Comparative Analysis: Stochastic Gradient Descend and Adam Optimizer for Neural Networks Training

Nattapoom Dumronglaohapun

April 2023

1 Introduction

In recent years, neural networks have achieved remarkable success in various machine learning tasks such as image classification, natural language processing, and speech recognition. However, training neural networks can be computationally expensive, especially for large datasets and complex architectures. To address this issue, many optimization algorithms have been proposed, including stochastic gradient descent (SGD) and Adam optimizer.

SGD is the canonical optimization algorithm for training neural networks. It works by updating the model's parameters in the direction of the negative gradient of the loss function with respect to the parameters, calculated using a mini-batch of training data. The "stochastic" part refers to the fact that the mini-batch used for each update is randomly sampled from the training data. This randomness introduces noise in the optimization process, which can help prevent the model from getting stuck in local minima.

On the other hand, Adam is an Adaptive optimization algorithms that is also commonly used across a variety of applications. Still, SGD is the most widely used optimization algorithm in deep learning> However, adaptive methods such as Adam have been found to perform better than SGD in some important tasks, such as attention models. However, the situations in which SGD performs poorly in comparison to Adam are not yet fully comprehended.

Moreover, While these algorithms have shown promising results, their performance can vary significantly depending on the dataset, network architecture, and hyperparameters. Therefore, the aim of this report is to compare and contrast the performance of SGD and Adam optimizer for neural network training across different types of problems such as binary classification, multi-class image classification, and natural language processing. The project will explore how the performance of these algorithms varies with different network architectures and hyperparameters, such as learning rate or batch size.

2 Methodology

Here are the methodology employed in this study.

2.1 Datasets

In this study, we used three datasets obtained from the `tensorflow-datasets` module, which is a library that provides a variety of publicly available datasets that can be used for machine learning tasks. The datasets that we used are titanic survival status, mnist handwriting digits, and imdb movie reviews. The titanic survival dataset is a binary classification problem, which involves predicting whether a passenger on the Titanic survived or not based on given features. The MNIST dataset is a well-known dataset in the field of computer vision and is used for multi-class image classification tasks. It consists of a set of grayscale images of handwritten digits from 0 to 9. Lastly, the imdb_reviews dataset is a popular dataset for sentiment analysis, which is a task of determining whether a given text expresses positive or negative sentiment.

2.2 Implementation

We utilized the TensorFlow Python module to implement the neural network models. TensorFlow is a widely used open-source library for machine learning and deep learning, which provides a comprehensive set of tools and functionalities to build and train neural networks. For binary classification, we used a simple feed-forward neural network architecture, which is a basic type of neural network that consists of an input layer, one or more hidden layers, and an output layer. For multi-class image classification, we used a Convolutional Neural Network (CNN) architecture, which is a type of neural network that is designed to work with image data. CNNs use a set of convolutional layers, pooling layers, and fully connected layers to learn features from images and classify them. Lastly, for sentimental analysis, we used a Recurrent Neural Network (RNN) architecture, which is a type of neural network that is designed to work with sequential data such as text. RNNs use a set of recurrent layers and fully connected layers to learn the features of the text and classify them.

2.3 Experiments

To evaluate the performance of the neural network models, we conducted experiments with varying hyperparameters, specifically batch size and learning rate. Batch size refers to the number of training samples that are used in one forward/backward pass during training, and learning rate refers to the step size at which the model learns from the training data. For each type of problem, we conducted experiments using batch sizes of 32, 64 and 128, and learning rates of 0.1, 0.01, and 0.001, comparing the performance of the models using both the SGD and Adam optimizer, which are commonly used optimization algorithms for training neural networks. Then, we evaluated the performance of the models based on their loss and accuracy, which are commonly used metrics for measuring the performance of neural network models.

2.4 Data Collection

In this study, we collected the loss and accuracy of each model variant, as well as the time taken for each experiment. We presented the data in both tabular and graphical formats to allow for a clear illustration and visualization of the observed trends. Collecting the data allows us to identify patterns and trends in the performance of the models and compare the effectiveness of different hyperparameters and optimization algorithms.

3 Results

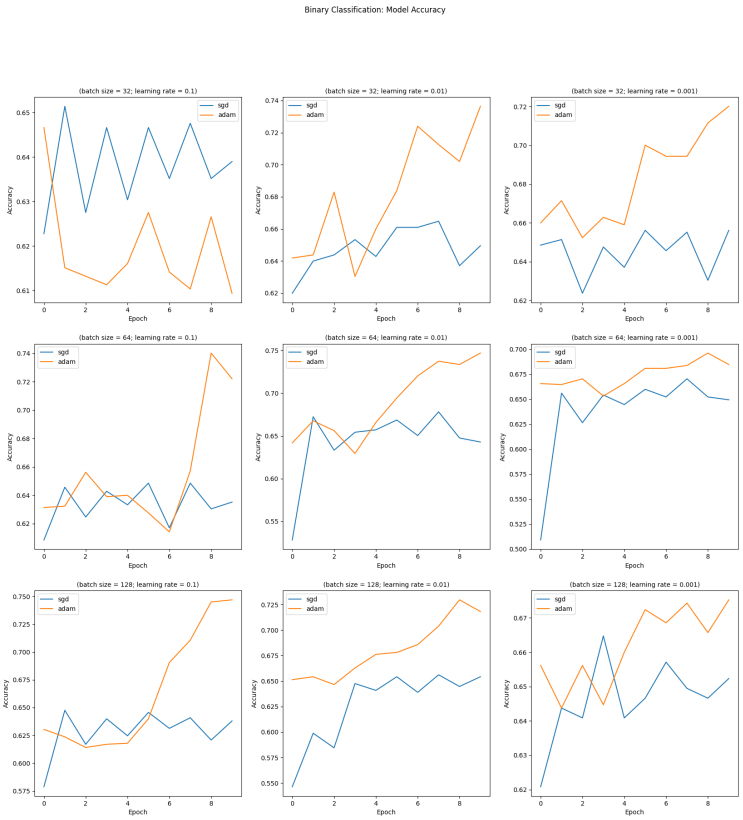
In order to evaluate the performance of the neural network models accurately, several metrics were used. These metrics included the loss and accuracy of the models, as well as the time taken for each experiment. The experiments were designed to test the models under a variety of conditions, including different batch sizes and learning rates. Furthermore, the models were trained using two different optimizers: the stochastic gradient descent (SGD) and the adaptive moment estimation (Adam) optimizers. By conducting experiments under these various conditions, we would expect to gain a more comprehensive understanding of the strengths and weaknesses of each model, as well as to identify which combinations of hyperparameters resulted in the best performance.

3.1 Binary Classification

For the binary classification problem using the titanic survival dataset, we trained the feed-forward neural network model with batch sizes of 32, 64, and 128 and learning rates of 0.1, 0.01, and 0.001. We compared the performance of the model using both the SGD and Adam optimizer. The results showing the time, loss and accuracy of the models trained with different hyperparameters and optimization algorithms are summarized in Table 1.

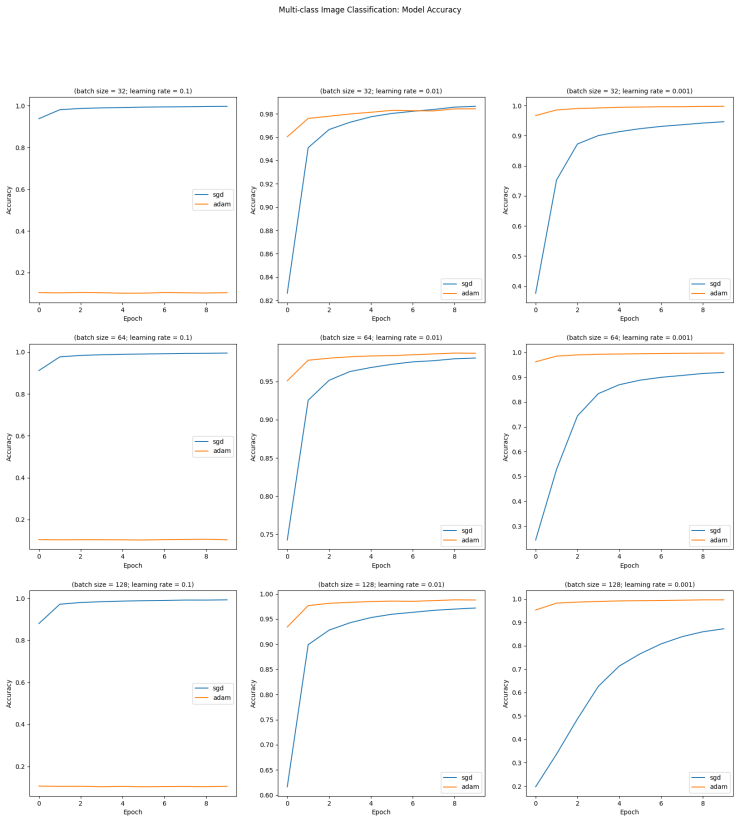
Batch Size	Learning Rate	Optimizer	Loss	Accuracy	Time
32	0.1	SGD	0.745	0.636	2.920
		Adam	0.654	0.636	3.698
	0.01	SGD	0.618	0.621	4.082
		Adam	0.485	0.780	3.553
	0.001	SGD	0.695	0.621	2.868
		Adam	0.513	0.682	4.802
64	0.1	SGD	0.628	0.636	3.022
		Adam	0.502	0.758	2.960
	0.01	SGD	0.627	0.629	3.345
		Adam	0.492	0.795	3.229
	0.001	SGD	0.753	0.674	2.497
		Adam	0.543	0.689	3.103
128	0.1	SGD	0.636	0.644	4.713
		Adam	0.582	0.750	2.692
	0.01	SGD	0.638	0.621	2.193
		Adam	0.497	0.705	2.662
	0.001	SGD	0.640	0.636	3.542
		Adam	0.557	0.689	2.678

Table 1: Performance of the feed-forward neural network model on the binary classification problem using the titanic survival dataset with varying batch sizes, learning rates, and optimizers.



Batch Size	Learning Rate	Optimizer	Loss	Accuracy	Time
32	0.1	SGD	0.040	0.990	82.754
		Adam	2.307	0.113	86.014
	0.01	SGD	0.055	0.983	82.123
		Adam	0.104	0.976	89.026
	0.001	SGD	0.189	0.946	82.779
		Adam	0.049	0.991	90.616
64	0.1	SGD	0.038	0.988	57.286
		Adam	2.312	0.099	57.438
	0.01	SGD	0.072	0.978	52.116
		Adam	0.086	0.980	53.568
	0.001	SGD	0.274	0.921	52.143
		Adam	0.036	0.990	57.751
128	0.1	SGD	0.074	0.976	40.361
		Adam	2.313	0.096	41.364
	0.01	SGD	0.098	0.973	42.382
		Adam	0.088	0.978	41.901
	0.001	SGD	0.440	0.881	42.922
		Adam	0.040	0.989	49.874

Table 2: Performance of the convolutional neural network model on the multi-class image classification problem using the MNIST with varying batch sizes, learning rates, and optimizers.



3.2 Multi-calss Image Classification

For the multi-class image classification problem, we used the MNIST dataset, which consists of 60,000 training images and 10,000 test images of handwritten digits from 0 to 9. We trained a CNN with two convolutional layers, two max pooling layers, and two fully connected layers. The input images were resized to 28x28 pixels and normalized.

The performance of the CNN was evaluated using both the SGD and Adam optimizer, with varying batch sizes and learning rates. The results of the experiments are summarized in Table 2.

3.3 Sentimental Analysis

For the sentiment analysis task, we used the IMDb dataset, which consists of movie reviews labeled as either positive or negative. And the results are as follows.

```
====================
sgd: batch_size = 32; learning_rate = 0.1
====================
--------------------
loss: 0.42167720198631287, accuracy: 0.7789999842643738
--------------------
====================
adam: batch_size = 32; learning_rate = 0.1
====================
--------------------
loss: 0.7367627024650574, accuracy: 0.4984000027179718
--------------------
====================
sgd: batch_size = 32; learning_rate = 0.01
====================
--------------------
loss: 0.560792088508606, accuracy: 0.722599983215332
--------------------
====================
adam: batch_size = 32; learning_rate = 0.01
====================
--------------------
loss: 0.4921780824661255, accuracy: 0.8633999824523926
--------------------
====================
sgd: batch_size = 32; learning_rate = 0.001
====================
--------------------
loss: 0.6928278207778931, accuracy: 0.49540001153945923
--------------------
====================
adam: batch_size = 32; learning_rate = 0.001
====================
--------------------
loss: 0.6226915121078491, accuracy: 0.873199999332428
--------------------
```

Note that the results of the Sentimental Analysis model is not completed due to insufficient memory. So, the kernel stopped while running. This seems to happen when we change the batch size from 32 to 64.

Also, it is also worth noting that the sentimental analysis task is inherently more difficult than the binary classification and multi-class image classification tasks. This is because sentiment analysis involves analyzing the meaning and context of a piece of text, which can be subjective and complex.

4 Discussion

4.1 Binary Classification

As we can see from the results, the Adam optimizer outperformed SGD in almost every scenario, with the exception of when the batch size was set to 32 and the learning rate was set to 0.1. In this specific case, SGD performed slightly better than Adam.

There could be a few reasons why this occurred. Firstly, it is possible that the specific combination of batch size and learning rate values created a scenario in which SGD was better suited to the problem at hand. For example, when the batch size is small and the learning rate is large, the model may be more likely to converge faster with SGD compared to Adam, which could lead to better performance. Or perhaps, this just happened by chance.

4.2 Multi-class Image Classification

As demonstrated by the above results, the performance of the SGD optimizer compared to Adam in the MNIST CNN model is quite interesting.

More specifically, we observed that When the learning rate is set to 0.1, SGD outperforms Adam in all scenarios. In fact, it would be more accurate to say that the Adam does not work at all in such cases. This may be due to the fact that a high learning rate with Adam optimizer can lead to overshooting the minimum loss and result in poor convergence. In contrast, SGD optimizer with a high learning rate can still converge well as the updates to the weights are proportional to the gradient and the learning rate, which allows for more aggressive updates.

Also, when the learning rate is set to 0.01, both Adam and SGD have similar performance in most scenarios. This suggests that the learning rate is within a range that is suitable for both optimizers to perform well.

However, when the learning rate is set to 0.001, Adam significantly outperforms SGD in most scenarios. This could be because with such a low learning rate, SGD optimizer might get stuck in a suboptimal local minimum while Adam optimizer with its adaptive learning rate can still find a better minimum.

So, from this, it would be safe to say that in CNN model, a higher value of learning rate is preferred by SGD, while Adam optimizer would perform better with a lower value of learning rate.

4.3 Sentimental Analysis

The behavior of the SGD and Adam optimizers in the RNN model trained on the IMDb reviews dataset follows a similar trend to that observed in the CNN model trained on the MNIST dataset.

When the learning rate is set to a high value such as 0.1, the SGD optimizer performs better than the Adam optimizer. This could be because the SGD optimizer is more aggressive in its weight updates and can take larger steps towards the minimum of the loss function. This may allow the optimizer to escape local minima and converge faster towards the global minimum.

On the other hand, when the learning rate is set to a low value such as 0.001, the Adam optimizer outperforms the SGD optimizer by a large margin. This could be because the Adam optimizer is better equipped to handle small learning rates by adapting the learning rate on a per-parameter basis. This allows the optimizer to make more precise updates to the weights and avoid overshooting the minimum of the loss function.

When the learning rate is set to an intermediate value such as 0.01, both the SGD and Adam optimizers perform similarly, with a slight edge to the Adam optimizer. This suggests that the Adam optimizer is better suited for this task, as it is able to make more efficient use of the learning rate than the SGD optimizer.

5 Conclusion

All in all, from this report, we can conclude that the behavior of the SGD and Adam optimizers depend on various factors such as the learning rate, batch size, and the complexity of the problem at hand. In the binary classification task, Adam outperformed SGD in most scenarios except for one specific combination of batch size and learning rate. In the multi-class image classification task, SGD outperformed Adam when the learning rate was high, while Adam outperformed SGD when the learning rate was low. For an intermediate learning rate, both optimizers performed similarly with a slight edge to the Adam optimizer. In the sentimental analysis task, the behavior of the optimizers was similar to that observed in the multi-class image classification task. Overall, the results suggest that the choice of optimizer should be carefully considered based on the specific problem and the values of the hyperparameters.

References

[1] Zhang, Z. (2018, June). Improved adam optimizer for deep neural networks. In 2018 IEEE/ACM 26th international symposium on quality of service (IWQoS) (pp. 1-2). Ieee.

[2] Desai, C. (2020). Comparative analysis of optimizers in deep neural networks. International Journal of Innovative Science and Research Technology, 5(10), 959-962.

[3] Vani, S., & Rao, T. M. (2019, April). An experimental approach towards the performance assessment of various optimizers on convolutional neural network. In 2019 3rd international conference on trends in electronics and informatics (ICOEI) (pp. 331-336). IEEE.