COMP551 Winter 2022
MiniProject 4 Report - Group 70
CNN with SVM

George Qiao 260779462 **Atia Islam 260737946** **Vraj Patel 261022581**

1 Introduction

This project aims to reproduce and validate the results of a 2019 study by Abien Fred M. Agarap titled "An Architecture Combining Convolutional Neural Network (CNN) and Support Vector Machine (SVM) for Image Classification" [1]. The goal is to compare the conventional use of softmax function for multi-class classification to a proposed alternative that uses SVM as the output layer of CNN. The popular MNIST[2] and Fashion-MNIST[3] datasets are used as image classification benchmarks. The study uses a simple 2-convolutional layer with max-pooling model with a fixed set of manually-selected hyperparameters as the baseline, controlling all variables except for the softmax/svm output layer. The original study found that CNN-softmax and CNN-SVM has comparably high performance on the two datasets, while admitting its limitations such as having no hyperparameter tuning or the simplicity of the CNN architecture.

2 Scope of reproducibility

The main claim which the original study centered around and which this project seeks to validate is that CNN-SVM is a solid alternative to the commmonly used CNN-softmax architecture in terms of image classification performance. The image classification performance is measured as test accuracy on the MNIST and Fashion-MNIST datasets. For simplicity, the original study chose a 2-layer CNN model with max pooling and manually fixed a set of hyperparameters, leaving the output layer structure as the only independent variable. The main results of the original study are shown in Figure 1.

Table 1: Main results of [1]

Dataset	CNN-softmax	CNN-SVM
MNIST	99.23%	99.04%
Fashion-MNIST	91.86%	90.72%

The original study itself sought to review and validate the results of a 2013 study that proposed the CNN-SVM architecture [4]. The original study acknowledged the limitations of its relatively simple investigation, namely the lack of sophistication in the chosen CNN model and the arbitrariness of hyperparameter selection. In additional to reproducing the main claim, this project will perform some hyperparameter tuning to address the second limitation as stated above. We consider the first limitation out-of-scope of the course and too computationally-demanding, thus we leave it as an opportunity for future investigations.

3 Methodology

We initially intended to use the author's code published on GitHub [5]. However, we found that his code uses a few APIs (main in Keras) that are now deprecated and do not have exact equivalents in new APIs. Instead of rummaging through his code and modifying it to fit the new APIs, we decided to build on our experiment with CNN and Keras from mini project 3 and build our own simple CNN.

3.1 Model description

The CNN model presented in the original paper consists of 2 convolutional layers (with 32 and 64 5x5 filters respectively, both using ReLU activation function), a max pooling layers after each convolutional layer (2x2, 1 stride), and 1 dropout

layer in between the fully connected layers (1024 hidden units) and the output layer (10 output classes). See the Appendix for a demonstration of model shape. The model has a total of 21,297,034 parameters.

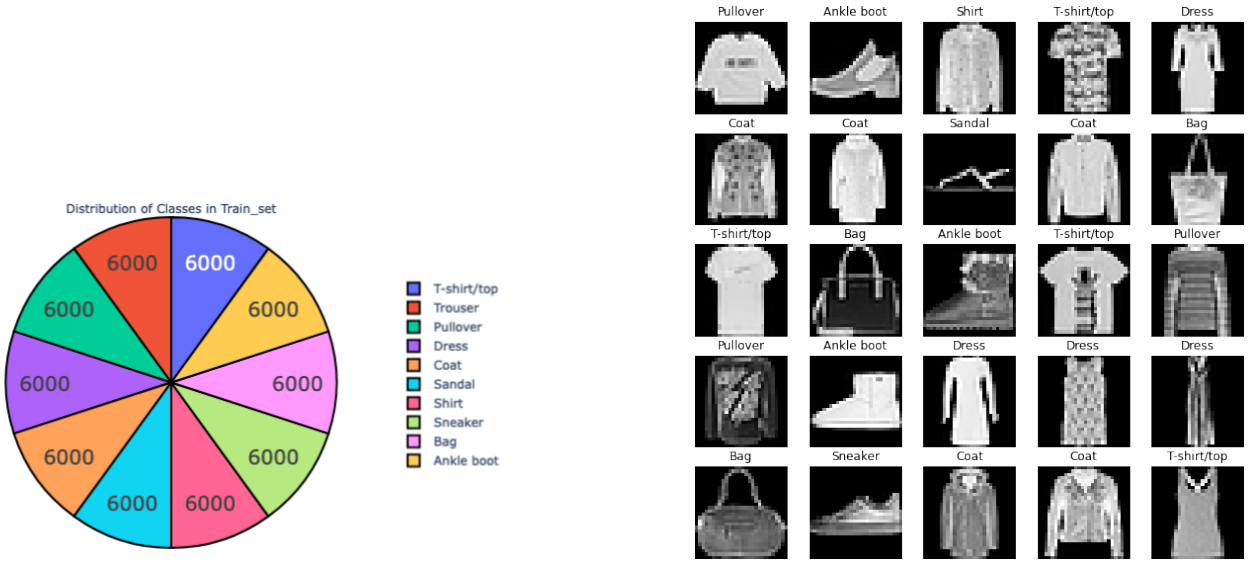
$$J_{CNN-SVM} = \sum_{n} max(0, 1 - y_n^{'}(w^T x_n - b)) + C||w||_2^2 \tag{1}$$

CNN-SVM essentially replaces the softmax function in CNN-softmax with a L2-regularized hinge loss function (see Equation 1, where C is the penalty parameter that determines the strength of the regularization relative to the hinge loss).

3.2 Datasets

The MNIST[2] is a database of handwritten digits in 28x28 greyscale images, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image, which is labeled in one of 10 classes. Both train and test sets have balanced classes.

Fashion MNIST-dataset[3] is presented by Zalando Research as an upgrade from the original MNIST dataset. Exactly like MNIST, Fashion-MNIST consists of 60,000 28x28 greyscale images in the training set and 10,000 in the test set, with each label is one out of 10 balanced classes.



(a) Class distribution (same for MNIST)

(b) Some samples

Figure 1: Fashion-MNIST Dataset

For both datasets, the only data-preprocessing done is normalization of all pixels values to the [0, 1] range.

Links to download the datasets can be found here: MNIST, Fashion-MNIST.

3.3 Hyperparameters

Four key hyperparameters in this experiment, identified by the original author, are the batch size, learning rate, dropout rate and the penalty parameter. The manually selected set of hyperparameters are listed in Table 2. Additionally, the original author decided on a maximum timestep of 10,000. We believed epoch to be a better measure of training progression and used a maximum epoch of 25, which for a batch size of 128 is roughly equivalent to (slightly over) 10,000 steps.

As mensioned in Section 2, the original study did not perform any hyperparameter search, since its focus was on comparing CNN-softmax and CNN-SVM on a fixed set of hyperparameters. This project performs some manual hyperparameter search in an attempt to address its lack in the original paper. We have done in total 52 experiments to see if the equivalence of CNN-softmax and CNN-SVM still hold at different hyperparameter values. The results are presented in Section 4.2.

Table 2: Hyper-parameters used for the baseline models

Hyper-parameters	CNN-softmax	CNN-SVM
Batch size	128	128
Dropout rate	0.5	0.5
Learning rate	0.001	0.001
Penalty parameter	N/A	1

3.4 Experimental setup and code

The models are implemented primarily using the tensorflow.keras library. The CNN-softmax model is a typical sequential model with the layers described in Section 3.1. For CNN-SVM implementation, since the sequential model API does not allow a custom-defined loss function that also takes the model weights as parameters, it is split into two parts with hinge loss in the compile() function and L2 regularization in the output layer. Both models are trained from scratch with default initialized parameters. Hyperparameter search is done manually for a few select values due to time/resource constraints and the fact that this is an exploratory venture.

Please see the attached .ipynb files. All CNN-softmax experiments are in CNN_softmax.ipynb; CNN_SVM_MNIST.ipynb and CNN_SVM_Fashion.ipynb contain CNN-SVM experiments on the two datasets respectively.

3.5 Computational requirements

We ran the experiments mostly on the default Google Colab runtime with GPU. Each individual run (25 epochs) takes around 5-10 minutes to finish. The experiments could be much accelerated on machine with better computational resources. The models are trained from scratch with default initialized weights.

4 Results

4.1 Results reproducing original paper

Table 3: Reproduced main results (test accuracy)

Dataset	CNN-softmax	CNN-SVM
MNIST	99.07%	99.24%
Fashion-MNIST	91.74%	91.57%

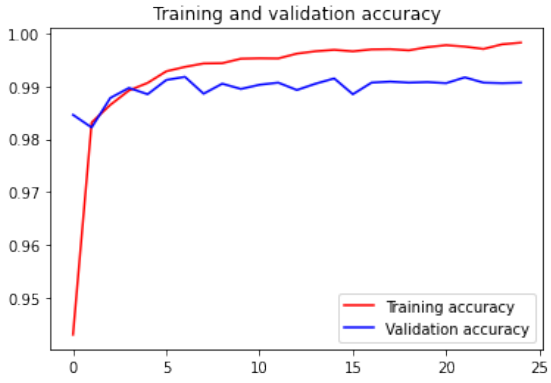
Table 3 shows the test accuracy of the two baseline models on the two datasets. In contrast with the original results (Table 1), our results instead show that CNN-SVM slightly outperforms CNN-softmax on MNIST and is closer to CNN-softmax performnce on Fashion-MNIST. However, the general trend seen here conforms to the original: test accuracies of CNN-softmax and CNN-SVM are both very high on both datasets. This validates the original claim that CNN-SVM is a solid alternative to CNN-softmax.

As in the original paper, we hereby reproduce the training and test accuracy as well as loss vs. epoch (timestep) plots.

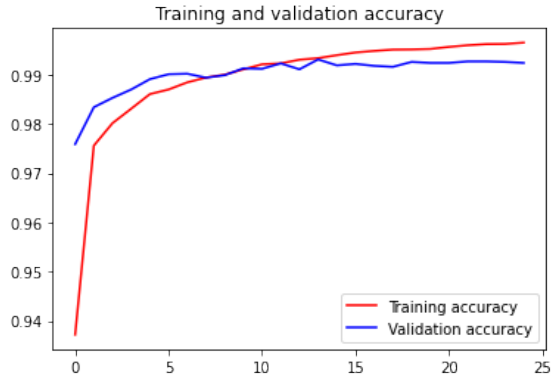
*Note: We apologize for the axis scale mismatch between the subplots. We found out late and unfortunately did not have time to rerun experiments.

These figures show very high similarity between behaviours of CNN-softmax and CNN-SVM. For both models on both datasets, the training accuracy reaches very high level at the end of the runs (>99% for MNIST and >94% for Fashion-MNIST), while the training accuracy flattens out before slightly decreasing at the end due to overfitting. The loss decreases rapidly during the first few epochs and stay at a very low level (0.05 for MNIST and 0.1 for Fashion-MNIST), with the only exception being CNN-softmax on Fashion-MNIST having a slightly higher and increasing loss up to 0.5.

In addition, CNN-SVM has demonstrated its advantage in terms of training time. Experimental data from our project shows that, for SVM it takes 7 seconds to run one epoch iteration if we compare the results of softmax with the same dataset and parameter it takes 9 seconds. This mean CNN-SVM is >20% faster than CNN-softmax for nearly the same test results, making it useful in situations of large datasets or complex models.

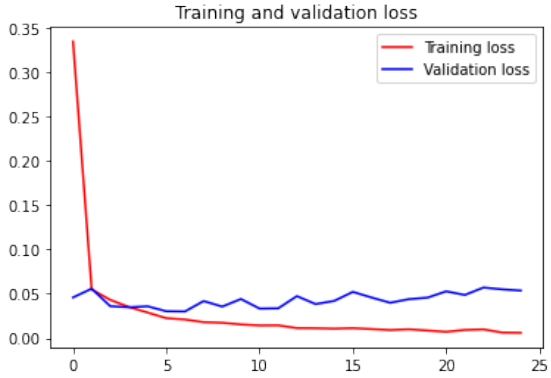


(a) CNN-softmax

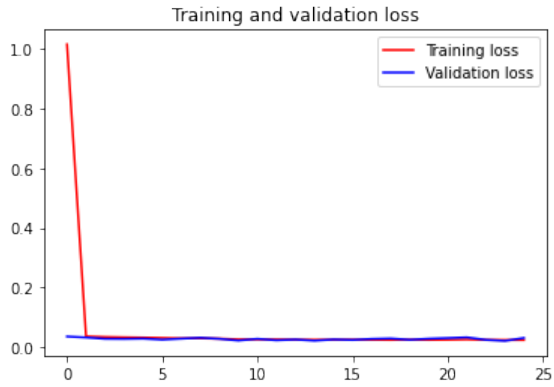


(b) CNN-SVM

Figure 2: Training and test accuracy vs. epoch, MNIST dataset

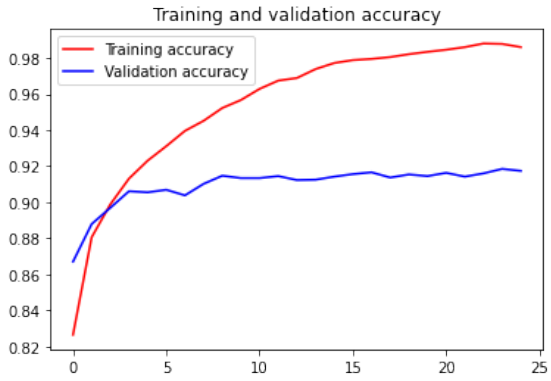


(a) CNN-softmax

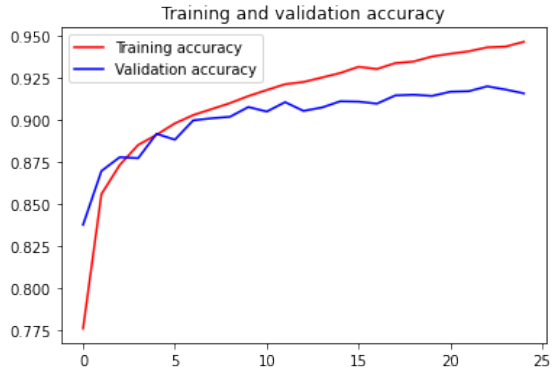


(b) CNN-SVM

Figure 3: Training and test loss vs. epoch, MNIST dataset



(a) CNN-softmax

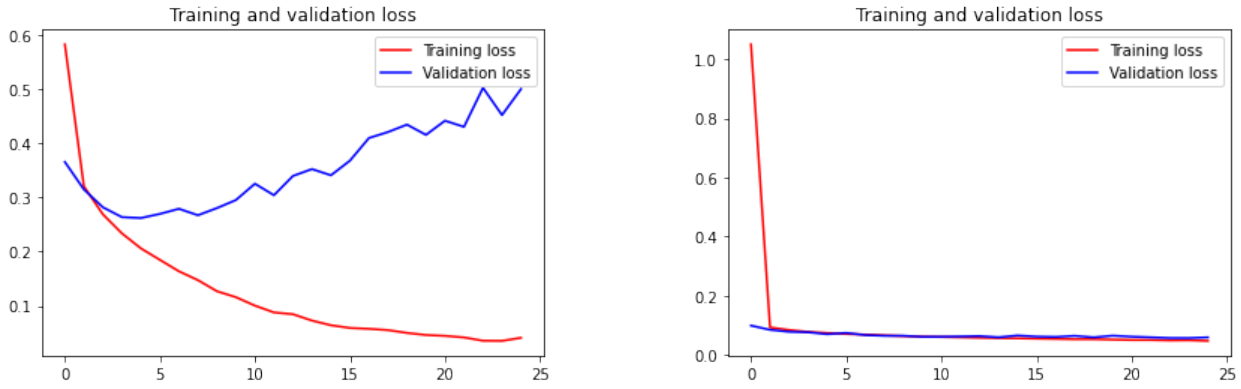


(b) CNN-SVM

Figure 4: Training and test accuracy vs. epoch, Fashion-MNIST dataset

4.2 Results beyond original paper

The following four hyperparameters were manually tuned for a few representative values each to see how the equivalence of the two model hold up under different conditions. Overall, from Table 4 to 6, we see great symmetry between results of CNN-softmax and CNN-SVM despite all the hyperparameter variations.



(a) CNN-softmax

(b) CNN-SVM

Figure 5: Training and test loss vs. epoch, Fashion-MNIST dataset

4.2.1 Dropout rate

Table 4: Effect of varying dropout rate

Dataset	Dropout rate	CNN-softmax	CNN-SVM
MNIST	0.4	98.83%	99.18%
	0.5	99.07%	99.24%
	0.6	99.28%	99.26%
	0.7	99.04%	99.20%
Fashion-MNIST	0.4	92.18%	91.89%
	0.5	91.74%	91.57%
	0.6	93.08%	91.49%
	0.7	92.72%	91.65%

We used different dropout rates to see the effects of that on the results but as we can see that there are no significant changes on both models.

4.2.2 Learning rate

Table 5: Effect of varying learning rate

Dataset	Learning rate	CNN-softmax	CNN-SVM
MNIST	0.001	99.07%	99.24%
	0.002	98.62%	99.18%
	0.005	98.35%	98.94%
	0.01	96.78%	9.57%
Fashion-MNIST	0.001	91.74%	91.57%
	0.002	90.49%	91.04%
	0.005	88.55%	89.88%
	0.01	82.82%	84.56%

Learning rate is the most important hyperparameter that we tuned since it greatly affects the convergence of the gradient descent to an optimum point. At higher learning rate gradient descent algorithm is easy to overshoot and even diverge, reducing the test accuracy, which we indeed see in Table 5. CNN-SVM completely diverges (accuracy of 9.57%) on MNIST with a high learning rate of 0.01.

Table 6: Effect of varying batch size

Dataset	penalty	CNN-softmax	CNN-SVM
MNIST	64	98.94%	99.33%
	128	99.07%	99.24%
	256	98.96%	99.11%
Fashion-MNIST	64	91.39%	91.64%
	128	91.74%	91.57%
	256	91.21%	91.00%

4.2.3 Batch size

With different batch sizes, the differences in the test accuracy are in that of $\pm 1\%$ range. We can say that different batch sizes doesn't change the results for our models. The main difference with batch size lies in the fact that larger batch size trains the model faster but requires more memory.

4.2.4 Penalty parameter

Table 7: Effect of varying penalty parameter

Dataset	penalty	CNN-SVM
MNIST	0.01	99.28%
	0.1	99.33%
	0.5	99.48 %
	1	99.24%
Fashion-MNIST	0.01	91.49%
	0.1	91.71%
	0.5	91.90 %
	1	91.57%

We varied the penalty parameter from the original author's 1 (very high amount of regularization) down to 0.01 (a typical order of magnitude we see for this parameter). We found that this variance is not very significant on the resulting test accuracy. We suspect that greater variations of this penalty parameter are needed to see changes in test accuracy.

5 Discussion

Our experimental results fully support the orginal author's claim that CNN-SVM is a great alternative to CNN-softmax in image multi-class classification tasks, at least in situations where the CNN architecture is simple. Moverover, our results furthers the strength of this argument by showing that varying the key hyperparameters does not break the symmetry between the performances of the two models.

Nonetheless, we need to be wary of the limitation of this study, the greatest of which is that we have only tested simple CNN models. Further work with more sophisticated CNN models (and also more extensive hyperparameter search) needs to be done to increase the generality of this claim.

5.1 What was easy

The easy parts of this project was to implement the CNN-softmax model. This model is quite basic, and the TensorFlow APIs as well as online documentations make this task quite simple. It is also simple (yet time-consuming) to perform hyperparameter tuning.

5.2 What was difficult

Since the original author's code uses deprecated APIs, reusing that code would require us to do some heavy fixing. We opted to implement our own models using Kera library. This presented the challenge of defining a custom L2-regularized hinge loss that fits within the Kera framework, which we ultimately circumvented by using the Kera layer regularizer.

References

[1] A. F. Agarap, "An architecture combining convolutional neural network (cnn) and support vector machine (svm) for image classification," *arXiv preprint arXiv:1712.03541*, 2017.

[2] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[3] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," 2017.

[4] Y. Tang, "Deep learning using linear support vector machines," *arXiv preprint arXiv:1306.0239*, 2013.

[5] A. F. Agarap, "Afagarap/cnn-svm v0.1.0-alpha," Dec. 2017.

Appendix

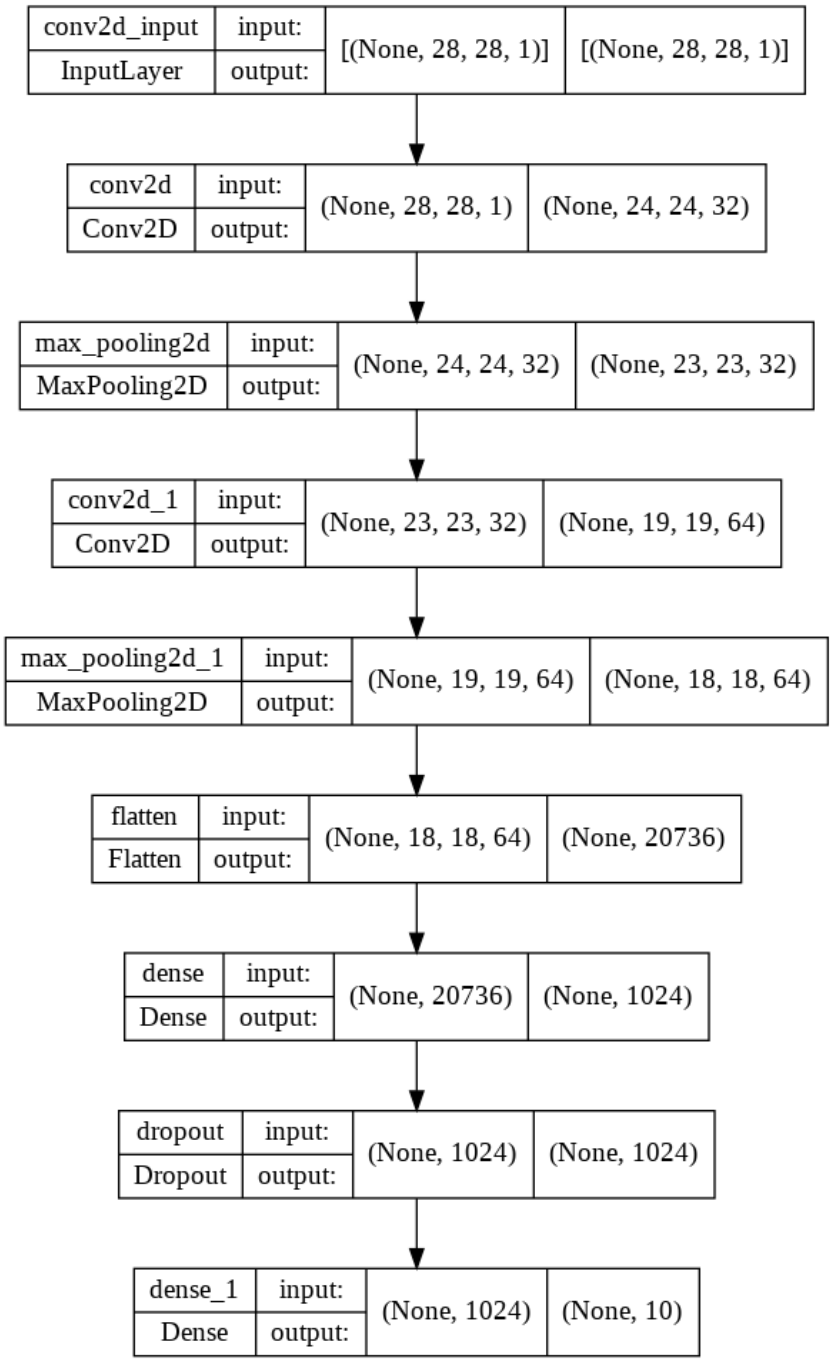


Figure 6: CNN model shape