Machine Learning Assignment 2 Report

Names and Student Numbers

Adam Gordon – 2275253
Jaden Harris – 2169997
Zagesh Parshotam – 1845347
Que Sera Subramoney – 1611459

Algorithm/Model used and why

The model we have decided to use to classify the data is a Convolutional Neural Network (CNN). After assessing the input data, it became clear that the data was, in short, a set of images represented by numerous triplets of values that signified the RGB values of given pixels. These pixels, in turn, create images. We then reshaped the data into 28x28x3 matrices, we discovered that this was the correct dimensions as dividing the total 2352 values by 3 (as each pixel is made up of 3 RGB values) results in 784, we then square root 784 which leaves us with a 28x28x3 matrix where each value in the matrix represents a pixel made up of an RBG value(3 values), we then proceeded to convert the data from RGB to grayscale reducing to a 28x28x1 matrix to be fed into the network. After assessing the labels of these inputs, we realized that these images were of handwritten numbers between 0-9. Therefore, the task was to classify images into sets of different classes. CNNs are feed forward Neural Networks that are fully connected. They are very effective in the reduction process of parameters without losing the quality of their input. Due to the fact that each pixel, in an image, is considered as a feature, images have high dimensionality and that is why reduction of parameters is needed in image classification, therefore making CNNs powerful image classifiers. In conclusion, CNNs are generally the first choice when building an image classification model. Due to all the information above about CNNs, it made logical sense that we would choose this model for the assignment.

Design of model

We have created a network with 2 convolutional layers, 2 pooling layers, 2 dropout layers, 2 fully connected layers and the RELU as our activation function. The convolutional layer's function, in short, is to transform the

input matrix in order to extract features. This input matrix is in the form of pixel values. The output from the convolutional layer is a feature map. We added and decreased the number of convolutional layers in our network and recorded the accuracy. We found that 2 layers was the best choice in terms of simplicity and accuracy. The pooling layer's function is to reduce the size of the feature map that is generated by the convolutional layer. It isused as it reduces the parameters that need to be learned by the network, therefore reducing computation. We repeated the same process for finding the most optimal number of pooling layers and it was also 2. Following the Convolutional layers and the pooling layers, the next layer is the dropout layer. The dropout layer randomly drops neurons from the neural network during each iteration of training, in order to avoid overfitting. We found 2 dropout layers to be optimal. Forming the last few layers of our network are the fully connected layers. The fully connected layers are simple feed forward neural networks that take in the output of the pooling layers as their input. Repeating the same process for how many layers to use and the outcome was 2 layers. For our parameters of our convolutional layers, we have chosen to use a combination of a kernel size of 3, a padding of 1 and a stride of 1. This combination of parameters allows for the dimension of our input matrix to be the same as the dimension of our output matrix. This combination of parameters is called a same convolution. We felt it was logical and sensible to use a same convolution in our network. The RELU activation function brings non-linearity to the network and does not saturate, meaning it does not have a maximum value as it will return any positive input it is given with no upper positive limit. In addition, this type of return is exactly the type of output we need for these specific class labels as our classes are all positive values, ranging from 0-9. Therefore, the RELU activation function was the clear choice for our model.

We have chosen a learning rate of 0.01, batch training data of size 1000, batch testing data of 1000, the log interval of 10, 200 epochs of training and a momentum of 0.9 for the SGD optimizer as our hyperparameters. We chose to use the SGD (stochastic gradient descent) optimizer as after testing, it performed better than the other optimizers we implemented like the Adam optimizer. The Adam optimizer converged faster but over time the SGD optimizer retuned a higher accuracy when training. We used a momentum of 0.9 with the SGD optimizer as this led to faster convergence by accelerating the gradient vectors in the correct directions.

Tips/Tricks

After realizing that our data shared great similarity with the MNIST handwritten digit classification dataset, we heavily relied on the resources surrounding the famous image classification problem to aid us in constructing our own model. The resources were vast and helpful. In addition to training our model on the training data that we split from the given input data for this assignment, we also trained our model on the MNIST dataset and then tested it on the split test data. This additional training on the MNIST dataset heavily boosted the accuracy of our model. Realizing the similarity between the two and using the resources and datasets out there, was definitely a helpful trick for designing a very accurate model.

Expected Accuracy

94%

References

https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

https://nextjournal.com/gkoehler/pytorch-mnist