Abstract

Understanding facial expressions is one of the key factors of social interactions and reading social cues. It is an important skill in communicating with others and being able to relate to those around you. While understanding facial expressions is not uniquely a skill only humans have, it is a sign of intelligence. In this application of artificial intelligence and classification techniques we try to identify expressions in facial images. We hope to be able to identify facial expressions by training our neural network to look for specific characteristics in an expression. For example to identify a "happy" expression we can train our neural network to look for features such as observation of teeth, dimples in the cheeks, and reduction in the size of the eyes. Through identification of a facial expression we can then classify the emotion of the person behind the image and slightly move one step forwards towards general intelligence.

Introduction

The problem of learning facial expressions or learning the mapping from a facial image to a human emotion is an image classification problem. Our given input is pixel data in the form of an array of integer values. Each integer value (in the range of 0 to 255) represents the grayscale value of a pixel, where 0 is black and 255 is white. The corresponding target label for each image is a discrete integer value (in the range of 0 to 6) representing the emotion of the person in the image. (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). Our goal is to develop a machine learning algorithm to learn this mapping from raw pixel data to emotion. The difficulty arises in how to parameterize the algorithm to learn features from the pixel data. While we see an image, the computer simply sees a 2,304 sized array of integers. This presents a challenge to the algorithm to identify and learn the features that affect the final predicted label. The current state-of-the-art approach to general image classification currently utilizes convolutional layers to extract features from an image in their neural network architecture. The network then takes theses features as input to a fully connected network. We'll be describing different techniques and approaches in applying convolutional neural networks to the problem of facial expression classification.

Data Analysis

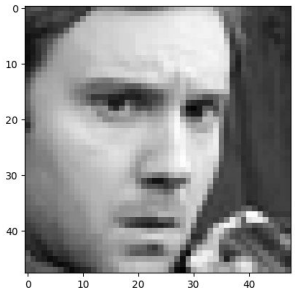

Figure 1: Preview of the Data.

The training data consists of 28,709 48x48 (grayscale) images similar to the one above. For each of the images, there is an associated integer label that describes the expression the face is showing. (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). The data comes from a variety of sources and as a result some images have a disruptive text watermark on it while others have a side view of a person's face. This presents a challenge to the neural network to separate watermark and face from the image and to

adjust and generalize facial recognition to side views of a face.

In general it is difficult to solve this classification problem using any machine learning technique considering how it is often difficult for humans to determine the facial expression and emotion of another person. What one person may consider to be sad, another may consider the expression to be neutral. Due to these inconsistencies there exist some data in this challenge where there is no consensus on the emotion or expression displayed by the person. The image below is an example of this, most would consider this expression to be neutral, however in the dataset this image was categorized as Angry.

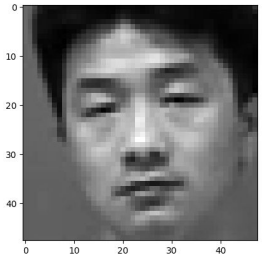

Figure 2: Example of a misclassified expression in the data

The table below shows the mean probability of classifying the image correctly in both the train and test datasets. *Figure 3*

	Average of Probability	Average of Probability
0	0.601143169	0.989626145
1	0.657541993	0.995128767
2	0.490149661	0.984858124
3	0.849857101	0.996594312
4	0.449727947	0.977925514
5	0.807300705	0.992135921
6	0.664949236	0.997012957

Train accuracy per class *Test accuracy per class*

From this table, it can be shown that on average class 2 (Fear) and 4 (Sad) are the hardest to classify while classe 3 (Happy) and 5 (Surprised) are the easiest to classify. It also shows that the model is excellent in classifying training data. The training data appears to preserve similar structures and hence able to have high training accuracy.

The distribution of the different classes in the training data are shown below.

Class	Outliers	Total samples	%outliers
0	24	3994	0.600901
1	2	436	0.458716
2	33	4097	0.805467
3	4	7215	0.05544
4	43	4829	0.890454
5	9	3170	0.283912
6	4	4964	0.08058

Figure 4: Distribution of classes in the training data

This shows that class 1 is underrepresented in the training data. This may cause issues with prediction. The distribution of the different classes and the percentage of outliers in the test data are shown below as well.

Class	Outliers	Total	%outlier
0	120	401	29.92519
1	7	41	17.07317
2	181	422	42.891
3	94	845	11.12426
4	290	584	49.65753
5	52	376	13.82979
6	150	557	26.92998

Figure 5: Distribution of classes/outliers in test data

We define an image as an outlier if the correct label is not contained in the top 3 probabilities of the softmax output. The test table shows that a substantial amount of outliers appear in the test set, which explains the low test accuracy we are experiencing (68%). Convolutional neural nets do not function well if the dataset is skewed or contains a large amount of anomalies. The dataset we are dealing with here does contain these issues.

Model

We started with the most basic convolutional neural network model and iteratively modified it until we increased our train and test accuracy. Initially, we started off with four convolutional layers, each followed by a max pooling layer. We then flattened the filters after the four convolutional layers. Next we added two fully connected layers, first being 256 neurons in size and the second being 512 in size. Lastly, we added an output layer with 7 neurons, each representing one of the 7 possible target labels. Each layer uses the ReLU activation function except for the last output layer that uses softmax activation.

We found that this basic convolutional model gave us around 90% training accuracy and a low 56% test accuracy. We first tried to improve the test accuracy by adding a dense layer of 100 neurons between our softmax output layer and our 512 neuron sized dense layer. This slightly increased our test accuracy by 2%.

To prevent overfitting, we added additional parameters to our model. First we added batch normalization and dropout of 50% to all layers. We also added Gaussian Noise between the first and second convolutional layers and between the third and fourth convolutional layers. All three of these techniques increased the test accuracy by 5%. While this was a modest improvement, our model was still significantly overfitting the training data. The next technique we tried was to modify the training data itself. With images we can rotate, flip, and shift the pixels of the image without changing the label. This is known as data augmentation and we were able to do this through the Keras ImageDataGenerator class. The generator takes an image and augments it in a way so that the original structure of the image is the exact same, however the image has been slightly transformed. In the figure 1 image example, the image was simply translated a few pixels up to the top, and the new space at the bottom was filled by having the last row of pixels repeated. This data augmentation increased our test accuracy by 5%.

Final Network Architecture

Our final model architecture consists of four convolutional layers, each followed by a max pooling layer. Each convolutional and max pooling layer pair was followed by batch normalization and dropout of 50%. In-between the first and second convolutional layer and in-between the third and fourth convolutional layer we added Gaussian Noise to the inputs. Following the four convolutional layers we have three fully connected layers of size 256, 512, and 100. Each followed by batch normalization and dropout of 50%. Finally we have an output layer of 7 neurons with softmax activation.

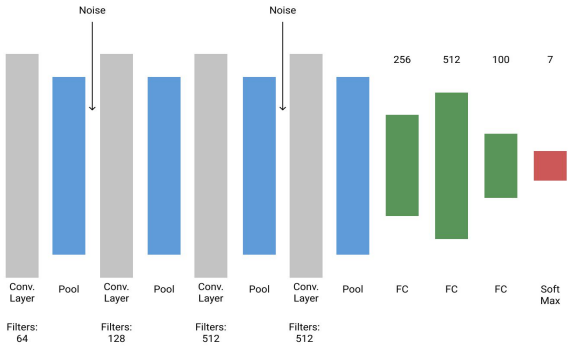

Figure 6: Conv Net Model Architecture

Training

To train our model we separated the learning into two phases. The first phase was to fit the model to the augmented training data as described before. For this we trained

our model for 30 epochs until our loss and accuracy converged.

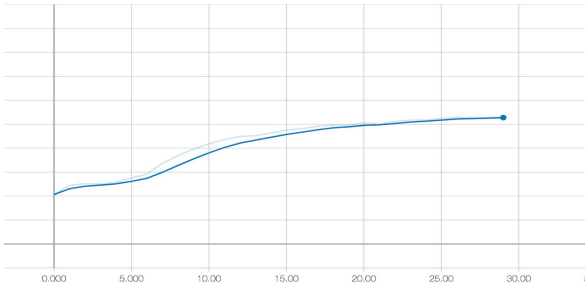

Figure 7: Increasing train accuracy over 30 epochs

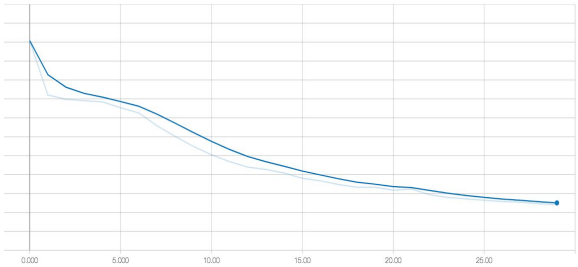

Figure 8: Decreasing training loss over 30 epochs

The second phase was to fit the model with normal pixel data and for we did this for 50 epochs. For both training phases we added class weights to adjust and increase the weights of underrepresented classes in the loss function.

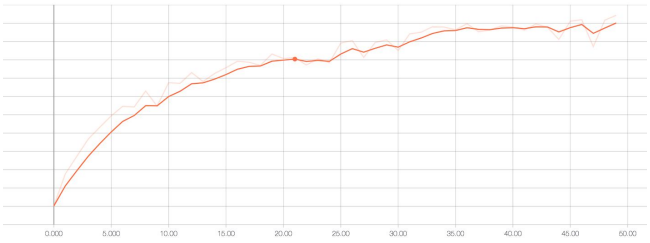

Figure 9: Increasing training accuracy over 50 epochs

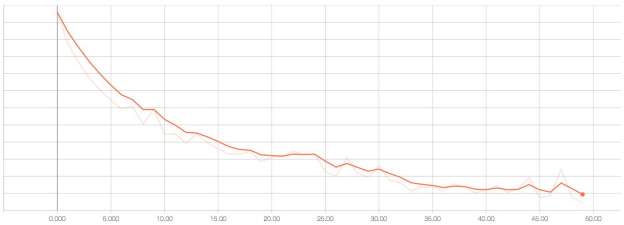

Figure 10: Decreasing training loss (error) over 50 epochs

Results

After both training phases, we were able to obtain 93.7% training accuracy with a loss of 0.1925 and a overall test accuracy of 67% averaged over all 7 classes.

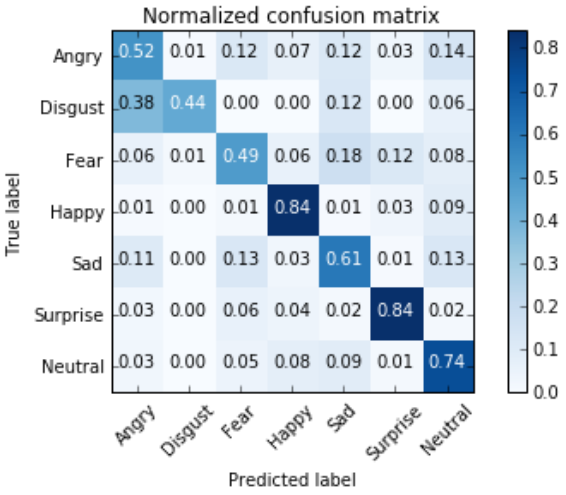

Figure 11: Confusion matrix of our predictions

Conclusion

The classification of expressions from the facial images of individuals is non-trivial. The underlying dataset had inconsistencies and there was a significant difference between the training and test data which would explain the model's inability to generalize well to the test data. To counter this we added various techniques to avoid overfitting which overall improved our model by 12%. As a result, our current network architecture gave 93% training accuracy and 68% test accuracy. We also observed that some classes have high test accuracies such as 'Happy' and 'Surprise' with both having a 84% test accuracy. With respect to the Kaggle competition, if we were to have competed when the contest was active we would have scored top 5 in the contest.

References

[1] https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge

[2] Alizadeh, Shima, and Azar Fazel. “Convolutional Neural Networks for Facial Expression Recognition.” *Computer Vision and Pattern Recognition*, 22 Apr. 2017.