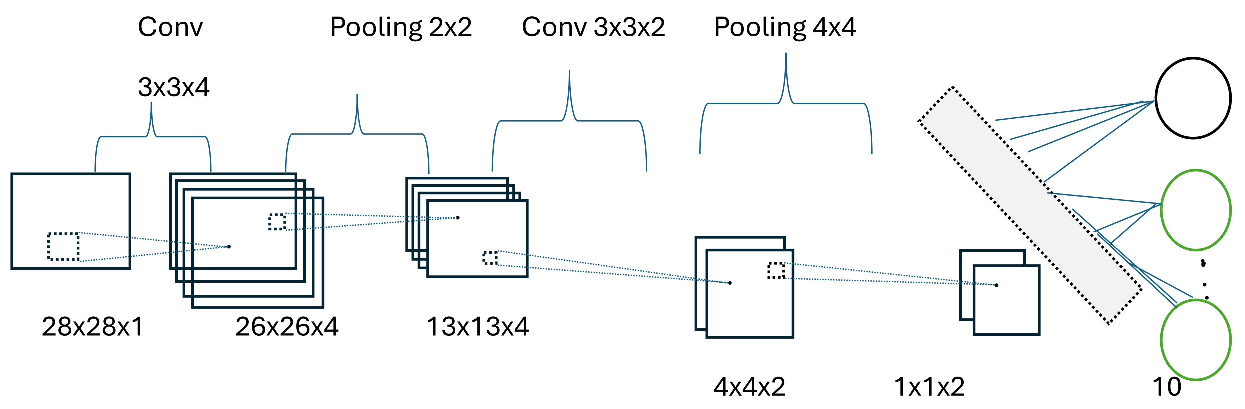
REPORT

CNN Architecture:



Introduction-

We analyze the performance of Convolutional Neural Networks (CNNs) on the MNIST dataset in this research. The goal is to comprehend how training time and prediction accuracy are impacted by changes in parameters like kernel size and pooling functions.

Results

Initial Model:

Training Accuracy: After 10 epochs, it rises from 61.64% to 91.08%.

Validation Accuracy: After 10 epochs, it rises from 82.69% to 91.34%.

Test Accuracy: Completes the test set with 91.79% accuracy.

Training Time: Every epoch is about 10 seconds.

Model of Variation:

Training Accuracy: After 10 epochs, it rises from 44.23% to 84.22%.

Validation Accuracy: After 10 epochs, it rises from 62.96% to 84.48%.

Test Accuracy: Completes the test set with 85.76% accuracy.

Training Time: Every epoch is about 10 seconds.

```
ConvNet(
 (first_layer): Sequential(
  (0): Conv2d(1, 4, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU()
  (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
 )
 (second_layer): Sequential(
  (0): Conv2d(4, 2, kernel_size=(3, 3), stride=(1, 1))
  (1): ReLU()
  (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
 )
 (additional_pooling): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
 (fully_connected): Linear(in_features=18, out_features=10, bias=True)
)
Output Size after First Layer: torch.Size([1, 4, 14, 14])
Output Size after Second Layer: torch.Size([1, 2, 6, 6])
Output Size after Additional Pooling: torch.Size([1, 2, 3, 3])
Output Size before Fully Connected Layer: torch.Size([1, 18])

Starting training with kernel size 3 on second layer.
Epoch 1, Training Loss: 1.1665, Training Accuracy: 61.64%, Validation Loss: 0.5970, Validation Accuracy: 82.69%, Time: 10.14s
Epoch 2, Training Loss: 0.5002, Training Accuracy: 85.14%, Validation Loss: 0.4521, Validation Accuracy: 86.77%, Time: 10.09s

Epoch 3, Training Loss: 0.4067, Training Accuracy: 87.72%, Validation Loss: 0.3845, Validation Accuracy: 88.59%, Time: 9.88s

Epoch 4, Training Loss: 0.3595, Training Accuracy: 88.98%, Validation Loss: 0.3494, Validation Accuracy: 89.42%, Time: 10.27s

Epoch 5, Training Loss: 0.3332, Training Accuracy: 89.71%, Validation Loss: 0.3267, Validation Accuracy: 90.21%, Time: 9.92s

Epoch 6, Training Loss: 0.3182, Training Accuracy: 90.12%, Validation Loss: 0.3082, Validation Accuracy: 90.49%, Time: 9.63s

Epoch 7, Training Loss: 0.3047, Training Accuracy: 90.56%, Validation Loss: 0.2989, Validation Accuracy: 90.85%, Time: 9.38s

Epoch 8, Training Loss: 0.2951, Training Accuracy: 90.74%, Validation Loss: 0.2967, Validation Accuracy: 90.92%, Time: 9.20s

Epoch 9, Training Loss: 0.2893, Training Accuracy: 90.95%, Validation Loss: 0.2863, Validation Accuracy: 91.07%, Time: 9.55s

Epoch 10, Training Loss: 0.2834, Training Accuracy: 91.08%, Validation Loss: 0.2837, Validation Accuracy: 91.34%, Time: 9.92s

Test Accuracy: 91.79%

ConvNet(
 (first_layer): Sequential(
  (0): Conv2d(1, 4, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU()
  (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
 )
 (second_layer): Sequential(
  (0): Conv2d(4, 2, kernel_size=(5, 5), stride=(1, 1))
  (1): ReLU()
  (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
 )
 (additional_pooling): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
 (fully_connected): Linear(in_features=8, out_features=10, bias=True)
)

Output Size after First Layer: torch.Size([1, 4, 14, 14])

Output Size after Second Layer: torch.Size([1, 2, 5, 5])

Output Size after Additional Pooling: torch.Size([1, 2, 2, 2])

Output Size before Fully Connected Layer: torch.Size([1, 8])
```

```
Starting training with kernel size 5 on second layer.

Epoch 1, Training Loss: 1.6113, Training Accuracy: 44.23%, Validation Loss: 1.1331, Validation Accuracy: 62.96%, Time: 10.01s

Epoch 2, Training Loss: 1.0321, Training Accuracy: 65.75%, Validation Loss: 0.9116, Validation Accuracy: 70.55%, Time: 10.63s

Epoch 3, Training Loss: 0.8713, Training Accuracy: 71.49%, Validation Loss: 0.7887, Validation Accuracy: 75.23%, Time: 10.33s

Epoch 4, Training Loss: 0.7539, Training Accuracy: 76.07%, Validation Loss: 0.6900, Validation Accuracy: 78.69%, Time: 9.72s

Epoch 5, Training Loss: 0.6742, Training Accuracy: 78.79%, Validation Loss: 0.6250, Validation Accuracy: 80.56%, Time: 9.62s

Epoch 6, Training Loss: 0.6178, Training Accuracy: 80.66%, Validation Loss: 0.5793, Validation Accuracy: 82.13%, Time: 9.58s

Epoch 7, Training Loss: 0.5768, Training Accuracy: 82.07%, Validation Loss: 0.5470, Validation Accuracy: 82.39%, Time: 9.81s

Epoch 8, Training Loss: 0.5446, Training Accuracy: 83.00%, Validation Loss: 0.5267, Validation Accuracy: 83.41%, Time: 9.95s

Epoch 9, Training Loss: 0.5219, Training Accuracy: 83.80%, Validation Loss: 0.5072, Validation Accuracy: 84.00%, Time: 10.15s

Epoch 10, Training Loss: 0.5041, Training Accuracy: 84.22%, Validation Loss: 0.4946, Validation Accuracy: 84.48%, Time: 10.28s

Test Accuracy: 85.76%
```

Analysis

Variation in Kernel Size's Effect:

There is a drop that is noticeable in accuracy when the second convolutional layer's kernel size is increased from 3 to 5.

Compared to the baseline model, the model with a kernel size of 5 learns more slowly and ends up with a lower final accuracy.

This implies that the model's capacity to learn discriminative features may be hampered by higher kernel sizes, which may cause a loss of spatial information.

Comparison of Training Times:

The comparable training times of the two models suggest that the training time is not considerably impacted by the increase in kernel size.

However, because learning proceeds more slowly, the variation model might take longer to converge.

Conclusion-

- In terms of accuracy, the variation model with a kernel size of 5 performs worse than the baseline model with a kernel size of 3.
- Greater kernel sizes may result in slower learning and a loss of spatial information, which would be detrimental to the model's performance.
- Additional testing with various kernel sizes and architectures may shed light on how to best optimize the CNN for the MNIST dataset.