Neural Networks and Deep Learning Programming Project
Salva Karimisahari 240753803

1. Data Loader

The data ladder was defined with inspiration from ECS7026P labs. Images are converted to pytorch tensors using ToTensor() and additionally normalized with mean values (0.4915, 0.4823, 0.4468) and standard deviation values (0.2470, 0.2435, 0.2616) derived from Cifar10 explanation with pytorch in order to save computational time and memory and not iterate over the entire dataset. The batch size has been set to 128.

2. Basic Architecture

The base model follows the assignment's required architecture and is composed of:

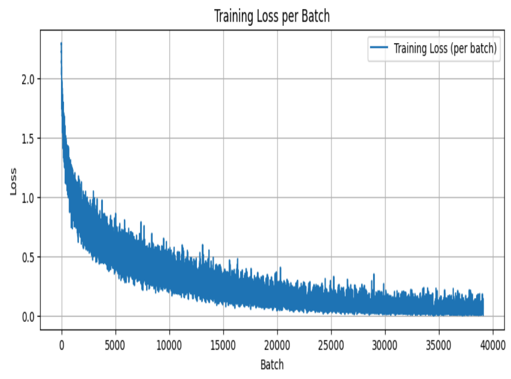
- K = 5 intermediate blocks.
  - Each block has 3 convolutional layers with kernel sizes: 3×3, 5×5, and 1×1 respectively.
  - Each convolutional layer is followed by BatchNorm and ReLU activation.
  - The weight vector [a1, a2, a3] is normalized using softmax.
- Each block contains L = 3 convolutional layers
- Convolutional outputs are weighted and summed using coefficients produced by a fully connected layer that gets a vector of channel means as input.
- The output block receives the final image output and averages each channel to form a vector, which is passed through two fully connected layers to generate a 10-dimensional logits vector.

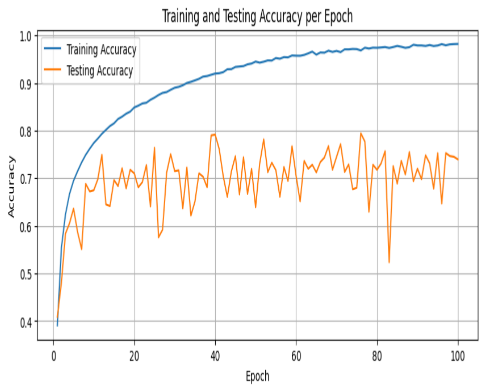
3. Training and testing

Aiming to reach about 70% accuracy, the number of intermediate blocks gradually increased from 3 to 5 and changes in the number of epochs aided in a desirable outcome without making changes to the basic structure. These were the configurations used:

Optimizer	Learning rate	Batch size	Epochs
Adam	0.001	128	100

Also a seed has been set in this experiment to brain the same result in different runs.
Results were obtained as following:





Best training accuracy: 97.43%
Best testing accuracy: 76.39% (epoch 25)
Final test accuracy: 79.43%

Overfitting begins around epoch 40, with training accuracy continuing to increase but test accuracy plateauing.

4. Improvements

To enhance the model's test accuracy, a series of architectural and training-related changes were explored. The key focus areas were:

- Regularization to reduce overfitting (dropout, weight decay)
- Depth and width of the network
- Empirical evaluation based on training and testing performance trends

Results were evaluated using test accuracy, training trends, and visual inspection of loss/accuracy curves. Each improvement was tested independently to isolate its impact.

4.1 dropout and weight decay

- Added Dropout (p=0.3) after the first fully connected layer in the output block.
- Added Weight Decay (5e-4) to the Adam optimizer.

The model appeared to be overfitting, so adding dropout layers was considered to be a good practice. Weight decay was also mentioned as a regularization method in week 6 will encourage smaller weights as it adds the weights to the loss function and encourages smaller (positive or negative) weights. These methods reduced the test accuracy to 77% meaning the combination of both of them was probably too aggressive for the learning process. Adding a dropout layer by itself also wasn't beneficial as it reduced the accuracy to 78%. Early stopping was also considered but given that the best results appeared in later epochs, it was best to exclude this method to let the model reach its full potential across all epochs.

4.2 Inspiration from VGG and adding blocks and layers

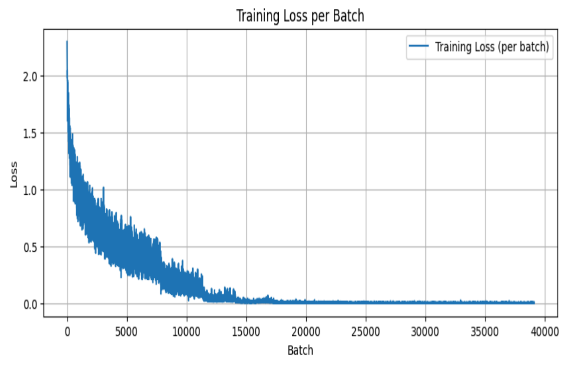
- Increased the number of intermediate blocks to K = 6
- Increased the number of convolutional layers per block to L = 4
- Used 64 output channels per conv layer
- Switched to VGG-style 3×3 convolutional filters with padding=1
- Removed dropout to avoid underfitting

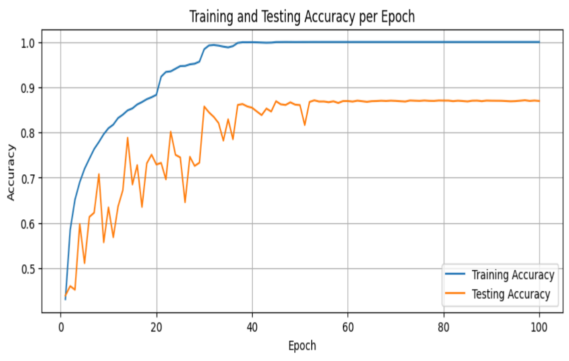
Increasing depth and filter consistency like VGG networks improved feature learning and resulted in a test accuracy of 85%

4.3 Experimenting LR scheduler and Batch Norm

- Learning rate scheduling with cosine annealing to start with a high learning rate (faster convergence) and gradually reduce the learning rate to fine tune the model in later epochs
- Using batch normalization to normalize the activation functions during training as a mild form of regularization

This approach was very beneficial as it increased the final test accuracy to 87% and reduced overfitting as the training and testing accuracy rose at a similar speed





4.4 Data Augmentation

Utilizing different transformers from the torchvision.transforms module for data augmentation, helped ensure the model would generalize better on the test set [source].

- RandomRotation(15): Rotates the image with a range of 15 degrees to aid the model to learn orientation-invariant features.
- RandAugment(): Applies a random selection of image transformers
- RandomHorizontalFlip(): Images flip horizontally with a 50% probability
- RandomCrop(32, padding=4): Adds a padding with 4 pixels on all sides and crops bag to the original size to shift the focus from the center of the image.
- Normalize(mean, std): Utilized dataset specific mean and std to scale pixel values to a zero centered distribution

A combination of these methods pushed the test accuracy to 88.6%

4.5 Increasing output channels and epochs, switching to SGD with momentum

Throughout different experiments, it was observed that increasing the number of epochs (increasing the chance of updating the weights, specifically well combined with cosine annealing) and increasing output channels (more representation power after each layer) resulted in an increment in the test accuracy with most architectures.

Since the accuracy on the long run was being evaluated, and the ADAM optimizer is mainly better for fast convergence, SGD with a momentum of 0.9 improved the accuracy on the test set. (Inspired from most of the implementations in paperswithcode)

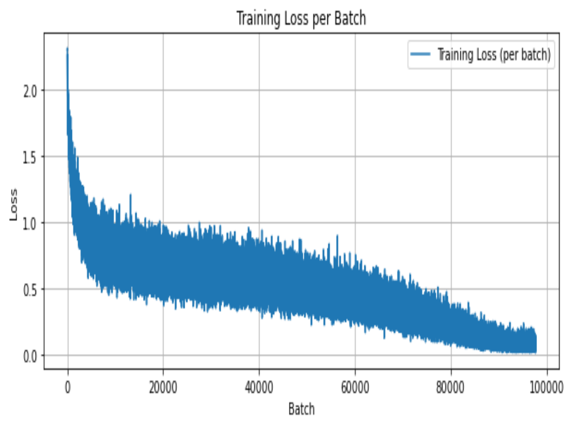
Switching the ADAM optimizer with SGD with momentum while increasing the number of epochs to 250 from 100 and using 128 output channels, resulted in a 92.9% accuracy.

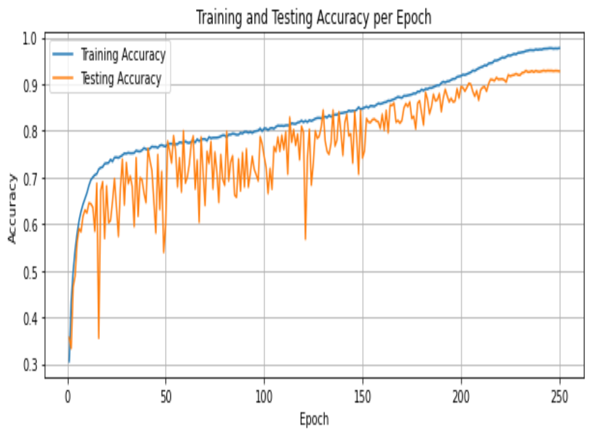
The final version performed better than previous implementations because data augmentation coupled with SGD and more epochs allowed the model to explore wider solutions and generalize better to converge to a more optimal minimum.

Best Testing Accuracy: 0.9291

Final model Architecture

Optimizer	LR Scheduler	Epochs	Intermediate blocks	Conv layer in each block	Output channels	Output hidden units	Batch size	Data Augmentation
SGD Momentum = 0.9 Weight_decay = 5e-4	Cosine Annealing	250	6	4	128	128	128	RandomRotation RandAugmentation RandomCrop HorizontalFlip


Training Loss per Batch
Training Loss (per batch)
Loss
Batch


Training and Testing Accuracy per Epoch
Training Accuracy
Testing Accuracy
Accuracy
Epoch