CPSC 8810 - Deep Learning: Midterm Project

Yadnyesh Y. Luktuke C96091890

Eshaa D. Sood C16170243

March 15, 2019

1 Introduction

The task for this project was to design and implement a deep neural network system for classifying different categories of images. The most important part of a system that can classify images or in fact any particular data is the availability of good features. Good features are said to be representative of the given data, if they allow a model to correctly distinguish between different categories of images. Deep neural networks are a special category of models that learn a representative set of features from the given data. This ability to learn the best set of features for a given task allow deep neural networks to achieve better accuracy than most models that rely on hand - made features which may not be very representative of the data. For this project, we were given a set of bullying images which were separated into the following 9 categories; gossiping, isolation, laughing, pulling hair, punching, quarrelling, slapping, stabbing and strangling. The goal of the project was thus to study the design and implementation of deep neural networks, and build a system that could successfully identify an image as belonging to either of the categories mentioned above. Additionally it was important to classify an external image that did not belong to any category mentioned above as a non - bullying image.

2 Methods

2.1 Pre - Processing

The first part of any image classification system is image pre - processing. This is because input images often originate from different sources, have different sizes and carry different information, such as greyscale and colour images. Hence it is important to standardize each input image to a neural network system. For this project we resized each input image to the size 224×224 as suggested in [2]. If an image was originally provided as a greyscale, the given information was copied twice and the image size appended to three channels, with each channel containing identical information. This was done to keep the input to the model consistent as a three channel input. Colour images contain three channels of information, with each channel containing information of the Red, Green and Blue components of each image pixel respectively. The images that contained an additional channel of information were treated as outliers and removed. Once the image size and channels have been standardized, it is important to ensure that the actual information in the

pixels themselves is standardized. This is done to allow the neural network to process the information correctly, and in turn to improve the accuracy of the final model. Standard scaling was used to transform the image data from raw pixels to the required input to the neural network. It can be represented as follows:

$$x_{transformed} = \frac{(x_{original} - \mu_{channel})}{\sigma_{channel}} \tag{1}$$

Here $x_{original}$ represents the pixel in each channel of the image, while $\mu_{channel}$ and $\sigma_{channel}$ represent the mean value and standard deviation of each channel respectively.

2.2 Network Architecture

Our network was inspired by the Convolutional Networks developed in [1] and [2]. It also borrowed from the concept of dimensionality reduction using Principal Component Analysis as explained in [6]. The concept was to continuously reduce the number of features at each stage, till the most representative features were retained by the network. This is done by using convolutional and max - pooling layers as explained in [1]. The convolutional layer consists of a stack of weights or filters that become sensitive to local features within the image, following this max - pooling retains only the strongest responses after the image has been filtered using each filter. Since the number of filters is often greater than the number of input channels, this operation transforms the input image into an image having greater number of channels (depth). The max - pooling layer reduces the size of output image by a factor of 2 along each dimension. It must be noted that this is because the value for the parameter 'stride' chosen was $[1, 2, 2, 1]$. The enthusiastic reader is encouraged to try different values of the middle parameters in order to get different filtered image sizes. The outer parameters need to be set to 1 in order to comply with the internal workings of TensorFlow. Another parameter which can be tuned for the convolutional network is the size of the receptive field or window, [1], [2].

For this project this value was set to 5 for the first convolutional layer and reduced to 3 for the successive convolutional layers. Each convolutional layer was followed by a max - pooling layer. Chaining three such convolutional and max - pooling stacks, results in a filtered image having size 28×28 and as many channels as specified by the depth of the final convolutional layer. This resulting image is flattened and used as input to the fully connected layers. Our model had two such layers having 200 and 100 neurons respectively. The final layer is known as the output layer, and it consists of neurons equal to the number of categories being predicted. This is due to the use of one - hot encoding [1], which converts the discrete label given for each image into a codeword containing a 1 in the respective index of the new label and 0 elsewhere. The advantage of this strategy is explained a little later in this report. The final network is shown in Table 1. Biases of size equal to the last dimension of each layer were added to each respective layer shown in Table 1. It must be noted that the size of the input to each subsequent convolutional layer reduces due to the max - pooling layer preceding it. Also, the number of neurons in the output layer depends on the classification type. For this project, we designed a network containing 10 neurons in its output layer, equal to the number of categories being predicted.

Layer	Size	Input Image Size
Convolutional 1	[5, 5, 3, Depth 1]	[224, 224, 3]
Convolutional 2	[3, 3, Depth 1, Depth 2]	[112, 112, Depth 1]
Convolutional 3	[3, 3, Depth 2, Depth 3]	[56, 56, Depth 2]
Fully Connected Layer 1	[28 $\times$ 28$\times$ Depth 3, 200]	[28 $\times$ 28, Depth 3]
Fully Connected Layer 2	[200, 100]	[200]
Output Layer	[100, 10]	[100]

Table 1: Network Architecture

2.3 Training

2.3.1 Mini - Batch Size

Training a neural network consists of feeding it the input data in a fixed set of subsequent samples also known as batches or mini - batches [1]. Such a strategy allows the network to update its parameters in response to a small set of samples, which speeds up the learning process. Choosing batch size as a multiple of 2 also allows parallelization of the data on systems using GPU's [7].

2.3.2 Learning Rate

The learning rate (η), controls the actual weight updates in the weight update equations. Choosing a value close to 1.0 results in faster updates, but causes the weights to oscillate about their mean value. On the contrary choosing a very small value results in very slow weight updates and slower convergence of the network parameters in general.

2.3.3 Dropout Probability

Dropout is the process of regularizing a neural network. During training a random number of neurons, determined by the dropout probability (p_{keep}) are turned off to prevent network parameters from co - adapting. For this project dropout was applied to only the fully connected layer weights. Since the number of parameters in the convolutional layer weights is small as compared to those in the fully connected layers, such a strategy prevents the useful weights in the convolutional layers from being discarded [1].

2.3.4 Choosing Network Architecture

Perhaps the most important part of the training process is choosing the parameters Depth 1, Depth 2, Depth 3 of the convolutional layers and the activation functions in each layer. For this project the depth of each successive convolutional layer were set as 10, 8 and 6. Hence the number of neurons in the fully connected layers were fixed to high values to allow the network to learn complex features that were produced as the output of the final convolutional layer. All neural network layers except the output layer had a Rectified Linear Unit activation function at their output, in order to allow us to design a network of sufficient depth [1]. The output of the output layer was fed through a

softmax activation function to the further routines.

2.3.5 Number of non - bullying images

The Stanford 40 Actions dataset [3] was chosen for the non - bullying images. It contains 9532 images of people performing 40 different actions, none of which were associated with the bullying actions provided earlier. During each training session, a random number of images was selected from this dataset (after shuffling), and appended to the dataset that already contained the bullying images. The number of images chosen was kept as an open parameter, that could be selected uniquely for every model.

2.4 Other Architectures

As our initial start point, we experimented with training a network similar to the VGG13 network as given in the [2]. However it did not result in good accuracy. Changing parameters such as the number of convolutional filters in each layer, adapting parameters such as the learning rate, p_{keep}, as well as using methods such as batch normalization [1] did not help to improve the accuracy of the model. Hence this approach was not considered any further.

2.5 Model Evaluation

In order to measure the performance of the network, the chosen loss function was the cross entropy between the predicted labels and actual label of the image. Here we come back to the number of neurons in the output layer as mentioned in Sub - Section 2.2, choosing the activation function of the output layer as softmax enables the network to predict the probability of the image belonging to each class. When the one hot encoded true label of the image is also treated as a probability, these vectors can be compared through the use of cross entropy. This is consistent with the networks mentioned in [1]. In order to assess the ability of the network to identify non - bullying images, we calculated the True Negative Rate of the classification given by the following equation:

$$TrueNegativeRate = \frac{TrueNegative}{TrueNegative + FalsePositive} \tag{2}$$

3 Results

In order to evaluate the network performance, and to ensure that the network was not overfitting to the training data provided, the dataset was randomly sampled into training and validation data. We experimented with different mini - batch sizes such as 32, 64 and 128. Choosing 32 made the training iterations run faster, but there was a noticeable decrease in the training and validation accuracy for each epoch. On the other hand, choosing 64 and 128 gave a network with higher accuracy for the training and validation data sets. Since there was no difference in the accuracy of the network trained with a batch size of 64 and 128, we chose the mini - batch size as 64 to speed up the training process. The values of η and p_{keep} were set to 0.001 and 0.7 respectively. The number of non - bullying images selected was varied from the following set of values [10, 20, 100, 200, 500, 1000, 2408] with the last number being equal to the

Non-Bullying	Gossiping	Isolation	Laughing	Pulling Hair	Punching	Quarrel	Slapping	Stabbing	Strangle
937	0	1	0	0	1	0	0	0	1
13	409	0	0	0	0	0	0	0	0
4	0	250	0	0	0	0	0	0	0
1	0	0	157	0	0	0	0	0	0
3	0	0	0	261	0	0	0	0	0
15	0	0	0	0	365	0	0	0	0
2	0	0	0	0	0	250	0	0	0
11	0	0	0	0	0	0	210	0	0
2	0	0	0	0	0	0	0	141	0
12	0	0	0	0	0	0	0	0	358

Table 2: Cross Confusion Matrix for All Categories

number of bullying images in the training dataset. It was observed that choosing an equal number of non - bullying and bullying images resulted in the best network performance. Figure 1 displays the accuracy obtained during the training of the neural network for the categorical classification task. It can be seen that the network achieved good accuracy within a few epochs. This is a surprising result, since the complexity of the network was low as compared to the more complex networks in [1] and [2]. It is thought that the variability of the images in the dataset is small, which allowed the network to learn a good representative set of features for most images within the dataset.

Table 2 displays the cross confusion matrix obtained after evaluating the network on the training set images and 1000 images randomly selected from the Stanford 40 actions dataset [3]. It can be seen that the network is able to categorize the images very well. Particularly note that a large number of non - bullying images were correctly classified. Hence the True Negative Rate as calculated by Equation 2 is 93.7 %.

4 Conclusion

For this project we designed and built a deep neural network using TensorFlow for the purpose of identifying images as either bullying or non bullying. Further each image was classified as belonging to a specific category of bullying. The results obtained for this experiment suggest that the designed network needs to be made more complex either through the use of more layers or different layers, which could possibly improve the performance of the network. However an important part of training a more complex network is the availability of more data. Hence data augmentation will be needed in the next stage of this project. Additionally we also need better architectures in order to predict the victim and the culprit associated in each bullying action image. It is thought that a neural network using an Attention Model will be useful in this task.

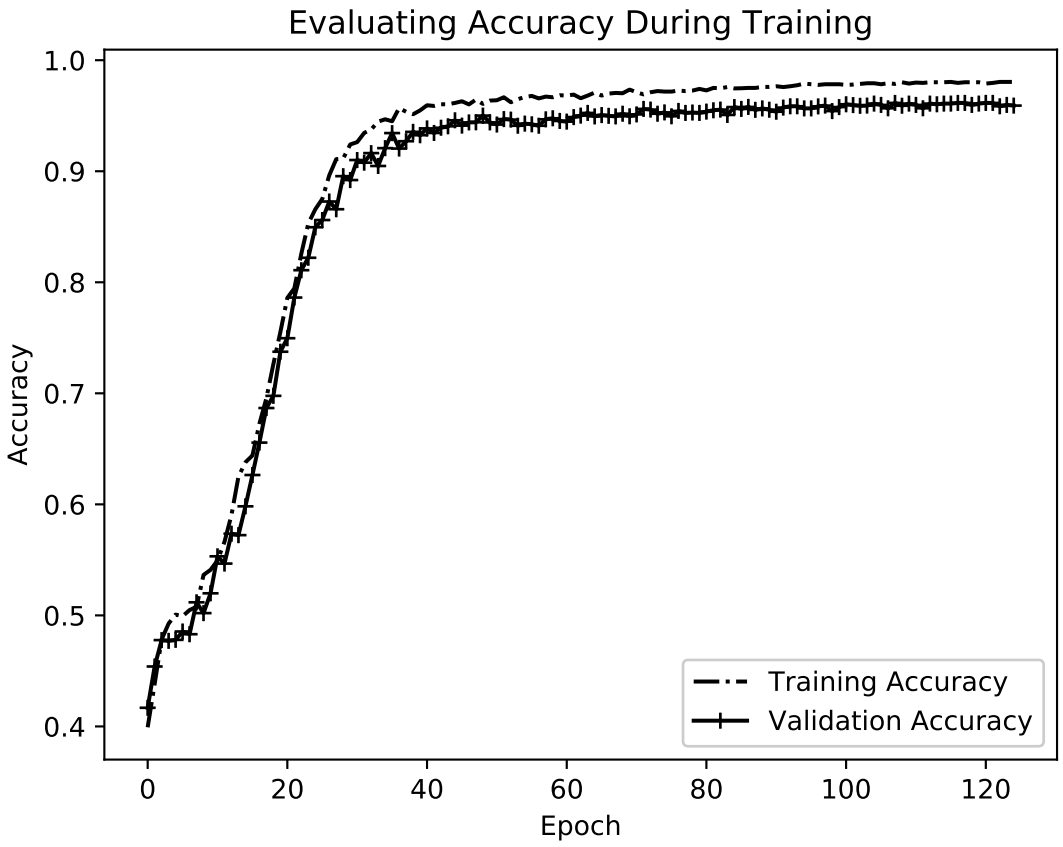

Figure 1: Accuracy during training: Categorical Classification

References

[1] Google Cloud Platform, "Tensorflow and deep learning, without a PhD", available at "https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#0", Online; Accessed Jan 20, 2019

[2] K.Simoyan, A.Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", available at "http://arxiv.org/abs/1409.1556", Online; Accessed Feb 28, 2019

[3] B. Yao, X. Jiang, A. Khosla, A.L. Lin, L.J. Guibas, and L. Fei-Fei, "Human Action Recognition by Learning Bases of Action Attributes and Parts", International Conference on Computer Vision (ICCV), Barcelona, Spain. November 6-13, 2011.

[4] S. van der Walt, S. Chris Colbert and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation", Computing in Science & Engineering, 13, 22-30 (2011).

[5] A.Clark and contributors, F.Lundh and contributors, "Pillow and Python Imaging Library (PIL)", available at https://pillow.readthedocs.io/en/stable/index.html, Online; Accessed Feb 28, 2019

[6] S.Theodoridis, K. Koutroumbas, "Pattern Recognition, Fourth Edition", Academic Press, Inc. 2008

[7] A. Ng, "Mini Batch Gradient Descent", available at "https://www.youtube.com/watch?v=4qJaSmvhxi8", Online; Accessed Jan 20, 2019