Emotions Recognition with Convolutional Neural Networks

James Holman (z5143237), Nihit Vyas (z5140232), Vidika Badal(z5129296)

Abstract

In the recent years, Artificial Neural Networks have been at the forefront of the development of smart-learning systems. Specific strains of Neural Networks such as *Convolutional Neural Networks* (CNNs) have evolved out of research to specialise to areas such as image recognition, and physical hardware has only become more powerful in the last few years, accommodating this development. With the launch of Google's Tensor Processing Units, Neural Networks based smart systems are now being embedded into a cornucopia of devices including smartphones.

This project aims to implement a Convolutional Neural Network based Facial and Emotions Recognition System with potential applications in measuring customer satisfaction for businesses, surveillance and behavioural analysis for law-enforcement agencies and automatic photography based on particular emotions of the users. The implemented system has been proposed by Gudi [1] using the FERC-2013 Facial Emotions Recognition Dataset.

Related Work

On Friday 12th of April 2013, Kaggle set the challenge of identifying emotions from a large dataset of expressions (FERC-2013). Since then there have been further attempts by other groups to outperform the winners of the challenge. In 2016 Correa et al. also were able to train a CNN model implementing alexnet to produce some accuracy in emotion recognition [2]. Furthermore, in other areas of CNN based image recognition there has been great progress in new layer architectures, such as Google's inception-v4 [3]. Research in these other areas of image recognition result in new techniques being applied within emotion recognition. Our goal was to implement a CNN for emotion recognition, inspired by some of this work while experimenting optimization techniques.

Overview of Convolutional Neural Networks

Convolutional Neural Networks consider 3 (width, height and depth) dimensional volumes of neurons that advance on the parameter sharing scheme, which minimizes the parameters needed for a network. So, neurons in one layer are connected to a small regional layer before it. Convolutional Neutral Networks are divided into major 4 layers:

1. Convolutional Layer
2. Pooling Layer
3. ReLu Layer
4. Fully-Connected Layer

These layers constitute the architecture of a Convolutional Neural Network.

The parameters of convolutional neural networks consist of learnable filters that extend to the depth. The filter is convoluted to the width and height, which produces a 2-dimensional activation map in the forward pass. The network learns the filter and activates itself when a relevant feature is present in the input.

Three features that control the size of the output volume of ***Convolution Layer*** are: the depth, stride, that is the distance that a filter moves a spatial arrange, and zero padding, which is padding zeros to the input. The spatial size of the output volume is a function of input volume(W), the field size of the convolutional layer (F), the stride(S) and zero padding(P).

$$Output\ volume\ dimensions = \frac{(W - F + 2P)}{S + 1}$$

The important thing to keep in mind that if the output volume dimension is not an integer then the stride used is incorrect.

The ***Pooling Layer***, is responsible for down sampling, that is reducing the spatial size which results in reduced parameters and hence less computation. This also controls the overfitting. The layer operates on every depth slice and resizes it using a MAX operation. It requires two hyper parameters, the field size of the convolutional layer (F) and stride(S) which produces the volume W2,H2,D2 which are:

$$W_2 = \frac{(W - F)}{S + 1}$$

$$H_2 = \frac{(H - F)}{S + 1}$$

$$D_2 = D(\text{Depth of the volume})$$

ReLu Layer, called the Rectified Linear Unit, even though is addressed as layer but it technically is the activation map. It computes:

$$f(x) = max(0, x)$$

A ***Fully Connected Layer*** have connections to all activations in the previous layer, which can be computed by matrix multiplication followed by a bias offset.

Implementation

The emotion recognition system is developed in Python3.6 using Tensorflow and OpenCV. The performance of the trained model depends highly on the available image dataset. A number of datasets are available for image classification tasks with images ranging from that of a few high-resolution to several thousand low-resolution images. The FERC-2013 dataset contains 28,709 examples of 48x48 pixel grayscale images of faces. The images consist of centred faces occupying approximately same amount of space in each image. The images are classified into seven different expressions (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprised, 6=Neutral).

Figure 1- Samples from FERC-2013

The CNN is programmed using TFLearn library together with Tensorflow on Python3.6. The library reduced the complexity of the code by allowing to create neural layers instead of individual neurons. Tensorflow also makes it easy to train and reuse models and the documentation available for Tensorflow makes it easy to understand the implementation of individual layers.

The network architecture is based on the work of Gudi [1] and Correa [2] which also uses the FERC-2013 dataset to recognise 7 emotions. The input layer of the network is of size 48x48 similar to the dimensions of the input data. The network consists of three convolutional layers, a local contrast normalization layer, two max-pooling layers and one fully connected layer.

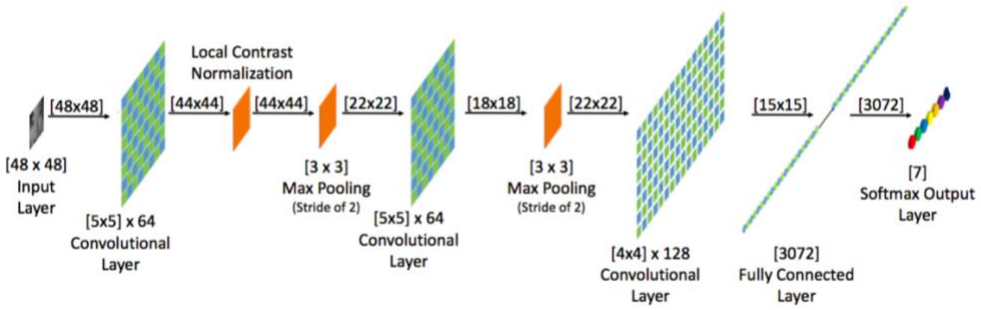

Figure 2- Network Architecture

The original network decreases the learning rate linearly. For this implementation, the learning rate with momentum optimiser is found to be converging faster. We have experimented with different optimisers namely, RMSprop, Adam and Momentum and the best results have been obtained with Momentum optimiser.

Optimizers and Activation Functions:

An important part of any neural network is backpropagation. It is used as a gradient descent, in order to find a global minimum of the loss function in a hyper-parametric space. The loss function used in this study was categorical cross entropy. There were three different

optimizers used in this implementation. Momentum, RMSProp, and ADAM. The way these work is by using past steps of gradient descent to influence how far the next step should travel. When a step is made, and backpropagation performed, for each item in the training set, it is stochastic gradient descent. Our implementation used mini-batch gradient descent with a batch size of 50, meaning that after 50 samples the backpropagation was performed and the step made. Momentum operates by calculating a velocity from the steps. For example, for a weight w if it is updating through back propagation and are set to minimize loss and training time.

$$V_{dw} = \beta V_{dw} + (1 - \beta) V_{dw}$$

$$w := w - \alpha V_{dw}$$

Where dw is the derivative of w. RMSprop, meanwhile uses the square root of a function on the derivative of w.

$$S_{dw} = \beta S_{dw} + (1 - \beta) dw^2$$

$$w := w - \alpha \frac{dw}{\sqrt{S_{dw} + \epsilon}}$$

The activation function serves to constrain the outputs of neurons and allow for backpropagation to take place via derivation. A sigmoid function is a good example of this, however sigmoid function was not used in this implementation; rather ReLU and Softmax. ReLU has the advantage of being fast to calculate, as it is linear and requires no exponential functions.

OpenCV offers a variety of libraries to perform object and facial recognition. For our project, we used it Harr Cascade Classifiers. The approach behind a cascade function is to train it using a set of positive (images with faces) and negative images (images without faces), which can be used to detect features on faces. To extract features from an image, windows of different sizes are placed on different locations on the image, as shown in Figure-3. From the features extracted, most features are considered irrelevant. The process to eliminate irrelevant features is done using the Adaboost technique, this technique selects only the features that potentially improve the accuracy of the classification. Even though Harr computationally complex, but offers higher accuracy in facial recognition. OpenCV provides a detector and trainer for facial recognition.

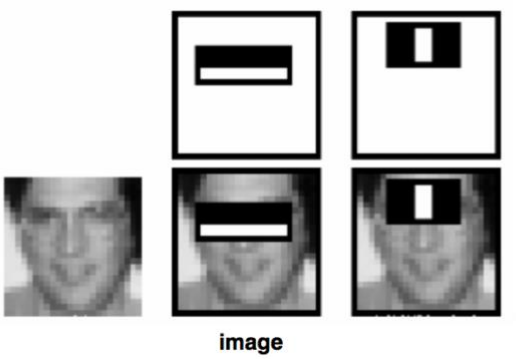

Figure 3- Haar Cascade Windows for facial recognition.

Results:

The network is trained with 11214 training and 2804 testing samples for 100 epochs. The performance with Momentum optimizer was first found and is given in the below figure with input on the vertical axis and the result on the horizontal axis. According to the performance matrix, higher accuracy rates are detected for happy (95%), neutral (88%) and surprised (95%).

The overall accuracy of the model is found to be 86%. The network is tested with emotions detection on using real time video capture with OpenCV. OpenCV tracks the biggest appearing face on the live video and feeds the data to the network from where the output layer values are obtained. The highest value output is assumed to be the current emotion of the user. Screenshots for some emotions are presented in the appendix.

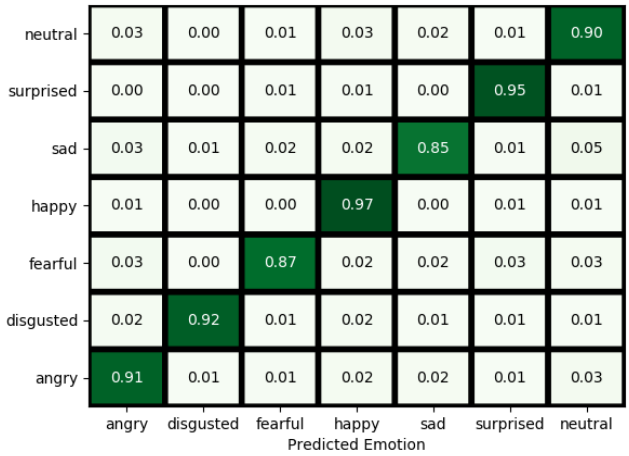

Figure 4 - Performance matrix with Momentum

When comparing the three optimizers, it was found that RMSProp and ADAM achieved a better accuracy over a lesser number of epochs.

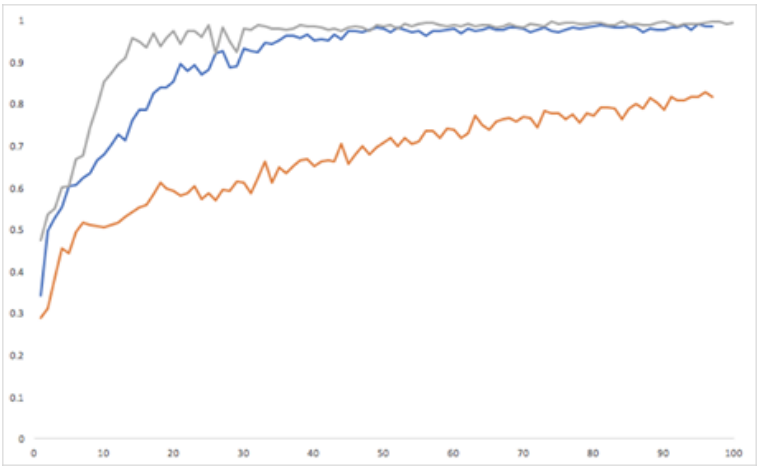

Figure 5- Accuracy vs Epochs graph with different optimizers. Grey(Adam), Blue(RMSprop), Orange(Momentum)

Future Work

In future our team would like to experiment with more changes to the stack of CNN layers. Also, we would like to test the model on a wider range of face photo datasets and add a condition for not finding a face in the image. Currently the implementation can also detect multiple faces but is only able to evaluate the expression of one with the largest size in the feed. A future implementation might attempt to maximise the accuracy of detecting several faces at once. This would be useful for obtaining analytics for customer satisfaction in commercial settings such as retail. It would also enable the model to be used to automatically take group photos when, say, the entire group is all smiling. Furthermore, some form of smoothing might want to be introduced, as currently the model's prediction can jitter between emotions especially during the subjects' transition between expressions or brief motion-based blurring. This would further help in making the tool be more practically applicable in the customer satisfaction case, but also be essential for applications such as "taking photos when the on-screen subject smiles".

Bibliography

[1] A. Gudi, "Recognizing Semantic Features in Faces using Deep Learning," University of Amsterdam, Amsterdam, 2014.

[2] A. J. M. O. Enri Correa, "Emotion Recognition using Deep Convolutional Neural Networks," 2016.

[3] S. I. V. V. A. A. Christian Szegedy, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," Cornell University, 2016.

Appendix

Link to files -
https://drive.google.com/open?id=1tsO3sWSDCYJqKmM7wBy3XNcRS3q1ZVwQ