COMP-551 Mini Project 4

Kaicheng Wu
kaicheng.wu@mail.mcgill.ca
Mathieu-Joseph Magri
mathieu-joseph.magri@mail.mcgill.ca
Mohammad Sami Nur Islam
mohammad.sami.islam@mail.mcgill.ca

School of Computer Science
McGill University
Canada
April 26th 2023

Reproducibility Summary

Scope of Reproducibility

The main claims of the author are as follows: compared to using a top softmax layer, an SVM top layer increases accuracy on the MNIST dataset, on the CIFAR-10 dataset, and on the ICML2013 workshop recognizing faces dataset, therefore demonstrating greater performance on all models used in various classification tasks.

Methodology

We implemented all models as per the instructions of the original author. Google Colab was used as well, and total runtime to run all experiments was about 10 hours.

Results

For the first claim, in our implementation, an accuracy of 95.38% was obtained when using a Softmax layer while an accuracy of 95.64% was obtained when using an SVM. SVM does indeed outperform the Softmax layer like claimed in the original paper. For the second claim, in our implementation, an accuracy of 81.26% was obtained when using a Softmax layer while an accuracy of 70.45% was obtained when using an SVM. The claim which was obtained in the original paper was sadly not obtained. For the third claim,in our implementation, an accuracy of 30.04% was obtained when using a Softmax layer while an accuracy of 50.82% was obtained when using an SVM. As we can see, SVM does indeed outperform the Softmax implementation.

What was easy

The author's paper is generally clear and comprehensible, making it easy to follow and implement with commonly used libraries like tensorflow or pytorch. In the instance of CIFAR-10, the author provided specific details such as the number of layers, hidden units, and dropout rates, which facilitated the organization of the algorithm. The facial recognition algorithm also followed a similar format. Similarly, for the MNIST algorithm, we were given crucial information such as the learning rate, batch sizes, and number of epochs.

What was difficult

To acquire the original data-sets for the facial recognition algorithms, we encountered several obstacles. The data-sets were only accessible through the ICML 2013 challenges, which proved to be a challenging task. Furthermore, we faced difficulties executing the code provided by the author, as the algorithm was written in CUDA and C++, and none of us had access to an NVIDIA GPU. Consequently, we opted to use COLAB, which posed its own set of difficulties owing to some conflicts with Matlab.

Communication with original author

No communication was done with the original author.

1 Introduction

Convolutional and fully-connected neural networks have been used and trained on many different sort of computational tasks with flying results. These tasks include image classification, speech recognition, bio-informatics and natural language processing. In many of the classification tasks, a majority of the previously mentioned models employ the usage of a softmax activation function for predicting and minimizing cross-entropy loss.

In the paper we are reproducing, the author demonstrates that using a support vector machine (SVM) instead can bring many benefits in many of these different tasks. Indeed, the author affirms that using linear SVMs instead of a softmax function can give great gains on commonly used deep learning datasets such as MNIST, CIFAR-10, and the ICML 2013 Representation Learning Workshop's face expression recognition challenge.

2 Scope of reproducibility

Over the many years of model experimentation, the softmax function was deemed to be a very good and high performing activation function. In this analysed paper, the author goes ahead and prove that, although the softmax function is indeed quite accurate for many classification tasks, using a support vector machine induces better performance for the same classification tasks. The author shows that for some architectures, using a linear SVM top layer can be greatly beneficial instead of using a softmax top layer. The author optimizes the primal problem of the SVM and the gradients can then be backpropagated to learn lower level features. The main claims on the different classification tasks attempted by the author are as follows:

- Compared to using a top softmax layer, an SVM top layer increases accuracy on the MNIST dataset, therefore demonstrating greater performance.
- Compared to using a top softmax layer, an SVM top layer increases accuracy on the CIFAR-10 dataset, therefore demonstrating greater performance.
- Compared to using a top softmax layer, an SVM top layer increases accuracy on the ICML2013 workshop recognizing faces dataset, therefore demonstrating greater performance.

Furthermore, the author mentions that optimization was done using stochastic gradient descent on small minibatches, and that they believe that the performance gain by using a SVM top layer is due to the superior regularization effects of the SVM loss function, rather than an advantage from better parameter optimization.

3 Methodology

A mix of the author's code as well as re-implementing a few parts of the code was the selected approach. Indeed, when wanting to emulate the model used for the facial recognition dataset, it proved to be rather challenging since the original algorithm was written in CUDA and C++, and no group members had access to an NVIDIA GPU to then run the code. In consequence, Google Colab was used, which posed its own set of difficulties owing to some conflicts with Matlab. Finally, we decided to re-implement the entire algorithm using tensorflow on Colab.

For the CIFAR-10 dataset and the MNIST dataset, common libraries such as tensorflow and pytorch were used. For the CIFAR-10 dataset, specific details such as the number of layers, hidden units, and dropout rates were already provided by the author, therefore those were used for our results as well. The same logic can be applied to the MNIST dataset, where the author of the original dataset mention the used learning rate, batch sizes, and number of epochs, aiding us in our implementation.

For all runs, the run time in Google Colab was changed to GPU allowing a more realistic and quick approach to running the code.

3.1 Model descriptions

As per the instructions of the original author, we implemented all models.

For the facial recognition dataset, we created two convolutional neural networks - one with SVN and the other with softmax. Both models consisted of an image mirroring layer, similarity transformation layer, two convolutional filtering+pooling stages, and a fully connected layer with 3072 hidden units of the rectified linear type.

For the CIFAR-10 dataset, we followed a similar approach and designed standard CNNs. The first C layer had 32 5×5 filters with Relu hidden units, and the second C layer had 64 5×5 filters. The pooling layers used max pooling and downsized by a factor of 2. The penultimate layer had 3072 hidden nodes with Relu activation and a dropout rate of 0.2.

Lastly, for the MNIST dataset, we employed a simple fully connected model by performing PCA from 784 dimensions to 70 dimensions. Two hidden layers of 512 units each were followed by a softmax or an L2-SVM. The data was divided into 300 minibatches of 200 samples, and we trained using stochastic gradient descent with momentum for over 400 epochs, totaling 120K weight updates. We linearly decayed the learning rate from 0.1 to 0.0 and set the L2 weight cost on the softmax layer to 0.001.

Finally, none of the used models were pretrained, as all training was done on our end.

3.2 Datasets

A total of three different datasets were used: the MNIST dataset, the CIFAR-10 dataset and the ICML2013 workshop recognizing faces dataset. The MNIST dataset is a handwritten digit classification dataset. It is a 10 class problem (digits from 0 to 9). It contains 60 000 training examples and 10 000 test examples. Furthermore, in the original paper, Gaussian noise was added to the input to prevent overfitting and also happened to be critical in achieving good results.

The CIFAR-10 dataset (Canadian Institute For Advanced Research 10) is a 10 class object dataset with 50 000 images for training while it has 10 000 images that are reserved for testing purposes. The images used are coloured and are 32 x 32 in resolution. In the original paper, jitter and horizontal reflection was applied to the data randomly before the weight was updated using a minibatch of size 128.

For the ICML2013 workshop recognizing faces dataset, the data consists of 28 709 images in 48x48 format. Each image can be a face under 7 different types of facial expression (Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral). The validation set and test set contain the same amount of images, which is 3 589 total images each. Like the two previous sets, this dataset was used as part of a classification task. Finally, the original paper mentions preprocessing the data by subtracting the mean value of each image and then setting the image norm to be 100. Moreover, after that, each pixel is then standardized by removing its mean and dividing its value by the standard deviation of that specific pixel, and this is done across all training images.

The datasets can be downloaded from the following links:

MNIST Dataset

CIFAR-10 Dataset

ICML2013 workshop recognizing faces dataset

3.3 Hyperparameters

All hyper-parameters germane to the experiments we conducted were furnished in the author's original paper. Given the output's gratifying quality and apparent optimization, we perceived no necessity to engage in further hyper-parameter experimentation beyond a handful of manual trials with alternative values. Ultimately, we elected to adhere to all hyper-parameters utilized in the original publication.

3.4 Experimental setup and code

The experimental setup for all experiments is quite simple. There are a total of 5 Google Colab documents that were created. The first two Colabs use the CIFAR-10 dataset, where the first of the two implements a model with Softmax, while the second Colab implements a model with SVM. The third model is then uses the MNIST dataset, where both the Softmax and SVM implementations are done on the same file. Finally, the last two models deal with the facial recognition dataset, the first of these models dealing with implementing Softmax while the second implements SVM. Furthermore, all code was run in Colab changing the runtime setting to a GPU runtime which helps with the speed of execution of the code and allows us about 15 GB of RAM.

The specific measure used to evaluate all models and experiments is simply the accuracy value that each model produces, that is the number of good predictions made by the model over the total number of predictions made by the model. This specific measurement was used as it is also very much used in the original paper and accuracy is also a very easily understandable measure for all given its simplicity.

Following is all the links to our developed code:

CIFAR-10 with Softmax

CIFAR-10 with SVM

MNIST Dataset with both Softmax and SVM

Facial Recognition Dataset with Softmax

Facial Recognition Dataset with SVM

3.5 Computational requirements

As all experiments were conducted on Google Colab, no specific hardware is necessary. The entirety of the computation is performed on Google's virtual machine in the cloud, requiring no additional resources such as CPU, GPU, RAM, or storage. The only prerequisite is a reliable internet connection.

4 Results

Regrettably, despite our best effort, our experiments fell short of substantiating the author's assertion that linear SVM invariably outperforms softmax in all experiments, as according to our experiments, the softmax approach does sometimes have a higher accuracy compared to SVM, although SVM trains slightly faster.

4.1 Results reproducing original paper

4.1.1 Result 1

For the first experiment, we needed to determine whether or not the runs we committed follow the following claim: compared to using a top Softmax layer, an SVM top layer increases accuracy on the MNIST dataset, therefore demonstrating greater performance. In the original paper, an accuracy of 99.01% was obtained when using a Softmax layer while an accuracy of 99.13% was obtained when using an SVM. In our implementation, an accuracy of 95.38% was obtained when using a Softmax layer while an accuracy of 95.64% was obtained when using an SVM. As we can see, although the difference in accuracy is extremely minimal, SVM does indeed outperform the Softmax layer like claimed in the original paper.

4.1.2 Result 2

For the second experiment, we needed to determine whether or not the runs we committed follow the following claim: compared to using a top Softmax layer, an SVM top layer increases accuracy on the CIFAR-10 dataset, therefore demonstrating greater performance. In the original paper, an accuracy of 86.00% was obtained when using a Softmax layer while an accuracy of 88.01% was obtained when using an SVM. In our implementation, an accuracy of 81.26% was obtained when using a Softmax layer while an accuracy of 70.45% was obtained when using an SVM. As we can see, the claim which was obtained in the original paper was sadly not obtained following our experiments. SVM, in our case, does not outperform Softmax.

4.1.3 Result 3

For the third experiment, we needed to determine whether or not the runs we committed follow the following claim: compared to using a top softmax layer, an SVM top layer increases accuracy on the ICML2013 workshop recognizing faces dataset, therefore demonstrating greater performance.. In the original paper, an accuracy of 70.10% was obtained when using a Softmax layer while an accuracy of 71.20% was obtained when using an SVM. In our implementation, an accuracy of 30.04% was obtained when using a Softmax layer while an accuracy of 50.82% was obtained when using an SVM. As we can see, the claim which was obtained in the original paper is obtained where SVM does indeed outperform the Softmax implementation.

5 Discussion

All in all, two out of the three claims stated in the original paper were found to be accurate which isn't great nor terrible. According to our experiments, the softmax approach does sometimes have a higher accuracy compared to SVM, although SVM trains slightly faster. The strength of our approach was its simplicity as we very much followed the paper and used Google Colab, a type of software that can be used by everyone in which the only requirement is a solid internet connection. The main weakness in our approach is the fact that we did not have the same hardware specs as those that wrote the original paper which may explain the discrepancy in some of the obtained results. Furthermore, no additional experiments were made, which could have maybe added a bit more to this report.

5.1 What was easy

The author's paper is generally clear and comprehensible, making it easy to follow and implement with commonly used libraries like tensorflow or pytorch. In the instance of CIFAR-10, the author provided specific details such as the number of layers, hidden units, and dropout rates, which facilitated the organization of the algorithm. The facial recognition algorithm also followed a similar format. Similarly, for the MNIST algorithm, we were given crucial information such as the learning rate, batch sizes, and number of epochs, streamlining the training process.

5.2 What was difficult

To acquire the original data-sets for the facial recognition algorithms, we encountered several obstacles. The data-sets were only accessible through the ICML 2013 challenges, which proved to be a challenging task. We combed through numerous resources on Kaggle before finally locating the appropriate dataset.

Furthermore, we faced difficulties executing the code provided by the author, as the algorithm was written in CUDA and C++, and none of us had access to an NVIDIA GPU. Consequently, we opted to use Colab, which posed its own set of difficulties owing to some conflicts with Matlab. Eventually, we decided to re-implement the entire algorithm using tensorflow.

5.3 Communication with original author

Unfortunately, no communication was made or done with the original author of the paper we are reproducing. We also did not send the author our completed report to gain any feedback.

5.4 Statement of Work

All teammates contributed equally to the report. Mohammad Sami and Kaicheng focused on coding while Mathieu-Joseph focused on the report, however all dabled in every section.

References

Dodge, J. (2020, September 22). The Reproducibility Challenge as an Educational Tool.PapersWithCode. https://medium.com/paperswithcode/the-reproducibility-challenge-as-an-educational-tool-cd1596e3716c

Tang, Y. (2015). Deep Learning using Linear Support Vector Machines.https://arxiv.org/pdf/1306.0239.pdf