Application of Convolutional Neural Networks Towards the Classification of Human Emotions from Images:

Joseph Alverson (12218544)

Eugene Antwi Boasiako (12424421)

Noah Bohannon (12139288)

Purpose:

The purpose of this project is to utilize a Convolutional Neural Network (CNN) to classify the emotions shown on a human face. We have several large datasets that will be used to train the model and then the model will be applied to an image captured once the program detects a face. The program will then display this capture along with the emotion it believes to be most prevalently shown in the image. We implemented this algorithm to work as quickly as possible so that there would only be a short delay between the image capture and the final result displayed.

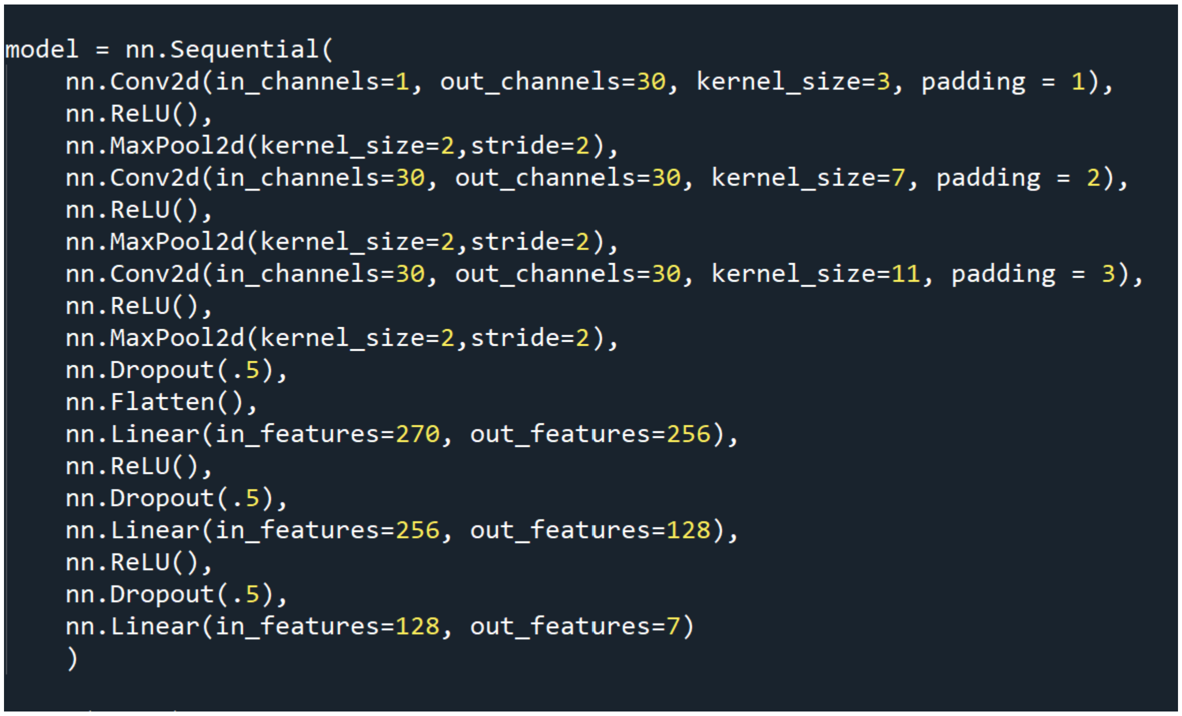
Related Work:

A. Verma et. al used a similar convolutional neural network in order to detect the emotions on the human face. Their network consisted of two convolution layers and two pooling layers and then those layers are flattened and then fully connected in the following layers. They also looked at several different types of architectures in their experiment including rectangular, modified triangular, and Venturi. They showed that the Venturi architecture had the best results with a training accuracy of 98.87%, a training loss of .0224, a testing accuracy of 86.78%, and a testing loss of .9693. For each of the seven emotions they classified their confusion matrix showed that each category had above an 85% correct classification.

M. Zadeh et. al. also used a CNN to classify the emotions on a human face. Their sample set was images of Japanese women showing different emotions. Where their method differed from Verma is that they used two Gabor filters to preprocess the image before feeding it into the CNN. The Gabor filters were used to filter the image by giving a higher response to points where the texture changes and at edges. This allowed for a larger focus to be placed on the more important features of the face that could then be used to determine the emotion. Applied two of these filters lead to a marked improvement in the performance of the CNN across all the epochs that they tested, ending with a thirty-epoch run that gave them an accuracy of 97.16% over the base CNN which had an accuracy of 91.16%. They also showed that this method drastically reduces the time across the higher accuracy ranges.

Method:

For our implementation, we used PyTorch and a six layer network for our convolutional neural network. Our architecture consists of 3 convolutional layers, 3 max pooling layers, and 3 fully connected layers, the settings for the network are referenced both in the code and below:



```
model = nn.Sequential(
    nn.Conv2d(in_channels=1, out_channels=30, kernel_size=3, padding = 1),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size=2,stride=2),
    nn.Conv2d(in_channels=30, out_channels=30, kernel_size=7, padding = 2),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size=2,stride=2),
    nn.Conv2d(in_channels=30, out_channels=30, kernel_size=11, padding = 3),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size=2,stride=2),
    nn.Dropout(.5),
    nn.Flatten(),
    nn.Linear(in_features=270, out_features=256),
    nn.ReLU(),
    nn.Dropout(.5),
    nn.Linear(in_features=256, out_features=128),
    nn.ReLU(),
    nn.Dropout(.5),
    nn.Linear(in_features=128, out_features=7)
    )
```

Figure 1: CNN settings

This network is trained on grayscale images, meaning that the input channel value is one when the CNN starts. The dataset we used was titled "facial emotion recognition" and can be found at the provided link: https://www.kaggle.com/chiragsoni/ferdata. Once the data was obtained, we used cross entropy loss to evaluate how far off our prediction was and to change the values. We then trained our data over 40 epochs and the data trained over 26 minutes. On the provided test set, our algorithm performed at 46.79 percent accuracy. As you can see, from the time it took to train, this is a lightweight CNN.

The facial detection and image capture portion of the program was implemented using OpenCV. The program detects the human face and then captures a screenshot of the face. The implementation uses the Haar Cascade Classifier for real-time face detection. A Haar Cascade classifier works by first calculating Haar features which are essentially calculations that are performed on adjacent regions at a specific location in a detection window. It involves summing the pixel intensities in each region and calculating the differences between sums and then using adaboost to choose the best features to train the classifiers. Pretrained Haar Cascade classifiers are available in OpenCV for use. As such our implementation of facial detection was based on the pretrained classifiers in OpenCV. The implementation involved converting an image to grayscale such that the RGB dimension of the image is lost. It essentially implies converting a 3-dimensional array into a 2-D array. Then the 2-D array is used for face detection and to draw a bounding box around the detected face after which a screenshot of the face is captured and saved for use with the CNN to predict emotion.

Once this image is passed to the CNN same preprocessing done to the test set is done to this image, which includes converting it to grayscale, trimming the image, and then converting it into a tensor. The CNN is then able to take this data and predict the emotion displayed on the face from the live camera feed capture. This picture is then displayed using OpenCV with the predicted label at the top.

Experiments and Results:

Number of Epochs:	Test Accuracy:
20	37.93%
40	46.79%
80	46.4%

Table 1: Table of Epochs vs. Accuracy

From the data above one can see that we were not able to meet the same accuracy metrics as the relevant work using our algorithm design. We saw a slight decrease in the accuracy when we went from 40 epochs to 80, however this was to be expected since we were overfitting the data with that many epochs. We settled on using 40 epochs because that gave us the highest accuracy out of the ones that we tested.

Discussion on the change in architecture or model to get better accuracy

What we were able to do was integrate the camera system of our device into the neural network to generate a prediction. This is a new and novel way for the user to interface with the machine and the CNN. While the accuracy of the predictions was lower than we had hoped, we still were able to integrate our facial detection and image capture system into the CNN to generate predictions. In doing this project, we learned a few lessons about deep learning. Our first takeaway was that deep networks like Alexnet and VGG are very computationally heavy and take a very long time to train. Those paradigms were our first choice for the specific type of algorithm to use, but their long training time made it hard to train and tweak hyperparameters. We figured we could do a lighter weight implementation and tweak the parameters to get it operating at an acceptable percentage. We now realize that, if we had more time and better hardware, the implementation of one of those algorithms could have brought us more success. We also could have done better with some more data to train on, this also could have helped boost the accuracy of this algorithm.

Bibliography:

A. Verma, P. Singh and J. S. Rani Alex, "Modified Convolutional Neural Network Architecture Analysis for Facial Emotion Recognition," 2019 International Conference on Systems, Signals and Image Processing (IWSSIP), 2019, pp. 169-173, doi: 10.1109/IWSSIP.2019.8787215.

M. M. Taghi Zadeh, M. Imani and B. Majidi, "Fast Facial emotion recognition Using Convolutional Neural Networks and Gabor Filters," 2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI), 2019, pp. 577-581, doi: 10.1109/KBEI.2019.8734943.