Report analyzing the impact of Batch normalization and different Gradient descent methods, including the role of Autoencoders in Traffic Flow Prediction

Introduction

This report highlights the performance of various gradient descent optimization methods, including Adam and Adagrad, applied to a traffic flow prediction model. We leveraged autoencoders for feature extraction and batch normalization to improve model stability. The primary goal was to evaluate how different gradient descent strategies affect both accuracy and loss across epochs.

Data Preprocessing

1. Dataset: The traffic dataset consists of features such as traffic flow, average speed, occupancy, temperature, humidity, wind speed, precipitation, visibility, and road conditions.

2. Preprocessing: One-hot encoding was applied to categorical variables like **Road Condition** and **Traffic Congestion**, while numerical features were standardized using a *StandardScaler*.

3. Target Variable: The categorical variable **Traffic Congestion** was encoded for binary classification.

Model Architecture

1. Autoencoder: A stacked autoencoder with dense layers was used to compress and reconstruct the input data. The final model exhibited a training loss of **0.2920** and a validation loss of **0.2935** after 50 epochs.

2. Neural Network: A feedforward neural network, equipped with batch normalization and trained with various gradient descent techniques, was employed for binary classification of **Traffic Congestion**.

Gradient Descent Optimization Approaches

The following strategies were applied:

1. Full Batch Gradient Descent: Adam and Adagrad optimizers with batch size equal to the dataset size.

2. Mini-Batch Gradient Descent: Adam and Adagrad optimizers with a batch size of 32.

3. Stochastic Gradient Descent: Adam and Adagrad optimizers with a batch size of 1 (SGD).

Results Overview

Accuracy Comparison

Optimizer	Full Batch Accuracy	Mini Batch Accuracy	SGD Accuracy
Adam	0.61	0.62	0.32
Adagrad	0.33	0.55	0.31

Loss Comparison

Optimizer	Full Batch Loss	Mini Batch Loss	SGD Loss
Adam	-50	-150	-12.78
Adagrad	-30	-500	-12.80

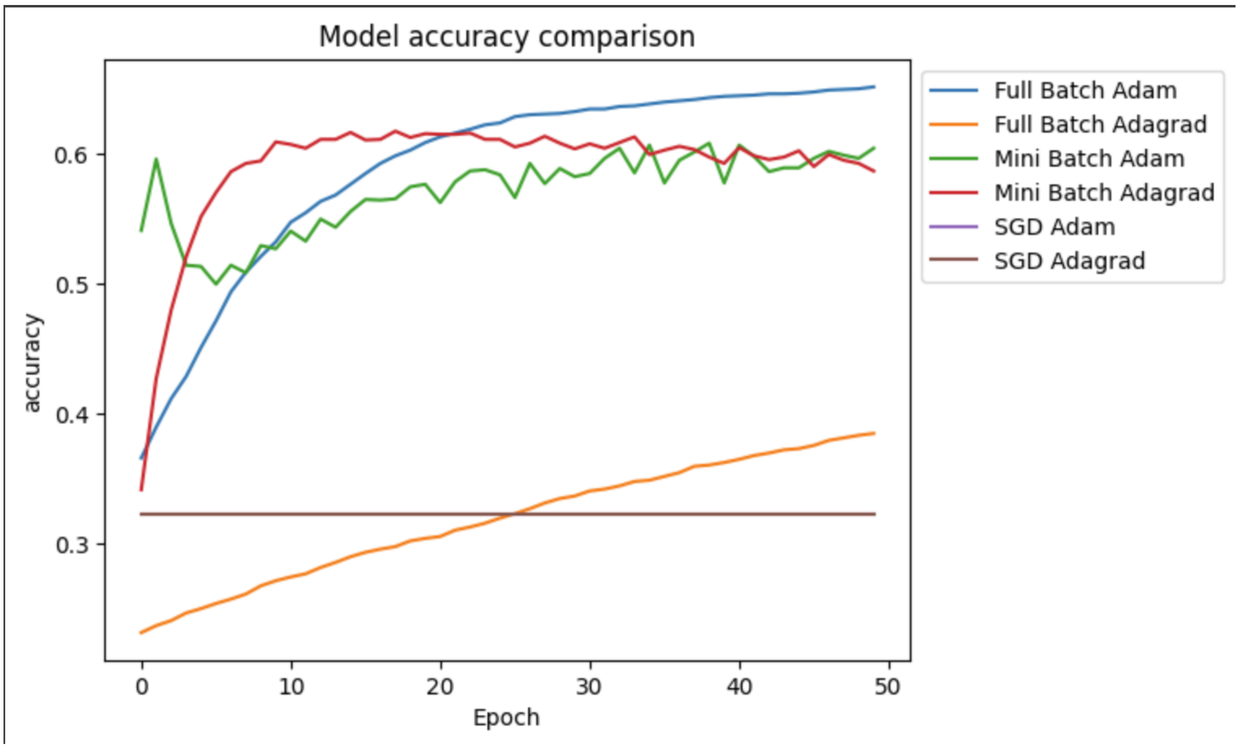
Key Observations

1. Accuracy: The mini-batch Adam optimizer exhibited the best performance, achieving an accuracy of around **0.62** across 50 epochs. Both full batch Adam and mini-batch Adagrad also showed competitive accuracy, while stochastic methods, particularly with Adagrad, struggled to maintain stable accuracy.

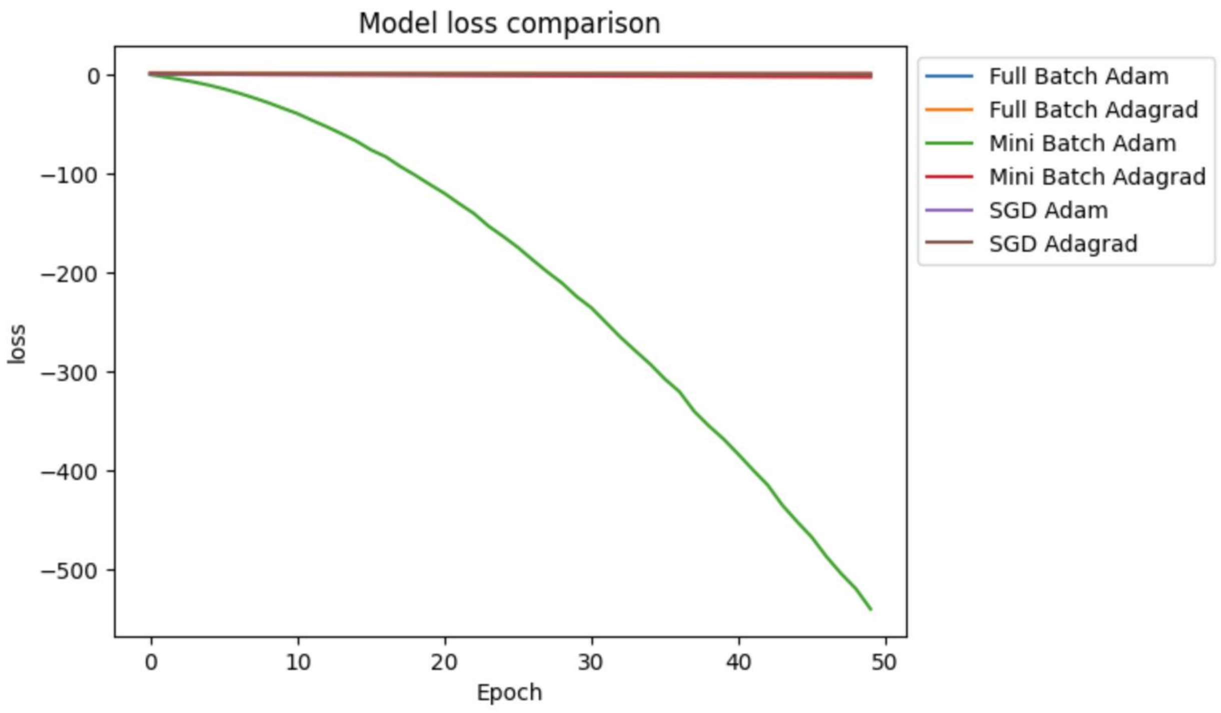
2. Loss: The loss curve for mini-batch Adagrad displayed the most dramatic decrease, reaching a loss of **-500** by epoch 50. However, stochastic methods (SGD) showed significantly higher final loss values, indicating slower convergence or suboptimal training behavior.

Visual Comparison

The attached plots clearly show how each method performs across epochs in terms of accuracy and loss:



1. The accuracy of mini-batch Adam stabilizes around 0.62, outperforming the others.



2. The loss for mini-batch Adagrad decreases sharply, reaching -500, demonstrating superior convergence.

Conclusion

The experiment demonstrates that mini-batch gradient descent with the Adam optimizer is the most effective approach for this traffic flow prediction task, yielding the highest accuracy. Meanwhile, mini-batch Adagrad excels in loss reduction, though its accuracy trails slightly behind Adam. Both full batch and stochastic gradient descent methods underperform in this scenario.

These insights suggest that a combination of mini-batch processing with adaptive optimizers like Adam could be a promising strategy for future work on traffic flow prediction and other time-series tasks.