Lu-Taylor Net: A Four Layer Convolutional Neural Network for Facial Expression Recognition

Areen Lu, Timothy Taylor
University of California, San Diego
arlu@ucsd.edu, tvtaylor@ucsd.edu

Abstract

In this report, we design a convolutional neural network (CNN) for the purpose of facial emotion recognition. We used grayscale images from the prominent FER-2013 dataset, which provides a challenging problem where even humans only achieve approximately 65% accuracy. The FER-2013 dataset contains images of seven different emotions, comprising 28,709 training images, and 7,178 validation images. Currently, the best models have about 77% accuracy, and our final model achieved an accuracy of 73.99%. Thus, our model is more accurate at detecting facial emotions than the average human. After testing various structures and architectures, as well as hyper-parameters for training, our final model design is composed of thirteen total layers: four convolutional layers with max-pooling layers to reduce noise, dropout layers to prevent overfitting, and a final two linear layers with softmax applied to the output. For the optimizer, we use RMSprop with a learning rate of 0.001.

0 Group Members

Areen Lu: Structured the training and validation data, handled data pre-processing, helped with network architecture, optimized the learning rates and optimization algorithm, and optimized hyperparameters for peak model accuracy.

Timothy Taylor: Optimized network architecture by testing various numbers of convolutional layers, kernel sizes, types of pooling layers, presence of dropout layers, numbers of neurons in the final linear layer, and various activation functions.

1 Introduction

Emotion recognition is an important and common domain within machine learning, allowing for the understanding and interpretation of human emotion states from modalities such as images, speech, and text. This has gained traction due its diverse applicability such as affective computing, human and computer interaction, marketing, and healthcare (Khare 2019). Emotion recognition contains such profound implications across domains such as enhancing human computer interaction to healthcare interventions as stated by Khare. And due to the development and refinement of such models that are extremely capable of accurately discerning and interpreting emotional states from visual cures, it shows a revolution in the interaction of artificial intelligence and human psychology.

However, despite the prominence of its field, facial emotion recognition has proved a challenging task, which we sought to tackle with our model. We worked with features from the FER-2013 dataset, which is a dataset of 35,887 images and seven labels. Currently, the performance benchmark for the FER-2013 is about 70%, showing the

difficulty of this classification task. The deep learning model that we chose to implement for this problem was a convolutional neural network, as it excels in capturing spatial patterns and hierarchical features from visual data, and in this report we outline the process of refining and optimizing our model's performance. To measure its performance we recorded three metrics: the maximum testing accuracy, training accuracy, and overall accuracy (accuracy over the whole dataset including testing and training data). Ultimately, we achieved 73.99% as our highest overall accuracy, making our model a solid performer for the task of emotion recognition.

2 Methods

2.1 Problem motivation and description

As stated above, the introductions of CNNs have revolutionized the domain of image processing and pattern recognition, allowing for solutions to complicated tasks such as emotion recognition from facial expressions in images. This code presented demonstrates the methodologies and foundational principles to develop CNN model tailors for emotion recognition. By using the FER dataset, we are able to use the power of deep learning to recognize intricate emotional nuances from facial expressions.

At the core, the CNN architecture outlined in the code displays a hierarchy of convolutional and pooling layers with dropout to mitigate overfitting. The model then is able to interactively learn and refine the internal representations of facial features, resulting in its discerning cues of various emotional states. The training process shall be guided with an optimizer and a loss function that will iteratively adjust model parameters to minimize classification errors and enhance the predictive performance of the model.

Emotion recognition contains such profound implications across domains such as enhancing human computer interaction to healthcare interventions as stated by Khare. And due to the development and refinement of such models that are extremely capable of accurately discerning and interpreting emotional states from visual cures, it shows a revolution in the interaction of artificial intelligence and human psychology.

In the following sections, we delve deeper into the intricacies of the CNN architecture, elucidate the nuances of the training, and evaluate the model's performances in differentiating between different emotional states from the FER 2013 data set. Through this report, we aim to contribute to the growing interest in emotional recognition research.

Lastly, we note that humans, evolutionarily designed to be good at recognizing emotions, can accurately label only about 65% of the pictures in this dataset, giving us a good benchmark to aim for regarding our model's performance. The current top performing model only has an accuracy of 76.82%. Thus, emotion recognition from faces is a difficult problem to solve, yet it has a variety of applications. Due to this, we seek to make a model that is at least as accurate as a human, so that we can push the boundaries of the application of emotion recognition.

2.2 Data processing and selection

We chose to work with the FER-2013 dataset for our classification problem.. This data set contains 35,685 examples of 48x48 pixel gray scale images of faces divided into a train and test data set, with 28,709 and 7,178 features respectively. The images are labeled by the emotion shown in the facial expressions, giving a total of seven categories for this dataset: "happiness", "neutral", "sadness", "anger", "surprise", "disgust", and "fear". The total size of the data set is 62.39 MB, and this data set was selected because of its extensiveness, both in regards to size and the number of categories, as well as the challenging problem it poses for highly accurate classification.

2.3 Model architecture

Due to this problem being an image classification class, a natural choice of architecture was a convolutional neural network in order to capture patterns in the spatial layout of the data. Convolutional neural networks are historically very well suited for image classification, and we tested various architectures within the convolutional neural network to optimize our model. We tested varied amounts of convolutional layers with different kernel sizes to best capture the patterns in our data, and we also tested various types and combinations of pooling layers. We tested the presence of max pooling and average pooling, two common methods for reducing noise, and we also tested the presence of dropout layers to reduce the risk of overfitting. Lastly, we tested the presence of one, or two linear layers before the final softmax output. Due to our problem being a multi-class classification task, we implemented a softmax function in the final layer of the network to assign a probabilistic measure to each label, which is known to improve accuracy in the multi-class setting.

2.4 Training procedure

During training, we optimized our models' performance in two ways. Firstly, we tested various optimizers including Adam, and root mean squared propagation (RMSprop), in order to maximize computational efficiency during training, while also looking for the best possible accuracy in our model’s predictions. Due to limitations in computing power, using more efficient training methods also enabled us to have a faster runtime and thus more testing time to optimize the model parameters and architecture. We also tested various hyper-parameters for training, such as learning rate and the number of epochs. We ultimately settled on RMSprop and a learning rate of 0.001. The model is trained using the training data, and performance is monitored using both training and validation sets. During training, we aim to minimize the categorical cross-entropy loss while maximizing classification accuracy.

2.5 Evaluation metrics

To evaluate the performance of our model, we use standard metrics such as accuracy and loss. Accuracy measures the proportion of correctly classified instances, while loss quantifies the dissimilarity between predicted and actual class labels. We compute these metrics on both training and validation data to assess the model's generalization capability. In addition to computing the accuracy on the training and validation data, we computed the models overall accuracy across the whole dataset, in order to compare our model’s performance to the benchmark metrics of other models, and humans, on the FER-2013 dataset.

3 Experiment

3.1 Optimizing data processing

During the optimization of data processing, extensive experimentation was conducted with various image editing functions to identify the most suitable preprocessing techniques for our model. Throughout this process, it became evident that grayscale images are prevalently employed in emotion detection models. This preference stems from their ease of processing and the observation that emotion detection tasks typically do not require significant alterations to the images provided to the model. These findings underscore the pragmatic choice of utilizing grayscale images, given their computational efficiency and the minimal need for intricate image transformations in

emotion detection tasks. This observation serves as a valuable insight guiding our approach to data preprocessing, facilitating streamlined model development and enhancing computational efficiency.

3.2 Optimizing training procedure

During the training process, we conducted experiments with different optimizers including Adam, Adamax, and RMSprop. After thorough evaluation, RMSprop emerged as the top performer, achieving the highest average accuracy of 74%. Remarkably, a learning rate of 0.001 yielded the best results across all three optimizer algorithms. Given RMSprop's superior performance, we further refined our approach by adjusting the batch size to a smaller value of 32, which is known to be beneficial for RMSprop optimization.

3.3 Optimizing network architecture

To optimize the network of the architecture, firstly we tested different numbers of convolutional layers and kernel sizes. We tested networks with 2, 3, 4, and 5 convolutional layers as well as kernel sizes of 2, 3, and 5, as well as a mixture of 3 and 5. For the mixture of kernel sizes, we used the kernel size of 5 in the first two layers, in an attempt to reduce noise and gather data on the bigger and overall patterns in the data, with a smaller kernel of 3 in the later convolutional layers. We ran each network with the varied architecture of the convolutional layers, and recorded the training, testing, and overall accuracy we achieved. We used the same performance to gather performance metrics for the different kernel sizes as well. From our results in *Table 1*, we see that 3 and 4 convolutional layers yielded the best performance in terms of overall accuracy. However, 4 convolutional layers had a better validation accuracy, meaning it performs better in the face of unseen data, and is probably less overfitted to the training data than the architecture with 3 convolutional layers. Thus, we find that 4 convolutional layers is a better choice for our model. In *Table 2* we see that a constant kernel size of 5 for each layer yielded the best performance for all metrics. Overall, we determined that four convolutional layers each with a kernel size of 5 was the optimal choice for our model's architecture.

Table 1: The Effect of Different Numbers of Convolutional Layers on Model Accuracies

Convolutional Layers	**Train Accuracy**	**Validation Accuracy**	**Overall Accuracy**
2	0.41	0.52	0.43
3	0.70	0.65	0.68
4	0.65	0.80	0.68
5	0.59	0.80	0.63

Table 2: The Effect of Different Kernel Sizes of Convolutional Layers on Model Accuracies

Kernel Sizes	**Train Accuracy**	**Validation Accuracy**	**Overall Accuracy**
(2, 2, 2, 2)	0.63	0.76	0.65
(3, 3, 3, 3)	0.62	0.80	0.65
(5, 5, 5, 5)	0.66	0.80	0.69
(5, 5, 3, 3)	0.65	0.78	0.68

Next, we chose to test the difference in performance that different pooling layers would cause. We knew that data from images is typically quite noisy, and average pooling as well as max pooling layers are a good way of reducing noise in data, by reducing the variance between nearby values. We tested the presence of three max pooling layers versus three average pooling layers, as well as a mixture of the two. Ultimately, as seen in *Table 3* we found that average pooling layers yielded the best overall accuracy, but the overall accuracy was only marginally higher (by 0.01) than the max pooling accuracy, whereas max pooling had much better validation accuracy (0.80 in comparison to 0.60). Thus, using max pooling likely makes our model less prone to overfitting, and better at performing in the face of new data, so we choose to just implement max pooling layers for our model.

Table 3: The Effect of Different Pooling Layers on Model Accuracies

Type of Pooling	**Train Accuracy**	**Validation Accuracy**	**Overall Accuracy**
Max Pooling	0.63	0.80	0.66
Average Pooling	0.69	0.60	0.67
Average and Max Pooling	0.59	0.80	0.63

In addition to reducing noise, we also wanted a way to reduce the possibility of overfitting to the training data, a notorious problem for neural networks. To combat this, we tested the implementation of dropout layers, which randomly set values to 0 at a given frequency in an attempt to reduce the possibility of overfitting in our model. We chose to add three dropout layers, the first two with a rate of 0.25 and the final with a rate of 0.5, as these rates are generally recommended for the input dropout rate (the first dropout layer), and hidden layer dropout rate (the later two dropout layers). We pay particular attention to the validation accuracy, as this is a direct measure of whether our model is overfitted to the training data, since it is new data that our model has never seen. From *Table 4*, we see that the presence of dropout layers improved the accuracy of our model as desired, not just in the validation accuracy, but in all metrics. Due to this, we chose to keep the dropout layers in our final implementation.

Table 4: The Effect of Dropout Layers on Model Accuracies

	Train Accuracy	**Validation Accuracy**	**Overall Accuracy**
Dropout Layers Present	0.66	0.80	0.68
No Dropout Layers	0.58	0.78	0.62

After testing the architecture of the layers in our model, we tested various activation functions for each layer. We tested the performance differences between using reLU activation functions for each layer, versus eLU or sigmoid. We chose to test eLU because of its allowance for negative values, and sigmoid because it is also a good candidate for activation function. From our results in *Table 5*, we see that reLU activation functions yielded much higher performance across all accuracy metrics, cementing the use of the reLU activation function in our model.

Table 5: The Effect of Different Activation Functions on Model Accuracies

Activation Function	**Train Accuracy**	**Validation Accuracy**	**Overall Accuracy**

reLU	0.66	0.80	0.69
eLU	0.61	0.67	0.62
Sigmoid	0.40	0.31	0.38

Lastly, we tested the effect of the number of linear layers near the output layer of our model. We compared the performance of one vs two linear layers, not less than one because we need one to flatten the outputs from the convolutional layers, and not more than 2 for the sake of runtime. In *Table 6* we see that having 2 linear layers improves accuracy on the training data, testing data, and the accuracy on the dataset as a whole. Thus, we include 2 linear layers in the final version of our model.

Table 6: The Effect of Numer of Linear Layers on Model Accuracies

Number of Linear Layers	**Train Accuracy**	**Validation Accuracy**	**Overall Accuracy**
1	0.65	0.75	0.67
2	0.66	0.80	0.69

4 Conclusion

Using our results above, we determined that the best architecture for our network would be described by the diagram in *Figure 1*.

Figure 1: Final Choice of Architecture for the CNN

Layer	**Output Shape**
Convolutional Layer (Kernel = 5, Activation = reLU	(44, 44, 32)
Convolutional Layer (Kernel = 5, Activation = reLU	(40, 40, 64)
Max Pooling (Kernel = 2)	(20, 20, 64)
Dropout Layer (Rate = 0.25)	(20, 20, 64)
Convolutional Layer (Kernel = 5, Activation = reLU	(16, 16, 128)
Max Pooling (Kernel = 2)	(8, 8, 128)
Convolutional Layer (Kernel = 5, Activation = reLU	(4, 4, 128)
Max Pooling (Kernel = 2)	(2, 2, 128)
Dropout Layer (Rate = 0.25)	(2, 2, 128)
Linear Layer (Activation = reLU)	(512)

Linear Layer (Activation = reLU)	(512)
Dropout Layer (Rate = 0.25)	(1024)
Output Layer (Linear, Activation = softmax)	(7)

Using this optimized model, we attained a peak test accuracy of 0.80, training accuracy of 0.7399, and an overall accuracy of 73.99%. Thus, our model performs quite well, comparable to or even better than a human, and it hits the performance benchmarks we aimed for when tackling this dataset. Our results demonstrate the potential of machine learning algorithms to adapt to, and even surpass humans in fields that we specialize in, such as facial emotion recognition. This has many applications as discussed before, such as in healthcare, where it could provide facilities with automated methods to regulate and attend to patients. Moreover, it shows that relatively simple neural networks, (13 total layers with 4 convolutional layers) can still perform extremely well in the face of difficult problems and data.

References

Khare, Smith K., et al. “Emotion Recognition and Artificial Intelligence: A Systematic Review (2014–2023) and Research Recommendations.” Information Fusion, vol. 102, 1 Feb. 2024, p. 102019, www.sciencedirect.com/science/article/pii/S1566253523003354#sec15, https://doi.org/10.1016/j.inffus.2023.102019.

“Emotion Detection.” www.kaggle.com, www.kaggle.com/datasets/ananthu017/emotion-detection-fer.