Adi Maman – adimaman22@gmail.com

Eran Fishbein – eranfish76@gmail.com

Deep Learning Project -

Neural Network Implementation for MNIST Classification

Introduction

The aim of this project was to design and experiment with a neural network implemented from scratch using PyTorch. This project provided a deep understanding of forward and backward propagation processes, training neural networks, and improving performance through techniques such as batch normalization and L2 regularization.

Implementation Details

1. **MNIST Dataset Loading and Preprocessing**
   - The MNIST dataset was divided into three subsets:
     - Train (80%)
     - Validation (20% of training set)
     - Test (20%)
   - **Data Preprocessing** included:
     - **Normalization**: Pixel values were scaled to the range [0, 1] by dividing by 255 (X/255).
     - **One-Hot Encoding**: Labels (Y) were encoded into a one-hot format.
     - **Flattening**: Each 28×28 grayscale image was reshaped into a one-dimensional vector of size 784.

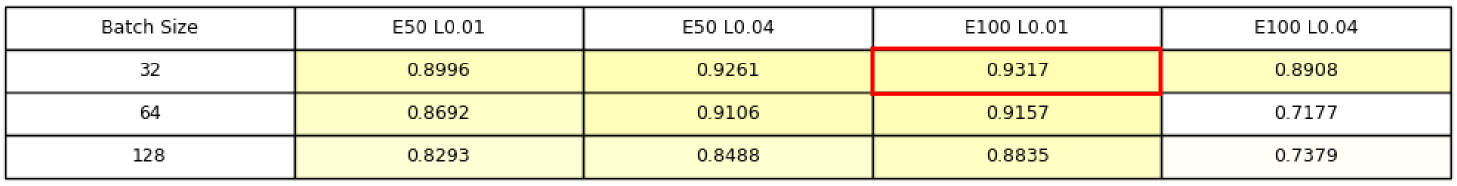
2. **Network Configuration:**

   A four-layer architecture was implemented with dimensions:
   [784 (input layer), 20, 7, 5, 10 (output layer)].
   - **Activation functions:**
     - ReLU for hidden layers.
     - SoftMax for the output layer.
   - **Training Hyperparameters:**
     - Learning Rate: 0.009
     - Batch Sizes: {**32**, 64, 128}
     - Maximum Number of Epochs: {50, **100**}
     - L2 Norm Regularization: {**0.01**, 0.04}

     * These hyperparameters were selected experimentally to optimize the model's performance.
   - **Stopping Criteria:** Training was stopped when there was no improvement in validation cost for 100 consecutive iterations with a threshold of 0.00001.

3. **Reasoning for Parameter Selection:**
   Based on the experimental results, we analyzed the impact of various hyperparameters on model performance, as visualized in the provided graphs.

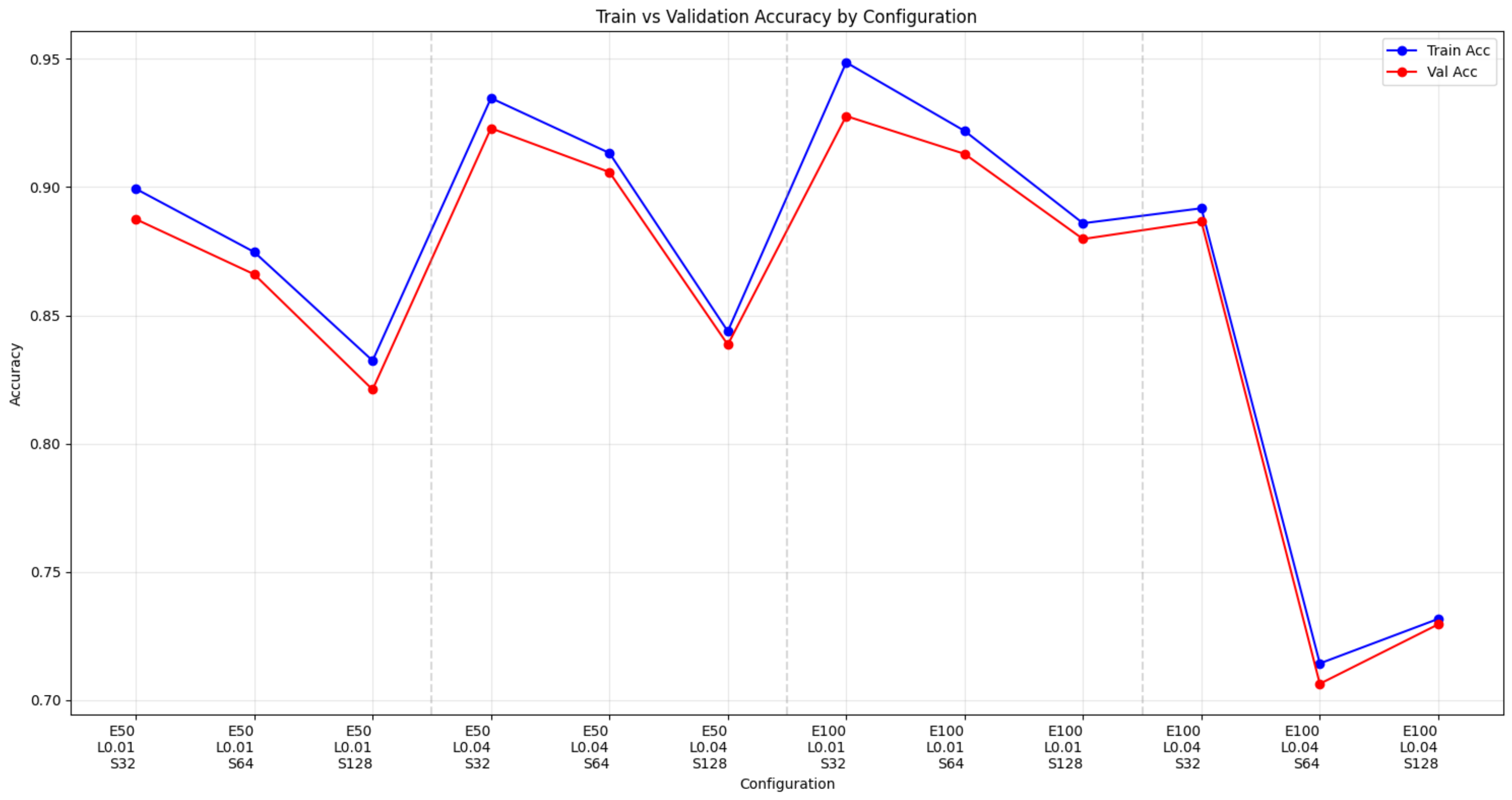
Test Accuracy for Different Configurations:



Batch Size	E50 L0.01	E50 L0.04	E100 L0.01	E100 L0.04
32	0.8996	0.9261	0.9317	0.8908
64	0.8692	0.9106	0.9157	0.7177
128	0.8293	0.8488	0.8835	0.7379

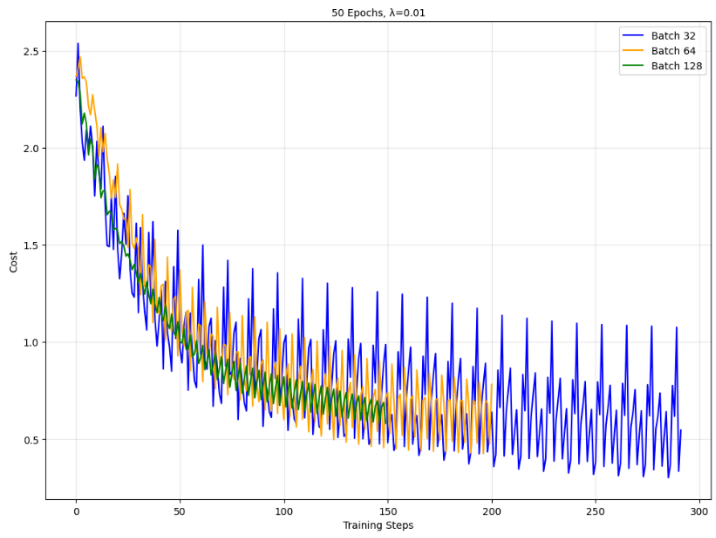
The table demonstrates that **batch size 32 consistently outperformed** larger batch sizes (64 and 128) across all configurations. Additionally, the combination of **100 epochs** and **L2 regularization (λ = 0.01)** achieved the highest test accuracy of 93.17%. This result highlights that smaller batch sizes, paired with sufficient training time and moderate regularization, provide the best balance between learning efficiency and generalization performance.

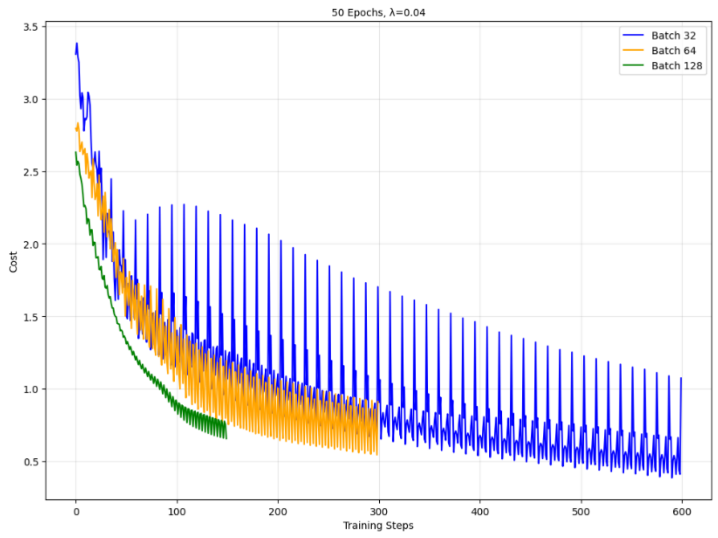
Train Vs. Validation Accuracy graph:

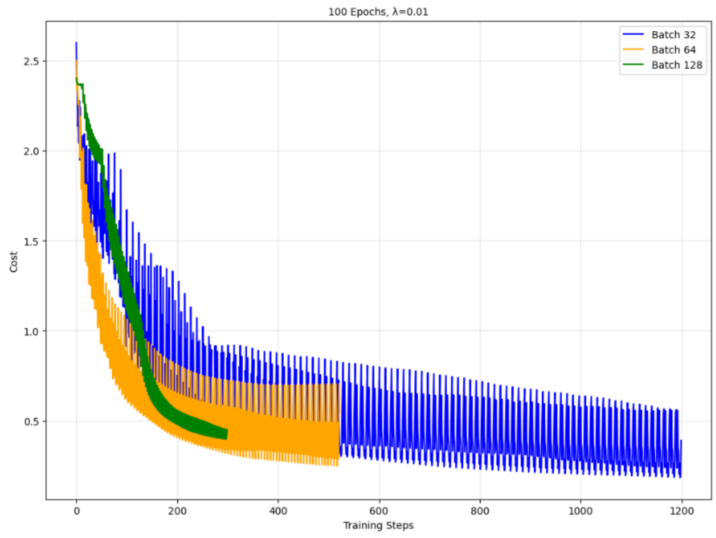


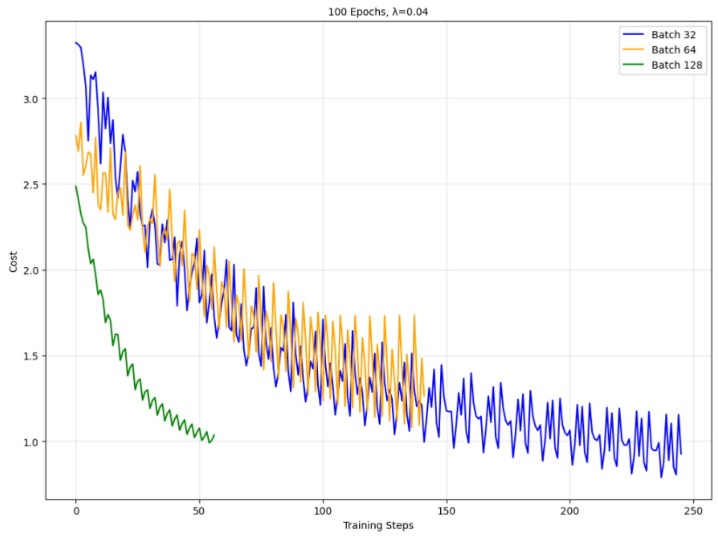
The graph emphasizes that the selected configuration (**batch size 32, 100 epochs, λ = 0.01**) outperformed all others in both training and validation accuracy. The consistently small gap between training and validation accuracies across the experiments indicates minimal overfitting.

Cost Vs. Training Steps graphs:









The cost graphs further validate that batch size 32 achieved smoother and faster convergence across all configurations. The best-performing combination of **100 epochs** and **λ = 0.01** (Bottom-Left Graph) demonstrated consistently lower costs during training compared to alternative configurations (batch sizes 64 and 128).

Conclusions on Parameter Selection:

During the experiments, we did not observe significant overfitting in the training or validation results that would justify using a larger regularization value (λ).

Therefore, we opted for a relatively small value of λ=0.01, which provided an effective balance between preventing overfitting and maintaining model flexibility.

4. **Batch Normalization:**
   - Application: Batch normalization was applied after activation (apply_batchnorm(A)) to improve stability and convergence.
   - Functionality: Batch normalization normalizes the activations of a layer to have a mean of 0 and a variance of 1.

5. **L2 Norm Regularization:**
   - **Lambda (λ)**: Experimentally set to 0.01 for optimization.
   - When λ=0, the regularization term is zero, meaning no regularization is applied.

6. **Run Results:**

Experiment	1	2	3	4
Apply Batchnorm	FALSE	TRUE	FALSE	TRUE
L2 Regularization (λ)	L2 = 0	L2 = 0.01	L2 = 0.01	L2 = 0
Batch Size	32	32	32	32
Epochs	99	99	99	99
Iterations	120001	120001	120001	120001
Train Accuracy	93.09%	93.42%	94.23%	89.49%
Validation Accuracy	91.53%	92.90%	92.73%	88.20%
Test Accuracy	91.93%	93.13%	92.93%	89.41%
Training Time (s)	138.77	172.91	142.4	173.57
Train Final Cost	0.613	263576.84	0.7044	0.8635
Validation Final Cost	0.3386	878.8498	0.263	0.3787

The table above summarizes the results of the four experiments, highlighting the effect of batch normalization and L2 regularization on accuracy, cost, and training time.

<u>**Conclusions**</u>:

Impact of L2 Regularization:

Adding L2 regularization (λ=0.01) demonstrated a clear improvement in generalization, as observed in Experiments 2 and 3, both of which achieved higher validation and test accuracies compared to experiments without regularization.

Specifically, in Experiment 3, L2 regularization improved train, validation, and test accuracies compared to Experiment 1 (No L2 Regularization, No Batch Normalization). While the final training cost increased slightly with L2 regularization, the validation cost decreased, indicating enhanced performance on unseen data.

These results highlight the effectiveness of L2 regularization in reducing overfitting while preserving computational efficiency. For scenarios requiring better generalization, Experiment 3 clearly outperformed Experiment 1.

Impact of Batch Normalization:

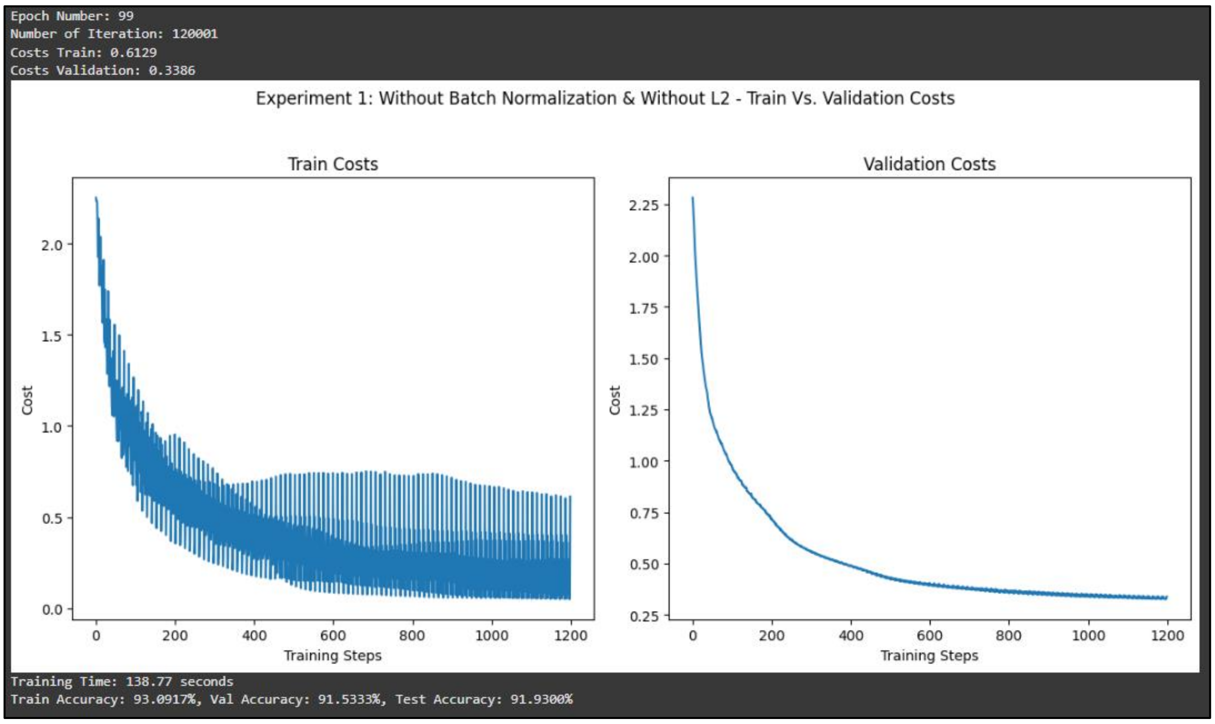
Batch normalization, when applied without regularization (Experiment 4), does not necessarily improve model performance. Instead, it led to lower accuracy and higher costs (Experiments 2 and 4).

Trade-offs in Combining Batch Normalization and L2 Regularization:

While Experiment 2 achieved the best accuracy, it also exhibited significantly higher training and validation costs compared to other experiments. This outcome suggests that the interaction between batch normalization and L2 regularization, while beneficial for accuracy, may introduce numerical instability, leading to irregular cost values. Despite this discrepancy, the model maintained strong generalization performance, indicating that the high cost values do not hinder its effectiveness.

Note: All cost values during training, logged every 100 training steps, are detailed in the attached notes for each experiment.

Experiment 1: Without Batch Normalization and Without L2 Regularization

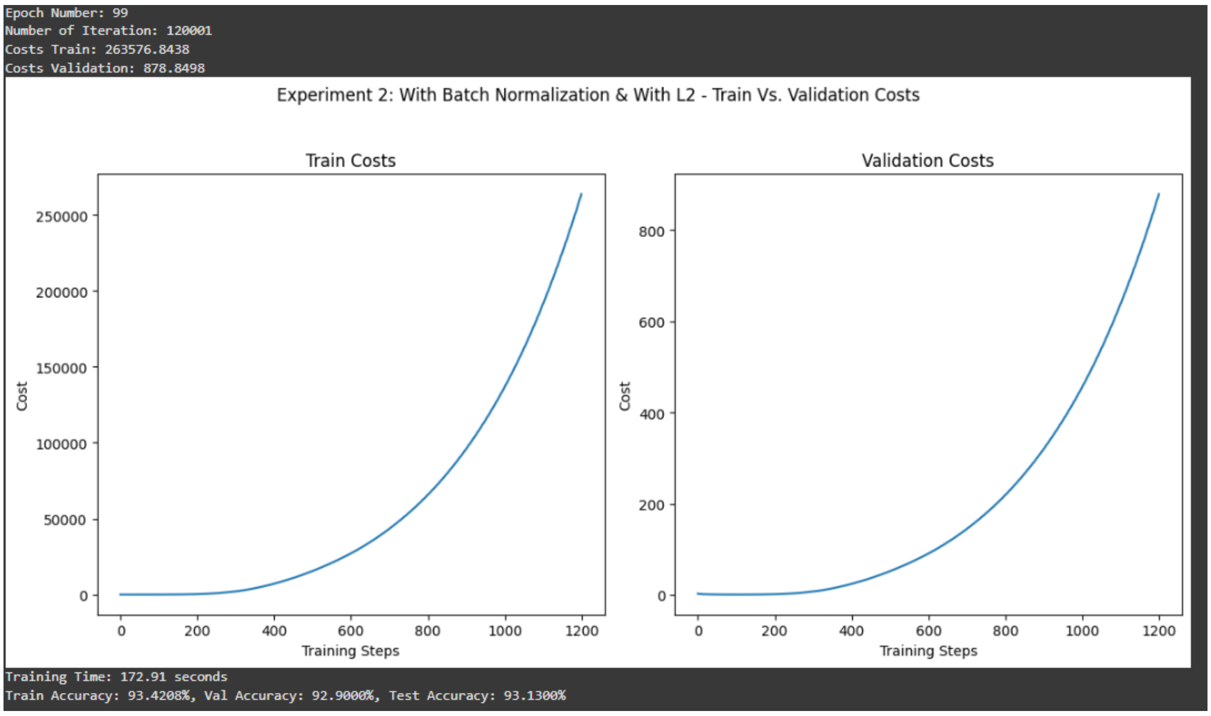


```
Epoch Number: 99
Number of Iteration: 120001
Costs Train: 0.6129
Costs Validation: 0.3386
```

Experiment 1: Without Batch Normalization & Without L2 - Train Vs. Validation Costs
Train Costs
Validation Costs
Cost
Training Steps

```
Training Time: 138.77 seconds
Train Accuracy: 93.0917%, Val Accuracy: 91.5333%, Test Accuracy: 91.9300%
```

Experiment 2: With Batch Normalization & With L2 Regularization



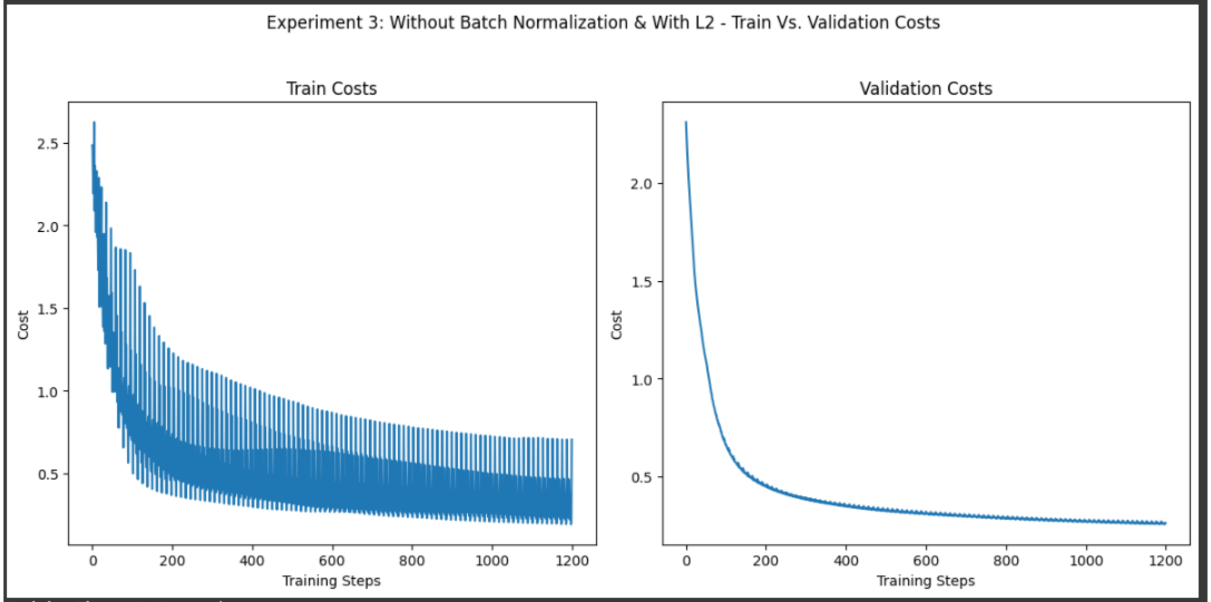
```
Epoch Number: 99
Number of Iteration: 120001
Costs Train: 263576.8438
Costs Validation: 878.8498
```

Experiment 2: With Batch Normalization & With L2 - Train Vs. Validation Costs
Train Costs
Validation Costs
Cost
Training Steps

```
Training Time: 172.91 seconds
Train Accuracy: 93.4208%, Val Accuracy: 92.9000%, Test Accuracy: 93.1300%
```

Experiment 3: Without Batch Normalization & With L2 Regularization

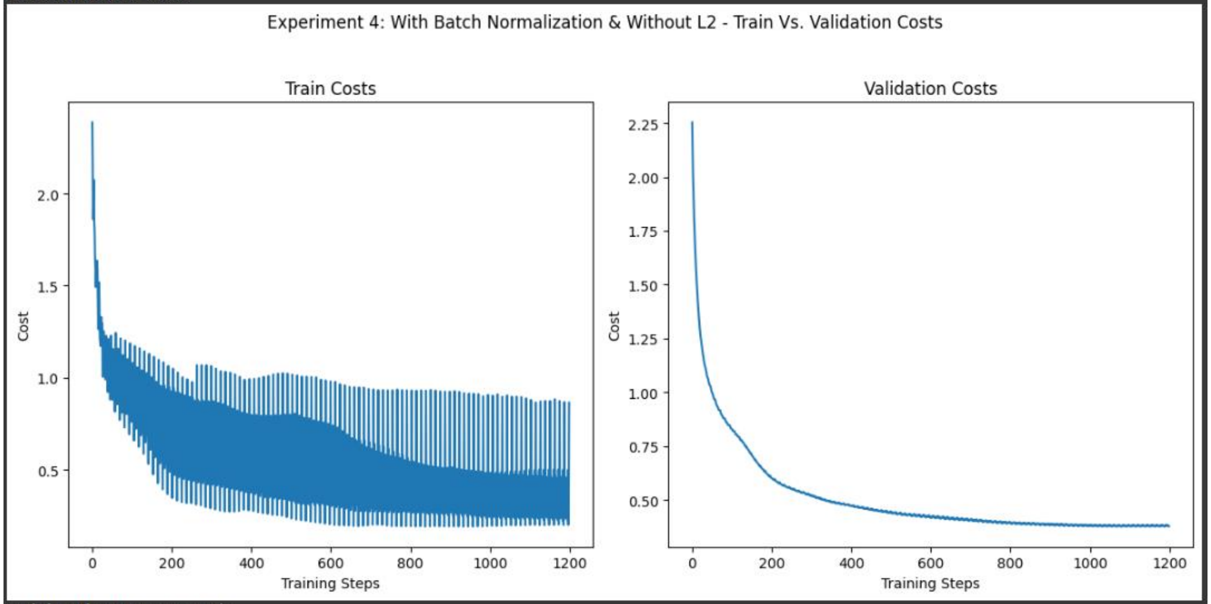
```
Epoch Number: 99
Number of Iteration: 120001
Costs Train: 0.7044
Costs Validation: 0.263
```



```
Training Time: 142.40 seconds
Train Accuracy: 94.2271%, Val Accuracy: 92.7333%, Test Accuracy: 92.9300%
```

Experiment 4: With Batch Normalization & Without L2 Regularization

```
Epoch Number: 99
Number of Iteration: 120001
Costs Train: 0.8635
Costs Validation: 0.3787
```



```
Training Time: 173.57 seconds
Train Accuracy: 89.4979%, Val Accuracy: 88.2000%, Test Accuracy: 89.4100%
```

Q4-5 - Effect of Batch Normalization:

Comparison of Experiments 1 (No Batch Normalization, No L2 Regularization) **and 4** (With Batch Normalization, No L2 Regularization)

Performance:

- Accuracy: Adding batch normalization in Experiment 4 reduced both training and generalization performance (validation and test accuracies) compared to Experiment 1.
- Cost: Batch normalization increased the training and validation final costs slightly. This indicates that while the weights stabilized during training, the learning process was less effective compared to experiment 1.

Runing Time:

Experiment 4 took 173.57 seconds, about **25% longer** than Experiment 1 (138.77 seconds). The additional computation for batch normalization layers accounts for this difference.

Convergence:

The cost graphs suggest that Experiment 1 converged faster and reached lower overall costs than Experiment 4.

This further supports the observation that batch normalization in this configuration might have hindered performance.

Q6 - L2 regularization:

Explanation of Changes for L2 Norm Functionality

1. Cost Function Modification (compute_cost()):

- **Purpose**: L2 regularization was incorporated to mitigate overfitting.
- **Updated Formula**: $\text{cost} = -\frac{1}{m}\sum(y\log(\hat{y})) + \underbrace{\frac{\lambda}{2m}\sum W^2}_{\text{Regularization}}$

- **Function Signature:**

  Original Signature: compute_cost(AL, Y)

  updated signature: compute_cost(AL, Y, parameters=None, L2=0.0)

- **New Parameters Added**:
  - Parameters: Dictionary containing the weights (W1, W2, ..., WL) of the network. Default is None.
  - L2: Regularization strength. Default is 0.

2. Gradient Update Modification (linear_backward()):

- **Purpose**: Adjusted the weight gradients to include the L2 regularization term.
- **Updated Formula**: $\text{Gradient with L2} = dW + \frac{\lambda}{m}W$

- **Function Signature:**

  original signature: linear_backward(dZ, cache)

  updated signature: linear_backward(dZ, cache, L2=0.0)

- **New Parameters Added**:
  - L2: Regularization strength. Default is 0.

- **Other Functions Updated**:

  To pass the L2 parameter seamlessly, these function signatures were updated:
  - linear_activation_backward(dA, cache, activation, L2=0.0)
  - l_model_backward(AL, Y, caches, L2=0.0)
  - l_layer_model(X, Y, layers_dims, learning_rate=0.009, num_iterations=100, batch_size=32, use_batchnorm=False, L2=0.0)

Weight Comparison: With and Without L2 Regularization

To evaluate the impact of L2 regularization on the network weights, we analyzed the weight statistics for **Experiment 1** (without L2) and **Experiment 3** (with L2).

The following metrics were examined for each layer:

	W1		**W2**		**W3**		**W4**	
	L2=0	L2=0.01	L2=0	L2=0.01	L2=0	L2=0.01	L2=0	L2=0.01
Mean	-0.0048	0.0007	0.0321	0.0198	0.1438	0.1589	0.0227	-0.0204
Std	0.3169	0.2279	0.6014	0.4227	0.8160	0.6116	0.7180	0.6047
Mean Difference $(W - W_{L2})$[1]	**-0.0055**		**0.0122**		**-0.0151**		**0.0432**	

Key Observations:

The application of L2 regularization (λ=0.01) introduced subtle adjustments to the weight distributions, with differences that are generally small and not consistently skewed in a specific direction.

The low regularization strength was effective in balancing performance, as it minimized overfitting without overly constraining the model.

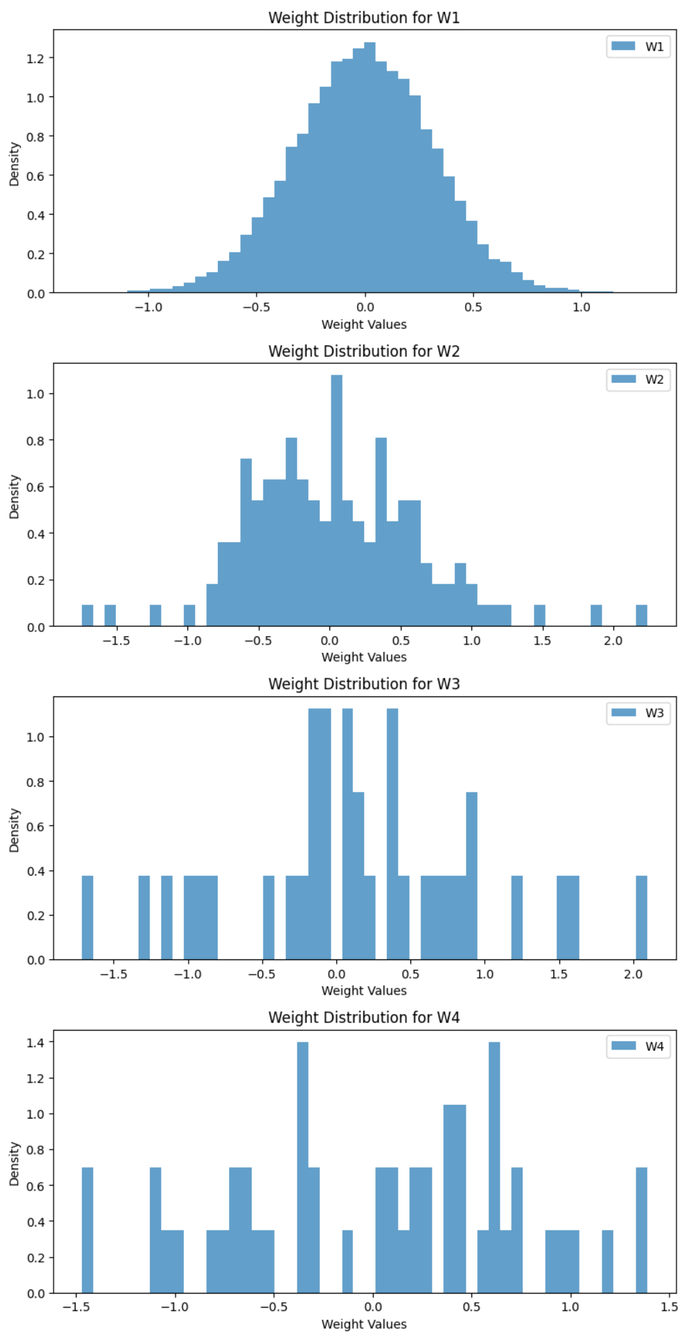
For the first layers, W1 and W2, the standard deviation of weights decreased, indicating tighter distributions. Conversely, in layers W3 and W4, the standard deviation increased slightly, reflecting the optimizer's adaptation to preserve flexibility in deeper layers.

These observations align with the model's generalization performance, confirming that the small λ value was sufficient to reduce overfitting while maintaining the network's ability to learn effectively. The regularization impact was mild and well-suited to the task, as the model did not exhibit significant overfitting initially.

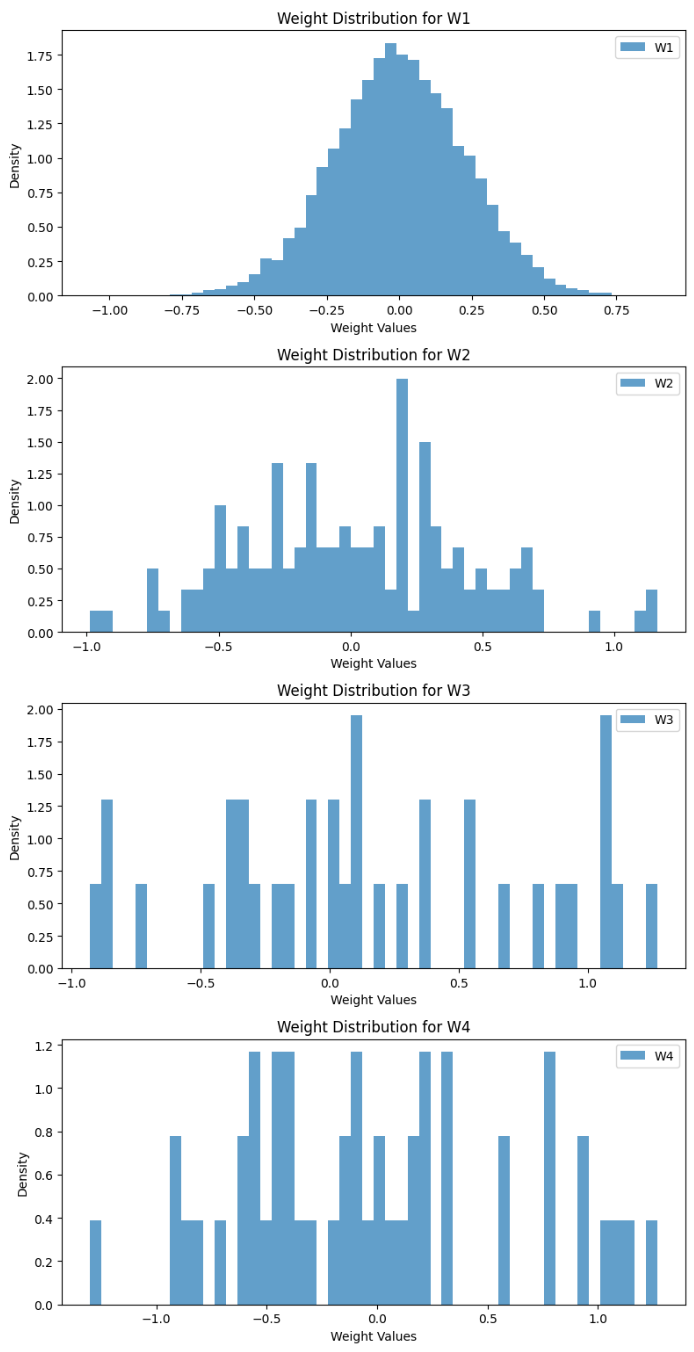
[1] The Weight Differences Matrix comparing weights between Not using L2 (Experiment 1) and Using L2 (Experiment 3) can be found in the appendices.

Weight Distribution Analysis:

Weights Distribution for Experiment 1: Without Batch Normalization & Without L2



Weights Distribution for Experiment 3: Without Batch Normalization & With L2



The weight distribution plots demonstrate relatively similar patterns between the scenarios with and without L2 regularization (λ=0.01). The differences are subtle, suggesting that the regularization had a mild impact on constraining the weights.

Note: The weight Differences Matrixes between Not using L2 (Experiment 1) and using L2 (Experiment 3) for each layer are detailed in the attached notes.