DL-Ops

Lab Assignment 7- Report

Debonil Ghosh
Roll No: M21AIE225
Executive MTech
Artificial Intelligence
Indian Institute of Technology, Jodhpur

Problem Statement:

Q1. In this assignment you are required to train a Convolutional Neural Network on two datasets using slurm and submit the jobs [100 marks]

a. If the last digit of your roll number is even:

Train ResNet-18 on FashionMNIST dataset [10 marks]

b. If the last digit of your roll number is odd:

Train MobileNet_V2 on CIFAR-10 dataset [10 marks]

● Download the datasets and extract them to folders on the GPU server and preprocess the data. [10 marks]

(Use default PyTorch dataloader function mentioning the dataset path and utilize

different transforms for preprocessing [compulsory])

● Set the loss function, optimiser, and metrics and compile the model [30 marks]

● Use Slurm to submit a job for training the model on the GPU server [15 marks]

● Also measure the training time (use timeit module for instance) for 10 and 15 epochs [10marks]

● Try to identify the set of hyperparameters that results in similar performance as

compared to the best performance in the previous step but with lower training time [25 marks]

Solution Google Colab Link:

https://colab.research.google.com/drive/1wNHfQh6cqsR5oNFQKmnmbyreEzvqBCOt?usp=sharing

Model Allotted:

MobileNet_V2 on CIFAR-10 dataset

[Roll number M21AIE225]

Mobile Net

MobileNet is a type of deep neural network architecture that was specifically designed for use on mobile and embedded devices with limited computational resources. It was developed by Google researchers in 2017 and has since become a popular choice for various computer vision tasks, such as image classification, object detection, and semantic segmentation.

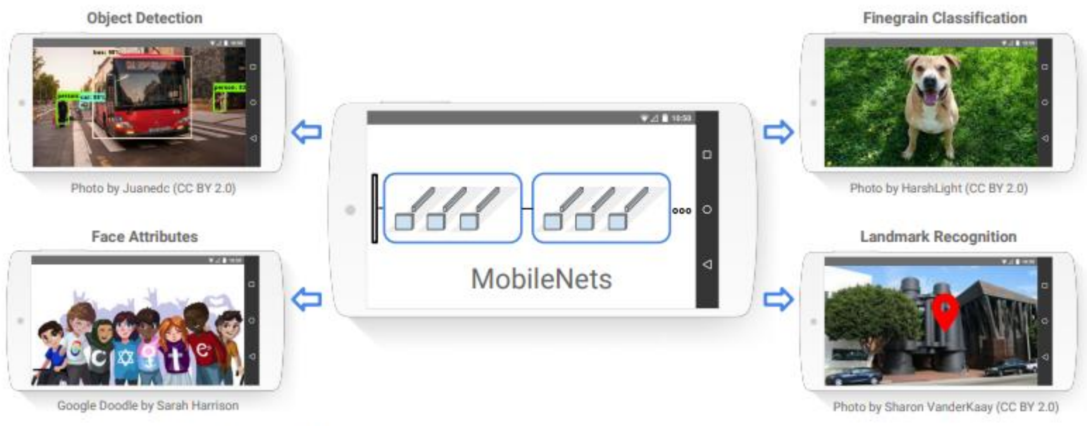

Object Detection
Photo by Juanedc (CC BY 2.0)
Face Attributes
Google Doodle by Sarah Harrison
MobileNets
Finegrain Classification
Photo by HarshLight (CC BY 2.0)
Landmark Recognition
Photo by Sharon VanderKaay (CC BY 2.0)

Figure 1. MobileNet models can be applied to various recognition tasks for efficient on device intelligence.

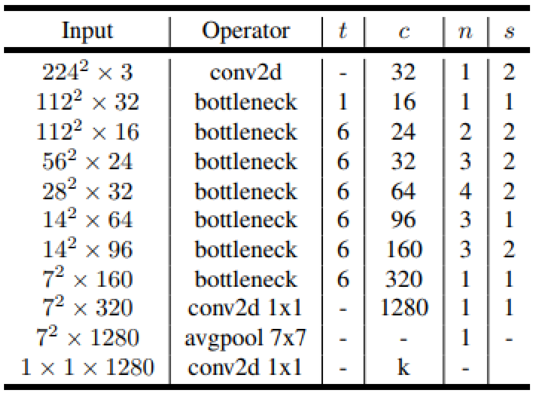
The MobileNet architecture is based on a combination of depthwise separable convolutions and pointwise convolutions. These operations split the convolution process into two parts, reducing the computational cost of the model significantly. Additionally, the MobileNet uses a "width multiplier" technique, which reduces the number of channels in each layer without affecting performance, further decreasing computational requirements.

MobileNet's compact size and efficiency make it ideal for deployment on resource-constrained devices such as smartphones, tablets, and embedded systems. It is available in several variations, including MobileNetV1, MobileNetV2, and MobileNetV3, each with specific improvements and trade-offs.

MobileNet_V2 Architecture:

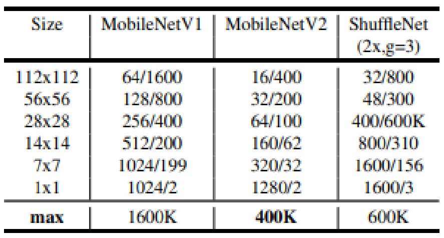
The architecture of MobileNet is based on a combination of depthwise separable convolutions and pointwise convolutions. Depthwise separable convolutions split the convolution operation into two separate operations: a depthwise convolution that applies a single filter to each input channel separately, and a pointwise convolution that applies a 1x1 filter to combine the outputs of the depthwise convolution. This reduces the computational cost of the convolution operation significantly.

In addition, MobileNet uses a technique called "width multiplier" to further reduce the computational cost of the model. This involves reducing the number of channels in each layer of the model by a certain factor, without significantly affecting its performance.



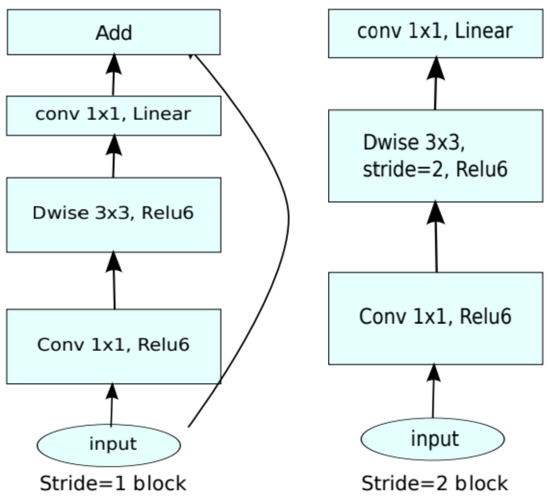
Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Table 2: MobileNetV2 : Each line describes a sequence of 1 or more identical (modulo stride) layers, repeated n times. All layers in the same sequence have the same number c of output channels. The first layer of each sequence has a stride s and all others use stride 1. All spatial convolutions use 3×3 kernels. The expansion factor t is always applied to the input size as described in Table 1.



Size	MobileNetV1	MobileNetV2	ShuffleNet (2x,g=3)
112x112	64/1600	16/400	32/800
56x56	128/800	32/200	48/300
28x28	256/400	64/100	400/600K
14x14	512/200	160/62	800/310
7x7	1024/199	320/32	1600/156
1x1	1024/2	1280/2	1600/3
max	1600K	**400K**	600K

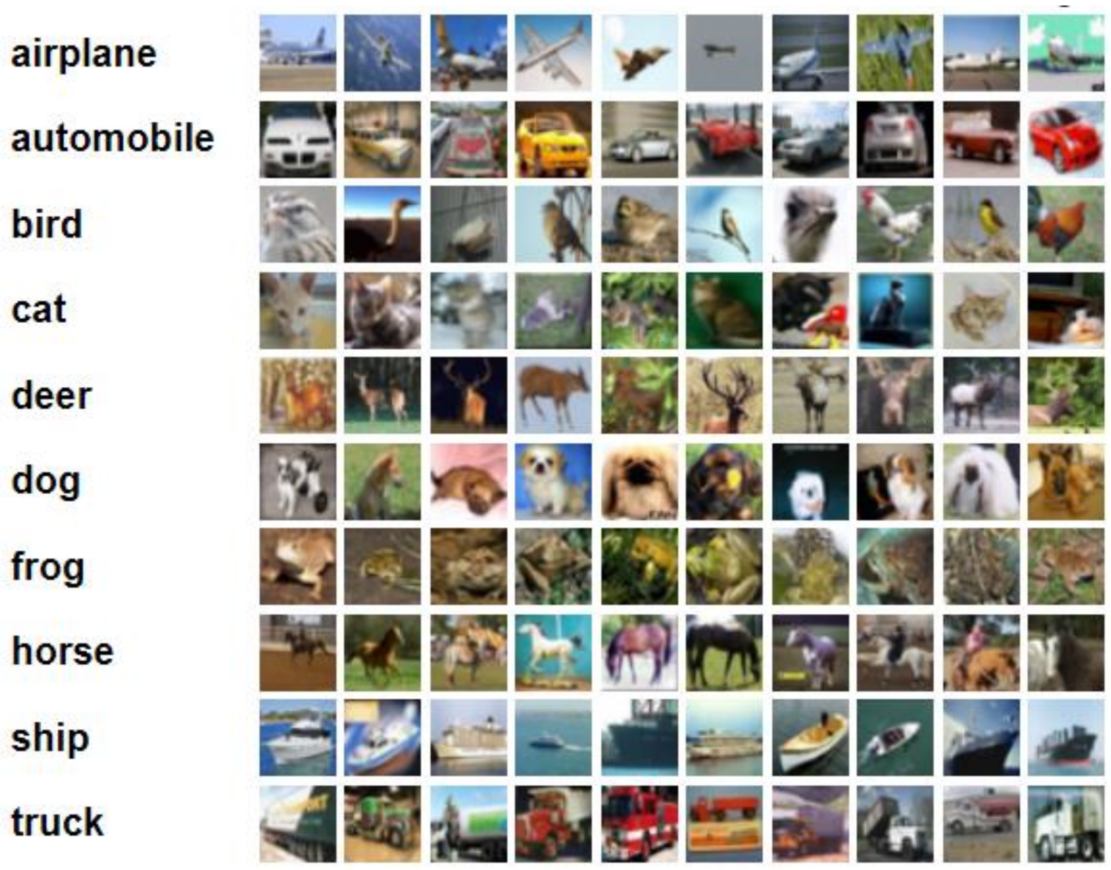
Table 3: The max number of channels/memory (in Kb) that needs to be materialized at each spatial resolution for different architectures. We assume 16-bit floats for activations. For ShuffleNet, we use $2x, g = 3$ that matches the performance of MobileNetV1 and MobileNetV2. For the first layer of MobileNetV2 and ShuffleNet we can employ the trick described in Section 5 to reduce memory requirement. Even though ShuffleNet employs bottlenecks elsewhere, the non-bottleneck tensors still need to be materialized due to the presence of shortcuts between the non-bottleneck tensors.



Convolution Blocks of Mobile Net V2

CIFAR 10 Dataset:

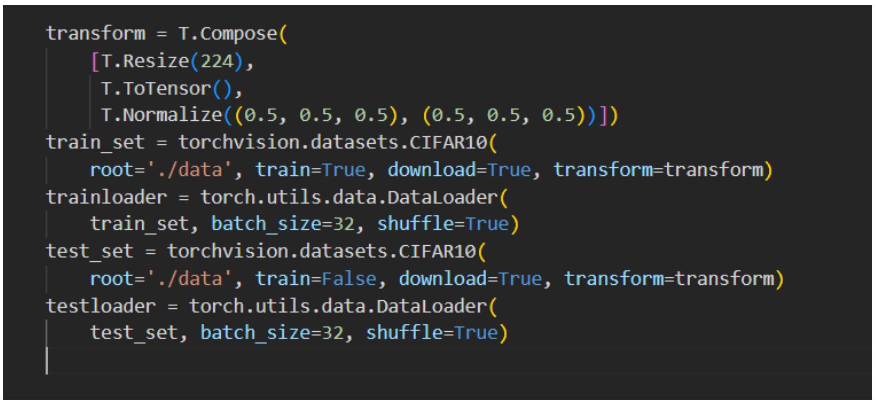
The CIFAR-10 dataset consists of **60000 32x32 colour images** in **10 classes**, with 6000 images per class. There are **50000 training** images and **10000 test images**.



Torch Transformations Applied:

- Resize to 224x224
- Normalize to (0.5, 0.5, 0.5), (0.5, 0.5, 0.5)
- Transform to tensor

Using default PyTorch data-loader function with the dataset path './data' and used above mentioned transforms for pre-processing.



```
transform = T.Compose(
    [T.Resize(224),
     T.ToTensor(),
     T.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])
train_set = torchvision.datasets.CIFAR10(
    root='./data', train=True, download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(
    train_set, batch_size=32, shuffle=True)
test_set = torchvision.datasets.CIFAR10(
    root='./data', train=False, download=True, transform=transform)
testloader = torch.utils.data.DataLoader(
    test_set, batch_size=32, shuffle=True)
```

Hyper Parameters Used:

Hyper Parameter	Value
Learning Rate	1e-2, 1e-3, 1e-4, 1e-5, 1e-6
Number of Epochs	10, 15
Batch Size	32
Loss Function	CrossEntropyLoss
Optimizer	Adam

Slurm Script Used:

```
#!/bin/bash
#SBATCH --job-name=m21aie225_lab7   # Job name
#SBATCH --partition=gpu2        #Partition name can be test/small/medium/large/gpu #Partition
"gpu" should be used only for gpu jobs
#SBATCH --nodes=1                       # Run all processes on a single node
#SBATCH --ntasks=1                      # Run a single task
#SBATCH --cpus-per-task=4       # Number of CPU cores per task
#SBATCH --gres=gpu:1            # Include gpu for the task (only for GPU jobs)
#SBATCH --mem=16gb                      # Total memory limit
#SBATCH --time=90:00:00         # Time limit hrs:min:sec
#SBATCH --output=m21aie225_lab7_%j.log # Standard output and error log
#date;hostname;pwd


module load python/3.8

python3 M21AIE225_Lab_Assignment_7.py
```

Highlighted modifications done in the given template.

Commands Used to submit batch:

```
[m21aie225@hpclogin ~]$ pwd

/iitjhome/m21aie225

[m21aie225@hpclogin ~]$ sbatch slurm_script.sh

Submitted batch job 40401

[m21aie225@hpclogin ~]$ squeue -j 40401

        JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)

        40401      gpu2 m21aie22 m21aie22  R       0:16      1 gpu2

[m21aie225@hpclogin ~]$ cat m21aie225_lab7_40401.log
```

Detail log of running:

```
Downloading: "https://download.pytorch.org/models/mobilenet_v2-7ebf99e0.pth" to
/iitjhome/m21aie225/.cache/torch/hub/checkpoints/mobilenet_v2-7ebf99e0.pth
cuda:0
Files already downloaded and verified
Files already downloaded and verified

***     Epoch = 10  Learning Rate = 0.01       ***


  0%|          | 0.00/13.6M [00:00<?, ?B/s]
  2%|▏         | 256k/13.6M [00:00<00:06, 2.30MB/s]
 11%|█         | 1.52M/13.6M [00:00<00:01, 8.39MB/s]
 42%|████▏     | 5.68M/13.6M [00:00<00:00, 24.0MB/s]
 73%|███████▎  | 9.92M/13.6M [00:00<00:00, 28.3MB/s]
100%|██████████| 13.6M/13.6M [00:00<00:00, 28.1MB/s]
Epoch: 1 (2m 0s)  Training Loss: 1.704,   Test Loss: 1.392,   Training acc: 0.36,   Test acc: 0.49,
Epoch: 2 (3m 40s)   Training Loss: 1.159,   Test Loss: 1.073,   Training acc: 0.59,   Test acc: 0.62,
Epoch: 3 (5m 23s)   Training Loss: 0.928,   Test Loss: 0.873,   Training acc: 0.68,   Test acc: 0.70,
Epoch: 4 (7m 8s)  Training Loss: 0.801,   Test Loss: 0.788,   Training acc: 0.72,   Test acc: 0.73,
Epoch: 5 (8m 51s)   Training Loss: 0.718,   Test Loss: 0.743,   Training acc: 0.75,   Test acc: 0.75,
Epoch: 6 (10m 35s)  Training Loss: 0.655,   Test Loss: 0.694,   Training acc: 0.77,   Test acc: 0.76,
Epoch: 7 (12m 21s)  Training Loss: 0.606,   Test Loss: 0.651,   Training acc: 0.79,   Test acc: 0.77,
Epoch: 8 (14m 5s)   Training Loss: 0.562,   Test Loss: 0.631,   Training acc: 0.81,   Test acc: 0.78,
Epoch: 9 (15m 48s)  Training Loss: 0.530,   Test Loss: 0.602,   Training acc: 0.82,   Test acc: 0.80,
Epoch: 10 (17m 28s)   Training Loss: 0.495,   Test Loss: 0.607,   Training acc: 0.83,   Test acc: 0.80,
Training completed in 17m 28s   Training Loss: 0.495,   Test Loss: 0.607,   Training acc: 0.83,   Test acc:
0.80,

***     Epoch = 15  Learning Rate = 0.01       ***

Epoch: 1 (1m 42s)   Training Loss: 1.613,   Test Loss: 1.286,   Training acc: 0.40,   Test acc: 0.53,
Epoch: 2 (3m 23s)   Training Loss: 1.135,   Test Loss: 1.058,   Training acc: 0.60,   Test acc: 0.63,
Epoch: 3 (5m 6s)  Training Loss: 0.954,   Test Loss: 0.916,   Training acc: 0.66,   Test acc: 0.68,
Epoch: 4 (6m 47s)   Training Loss: 0.822,   Test Loss: 0.841,   Training acc: 0.72,   Test acc: 0.71,
Epoch: 5 (8m 28s)   Training Loss: 0.730,   Test Loss: 0.746,   Training acc: 0.75,   Test acc: 0.74,
Epoch: 6 (10m 8s)   Training Loss: 0.665,   Test Loss: 0.694,   Training acc: 0.77,   Test acc: 0.76,
Epoch: 7 (11m 48s)  Training Loss: 0.616,   Test Loss: 0.646,   Training acc: 0.79,   Test acc: 0.78,
Epoch: 8 (13m 44s)  Training Loss: 0.580,   Test Loss: 0.646,   Training acc: 0.80,   Test acc: 0.78,
Epoch: 9 (15m 24s)  Training Loss: 0.543,   Test Loss: 0.616,   Training acc: 0.81,   Test acc: 0.79,
Epoch: 10 (17m 5s)  Training Loss: 0.512,   Test Loss: 0.606,   Training acc: 0.82,   Test acc: 0.79,
Epoch: 11 (18m 53s)   Training Loss: 0.490,   Test Loss: 0.619,   Training acc: 0.83,   Test acc: 0.79,
Epoch: 12 (20m 36s)   Training Loss: 0.464,   Test Loss: 0.597,   Training acc: 0.84,   Test acc: 0.80,
Epoch: 13 (22m 20s)   Training Loss: 0.435,   Test Loss: 0.566,   Training acc: 0.85,   Test acc: 0.81,
Epoch: 14 (24m 0s)  Training Loss: 0.424,   Test Loss: 0.556,   Training acc: 0.85,   Test acc: 0.82,
Epoch: 15 (25m 41s)   Training Loss: 0.402,   Test Loss: 0.551,   Training acc: 0.86,   Test acc: 0.81,
Training completed in 25m 41s   Training Loss: 0.402,   Test Loss: 0.551,   Training acc: 0.86,   Test acc:
0.81,

***     Epoch = 10  Learning Rate = 0.001       ***
```

```
Epoch: 1 (1m 40s)   Training Loss: 0.521,   Test Loss: 0.423,   Training acc: 0.83,   Test acc: 0.86,
Epoch: 2 (3m 20s)   Training Loss: 0.331,   Test Loss: 0.331,   Training acc: 0.89,   Test acc: 0.89,
Epoch: 3 (5m 1s)  Training Loss: 0.269,   Test Loss: 0.353,   Training acc: 0.91,   Test acc: 0.88,
Epoch: 4 (6m 42s)   Training Loss: 0.232,   Test Loss: 0.317,   Training acc: 0.92,   Test acc: 0.89,
Epoch: 5 (8m 22s)   Training Loss: 0.195,   Test Loss: 0.321,   Training acc: 0.93,   Test acc: 0.90,
Epoch: 6 (10m 3s)   Training Loss: 0.172,   Test Loss: 0.317,   Training acc: 0.94,   Test acc: 0.90,
Epoch: 7 (11m 44s)  Training Loss: 0.156,   Test Loss: 0.297,   Training acc: 0.95,   Test acc: 0.91,
Epoch: 8 (13m 32s)  Training Loss: 0.140,   Test Loss: 0.266,   Training acc: 0.95,   Test acc: 0.91,
Epoch: 9 (15m 13s)  Training Loss: 0.125,   Test Loss: 0.297,   Training acc: 0.96,   Test acc: 0.91,
Epoch: 10 (16m 54s)   Training Loss: 0.112,   Test Loss: 0.305,   Training acc: 0.96,   Test acc: 0.91,
Training completed in 16m 54s   Training Loss: 0.112,   Test Loss: 0.305,   Training acc: 0.96,   Test acc: 
0.91,

***      Epoch = 15  Learning Rate = 0.001      ***

Epoch: 1 (2m 41s)   Training Loss: 0.529,   Test Loss: 0.400,   Training acc: 0.82,   Test acc: 0.87,
Epoch: 2 (4m 21s)   Training Loss: 0.332,   Test Loss: 0.351,   Training acc: 0.89,   Test acc: 0.88,
Epoch: 3 (6m 3s)  Training Loss: 0.274,   Test Loss: 0.332,   Training acc: 0.91,   Test acc: 0.89,
Epoch: 4 (8m 2s)  Training Loss: 0.232,   Test Loss: 0.324,   Training acc: 0.92,   Test acc: 0.90,
Epoch: 5 (9m 46s)   Training Loss: 0.201,   Test Loss: 0.286,   Training acc: 0.93,   Test acc: 0.91,
Epoch: 6 (11m 29s)  Training Loss: 0.171,   Test Loss: 0.303,   Training acc: 0.94,   Test acc: 0.90,
Epoch: 7 (13m 11s)  Training Loss: 0.153,   Test Loss: 0.308,   Training acc: 0.95,   Test acc: 0.90,
Epoch: 8 (14m 58s)  Training Loss: 0.138,   Test Loss: 0.289,   Training acc: 0.95,   Test acc: 0.91,
Epoch: 9 (16m 41s)  Training Loss: 0.123,   Test Loss: 0.293,   Training acc: 0.96,   Test acc: 0.91,
Epoch: 10 (18m 23s)   Training Loss: 0.112,   Test Loss: 0.280,   Training acc: 0.96,   Test acc: 0.92,
Epoch: 11 (20m 7s)  Training Loss: 0.106,   Test Loss: 0.292,   Training acc: 0.97,   Test acc: 0.91,
Epoch: 12 (21m 51s)   Training Loss: 0.097,   Test Loss: 0.324,   Training acc: 0.97,   Test acc: 0.91,
Epoch: 13 (23m 34s)   Training Loss: 0.091,   Test Loss: 0.312,   Training acc: 0.97,   Test acc: 0.91,
Epoch: 14 (25m 16s)   Training Loss: 0.083,   Test Loss: 0.352,   Training acc: 0.97,   Test acc: 0.90,
Epoch: 15 (27m 8s)  Training Loss: 0.080,   Test Loss: 0.324,   Training acc: 0.97,   Test acc: 0.91,
Training completed in 27m 8s  Training Loss: 0.080,   Test Loss: 0.324,   Training acc: 0.97,   Test acc: 
0.91,

***      Epoch = 10  Learning Rate = 0.0001      ***

Epoch: 1 (1m 46s)   Training Loss: 0.494,   Test Loss: 0.297,   Training acc: 0.84,   Test acc: 0.90,
Epoch: 2 (3m 27s)   Training Loss: 0.184,   Test Loss: 0.239,   Training acc: 0.94,   Test acc: 0.92,
Epoch: 3 (5m 12s)   Training Loss: 0.108,   Test Loss: 0.234,   Training acc: 0.96,   Test acc: 0.92,
Epoch: 4 (6m 56s)   Training Loss: 0.066,   Test Loss: 0.238,   Training acc: 0.98,   Test acc: 0.93,
Epoch: 5 (8m 37s)   Training Loss: 0.047,   Test Loss: 0.250,   Training acc: 0.98,   Test acc: 0.93,
Epoch: 6 (10m 17s)  Training Loss: 0.038,   Test Loss: 0.255,   Training acc: 0.99,   Test acc: 0.93,
Epoch: 7 (12m 2s)   Training Loss: 0.033,   Test Loss: 0.247,   Training acc: 0.99,   Test acc: 0.93,
Epoch: 8 (13m 42s)  Training Loss: 0.029,   Test Loss: 0.268,   Training acc: 0.99,   Test acc: 0.93,
Epoch: 9 (15m 25s)  Training Loss: 0.027,   Test Loss: 0.271,   Training acc: 0.99,   Test acc: 0.93,
Epoch: 10 (17m 6s)  Training Loss: 0.022,   Test Loss: 0.275,   Training acc: 0.99,   Test acc: 0.93,
Training completed in 17m 6s  Training Loss: 0.022,   Test Loss: 0.275,   Training acc: 0.99,   Test acc: 
0.93,

***      Epoch = 15  Learning Rate = 0.0001      ***

Epoch: 1 (1m 42s)   Training Loss: 0.502,   Test Loss: 0.289,   Training acc: 0.84,   Test acc: 0.90,
Epoch: 2 (3m 30s)   Training Loss: 0.189,   Test Loss: 0.230,   Training acc: 0.94,   Test acc: 0.92,
Epoch: 3 (5m 12s)   Training Loss: 0.106,   Test Loss: 0.240,   Training acc: 0.96,   Test acc: 0.92,
Epoch: 4 (6m 53s)   Training Loss: 0.068,   Test Loss: 0.240,   Training acc: 0.98,   Test acc: 0.92,
Epoch: 5 (8m 34s)   Training Loss: 0.047,   Test Loss: 0.248,   Training acc: 0.98,   Test acc: 0.93,
Epoch: 6 (10m 14s)  Training Loss: 0.040,   Test Loss: 0.255,   Training acc: 0.99,   Test acc: 0.93,
Epoch: 7 (11m 56s)  Training Loss: 0.034,   Test Loss: 0.255,   Training acc: 0.99,   Test acc: 0.93,
Epoch: 8 (13m 37s)  Training Loss: 0.029,   Test Loss: 0.264,   Training acc: 0.99,   Test acc: 0.93,
Epoch: 9 (15m 17s)  Training Loss: 0.027,   Test Loss: 0.259,   Training acc: 0.99,   Test acc: 0.94,
Epoch: 10 (16m 58s)   Training Loss: 0.024,   Test Loss: 0.273,   Training acc: 0.99,   Test acc: 0.93,
Epoch: 11 (18m 38s)   Training Loss: 0.021,   Test Loss: 0.264,   Training acc: 0.99,   Test acc: 0.93,
Epoch: 12 (20m 19s)   Training Loss: 0.022,   Test Loss: 0.282,   Training acc: 0.99,   Test acc: 0.93,
Epoch: 13 (21m 59s)   Training Loss: 0.019,   Test Loss: 0.296,   Training acc: 0.99,   Test acc: 0.93,
Epoch: 14 (23m 39s)   Training Loss: 0.017,   Test Loss: 0.289,   Training acc: 0.99,   Test acc: 0.93,
Epoch: 15 (25m 20s)   Training Loss: 0.020,   Test Loss: 0.273,   Training acc: 0.99,   Test acc: 0.93,
Training completed in 25m 20s   Training Loss: 0.020,   Test Loss: 0.273,   Training acc: 0.99,   Test acc:
0.93,

***      Epoch = 10  Learning Rate = 1e-05      ***

Epoch: 1 (1m 42s)   Training Loss: 1.419,   Test Loss: 0.743,   Training acc: 0.60,   Test acc: 0.77,
Epoch: 2 (3m 25s)   Training Loss: 0.571,   Test Loss: 0.492,   Training acc: 0.82,   Test acc: 0.84,
Epoch: 3 (5m 7s)  Training Loss: 0.418,   Test Loss: 0.410,   Training acc: 0.86,   Test acc: 0.86,
Epoch: 4 (6m 48s)   Training Loss: 0.338,   Test Loss: 0.362,   Training acc: 0.89,   Test acc: 0.88,
Epoch: 5 (8m 28s)   Training Loss: 0.293,   Test Loss: 0.335,   Training acc: 0.90,   Test acc: 0.89,
Epoch: 6 (10m 9s)   Training Loss: 0.253,   Test Loss: 0.318,   Training acc: 0.92,   Test acc: 0.89,
Epoch: 7 (11m 51s)  Training Loss: 0.220,   Test Loss: 0.294,   Training acc: 0.93,   Test acc: 0.90,
Epoch: 8 (13m 33s)  Training Loss: 0.194,   Test Loss: 0.280,   Training acc: 0.94,   Test acc: 0.90,
Epoch: 9 (15m 14s)  Training Loss: 0.172,   Test Loss: 0.278,   Training acc: 0.94,   Test acc: 0.90,
Epoch: 10 (16m 56s)   Training Loss: 0.151,   Test Loss: 0.276,   Training acc: 0.95,   Test acc: 0.91,
Training completed in 16m 56s   Training Loss: 0.151,   Test Loss: 0.276,   Training acc: 0.95,   Test acc:
0.91,

***      Epoch = 15  Learning Rate = 1e-05      ***

Epoch: 1 (1m 42s)   Training Loss: 1.349,   Test Loss: 0.710,   Training acc: 0.62,   Test acc: 0.78,
Epoch: 2 (3m 26s)   Training Loss: 0.553,   Test Loss: 0.478,   Training acc: 0.83,   Test acc: 0.84,
Epoch: 3 (5m 8s)  Training Loss: 0.408,   Test Loss: 0.411,   Training acc: 0.87,   Test acc: 0.86,
Epoch: 4 (6m 49s)   Training Loss: 0.338,   Test Loss: 0.361,   Training acc: 0.89,   Test acc: 0.88,
Epoch: 5 (8m 30s)   Training Loss: 0.285,   Test Loss: 0.333,   Training acc: 0.91,   Test acc: 0.89,
Epoch: 6 (10m 12s)  Training Loss: 0.244,   Test Loss: 0.321,   Training acc: 0.92,   Test acc: 0.89,
Epoch: 7 (11m 53s)  Training Loss: 0.216,   Test Loss: 0.310,   Training acc: 0.93,   Test acc: 0.90,
Epoch: 8 (13m 34s)  Training Loss: 0.189,   Test Loss: 0.285,   Training acc: 0.94,   Test acc: 0.90,
Epoch: 9 (15m 15s)  Training Loss: 0.168,   Test Loss: 0.277,   Training acc: 0.95,   Test acc: 0.90,
Epoch: 10 (16m 56s)   Training Loss: 0.148,   Test Loss: 0.271,   Training acc: 0.95,   Test acc: 0.91,
Epoch: 11 (18m 38s)   Training Loss: 0.129,   Test Loss: 0.271,   Training acc: 0.96,   Test acc: 0.91,
Epoch: 12 (20m 20s)   Training Loss: 0.114,   Test Loss: 0.269,   Training acc: 0.96,   Test acc: 0.91,
Epoch: 13 (22m 1s)  Training Loss: 0.101,   Test Loss: 0.262,   Training acc: 0.97,   Test acc: 0.91,
Epoch: 14 (23m 43s)   Training Loss: 0.086,   Test Loss: 0.253,   Training acc: 0.97,   Test acc: 0.92,
Epoch: 15 (25m 24s)   Training Loss: 0.077,   Test Loss: 0.258,   Training acc: 0.98,   Test acc: 0.92,
Training completed in 25m 24s   Training Loss: 0.077,   Test Loss: 0.258,   Training acc: 0.98,   Test acc:
0.92,
***     Epoch = 10  Learning Rate = 1e-06      ***

Epoch: 1 (1m 40s)   Training Loss: 2.263,   Test Loss: 2.193,   Training acc: 0.18,   Test acc: 0.28,
Epoch: 2 (3m 21s)   Training Loss: 2.106,   Test Loss: 2.009,   Training acc: 0.38,   Test acc: 0.46,
Epoch: 3 (5m 2s)  Training Loss: 1.892,   Test Loss: 1.770,   Training acc: 0.52,   Test acc: 0.57,
Epoch: 4 (6m 43s)   Training Loss: 1.638,   Test Loss: 1.513,   Training acc: 0.60,   Test acc: 0.63,
Epoch: 5 (8m 23s)   Training Loss: 1.396,   Test Loss: 1.294,   Training acc: 0.65,   Test acc: 0.66,
Epoch: 6 (10m 4s)   Training Loss: 1.200,   Test Loss: 1.115,   Training acc: 0.69,   Test acc: 0.70,
Epoch: 7 (11m 45s)  Training Loss: 1.047,   Test Loss: 0.986,   Training acc: 0.72,   Test acc: 0.73,
Epoch: 8 (13m 27s)  Training Loss: 0.927,   Test Loss: 0.889,   Training acc: 0.74,   Test acc: 0.74,
Epoch: 9 (15m 10s)  Training Loss: 0.838,   Test Loss: 0.811,   Training acc: 0.76,   Test acc: 0.76,
Epoch: 10 (16m 51s)   Training Loss: 0.766,   Test Loss: 0.749,   Training acc: 0.77,   Test acc: 0.77,
Training completed in 16m 51s   Training Loss: 0.766,   Test Loss: 0.749,   Training acc: 0.77,   Test acc:
0.77,

***     Epoch = 15  Learning Rate = 1e-06      ***

Epoch: 1 (1m 41s)   Training Loss: 2.249,   Test Loss: 2.175,   Training acc: 0.19,   Test acc: 0.29,
Epoch: 2 (3m 21s)   Training Loss: 2.089,   Test Loss: 1.992,   Training acc: 0.39,   Test acc: 0.49,
Epoch: 3 (5m 3s)  Training Loss: 1.871,   Test Loss: 1.745,   Training acc: 0.54,   Test acc: 0.59,
Epoch: 4 (6m 45s)   Training Loss: 1.613,   Test Loss: 1.484,   Training acc: 0.62,   Test acc: 0.65,
Epoch: 5 (8m 28s)   Training Loss: 1.366,   Test Loss: 1.257,   Training acc: 0.67,   Test acc: 0.68,
Epoch: 6 (10m 11s)  Training Loss: 1.164,   Test Loss: 1.080,   Training acc: 0.70,   Test acc: 0.71,
Epoch: 7 (11m 51s)  Training Loss: 1.016,   Test Loss: 0.959,   Training acc: 0.72,   Test acc: 0.72,
Epoch: 8 (13m 32s)  Training Loss: 0.902,   Test Loss: 0.864,   Training acc: 0.74,   Test acc: 0.74,
Epoch: 9 (15m 13s)  Training Loss: 0.818,   Test Loss: 0.788,   Training acc: 0.76,   Test acc: 0.76,
Epoch: 10 (16m 54s)   Training Loss: 0.754,   Test Loss: 0.741,   Training acc: 0.77,   Test acc: 0.77,
Epoch: 11 (18m 34s)   Training Loss: 0.703,   Test Loss: 0.699,   Training acc: 0.78,   Test acc: 0.78,
Epoch: 12 (20m 15s)   Training Loss: 0.659,   Test Loss: 0.655,   Training acc: 0.80,   Test acc: 0.79,
Epoch: 13 (21m 57s)   Training Loss: 0.624,   Test Loss: 0.620,   Training acc: 0.80,   Test acc: 0.80,
Epoch: 14 (23m 41s)   Training Loss: 0.594,   Test Loss: 0.598,   Training acc: 0.81,   Test acc: 0.81,
Epoch: 15 (25m 24s)   Training Loss: 0.565,   Test Loss: 0.576,   Training acc: 0.82,   Test acc: 0.81,
Training completed in 25m 24s   Training Loss: 0.565,   Test Loss: 0.576,   Training acc: 0.82,   Test acc:
0.81,

 Overall Summary :

|   Learning Rate |   Epoch |   Training Time |   Training Accuracy |   Test Accuracy |
|----------------:|--------:|----------------:|--------------------:|----------------:|
|          0.01   |      10 |         1048.41 |            0.827735 |        0.80022  |
|          0.01   |      15 |         1541.13 |            0.860905 |        0.814397 |
|          0.001  |      10 |         1014.4  |            0.961472 |        0.911342 |
|          0.001  |      15 |         1628.36 |            0.971509 |        0.907947 |
|          0.0001 |      10 |         1026.97 |            0.992682 |        0.928115 |
|          0.0001 |      15 |         1520.75 |            0.992962 |        0.93131  |
|          1e-05  |      10 |         1016.06 |            0.951156 |        0.905751 |
|          1e-05  |      15 |         1524.59 |            0.976408 |        0.915535 |
|          1e-06  |      10 |         1011.81 |            0.773093 |        0.773962 |
|          1e-06  |      15 |         1524.61 |            0.819778 |        0.812001 |
```

Final Results:

Learning Rate	Epoch	Training Time	Training Accuracy	Test Accuracy
0.01	10	1048.406607	0.827735125	0.800219649
0.001	10	1014.398538	0.961472329	0.911341853
0.0001	**10**	**1026.972617**	**0.992682342**	**0.928115016**
1.00E-05	10	1016.061641	0.95115563	0.905750799
1.00E-06	10	1011.813282	0.77309261	0.773961661
0.01	15	1541.126099	0.860904511	0.814396965
0.001	15	1628.355758	0.971509117	0.907947284
0.0001	**15**	**1520.749465**	**0.992962252**	**0.931309904**
1.00E-05	15	1524.58961	0.97640755	0.915535144
1.00E-06	15	1524.606232	0.819777671	0.812000799

To identify the set of hyperparameters that results in similar performance as compared to the best performance in the previous step but with lower training time, we can try the following strategies:

1. **Adjust the learning rate**: The learning rate controls the step size in gradient descent and can have a significant impact on training time. If the learning rate is too high, the optimizer may overshoot the optimal weights and take longer to converge. On the other hand, if the learning rate is too low, the optimizer may get stuck in local optima. Therefore, we can try adjusting the learning rate to find the optimal value that achieves similar performance with lower training time.

2. **Change the batch size**: The batch size determines how many samples are processed in each training iteration. Larger batch sizes can result in faster training times, but they may also lead to overfitting. Conversely, smaller batch sizes can improve generalization but may require more training iterations, which can increase training time. Therefore, we can try adjusting the batch size to find the optimal value that achieves similar performance with lower training time.

3. **Modify the architecture**: The complexity of the model architecture can also affect training time. If the model is too complex, it may require more time to train. However, a simpler model may not be able to capture the underlying patterns in the data. Therefore, we can try modifying the architecture by adding or removing layers, adjusting the number of neurons, or changing the activation functions to find the optimal configuration that achieves similar performance with lower training time.

4. **Try different regularization techniques**: Regularization techniques such as L1, L2, or dropout can help prevent overfitting, but they can also increase training time.

Therefore, we can try experimenting with different regularization techniques to find the optimal combination that achieves similar performance with lower training time.

5. **Use transfer learning**: Transfer learning can be a useful technique to reduce training time. By using a pre-trained model, we can leverage the knowledge learned from a similar task and fine-tune the model on our dataset. This can result in faster convergence and improved performance. Therefore, we can try using transfer learning to find the optimal pre-trained model and fine-tuning strategy that achieves similar performance with lower training time.

Due to time constraint, in this assignment, I have varied only first option, that is learning rate and number of epochs as mentioned in the question. Tried learning rates 1e-2, 1e-3, 1e-4, 1e-5, 1e-6 and found that **1e-4** gives Optimum results in both epoch count 10 and 15. So from this result, this can be concluded that, when **learning rate set at 1e-4**, **spending about 500 more seconds in training with epoch 15, in compare to epoch 10, is raising accuracy of the model 0.32 % only. So optimal hyperparameter set can be following:**

Hyper Parameter	Value
Learning Rate	1e-4
Number of Epochs	10
Batch Size	32
Loss Function	CrossEntropyLoss
Optimizer	Adam

References:

1. Lectures of DL-Ops class by Professors
2. Implementation Session by Teaching Assistants