for our optimizer, which adjusts parameters individually and adjusts learning rates based on error results. This provides more flexibility and faster convergence than optimizers like SGD, which depend heavily on parameter initialization and contain fixed steps.

5. Cross-Validation and Performance Analysis

Figures 5.1 to 5.3 display the accuracy, loss, and final outputs of our model across five epochs.

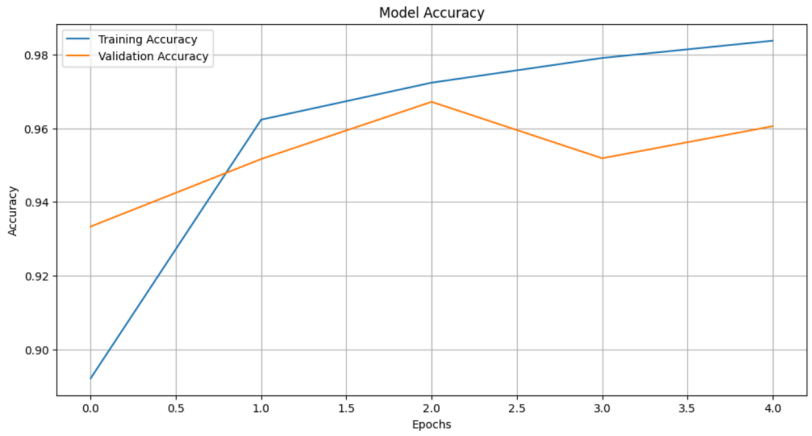

FIG. 5.1: Training and validation accuracies over five epochs.

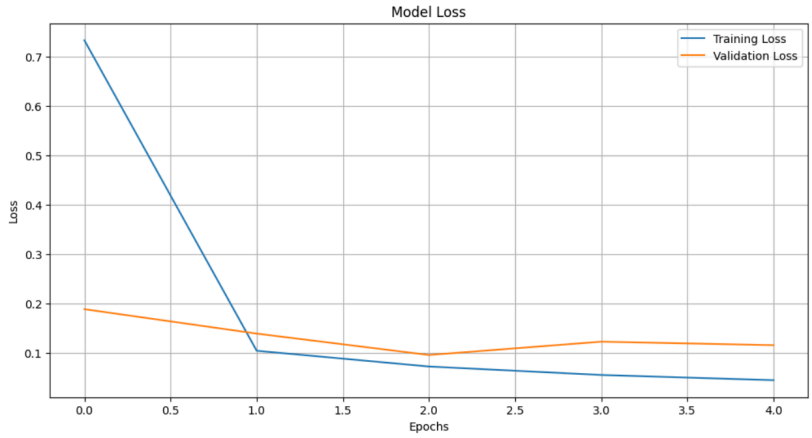

FIG. 5.2: Training and validation losses over five epochs.

Epoch	Time	Step Time	Accuracy	Loss	Val Accuracy	Val Loss
1	63s	51ms/step	0.8058	2.0858	0.9333	0.1885
2	60s	50ms/step	0.9611	0.1103	0.9517	0.1391
3	59s	49ms/step	0.9733	0.0708	0.9672	0.0958
4	60s	50ms/step	0.9809	0.0531	0.9519	0.1227
5	63s	52ms/step	0.9845	0.0435	0.9606	0.1156

Model Result
Loss: 0.1156
Compile Metrics: 0.9606

FIG. 5.3: Prediction data over five epochs.

Peak validation accuracy (0.9672) for our model occurs at epoch three. Validation accuracy increases steadily until epoch four, where there is a slight accuracy dip to 0.9519, before rising to 0.9606 at epoch five. These minor fluctuations show a variance in generalization, but do not contain a significant gap to conclude the model is overfitting. With relatively similar training and validation accuracy scores, the final CNN model is able to perform and classify well while maintaining low bias and variance.

6. Conclusion

Our experimentations with various parameters within convolutional and recurrent neural network models for emotion classification through facial recognition for images were very insightful towards our understanding of CS178 course subjects. We concluded that CNN models were the most effective in understanding facial feature patterns and accurately classifying images with relatively fast computation speeds. Added complexities, like convolutional layer filters and LeakyReLU activation, coupled with normalization techniques, like pooling and batch normalization, balanced total performance decently, resulting in a peak validation accuracy of 0.9672.

Of course, the progression of our model was not entirely linear. Our initial biases towards certain parts of development, like preferring uniform data distributions over variation, resulted in lower accuracies than we hoped. Similarly, experimenting with complex models, like RNN LSTM units, did not improve classification. However, the final model performs proper complexity balancing and efficient computations without compromising final validation accuracy.

In contrast, the progression of our learning was positive throughout our development. Through experimentation, we learned how to best utilize powerful neural networks, like RNNs, to predict outputs based on previous inputs in memory. While interesting to implement, we also learned that complexity does not necessarily lead to the best predictions. We learned to optimize CNN frameworks to yield accurate predictions for our data, which is focused on static images, not continuous frames. Our process towards developing our final CNN model demonstrates the tradeoffs between complexity and normalization, but succeeds in effectively balancing both.

7. Acknowledgements

Maggie Zhang: Write-up workflow, data visualization and analysis, parameter fine-tuning
Josh Talla: Data processing workflow, batch-normalization, parameter fine-tuning
Brandon Sun: CNN model workflow, CNN activation, heatmap creation, parameter fine-tuning

We equally contributed to the development of this model and write-up.