COMP551 Mini-Project4 Report

Leila Wang, Raphael Wang, Oliver Wang

April 27, 2023

1 Introduction

The paper Deep Learning using Linear Support Vector Machines Tang [2015] investigates the use of linear support vector machines (SVMs) as a deep learning classification job alternative to the softmax activation function. The author showed that replacing the softmax layer with an SVM results in considerable performance improvements on common datasets like MNIST, CIFAR-10, and a facial expression recognition task. Instead of the cross-entropy loss employed by the softmax function, the SVM is trained using a margin-based loss. The author contended that the improved performance is due to the SVM loss function's greater regularization effects, rather than better parameter optimization. The paper gives a summary of previous work on integrating neural nets and SVMs, with a focus on the L2-SVM model.

2 Scope of reproducibility

We replicated the majority of the experiments using the code provided by the author. However, rather than testing the CNN-SVM and CNN-softmax models on three benchmark datasets as in the original paper, we only tested them on the Cifar-10 and MNIST datasets. On both datasets, we tested the claims that

- Replacing the softmax layer with a layer of SVM results in considerable performance improvements on common machine learning datasets like MNIST, CIFAR-10.
- The greater performance of the CNN-SVM model is attributable to the greater regularization effects of the SVM loss function (squared hinge), rather than better parameter optimization.

3 Methodology

3.1 Model Description

For the CIFAR-10 experiment, the CNN-SVM model is composed of two convolutional layers, followed by max-pooling layers, a fully connected layer, a

dropout layer, and an SVM layer. The second pooling layer's output is flattened and passed through a fully linked layer with 3072 units and a ReLU activation function. For multiclass classification, the final SVM layer employs the squared hinge loss function with L2 regularization. The CNN-SoftMax features a similar architecture with the difference of a dense output layer with SoftMax activation and categorical cross-entropy loss function.

For the MNIST experiment, we first performed PCA from 784 dimensions down to 70 dimensions. A simple fully-connected model composed of two hidden layers of 512 units each and followed by a softmax or an L2-SVM was used.

3.2 Datasets

The MNIST dataset, which contains 70,000 images of handwritten digits, is commonly used for testing image classification methods. 60,000 photos are utilized for training and 10,000 for testing. Each image is grayscale and 28x28 pixels in size, with a relatively uniform label distribution across all ten classes. CIFAR-10 is a collection of 60,000 color images of ten different types of small visuals, 50,000 of which are used for training and 10,000 for testing. Each image is 32x32 pixels in size, with uniform label distribution across all classes. CIFAR-10 also has a fixed train/test split of 5,000 images for testing and 45,000 images for training. The face expression identification challenge dataset contains 35,887 pictures of faces classified with one of seven different expressions. The distribution of labels is not uniform across all classes, with some classes containing more examples than others. For all three datasets, preprocessing procedures such as normalizing pixel values and resizing images are utilized. Each dataset is available for download from their respective websites.

3.2.1 CIFAR-10 Preprocessing

The dataset was divided into training and testing sets, with pixel values normalized to a range of 0 to 1. Data augmentation techniques were used to increase the diversity and robustness of the training dataset. The data augmentation sequential model was used specifically to randomly flip photos horizontally, rotate them by up to 0.2 radians, and zoom them by up to 10 percent. After that, the training data was shuffled and mapped to the augmented images and labels, and batched to a specific size (128).

3.2.2 MNIST Preprocessing

Several important steps are involved in the MNIST dataset data preprocessing steps. First, the data is vectorized, which means it is flattened into a one-dimensional array of 784-pixel values. Principal Component Analysis (PCA) is performed after vectorization to further preprocess the data by decreasing the dimensionality from 784 to 70 of the input features. This is done to simplify the data while maintaining as much of the original data's variation as possible.

The reduced feature set is then used to train a machine-learning model that can categorize digits reliably.

3.3 Hyperparameters

For the MNIST experiment, the hyperparameters of the CNN-SoftMax model are set manually according to the training instructions introduced in the paper. We trained using stochastic gradient descent with momentum on these 300 mini-batches for over 400 epochs. The learning rate is initially set to 0.1 and linearly decayed to 0.0. The L2 weight cost on the SoftMax layer is set to 0.001. Gaussian noise of standard deviation of 1.0 (linearly decayed to 0) is added. For the CNN-SVM model, the L2 weight cost is tuned by optimizing the accuracy over 20 epochs using 5-fold cross-validation. It turns out that L2 weight cost equaling 4.223e-05 reached the highest accuracy of 0.963 over 20 epochs using 5-fold cross-validation.

Similarly, for the CIFAR-10 dataset, the average validation accuracy across all folds is utilized as the optimization metric. Optuna is used to find the best values for the hyperparameters. For the CNN-SoftMax model, the best hyperparameters discovered were a learning rate of 0.000167 and a weight decay of 0.000156, with 69.3% test set accuracy. Tuning the SVM model, on the other hand, entails optimizing the learning rate and the regularization parameter C. The objective function is a combination of squared hinge loss and the L2 regularization term, with the L2 regularization term controlling the model's complexity and preventing overfitting. The best hyperparameters discovered were a learning rate of 0.001235 and a regularization parameter C of 0.000141, with a test set accuracy of 71.1 percent attained.

4 Results

4.1 Results reproducing original paper

The performance of the CNN-SVM and CNN-SoftMax models on the CIFAR-10 and MINST datasets was examined and compared. Results were reported in the form of loss/accuracy curves and classification reports.

4.1.1 Result 1: MNIST

By following the exact approaches introduced in the paper, we achieved an accuracy of 97.91% on the CNN-SVM model and 97.88% on the CNN-SoftMax model on the MINST dataset. Figures 1 and 2 illustrate the training and testing loss/accuracy curves of the two models. Both models have comparable training and testing accuracy, but the CNN-SVM model has slightly better accuracy than the CNN-SoftMax model. This confirms the author's claim that the CNN-SVM model may be better suited for simple classification tasks such as the MINST dataset.

Figures 3 and 4 are the classification reports of the CNN-SVM model and the CNN-SoftMax model performance on the MNIST dataset. The precision metric counts the number of successfully identified samples among all samples expected to be positive for a particular class. The recall metric calculates the percentage of correctly categorized samples among all samples in a particular class. The F1-score is the harmonic mean of precision and recall for a particular class, and it gives a single summary assessment of the model's performance.

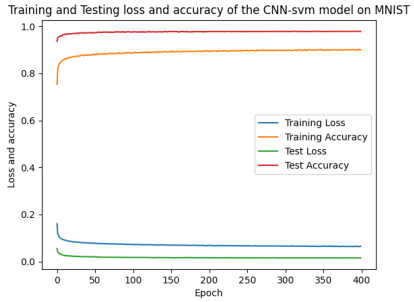

Figure 1: CNN-SVM Training and Validation Curves on MNIST

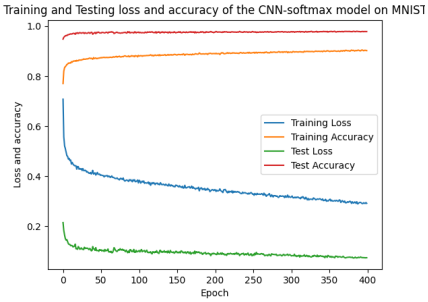

Figure 2: CNN-SoftMax Training and Validation Curves on MNIST

	precision	recall	f1-score	support
0	0.98	0.99	0.99	980
1	0.96	1.00	0.98	1135
2	0.99	0.98	0.98	1032
3	0.99	0.98	0.98	1010
4	0.99	0.97	0.98	982
5	0.99	0.98	0.98	892
6	0.98	0.99	0.98	958
7	0.97	0.96	0.96	1028
8	0.99	0.97	0.98	974
9	0.96	0.97	0.97	1009
accuracy			0.98	10000
macro avg	0.98	0.98	0.98	10000
weighted avg	0.98	0.98	0.98	10000

Figure 3: CNN-SVM Classification Report on MNIST

	precision	recall	f1-score	support
0	0.98	0.99	0.99	980
1	0.96	1.00	0.98	1135
2	0.99	0.98	0.98	1032
3	0.99	0.98	0.98	1010
4	0.99	0.96	0.98	982
5	0.98	0.98	0.98	892
6	0.98	0.99	0.98	958
7	0.97	0.96	0.97	1028
8	0.99	0.98	0.98	974
9	0.96	0.97	0.96	1009
accuracy			0.98	10000
macro avg	0.98	0.98	0.98	10000
weighted avg	0.98	0.98	0.98	10000

Figure 4: CNN-SoftMax Classification Report on MNIST

4.1.2 Result 2: CIFAR

On CIFAR-10, the CNN-softmax model (Figure 6) and CNN-svm model (Figure 5) have a relatively similar accuracy of 0.70 on the testing dataset. The CNN-svm model converges quicker, attaining a loss of 2.9452e-04 within a few epochs, but the CNN-softmax model converges slowly, with training and testing loss converging to 0.7647 and 1.0675 respectively.

Similarly, the classification report (Figures 7 and 8) shows the result of the reproduced models. After 100 epochs, both models reached an accuracy of 70%. To be more specific, the CNN-svm model did better than CNN-softmax on

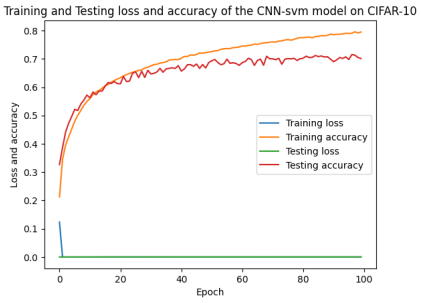

Figure 5: CNN-SVM Training and Validation Curves on CIFAR-10

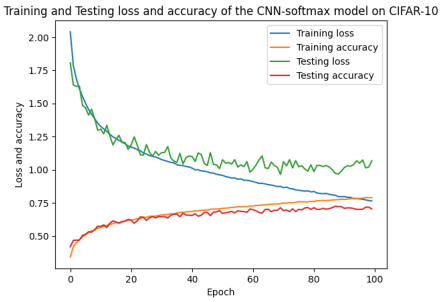

Figure 6: CNN-SoftMax Training and Validation Curves on CIFAR-10

classifying the third and sixth objects. Overall, the precision of prediction by CNN-svm is steady. Both the highest and lowerest precision of CNN-svm model is higher than those of CNN-softmax model

	precision	recall	f1-score	support
0	0.74	0.77	0.76	1000
1	0.77	0.86	0.81	1000
2	0.75	0.51	0.61	1000
3	0.65	0.38	0.48	1000
4	0.71	0.57	0.63	1000
5	0.66	0.62	0.64	1000
6	0.62	0.88	0.73	1000
7	0.68	0.82	0.75	1000
8	0.88	0.76	0.82	1000
9	0.62	0.85	0.72	1000
accuracy			0.70	10000
macro avg	0.71	0.70	0.69	10000
weighted avg	0.71	0.70	0.69	10000

Figure 7: CNN-SVM Classification Report on CIFAR-10

	precision	recall	f1-score	support
0	0.77	0.78	0.78	1000
1	0.73	0.89	0.80	1000
2	0.77	0.49	0.60	1000
3	0.59	0.49	0.54	1000
4	0.69	0.59	0.64	1000
5	0.71	0.54	0.62	1000
6	0.53	0.91	0.67	1000
7	0.80	0.74	0.77	1000
8	0.86	0.80	0.83	1000
9	0.72	0.80	0.75	1000
accuracy			0.70	10000
macro avg	0.72	0.70	0.70	10000
weighted avg	0.72	0.70	0.70	10000

Figure 8: CNN-SoftMax Classification Report on CIFAR-10

5 Discussion

5.1 Challenges

Several difficulties arose during our attempt to replicate the results provided in the paper. First of all, the paper did not provide clear directions on how to repeat the studies, making the reproduction process difficult. Second, the author's code contained a TensorFlow version conflict, as the version utilized by the author was too low. As a result, we had to downgrade TensorFlow and make many changes to the code to be compatible with the old version. Furthermore, while we were able to successfully replicate the author's approach, we were unable to reach the same degree of accuracy described in the paper. This could be due to a variety of causes, including variances in technology, variations in the dataset, or discrepancies in the pre-processing methods utilized. Furthermore, we

discovered that the SVM layer used in the paper was not directly implemented in TensorFlow, so we modified the dense layer with the Keras API to have a linear activation function and squared hinge loss to simulate the SVM layer.

5.2 Future Investigation

The project's major takeaway is that using DLSVM instead of softmax can improve classification performance on both standard and current datasets. According to the study, DLSVM beat MNIST, CIFAR-10, and the facial expression recognition challenge dataset. The move from softmax to SVMs is simple and can be advantageous for classification issues. Future studies on the use of DLSVM for classification tasks can look at datasets other than those covered in this article, such as natural language processing datasets like the Stanford Sentiment Treebank or image datasets like ImageNet. Another intriguing research direction is to increase DLSVM performance on tasks with limited data using other methodologies like data augmentation or transfer learning.

6 Statement of Contributions

Every group member cooperated and contributed to each part of this project

References

Yichuan Tang. Deep learning using linear support vector machines, 2015.