## 1st Trial

Finished Training

Test loss: 1.136 | Test acc: 61.60%

Model saved to weights/checkpoint.pth

## 2nd Trial

We’ll do 5 clear upgrades:

1. Use nn.ReLU() layers instead of F.relu(). Why? It's cleaner, easier to register in the model, and more modular.

2. Add a third Conv2d layer. Why? Deeper networks learn more complex patterns.

3. Add Batch Normalization. Why? Helps the model converge faster and generalize better.

4. Add Dropout to FC layers. Why? Prevents overfitting.

5. Recalculate the flattened shape. With extra conv and pooling, the final output size changes. You must update the fc1 input size accordingly.

Output:

Finished Training

Test loss: 0.982 | Test acc: 75.70%

Model saved to weights/checkpoint.pth

Problem:

After epoch 5, val loss starts rising while val accuracy barely improves → sign of slight overfitting.

Model is learning well, but it might now benefit more from regularization or data augmentation.

## 3rd Trial

Solution: Safe Boost (Minor Tweaks)

Add data augmentation to prevent overfitting and improve generalization.

Transform pipeline in train.py was updated with RandomHorizontalFlip and RandomCrop.

Output:

Finished Training

Test loss: 0.676 | Test acc: 76.85%

Model saved to weights/checkpoint.pth

Effect:

Slightly lower training accuracy → less overfitting.

Validation and test accuracy improved → generalization is stronger.

Val and test loss decreased — model is now more confident and correct.

Why It Worked:

Random transformations make the model see slightly different versions of the same images, forcing it to learn more robust patterns.

## 4th Trial

Removed data augmentation and applied config tuning alone (e.g., more epochs, different batch size, etc.)

Output:

Finished Training

Test loss: 1.384 | Test acc: 73.37%

Model saved to weights/checkpoint.pth

Observation:

Training loss dropped a lot — overfitting kicked in around epoch 13–14.

Tuning alone isn’t enough — data augmentation is essential to generalize better.

## 5th Trial

Combined data augmentation (from trial 3) with config tuning (from trial 4).

Output:

Finished Training

Test loss: 0.686 | Test acc: 76.15%

Model saved to weights/checkpoint.pth

Conclusion:

Best test accuracy: Augmentation only (76.85%)

Most stable val/test balance: Augmentation + Tuning

Augmentation is the MVP — helps generalize well, especially with a small CNN.

Tuning + Augmentation is stable and reliable.

Tuning alone leads to overfitting.

## 6th Trial

Epochs: 10 batch\_size: 64 num\_workers: 8

Output:

Finished Training

Test loss: 0.833 | Test acc: 70.51%

Model saved to weights/checkpoint.pth

## 7th Trial

Epochs: 20 batch\_size: 16 num\_workers: 4

Output:

Finished Training

Test loss: 0.631 | Test acc: 79.12%

Model saved to weights/checkpoint.pth

## 8th Trial

Epochs: 30 batch\_size: 16 num\_workers: 4

Architecture Upgrade:

Added a 4th convolutional block:

self.conv4 = nn.Conv2d(128, 256, kernel\_size=3, padding=1)

self.bn4 = nn.BatchNorm2d(256)

self.relu4 = nn.ReLU()

Updated fc1 input to 256 \* 2 \* 2 = 1024

Results by Epochs:

Epoch 10: 77.52%

Epoch 20: 79.90%

Epoch 29: 82.50%

Epoch 30: 81.76%

Final Output:

Finished Training

Test loss: 0.593 | Test acc: 81.01%

Model saved to weights/checkpoint.pth