Classifying Facial Expressions using Convolutional Neural Networks

AMTH 667 - Final Project Report
Prof. Steven Zucker
Muyi Aghedo

Introduction

Facial expression is perhaps the most unique method of communication between human beings (and perhaps other between other species as well), so I wondered about the plausibility of tackling the deciphering of facial expressions using computational vision techniques.

Initially, I wondered if I should perhaps use some dimensionality reduction or feature extraction techniques amongst the ones we learnt in class: principal component analysis, diffusion mapping, etc to preprocess the data and then perform some perceptron analysis, but after seeing how powerful convolutional neural networks could be, especially for images, I decided to pivot to those!

The end result is a trained model and corresponding script for which we can further train or evaluate our model on the validation set.

Dataset

I utilized a pre-existing dataset, found on Kaggle, available here (www.kaggle.com/datasets/jonathanoheix/face-expression-recognition-dataset).

The data labels about 36,000 48 x 48 grayscale images into 7 emotions: anger, disgust, fear, happiness, neutrality, sadness, and surprise. This dataset was well-maintained, organized, and suitable for my project. Example images from this dataset are shown below:

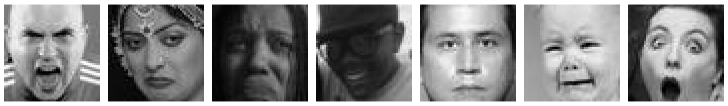
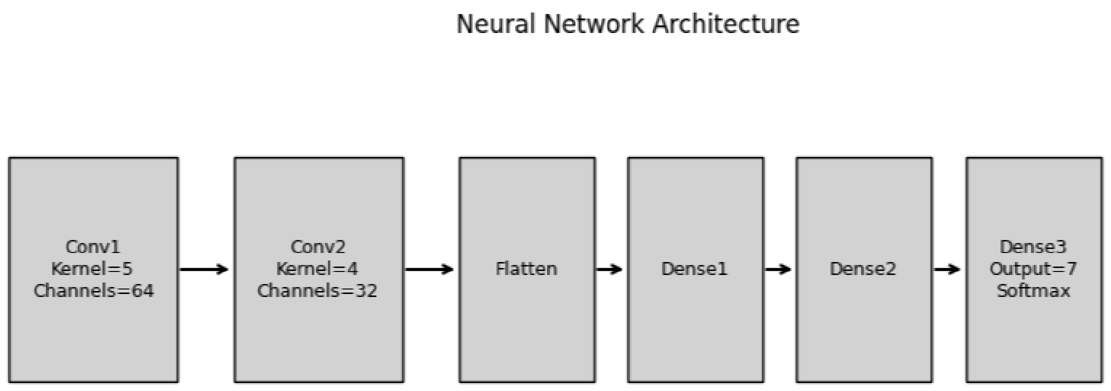


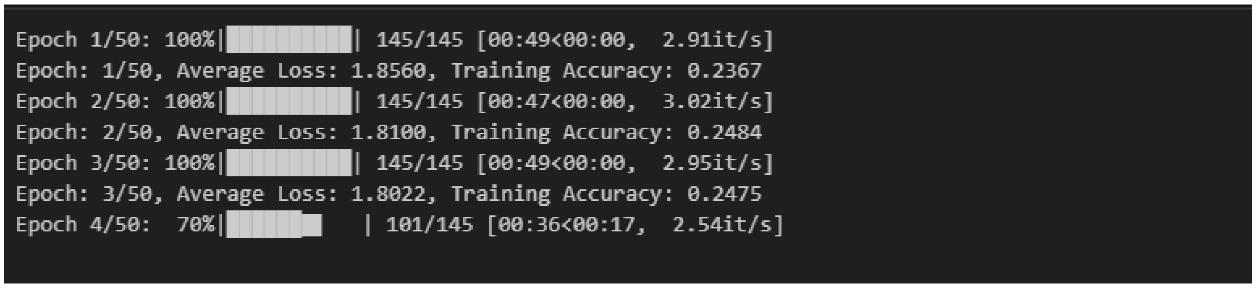
Figure 1. Sample Images From Dataset
(from left to right: anger, disgust, fear, happiness, neutrality, sadness, surprise)

Methodology

My chosen model architecture for this computer vision task was a convolutional neural network (CNN), which, as I've learned, are specifically designed to process grid-like data like images, and have demonstrated remarkable success in various image recognition and classification tasks. Using Python, I loaded the training and testing data into memory with a batch size of 150 images each. The network architecture consisted of two convolutional kernel layers. The first convolutional layer utilized a kernel size of 5 and had 64 output channels. This was followed by a second convolutional layer with a kernel size of 4 and 32 output channels. The output of these convolutional layers was then flattened before being fed into a standard feed-forward architecture with ReLU activation functions between three layers. The final dense layer had 7 output units, corresponding to the seven emotion classes, and employed a softmax activation function to produce a probability distribution over these classes.


Neural Network Architecture
Conv1 Kernel=5 Channels=64
Conv2 Kernel=4 Channels=32
Flatten
Dense1
Dense2
Dense3 Output=7 Softmax

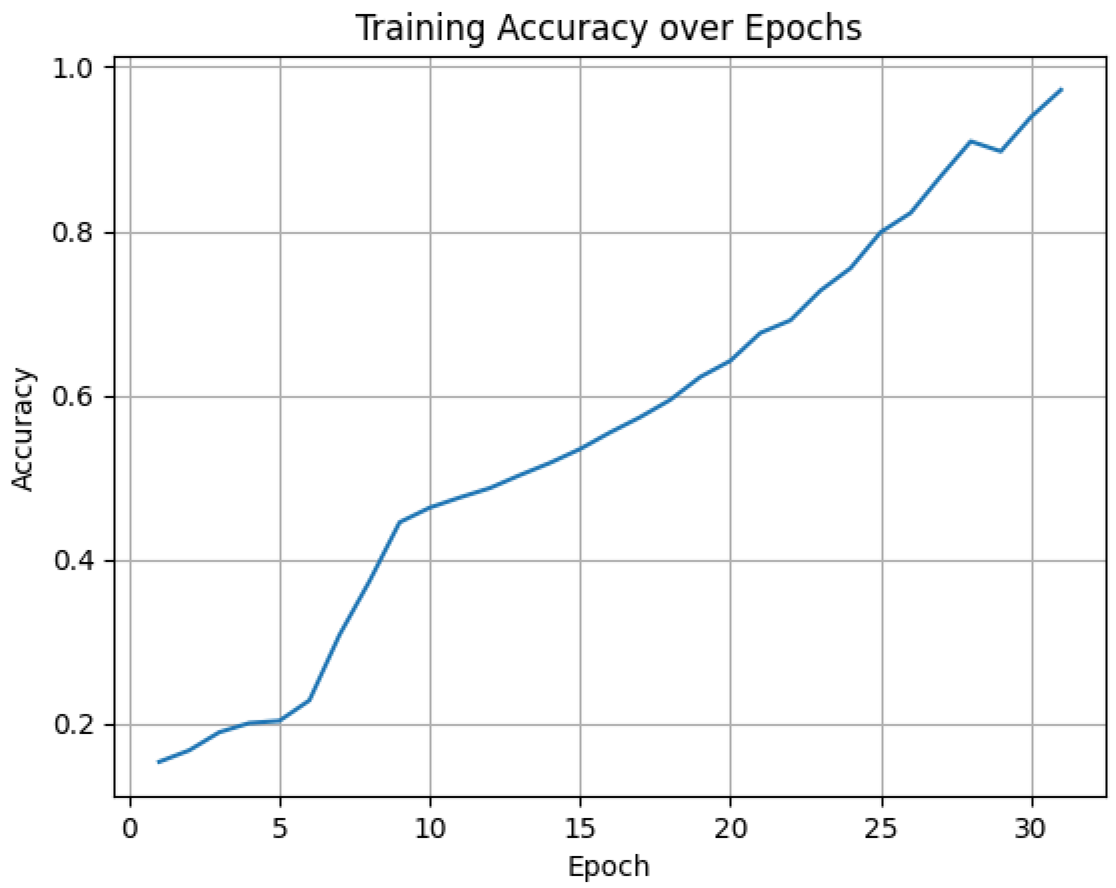
The implementation of this itself was pretty straightforward. Using PyTorch, I defined my model as a NN module that could be saved and loaded to continue training when necessary, and defined a training loop to optimize the optimization function (stochastic gradient descent, in my case) and minimize the loss, measured using PyTorch's cross-entropy loss. I stopped training until after 50 epochs, or until getting 95% accuracy on my training set, whichever came first. Below is some sample output from my training loop:



```
Epoch 1/50: 100%|██████████| 145/145 [00:49<00:00,  2.91it/s]
Epoch: 1/50, Average Loss: 1.8560, Training Accuracy: 0.2367
Epoch 2/50: 100%|██████████| 145/145 [00:47<00:00,  3.02it/s]
Epoch: 2/50, Average Loss: 1.8100, Training Accuracy: 0.2484
Epoch 3/50: 100%|██████████| 145/145 [00:49<00:00,  2.95it/s]
Epoch: 3/50, Average Loss: 1.8022, Training Accuracy: 0.2475
Epoch 4/50:  70%|███████   | 101/145 [00:36<00:17,  2.54it/s]
```

Results and Measures of Success:

My evaluation on my training set over epochs ended up being 0.97, or 97%. Here's a graph oif the training accuracy over time below:



On the validation set, I ended up getting an accuracy score of 0.518, or 51.8%. For what would otherwise be a 1 in 7 chance of correctness, I would say this score is pretty good! Perhaps with a larger batch size I would be less susceptible to overfitting, or maybe there are patterns in the validation set that my model still hasn't completely understood.

My measures of success included achieving a significantly higher accuracy than random chance (1/7 or approximately 14.3%) and demonstrating the CNN's ability to learn meaningful features from the image data, ideally reaching or exceeding the training accuracy target of 95%. With that, I think this project has been pretty successful!

Obstacles and Limitations

Several challenges and limitations were encountered during the course of my project:

- **Hyperparameter Tuning:** Deciding on the optimal hyperparameters for the convolutional neural network proved to be a significant hurdle. Without prior knowledge of how the model would behave with different configurations, the selection process involved a degree of trial and error. There was a constant uncertainty regarding whether the chosen hyperparameters would enable the model to learn effectively. Fortunately, the training process indicated that the model was indeed learning patterns from the data as anticipated.

- **A Really Long Training Time:** The sheer size of the training dataset significantly impacted the training duration. With each epoch taking just under 1 minute to complete, and a processing speed of approximately 2.8 iterations per second (with 145 iterations per epoch), the total training time for 50 epochs amounted to a considerable period. This issue was compounded by the lack of CUDA support on my GPU, which prevented the utilization of hardware acceleration for the computations. Furthermore, the logistical challenge of migrating the 38,000-image dataset to a cloud computing platform, which could have provided the necessary computational resources, further exacerbated this limitation.
- **Model Complexity (Or lack thereof):** The relatively shallow architecture of the CNN, consisting of only two convolutional layers and three dense layers, might have limited the model's capacity to learn highly complex and nuanced features present in the facial expressions. While the model showed signs of learning, there's a possibility that a deeper or more sophisticated architecture could have yielded better performance.
- **Data Variability:** Facial expressions are inherently complex and can vary significantly between individuals due to factors such as age, ethnicity, lighting conditions, and the subtlety of the expressed emotion. While the dataset was substantial, it may not have captured the full spectrum of this variability, potentially limiting the model's generalization ability to unseen data with different characteristics.
- **Interpretability:** As with many deep learning models, understanding the specific features learned by the convolutional layers and the decision-making process of the network remains a challenge. This lack of interpretability can make it difficult to diagnose potential issues or gain deeper insights into how the model is classifying emotions.

I think these all highlight areas for potential improvement in future iterations of this project, such as exploring more advanced model architectures, optimizing data handling for cloud platforms, and investigating techniques for better interpretability.