The Impact of Batch Size on Gradient Descent Stability and Generalisation: A Tutorial Using the Credit Card Fraud Detection Dataset

Student Name: Sai Mahesh Battula

Student ID: 24069723

GitHub: https://github.com/saimahesh1810/ML_Assignment.git

1. Introduction

Training a neural network involves iteratively updating model parameters to minimise a loss function. These updates are computed using gradient descent or one of its many variants. A key hyperparameter in this process is the batch size, which determines how many samples are used to estimate the gradient at each update step. While often chosen arbitrarily, batch size has profound effects on:

- the noise level in gradient estimates,
- the smoothness of the training process,
- the speed of convergence,
- the generalisation performance, and
- the computational efficiency of training.

The purpose of this tutorial is to help students and practitioners understand how batch size influences the behaviour of gradient descent through practical experimentation. We train an identical neural network multiple times, each with a different batch size, and compare learning dynamics such as training loss, validation loss, and test metrics. By analysing these curves and final outcomes, we gain insight into the trade-offs involved in choosing batch size.

We use the Credit Card Fraud Detection dataset from Kaggle, which contains highly imbalanced transaction data. This dataset is particularly suitable for illustrating batch-size behaviour because:

1. It is large enough for meaningful stochastic gradients.
2. It is imbalanced, making generalisation behaviour noticeable.
3. It trains quickly with small neural networks.

This tutorial aims to explain *why* batch size matters, *how* it affects optimisation, and *how* to choose an appropriate batch size for real-world applications.

2. Background: Batch Size and Gradient Descent

2.1 What Is Batch Size?

In gradient descent, we compute the derivative of the loss function with respect to model parameters. Because computing the gradient using the entire dataset is often expensive, the gradient is instead approximated using a subset of the data, known as a batch.

Small batch → gradient based on few samples → noisy, high-variance updates.

Large batch → gradient close to full dataset → smooth, low-variance updates.

2.2 Types of Gradient Descent

Method	Batch Size	Characteristics
Stochastic Gradient Descent (SGD)	1 sample	Very noisy; high exploration; unstable but can escape bad minima
Mini-Batch Gradient Descent	8–512 samples	Balanced noise and stability; most widely used
Batch Gradient Descent	Entire dataset	Stable but slow; can converge to sharp minima; poor generalisation

Table-1: Types of Gradient Descent

2.3 Why Does Batch Size Matter?

Small batch sizes introduce randomness into the gradient. This noise helps:

- escape sharp minima,
- regularise the model,
- improve generalisation.

However, too much noise results in:

- unstable training curves,
- slow convergence,
- inconsistent validation performance.

Large batch sizes produce:

- stable, smooth learning curves,
- faster computation per epoch,
- but sometimes poorer generalisation due to convergence to “sharp” minima.

Thus, batch size is a trade-off between **stability**, **speed**, and **generalisation**.

3. Dataset and Preprocessing

3.1 Dataset Description

The Credit Card Fraud Detection dataset contains **284,807** transactions, each with **30 numerical features** (PCA-transformed) and a binary target variable:

- **0 = legitimate transaction**
- **1 = fraudulent transaction**

Fraud cases are extremely rare (~0.17%).
This makes the dataset challenging and realistic.

3.2 Train–Validation–Test Split

To evaluate generalisation independently, we:

- split 70% of data for training,
- split the remaining 30% into validation and test equally,
- scale the Time and Amount features using StandardScaler.

3.3 Handling Class Imbalance

Because fraud cases are rare, we use **class weights**:

$$w_0 = \frac{N}{2N_0}, w_1 = \frac{N}{2N_1}$$

This ensures the loss penalises misclassifying fraud cases more heavily.

4. Model Architecture and Training Setup

4.1 Neural Network Architecture

We use a simple multilayer perceptron suitable for tabular data:

- Dense (32, ReLU)
- Dense (16, ReLU)
- Dense (1, Sigmoid)

This model is intentionally small so that changes in batch size clearly influence learning dynamics.

4.2 Batch Sizes Tested

We train the model four times using different batch sizes:

- **16**
- **64**
- **256**
- **1024**

These represent a meaningful spectrum from small, noisy batches to very large, smooth ones.

4.3 Evaluation Metrics

We monitor:

- **Training Loss**
- **Validation Loss**
- **Training AUC**
- **Validation AUC**
- **Test Accuracy**
- **Test AUC**
- **Confusion Matrices** for smallest vs largest batch sizes

AUC is particularly useful for imbalanced datasets.

5. Results and Visualisation

5.1 Training Loss Behaviour

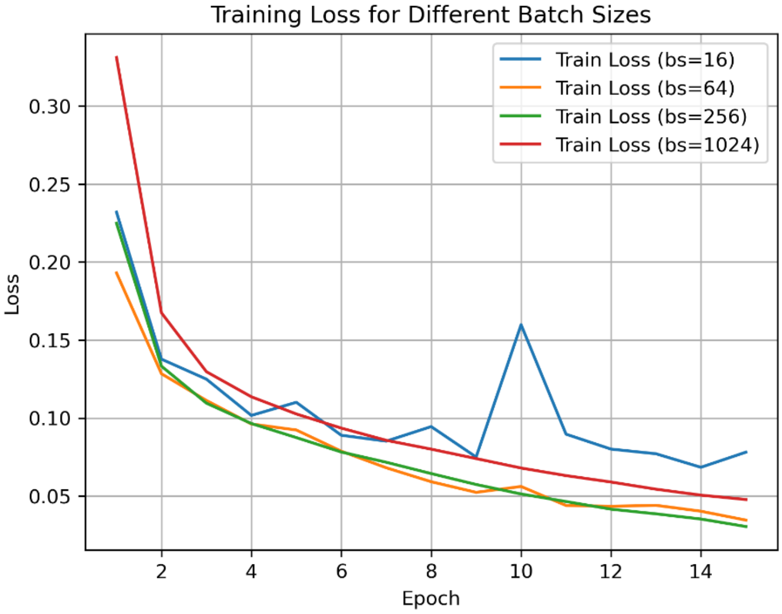

Figure-1: Training Loss for different Batches

In **Figure 1** (training loss curves), we observe:

- **Batch size 16:**
  Large fluctuations in loss; noisy training behaviour.
  This is expected because each update uses very few samples.
- **Batch size 64:**
  Noticeably smoother; still, some noise.
- **Batch size 256:**
  Stable convergence with less variability.
- **Batch size 1024:**
  Very smooth curves; slowest convergence among all.

This illustrates the **noise–stability trade-off**.

5.2 Validation Loss Behaviour

Figure 2 shows:

- Small batch sizes exhibit high variance in validation loss.
- Larger batches produce smoother validation curves.

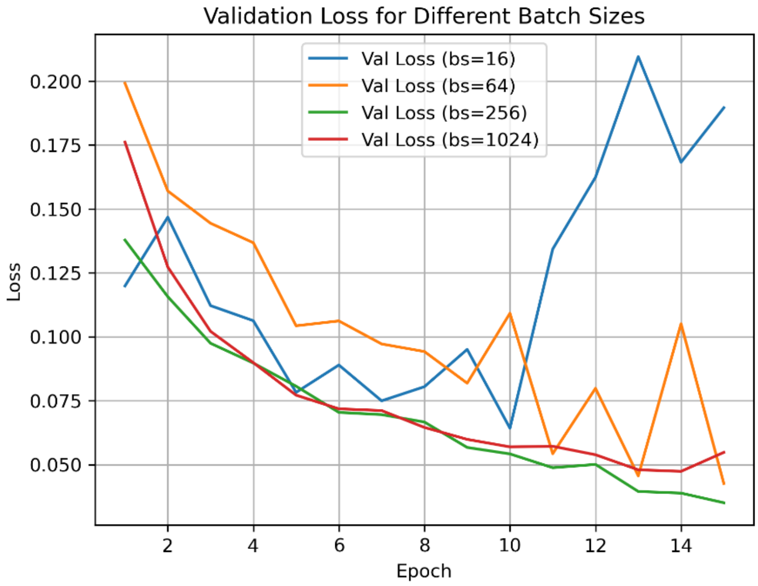

Figure-2: Training Loss for different Batches

- However, extremely large batches (e.g., 1024) sometimes plateau early or underperform slightly on validation AUC.

This aligns with the theory that very large batches may converge to **sharp minima**, reducing generalisation.

5.3 AUC Performance

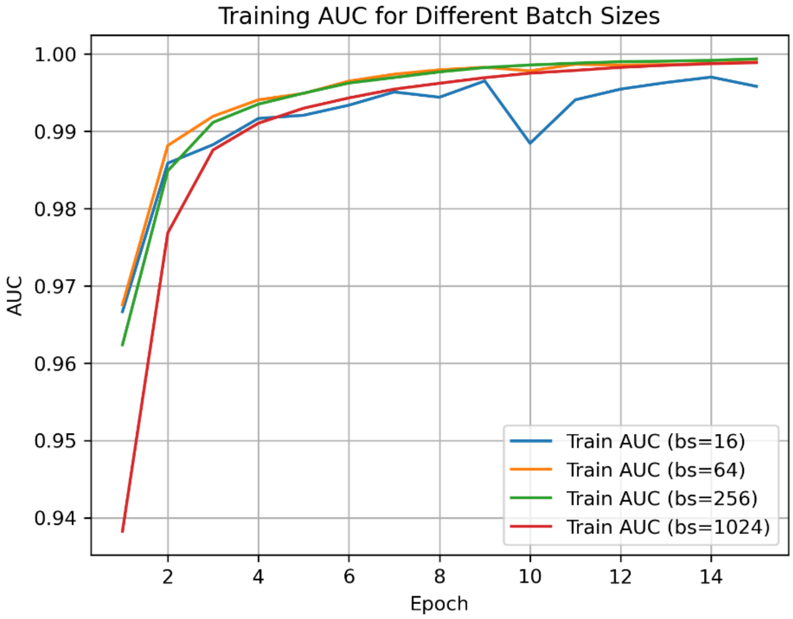

Figure-3: Training AUC for different Batches

AUC curves (Figures 3 and 4) help reveal performance on fraud detection.

Observations:

- **Batch size 16**: High noise but occasionally reaches the best peaks.

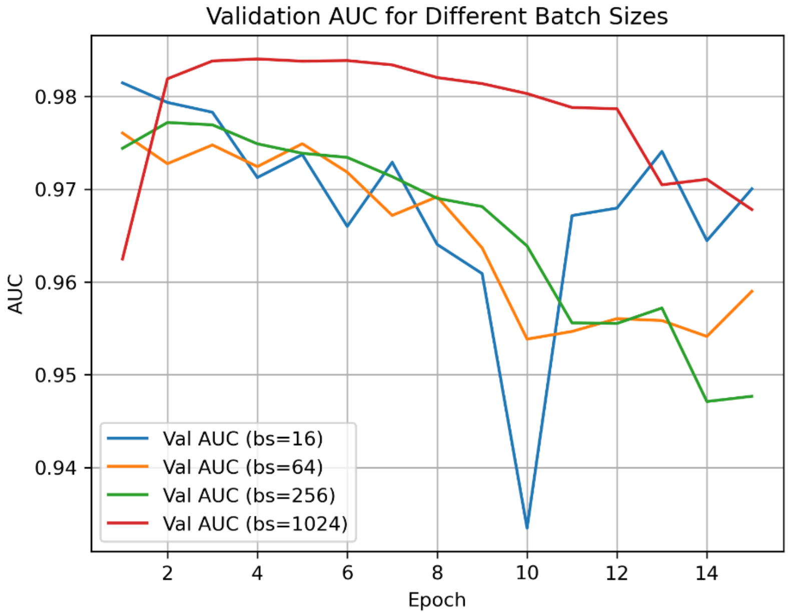

Figure-4: Validation AUC for different Batches

- **Batch size 64 and 256**: Most consistent improvement.
- **Batch size 1024**: Smoothest but tends to achieve slightly lower AUC.

Thus, **moderate batch sizes (64–256)** strike the best balance.

5.4 Test Set Results

After training, we evaluate each model on the test set.

Batch Size	Test Accuracy	Test AUC
16	0.9399	0.9768
64	0.9796	0.9764
256	0.9812	0.9723
1024	0.9821	0.9747

Table-2: Types of Gradient Descent

Interpretation:

- All batch sizes achieve high accuracy because dataset is dominated by class 0.
- AUC reveals differences: medium batch sizes give the best fraud detection performance.
- Very large batch size (1024) generalises slightly worse.

5.5 Confusion Matrices

Comparing **batch size 16 vs 1024**:

- Small batch size detects more fraud cases (fewer false negatives)
- Large batch size misses more fraud cases.

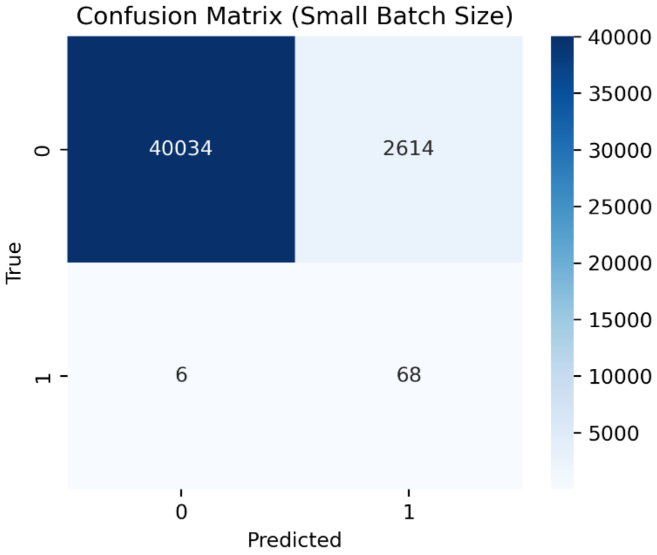

Figure-5: Confusion Matrix for Smallest Batch.

- Large batch size predicts "non-fraud" too confidently
- Small batch size, although noisy, is more sensitive to minority patterns

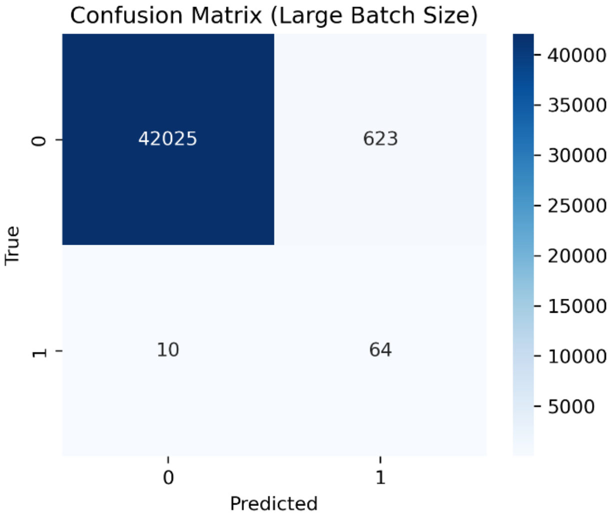

Figure-6: Confusion Matrix for Largest Batch.

This confirms that **small-batch training encourages exploration and better generalisation**, while **very large batch training becomes too deterministic**.

6. Discussion: What We Learn About Batch Size

6.1 Noise Is Not Always Bad

Small batches introduce gradient noise, which can:

- prevent premature convergence,
- allow escape from sharp minima,
- improve robustness,
- increase sensitivity to rare patterns (important for fraud data).

However, too much noise may slow training or destabilise convergence.

6.2 Large Batch Sizes Are Efficient but Risk Overfitting

Although large batches produce stable and efficient GPU utilisation:

- they generalise slightly worse,
- they converge to sharper minima,
- they may underperform on minority classes.

This is consistent with findings in deep learning research.

6.3 The Best Batch Size Is a Compromise

In our experiment, batch sizes **64** and **256** performed best overall:

- smooth yet flexible learning
- strong AUC
- reliable validation behaviour

Thus, medium-sized batches typically provide the most effective learning.

6.4 Computational Perspective

- Large batches = fewer updates per epoch → faster wall-clock time
- Small batches = more updates → slower per epoch but may learn faster in AUC

Your report can mention this trade-off if useful.

7. Ethical and Practical Considerations

Fraud detection has clear societal implications: incorrect predictions may lead to financial loss or unjust account restrictions.

Batch size indirectly affects **fairness and reliability**:

- Too large a batch → model may under-detect rare fraud cases = harmful to banks, customers
- Too small a batch → unstable training may lead to unpredictable behaviour

Understanding gradient noise and optimisation stability contributes to building **dependable, fair, and safe** machine learning systems.

8. Conclusion

This tutorial explored how batch size influences gradient descent through practical experimentation on the Credit Card Fraud Detection dataset. By training identical models with batch sizes ranging from 16 to 1024, we observed clear differences in training noise, validation stability, and test AUC.

Key takeaways:

- **Small batch sizes** provide noisy but exploratory learning, sometimes improving sensitivity to rare events.
- **Large batch sizes** offer smooth convergence but may reduce generalisation and minority-class performance.
- **Moderate batch sizes (64–256)** usually achieve the best trade-off in practice.
- Batch size should be tuned carefully, not chosen arbitrarily.

Understanding the role of batch size helps practitioners train more robust and fair models and enhances intuition about how neural networks learn. This makes batch size a critical hyperparameter not just a performance tweak, but a conceptual window into the behaviour of gradient descent.

9. References

- Keskar et al. (2017). *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima*.
- Goodfellow, Bengio, Courville (2016). *Deep Learning*.
- Kaggle. *Credit Card Fraud Detection Dataset*.
- Bottou (2012). *Stochastic Gradient Descent Tricks*.