Reproducibility in ML: COMP 551

Authors
Brandon Park
Yanis Mouazar
Sean Li

Reproducibility Summary

Scope of Reproducibility

The original paper states that most “deep learning” models use the softmax activation function for prediction in the last layer of a convolutional neural network. However, they prove an advantage in replacing the softmax layer with a linear support vector machine instead. They prove their results on popular datasets such as MNIST, CIFAR-10, and ICML 2013 face expression recognition challenge.

Methodology

To load, implement and replicate the fully-connected and convolutional neural network, we had to work within the confines of our experiment. We used a classification implementation of the MNIST dataset using linear SVM with Keras and a Tensorflow backend found under Papers With Code. We modeled our following experiments using this implementation and were able to conduct our experiments with the GPU allocated by Google Colab.

Results

By using the cited code, we were able to reproduce some results that were similar to the original paper. For instance, our reproduced models on the MNIST dataset showed an accuracy of 97 percent with the SVM layer and 96 percent with the softmax layer. Although the paper does not give their accuracy results specifically, they give us their correlated test error results which were 0.87 percent and 0.99 percent with SVM and softmax respectively. What our results couldn't reproduce was the performance on the CIFAR-10 dataset, where the SVM and softmax model achieved an accuracy of 10 percent and 45 percent respectively. The original paper had 14 percent and 11.9 percent test error on softmax and SVM, meaning our results should have a better performance with the SVM layer.

What was easy

For the most parts the models were smoothly re-implemented into our work, running the author's code for implementing SVM and softmax models was relatively easy as the code was well-documented and easy to follow. Re-implementing their method based on the description in the paper was also straightforward as the paper provided clear and detailed explanations of the methodology. Overall, these parts of the reproduction study were easy to apply to their problem.

What was difficult

Since the code was an official implementation, but a community one, it was difficult to reproduce the same conditions that they had set in their paper. For instance, for the CIFAR10 dataset, the code implementation did not have any convolutional layers to the code, which meant that we had to add these specifications ourselves: The Convolutional Net part of both the model is fairly standard, the first C layer had 32 5×5 filters with Relu hidden units, the second C layer has 64 5 × 5 filters. Both pooling layers used max pooling and downsampled by a factor of 2.

Submitted to COMP 551.

1 Abstract

Throughout the course, we have seen different methods to perform image classification such as deep convolutional neural networks, but we have mainly used the softmax activation function on the final layer for the task of multi-class classification. There exists other methods, and one of them is to use a Support Vector Machine as a final instead. To assess if this method is preferable to what we have done so far, we will follow the exact same steps by (Tang, 2013) in Deep Learning Using Linear Support Vector Machines and try to reproduce the same results. In this paper, the SVM model and softmax model were compared on 3 different datasets : MNIST, CIFAR-10 and ICML 2013 Representation Learning Workshop's face expression recognition challenge. By reproducing the same results as the paper on the MNIST and CIFAR-10 datasets, we reached the same conclusion : "for some deep architectures, a linear SVM top layer instead of a softmax is beneficial" (Tang, 2013).

2 Introduction

First, let's define what are the main differences between softmax and SVM. A softmax function internally normalizes a vector of K real number values into a vector of K real number values that sum up to 1. After accepting the vector as input, proportional to the exponentials of the input values. For instance, before applying the softmax function, some vector components can consist of any real number. However, after applying softmax, each value will be between 0 and 1 and add up to 1, so they can be interpreted as a probability function. This is essentially a generalization of the logistic regression for K different classes. Additionally, "larger input components will correspond to larger probabilities" (2023).

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

On the other hand, "The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points" (Gandhi, 2018). Here, we use the L2-SVM instead of the hinge loss, since it is differentiable and we need this specificity to be able to use backpropagation in our neural network.

$$\text{minimize} \qquad \frac{1}{2} \parallel \mathbf{w} \parallel^2 + \frac{C}{2}\sum_{i=1}^{M} \xi_i^2$$

This larger regularization with L2-SVM is believed to be more efficient, which would explain the better results than softmax. Also, since we are doing a multi-class classification, we use a one-vs-rest approach. It helps to extend SVM to K classes instead of 2 : "For K class problems, K linear SVMs will be trained independently, where the data from the other classes form the negative cases" (Tang, 2013).

2.1 Datasets

The CIFAR-10 dataset consists of a total of 60000 images as mentioned previously. There are 10 different classes that these images can fall under, with 6000 images per class. When extracting the data from its raw format, it is divided into another 6 different batches. 5 of them being training batches and the lone other being the testing batch. Each batch is divided evenly again with 10000 images in each, with no uniform distribution of image classes in each batch. However, the training batches combined have exactly 5000 images from each class.

The MNIST dataset consists of a total of 70000 images of hand-written numbers. There are 10 classes, one for each number from 0 to 9. The images are in black and white, meaning that we only need one number to represent every pixel on a greyscale. From the 70000 images, we selected 60000 to train our model, and the remaining 10000 to test it.

In order to properly process this into our implementation, we first needed to vectorize, normalize and one-hot encode our data. For CIFAR-10, each image can be represented by 3072 individual neurons (pixels). 32 x 32 x 3, 32 being the dimension of each image and 3 being the RGB values of each pixel. For each batch, a 10000 by 3072 matrix can be used to represent each image. For MNIST, every image is 28 x 28 pixels for a total of 784 pixels.

3 Results

3.1 Results reproducing original paper

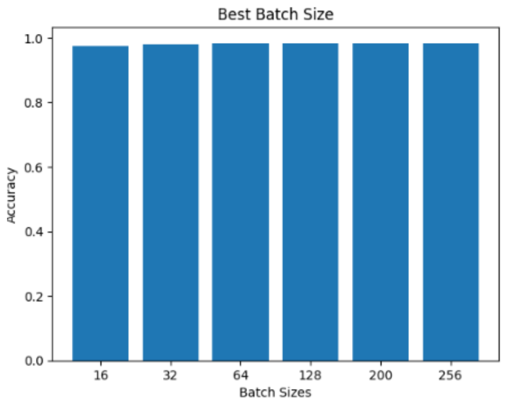
For the MNIST dataset, we used roughly the same settings as in the paper. Two hidden layers of 512 units, a batch size of 200, 300 minibatches. We also trained the model with stochastic gradient descent and learning rate of 0.1. It was said on the paper that momentum was also used, but without providing the exact value. Thus, we arbitrarily chose a momentum of 0.9. We made a few changes such as not applying PCA to the input layer, reducing epochs from 400 to 4 and not adding a Gaussian noise to the input. With these settings, we obtained an accuracy of 97 percent for the SVM and 96 percent for the softmax model.

For the CIFAR-10 dataset, we used the same convolutional neural network with 3 hidden layers as in the paper : the first Conv2D layer has 32 5×5 filters with Relu hidden units, the second Conv2D layer has 64 5×5 filters. The penultimate layer has 3072 hidden nodes and uses Relu activation with a dropout rate of 0.2. There is also a max-pooling layer after each Conv2D, with a factor of 2. The optimizer, batch size and number of epochs were not specified so we had to choose these parameters ourselves : Batch size of 128, 4 epochs and Adam optimizer. With these settings, the SVM and softmax model achieved an accuracy of 10 percent and 45 percent respectively.

3.2 Results beyond original paper

As suggested in the paper, we added some drop-out on the hidden layers to avoid overfitting on the MNIST dataset. We came to the conclusion that this was indeed effective in increasing the test accuracy. Thus, we added a dropout of 0.5 on both hidden layers.

We first decided to investigate the best batch size that could further our model's optimization using a baseline of 4 epochs on the MNIST dataset. The best batch size with a linear support vector machine and softmax layer were 200 and 256 respectively. We noticed using a mini batch of more data cases correlated with higher accuracy although the test accuracy plateaued at each size (Figure 1).



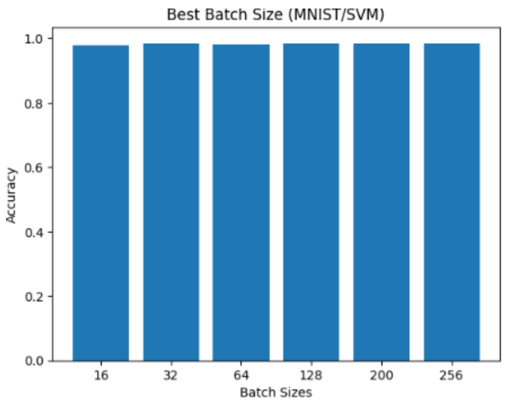
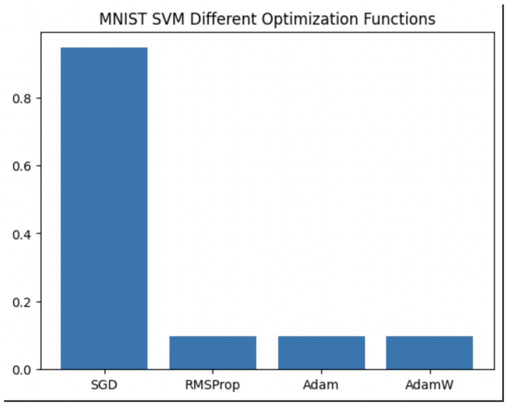


Figure 1: Best Batch Size using Softmax and SVM on MNIST

Different optimizers have different strengths and weaknesses and may perform better or worse depending on the specific problem we are trying to solve. We experimented on each model to test which optimizer function would yield the best results. For both the models using the MNIST data, SGD was the best optimizer function (Figure 2).



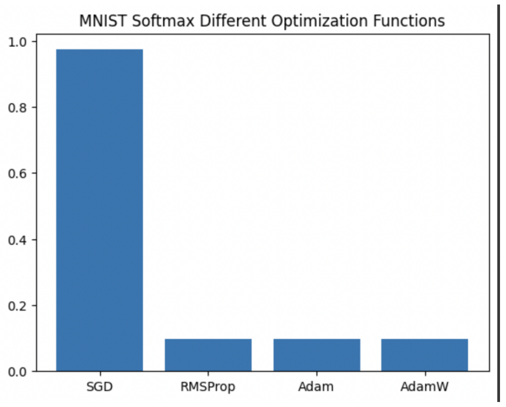


Figure 2: Best Optimizer using Softmax and SVM on MNIST

One reason why SGD might perform well is that it can handle large datasets efficiently. SGD updates our model's parameters using only one training example at a time, which makes it suitable for online learning and large datasets. Additionally, the stochastic nature of SGD can help our model escape local minima and find better solutions.

Changing the batch size on the cifar-10 dataset didn't improve the model's accuracy with the SVM layer. This can conclude the notion that our batch size does not really affect the testing and training accuracy and generalization is not really an issue with our trained model. In contrast, using a softmax layer showed that the best batch size would be a batch size of 200 (Figure 3), striking a fair balance between the accuracy and the estimate of the error gradient.

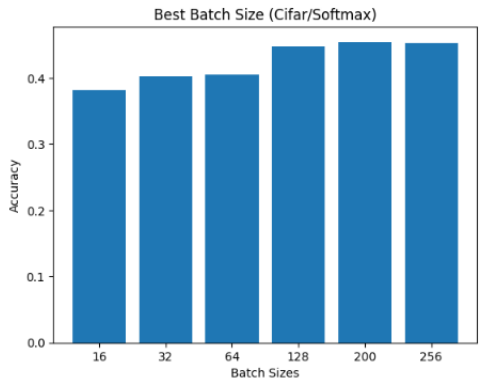


Figure 3: Best Batch Size using Softmax on CIFAR-10

Comparing however each test error allowed us to compare the accuracy of during their changes in batch size. The test error using SVM layer was on average 10.12 and using softmax layer was 1.63. This is where our replicated model does not correlate with the paper we attempted to reproduce. High variance and bias using the SVM layer could have been the underlying reason for this, and can be fixed for future reproduced results by adding more features into the CNN or reducing the regularization from the SVM.

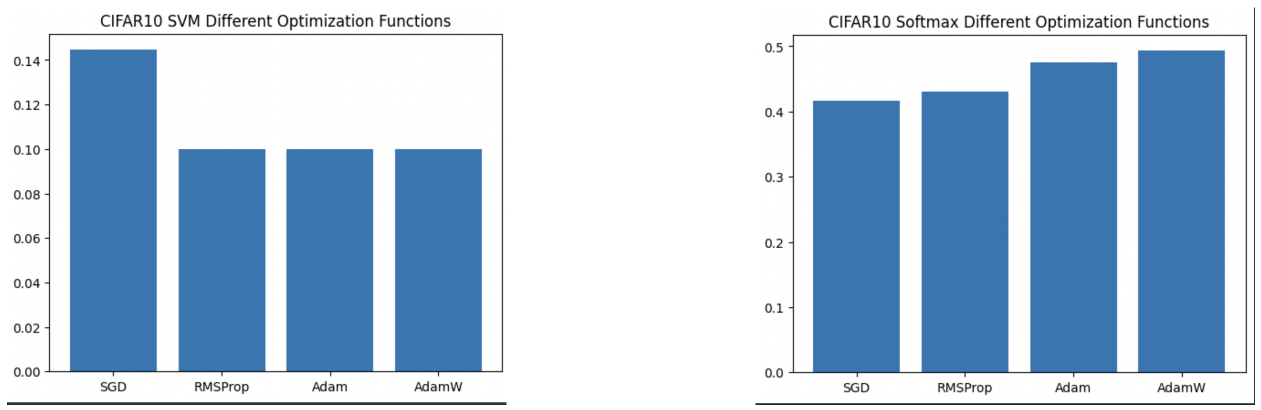


Figure 4: Best Optimizer using Softmax and SVM on CIFAR-10

Comparing different optimizers with the CIFAR-10 dataset, the SVM model had SGD as the best, and the Softmax model had the AdamW optimizer as the one the returned the highest accuracy. One reason why AdamW might perform well on our model and dataset is that it can effectively prevent overfitting while still benefiting from the adaptive learning rate and momentum of the original Adam algorithm. This can result in faster convergence and better generalization compared to other optimization algorithms.

Upon these ablation studies, the reproducibility of the paper reached approximately the same conclusion as the paper. Our accuracy when using the same models on the CIFAR-10 dataset was now improved, but still was not able to show that the SVM layer would outperform the softmax layer. Different fine-tuning parameters were not enough to overcome this obstacle, which can be fixed using different models for the dataset instead.

4 Discussion and Conclusion

In summary, it seems like a support vector machine as final layers are outperforming softmax activation function in some situations. The tests were only done of a very small and simple sample size of datasets. We would need to explore other more complex datasets to be certain that this statement is true in general. More specifically, the SVM layer and softmax layer used on the CIFAR-10 dataset were not able to conclude that the SVM layer is more advantageous than the softmax layer. In the future, we can attempt to replicate the results using a different implementation of the CNN and not base it on the model that worked really well with the MNIST dataset. Now that we know that some classification algorithms give better results in certain situation that the one we are used to to, we could try a variety of other classification methods as final layers such as a decision tree, a random forest or K-Nearest Neighbors, which on certain occasions, might be outperforming both softmax and SVM.

4.1 Challenges

We faced a few challenges when trying to reproduce the paper's results and when doing hyperparameter tuning. The first one was the computation cost. Indeed, the stated number of epochs was 400, which was computationally too heavy for us. Instead, we decided to use only 4 epochs as the accuracy was converging fast enough. We also had to neglect some of the neural networks settings specified in the paper that we were not able to understand such as "The L2 weight cost on the softmax layer is set to 0.001" or " downsampled by a factor of 2". Lastly, we had a hard time to find a good optimizer as, for some unknown reason, most of them were stuck at approximately 10 percent accuracy without ever increasing. We had to try many different ones with various changes in the parameters to find one that would work.

5 Statement of Contributions

Each group member shared the components of the project equally. Each member was designated with a task to work on, and continued this approach when conducting the tests. The write-up was written with our shared expertise

References

Gandhi, Rohith. Support Vector Machine — Introduction to Machine Learning Algorithms. Medium. 2018, June 7. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

Gupta, H. SVM classification using Keras [Computer software]. GitHub. 2019, April 23. https://github.com/hgupta01/svm$_{c}lassification_{k}eras$

Softmax function. Wikipedia. 2023, April 16 https://en.wikipedia.org/wiki/Softmax$_{f}unction$

Tang, Yichuan. Deep learning using linear support vector machines. 2015, Feb 15. arXiv preprint arXiv:1306.0239.