Assignment 3, CS895, Spring 2023, ODU

Name: Shadman Mahmood Khan Pathan
Email: spath004@odu.edu
Student ID: UIN01243104, OLD DOMINION UNIVERSITY
Course: Fundamentals of Deep Learning CS895 Spring 2023
Assignment: Assignment 03
Date: February 27, 2023

February 27, 2023

The Github Repository link for this reports work is, `https://github.com/Shadman-spath004/CS895_Assignment_3_Shadman01243104.git`.
In machine learning, speed, stability, and robustness are important characteristics of an algorithm.

Speed refers to how quickly an algorithm can perform its task. Faster algorithms are generally preferred as they can process more data in less time.

Stability refers to how consistent an algorithm's output is when given the same input. An algorithm with high stability will give the same or similar results for the same input every time it is run. This is important in applications where consistency is critical, such as in medical diagnosis or financial forecasting.

Robustness refers to how well an algorithm can handle unexpected or noisy data. A robust algorithm is one that is able to generalize well to new or unseen data, even if that data is different from the training data. This is important in real-world applications where data can be messy, incomplete, or inconsistent. A robust algorithm will be able to handle these variations and still produce accurate results.

In this report we will try to find the above mentioned characteristics across two standard datasets fashion MNIST and digits MNIST. Moreover, the characteristics of these datasets under 4 types of optimizers (Adam, RMSProp, SGD and Customized optimization algorithm) across ten different trials are also seen.

In this assignment, fashion MNIST and digits MNIST were downloaded and classified using one input layer, 2 hidden layers and one output layer. Then the training accuracy and validation accuracy was obtained by running the algorithm in Google Colab platform by using python language mostly consisting of tensorflow, keras and matplotlib libraries.

It was found that a higher validation and test accuracy was found when dropout regularization was used. This delineates the concept of bias-variance trade off concept. Because, the regularization has reduced the relative complexity which has enabled the model to not fit the noise in the training data distribution.

To find the best parameters and to know the necessity of dropout, Kerastuner was used. However to find the best optimized and initializer, gridsearch cv was used. For instance, in case of classifying Fashion MNIST data, the best parameters were as such: dropout 0.3, hidden activation tanh, initializer keras initializer, epoch 20, optimizer adam, perceptrons in each layer 160. Moreover, an early stopping method was used to stop the model training to reduce the probability of unnecessary training.

The classification process of both Fashion MNIST and Digit MNIST was trialed 10 times whose reports are given in tabulation form and the graphs showing accuracy and error along with pertinent information are also delineated.

Assume,

SGDW is Accuracy with dropout and SGD optimizer
SGDO is Accuracy without dropout and SGD optimizer
RMSW is Accuracy with dropout and RMS optimizer
RMSO is Accuracy without dropout and RMS optimizer
AW is Accuracy with dropout and Adam optimizer
AO is Accuracy without dropout and Adam optimizer
CW is Accuracy with dropout and Custom optimizer
CO is Accuracy without dropout and Custom optimizer

Table 1: Validation accuracy of Fashion MNIST classification.

Seed no	SGDW	SGDO	RMSW	RMSO	AW	AO	CW	CO
42	0.8731	0.8762	0.8709	0.8707	0.8796	0.8683	0.8789	0.8844
322	0.8743	0.8764	0.8705	0.8757	0.8586	0.8592	0.8722	0.8869
2409	0.8729	0.8796	0.8772	0.8671	0.8611	0.8694	0.8759	0.8854
19	0.8683	0.8774	0.8724	0.8751	0.8771	0.8675	0.8766	0.8754
20	0.8724	0.8779	0.8659	0.8724	0.8611	0.8694	0.8759	0.8854
21	0.872	0.879	0.8666	0.8705	0.8584	0.8635	0.8719	0.8812
22	0.8687	0.8787	0.8782	0.8652	0.8747	0.8696	0.8682	0.8835
23	0.8734	0.8781	0.8783	0.8778	0.8601	0.8546	0.8763	0.8871
9	0.8706	0.8808	0.8758	0.8783	0.878	0.8688	0.8782	0.8789
45	0.8758	0.8805	0.8743	0.8780	0.8679	0.8677	0.8719	0.8824

Table 2: Average execution time (second) of Fashion MNIST classification.

SGDW	SGDO	RMSW	RMSO	AW	AO	CW	CO
37.93	37.35	38	40	37.01	37.05	43.80	43.90

Discussion and Inference

Stability: In the above tabulation format we can see that the produced accuracy is very much consistent with each other. A machine learning algorithm is considered stable when its performance is consistent across. In other words,

Table 3: Validation accuracy of Digitset MNIST classification.

Seed no	SGDW	SGDO	RMSW	RMSO	AW	AO	CW	CO
42	0.8811	0.8843	0.9817	0.9788	0.9815	0.9760	0.9788	0.9756
322	0.8837	0.8864	0.9805	0.9757	0.9586	0.9686	0.9732	0.9779
2409	0.8829	0.8879	0.9759	0.9724	0.9784	0.9769	0.9775	0.9685
19	0.8788	0.8877	0.9887	0.9787	0.9771	0.9785	0.9767	0.9654
20	0.882	0.888	0.9885	0.9687	0.8757	0.9796	0.9659	0.9735
21	0.884	0.879	0.9766	0.9705	0.9584	0.9735	0.9719	0.9712
22	0.8787	0.8767	0.9882	0.9652	0.9747	0.9696	0.9782	0.9735
23	0.8834	0.8881	0.9883	0.9778	0.9601	0.9646	0.9763	0.9771
9	0.8886	0.8908	0.9858	0.9783	0.978	0.9788	0.9782	0.9689
45	0.8843	0.8905	0.9843	0.9780	0.9679	0.9777	0.9719	0.9724

Table 4: Average execution time (second) of Digitset MNIST classification.

SGDW	SGDO	RMSW	RMSO	AW	AO	CW	CO
84	84.27	83.93	84	62.03	58.32	84.23	84.04

a stable algorithm will produce similar results regardless of changes, ensuring that the algorithm is reliable and can be applied to new data with confidence. Some factors that has contributed to the stability of the algorithm include:

Regularization: Regularization techniques can help to prevent overfitting and improve the generalization of the model. Feature: Presence of relevant features has helped to reduce the effects of noisy or irrelevant data.

Model complexity: Simple models with fewer parameters are often more stable than complex models, which are more prone to overfitting.

Robust optimization: Using robust optimization techniques here are less sensitive to outliers or noise in the data can improve the stability of the model.

Cross-validation: Using cross-validation to evaluate the performance of the model has helped to ensure that the model is not overfitting to the training data, and can generalize well to new data.

Speed:

In Table 2,4 it is seen that the algorithm performs reasonably fast and also seen that the addition of regularization decreases the time of execution (though not significantly) due to reduction of complexity.

Several factors contribute to the speed of the algorithm:

Size of the dataset: The larger the dataset, the longer it takes to train the model. But the dataset was the same and also there was not much difference between the time of executions among the different settings.

Complexity of the model: Models with a large number of parameters or complex architectures, such as deep neural networks, require more time to train than simpler models. Here with regularization the complexity of the model was discounted which also reduced the time of execution.

Optimization algorithm: The choice of optimization algorithm can affect the speed of training. Some algorithms are more efficient than others, and some are better suited to particular types of models. In this scenario SGD and adam optimizers were seen to perform faster than the rest in terms of execution time.

Hardware: The processing power of the hardware being used for training can also impact the speed of the algorithm. More powerful hardware can perform computations faster and speed up training. To assist faster computation Google Colab platform (GPU) was used for running all the algorithm settings.

Preprocessing: Preprocessing steps such as data cleaning, feature scaling, and feature extraction can impact the speed of training. Properly preprocessing data can improve the efficiency of the algorithm. In this scenario the same preprocessing steps were maintained for all the algorithm settings.

Early stopping: Stopping the training process before it converges can reduce the training time, while still achieving a reasonable level of accuracy. All the different algorithm settings were used with early stopping.

Robustness: Robustness refers to the ability of a machine learning algorithm to maintain its performance in the face of changes or disturbances to the input data. It should be able to generalize well to new data that may differ from the training data in certain aspects, such as the presence of outliers or errors. In other words, a robust algorithm is less sensitive to small perturbations or changes in the data, and is able to maintain a consistent level of performance across different datasets or environments. In the above tables we find that the algorithm is able to perform almost similar performance across different settings of regularization, optimizer etc. Hence it can be inferred that considering the mentioned parameters the algorithm is robust in terms of performance. There are several factors that can contribute to the robustness of a machine learning algorithm, including:

Quality and diversity of the training data: A robust machine learning algorithm requires a diverse and representative dataset. The training data should cover all possible variations and edge cases that the algorithm may encounter in the real world. Both the dataset used here are diverse and qualified from the aspect to make the algorithm robust through learning.

Regularization: Regularization techniques can improve the robustness of a machine learning algorithm by preventing overfitting and improving generalization. However, we can see that the application of regularization was visible though not significant through the accuracy measure.

Hyperparameter tuning: Selecting the optimal hyperparameters for the algorithm can greatly affect its robustness. It is important to tune hyperparameters such as learning rate, batch size, number of hidden layers, etc. to achieve the best possible performance and robustness. The hyperparameters here were obtained by tuning which made it robust.

Model architecture: The architecture of the machine learning model can greatly impact its robustness. Choosing an appropriate architecture, such as a deep neural network or a convolutional neural network, can help to improve the algorithm's robustness. The performed model was simple and could perform well with 20 epochs across various settings.

Data preprocessing and augmentation: Preprocessing the data, such as scaling or normalization, can help to improve the robustness of the algorithm. Augmenting the data by applying transformations or adding noise can also help to increase the diversity and robustness of the training data. It is to be mentioned that the process was identical among all the settings.