CPSC 8810 - Deep Learning
Deep Learning Model to Detect CyberBully Actions in Images

Submitted By:
Vivek Koodli Udupa - (C12768888)
Shashi Shivaraju - (C88650674)

Clemson University
March 4, 2019

Abstract

This report explains the process involved in a Convolutional Neural Network to detect and classify various cyberbully actions in an image.

1 Introduction

This report considers the problem of detection and calssification of cyberbully actions caputired in an image using Deep learning models.

Cyberbullying is bullying that takes place over digital devices like cell phones, computers, and tablets. Cyberbullying can occur through SMS, Text, and apps, or online in social media, forums, or gaming where people can view, participate in, or share content. Cyberbullying includes sending, posting, or sharing negative, harmful, false, or mean content about someone else. Some cyberbullying crosses the line into unlawful or criminal behavior. With the prevalence of social media and digital forums, comments, photos, posts, and content shared by individuals can often be viewed by strangers as well as acquaintances. The content an individual shares online – both their personal content as well as any negative, mean, or hurtful content – creates a kind of permanent public record of their views, activities, and behavior. This public record can be thought of as an online reputation, which may be accessible to schools, employers, colleges, clubs, and others who may be researching an individual now or in the future. Cyberbullying can harm the online reputations of everyone involved – not just the person being bullied, but those doing the bullying or participating in it.[1]

Deep learning is a subset of machine learning where artificial neural networks, algorithms inspired by the human brain, learn from large amounts of data. The term 'deep learning' because the neural networks have various (deep) layers that enable learning. Deep learning allows machines to solve complex problems even when using a data set that is very diverse, unstructured and inter-connected. The more deep learning algorithms learn, the better they perform. [2]

This report describes modeling of a deep learning network based on the concept of convolutional neural network for detecting and classifying cyberbully actions for a given image. The cyberbullying actions considered in this project are laughing, pulling-hair, quarrel, slapping, punching, stabbing, gossiping, strangle and isolation. The designed model is trained using the provided image dataset which contain above mentioned 9 categories of cyberbully actions in them.

2 Methods

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery.[3] A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, ReLU layer i.e. activation function, pooling layers, fully connected layers and normalization layers.[4]

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

The implementation of our model based on CNN using an open source machine learning library PyTorch is described in the section below.

2.1 Implementation of CNN Model

Our model is implemented using the following layers:

1. **Convolutional Layer:** The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels). During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input.

2. **ReLU layer:** ReLU is the abbreviation of rectified linear unit, which applies the non-saturating activation function
$$f(x) = max(0, x) \tag{1}$$
It effectively removes negative values from an activation map by setting them to zero. It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer.

3. **Max Pooling:** Another important concept of CNNs is pooling, which is a form of non-linear down-sampling. Max pooling is the most common non-linear function for down-sampling. It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum.

4. **Fully Connected Layer:** Finally, after several convolutional and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular (non-convolutional) artificial neural networks. Their activations can thus be computed as an affine transformation, with matrix multiplication followed by a bias offset.

5. **Dropout Layer:** A single model can be used to simulate having a large number of different network architectures by randomly dropping out nodes during training. This is called dropout and offers a very computationally cheap and remarkably effective regularization method to reduce over fitting and generalization error in deep neural networks.[5]

The implementation details of our model is as follows:

1. **Image Pre-Processing:** The given input image is converted into mono-channeled, gray scale image of size 256 x 256. Then it is converted to a PyTorch tensor image and its values are normalized with a mean of 0.5 and Standard Deviation of 0.5.

2. **Convolution Layer 1:** The input to this layer is a preprocessed tensor image from the previous layer. This layer performs 2D convolution using a 3x3 kernel with stride set to 1 and padding enabled to produce an output which is a 16 channel feature map.

3. **ReLU Layer 1:** This layer applies a relu activation function to the 16 channel feature map.

4. **Max Pooling Layer 1:** This layer down-samples the 256 x 256 16 channel feature map to 128 x 128 16 channel feature map.

5. **Convolution Layer 2:** The input to this layer is the 16 channel 128 x 128 feature map from the previous layer. This layer performs 2D convolution using a 3 x 3 kernel with stride set to 1 and padding enabled to produce an output which is a 32 channel feature map.

6. **ReLU Layer 2:** This layer applies a relu activation function to the 32 channel feature map.

7. **Max Pooling Layer 2:** This layer down-samples the 128 x 128 32 channel feature map to 64 x 64 32 channel feature map.

8. **Flattening Layer:** This layer flattens the 2D feature map to 1D feature map.

9. **Dropout Layer:** This layer randomly zeros some of the element of the input tensor with probability 0.4.

10. **Fully Connected Layer 1:** This layer maps the feature map into 100 neurons.

11. **ReLU Layer 3:** This layer applies a relu activation function to the output of Fully Connected Layer 1.

12. **Fully Connected Layer 2:** This layer maps 100 neurons into 10 categories of classification.

2.2 Training the CNN Model

To train a deep learning model, the following parameters are considered:

1. **Epoch:** An epoch describes the number of times the algorithm sees the entire data set. So, each time the algorithm has seen all samples in the dataset, an epoch has completed.

2. **Batch Size:** The total number of training examples present in a single batch, wherein a batch is a subset of the entire data set.

3. **Iteration:** The number of batches needed to complete one Epoch.
4. **Learning Rate:** The learning rate or step size in machine learning is hyper-parameter which determines to what extent newly acquired information overrides old information.

The implemented model is trained using the below mentioned configuration:

1. Epoch = 50
2. Batch Size = 1
3. Learning Rate = 0.001

The model is trained with the given training dataset as per the below mentioned algorithm:

1. Initialize the CNN model with default parameters.
2. Create an instance of Adam optimizer for setting the learning rate
3. Create an instance of cross entropy loss
4. Initialize the optimizer with zero gradients
5. Feed a training input image from the current batch to the model to perform forward propagation
6. After the completion of forward propagation, calculate the cross entropy loss
7. Perform back propagation to minimize the loss
8. Update gradients
9. Iterate through step 4 for all the batches in the training dataset
10. Repeat the above steps for the given number of epochs
11. Save the trained model for testing purpose

Please refer the appendix for the python implementation of the above described model.

3 Results

The implemented model was trained using a learning rate of 0.001

4 References

[1] https://www.stopbullying.gov/cyberbullying/what-is-it/index.html

[2] https://www.forbes.com/sites/bernardmarr/2018/10/01/what-is-deep-learning-ai-a-simple-guide-with-8-practical-examples/#434cffaa8d4b

[3] https://en.wikipedia.org/wiki/Convolutional_neural_network#Convolutional

[4] https://cs231n.github.io/convolutional-networks/

[5] https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/

5 Appendix