**Exploring Optimization and Depth Limits in CNNs Using MNIST**

**1. Setup Overview**

* **Dataset**: MNIST (60,000 training samples, 10,000 test samples)
* **Framework**: PyTorch (Google Colab, GPU-enabled)
* **Optimizer**: Stochastic Gradient Descent (SGD) with momentum = 0.9
* **Epochs**: 50 per experiment (adjusted with early stopping where appropriate)
* **Data Split**:
  + Training: 60% (36,000)
  + Validation: 20% (12,000)
  + Testing: 20% (12,000)
* **Evaluation Metric**: Validation/Test Accuracy
* **Random Seed**: Fixed (42)
* **Model Architectures**: Vary by experiment, ranging from shallow CNNs to deep convolutional stacks

**2. Task 1 – Optimal Learning Rate Search**

**Objective**

Identify the best learning rate for a simple CNN on MNIST using SGD with momentum.

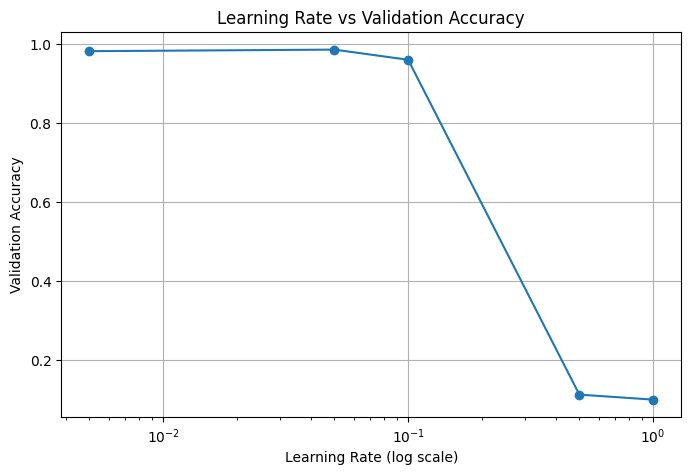
**Experimental Design**

* Model: 2 Convolutional Layers + 1 Fully Connected Layer (no BatchNorm or pooling)
* Learning Rates Tested: 1, 0.5, 0.1, 0.05, 0.005
* Fixed training length: 50 epochs per learning rate

**Results**

| **Learning Rate** | **Final Validation Accuracy** |
| --- | --- |
| 1.0 | 10.05% |
| 0.5 | 11.32% |
| 0.1 | 96.05% |
| 0.05 | 98.62% |
| 0.005 | 98.22% |

**Conclusion**: The optimal learning rate was 0.05.



**3. Task 2 – Maximum Depth Without Batch Normalization (Sigmoid Activation)**

**Objective**

Assess how deep a CNN can go using only Conv2D + Sigmoid, without any batch normalization or residual connections.

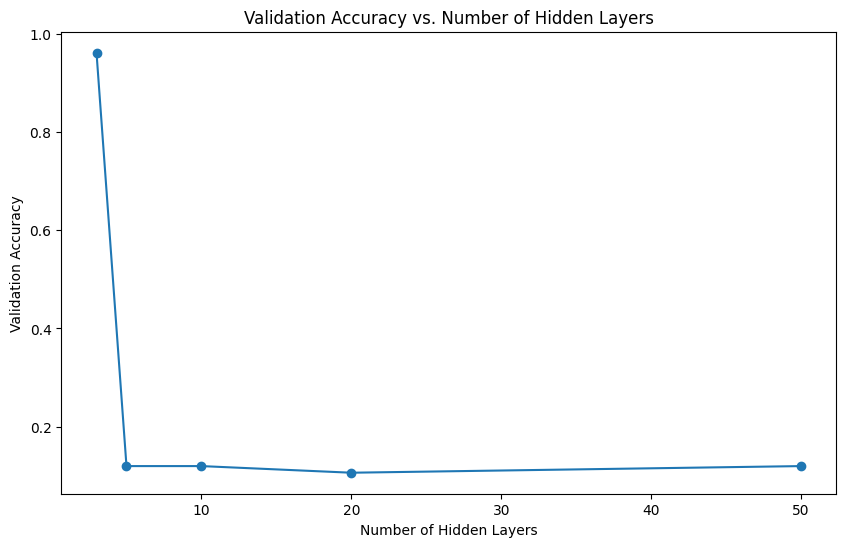
**Experimental Design**

* Activation: Sigmoid only
* Layer Depths Tested: 3, 5, 10, 20, 50
* Learning Rate: 0.05
* Training Length: 50 epochs

**Results**

| **Depth** | **Final Validation Accuracy** |
| --- | --- |
| 3 | 96.09% |
| 5 | 11.99% |
| 10 | 11.99% |
| 20 | 10.64% |
| 50 | 11.99% |

**Conclusion**: Networks with more than 3 Sigmoid-based layers failed to converge. The deeper the network, the more the accuracy collapsed due to vanishing gradients and lack of normalization.



**4. Task 3 – Maximum Depth with Batch Normalization and LeakyReLU**

**Objective**

Evaluate how far depth can be scaled when using Conv2D → BatchNorm → LeakyReLU.

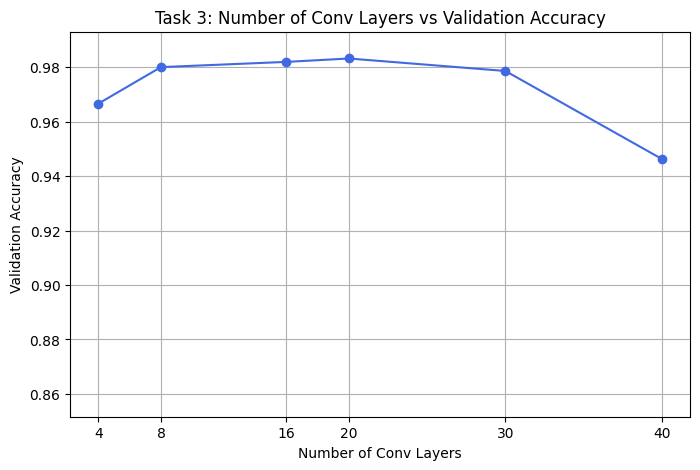
**Experimental Design**

* Architecture: Repeated Conv2D → BatchNorm → LeakyReLU blocks
* Depths Tested: 4, 8, 16, 20, 30, 40 layers
* Learning Rate: 0.05
* Early Stopping Enabled

**Results**

| **Depth** | **Best Validation Accuracy** |
| --- | --- |
| 4 | 97.10% |
| 8 | 98.27% |
| 16 | 98.31% |
| 20 | 98.32% |
| 30 | 97.91% |
| 40 | 95.22% |

**Conclusion**: Validation accuracy peaked between 16–20 layers. Further depth provided diminishing returns and started to degrade performance. Batch normalization and LeakyReLU were critical to trainability at higher depths.



**5. Task 4 – Optimal Batch Size Search**

**Objective**

Evaluate how different batch sizes affect test accuracy using the best-performing architecture (Task 3, depth 8 model).

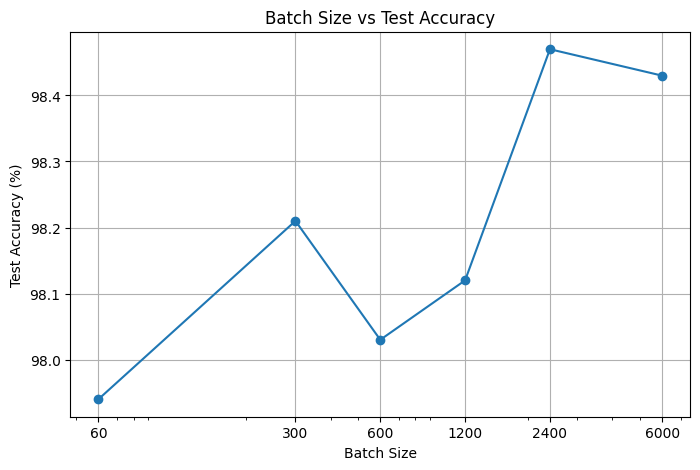
**Experimental Design**

* Batch Sizes Tested: 60, 300, 600, 1200, 2400, 6000 (corresponding to 0.1% to 10% of training data)
* Early Stopping Applied
* Test Accuracy used for final evaluation

**Results**

| **Batch Size** | **Training %** | **Test Accuracy** |
| --- | --- | --- |
| 60 | 0.1% | 97.94% |
| 300 | 0.5% | 98.21% |
| 600 | 1.0% | 98.03% |
| 1200 | 2.0% | 98.12% |
| 2400 | 4.0% | 98.47% |
| 6000 | 10.0% | 98.43% |

**Conclusion**: A batch size of 2400 (4% of the dataset) yielded the best test accuracy. This balance between stability and gradient noise led to optimal generalization.



**6. Key Observations**

* Learning rate of 0.05 consistently led to the best convergence speed and final accuracy.
* Sigmoid-only networks cannot scale deep without batch normalization.
* BatchNorm combined with LeakyReLU allows stable training up to at least 40 layers, with optimal results around 16–20 layers.
* Medium-sized batches (4–10% of data) provide better generalization than very small or very large batches.

**7. Challenges Encountered**

* **Colab GPU Limits**: Required the use of early stopping and reduced epoch counts for deeper networks to stay within runtime constraints.
* **Model Compatibility**: The saved depth8.pt model required precise architectural matching to avoid weight-loading errors.
* **Activation Behavior**: Sigmoid saturation led to early plateauing of gradients in deep networks.