```
import torch
import torchvision
import matplotlib.pyplot as plt
import numpy as np

# Fetch CIFAR-10 dataset
transform =
torchvision.transforms.Compose([torchvision.transforms.ToTensor(),])
trainset = torchvision.datasets.CIFAR10(root='./data', train=True,
download=True, transform=transform)
```

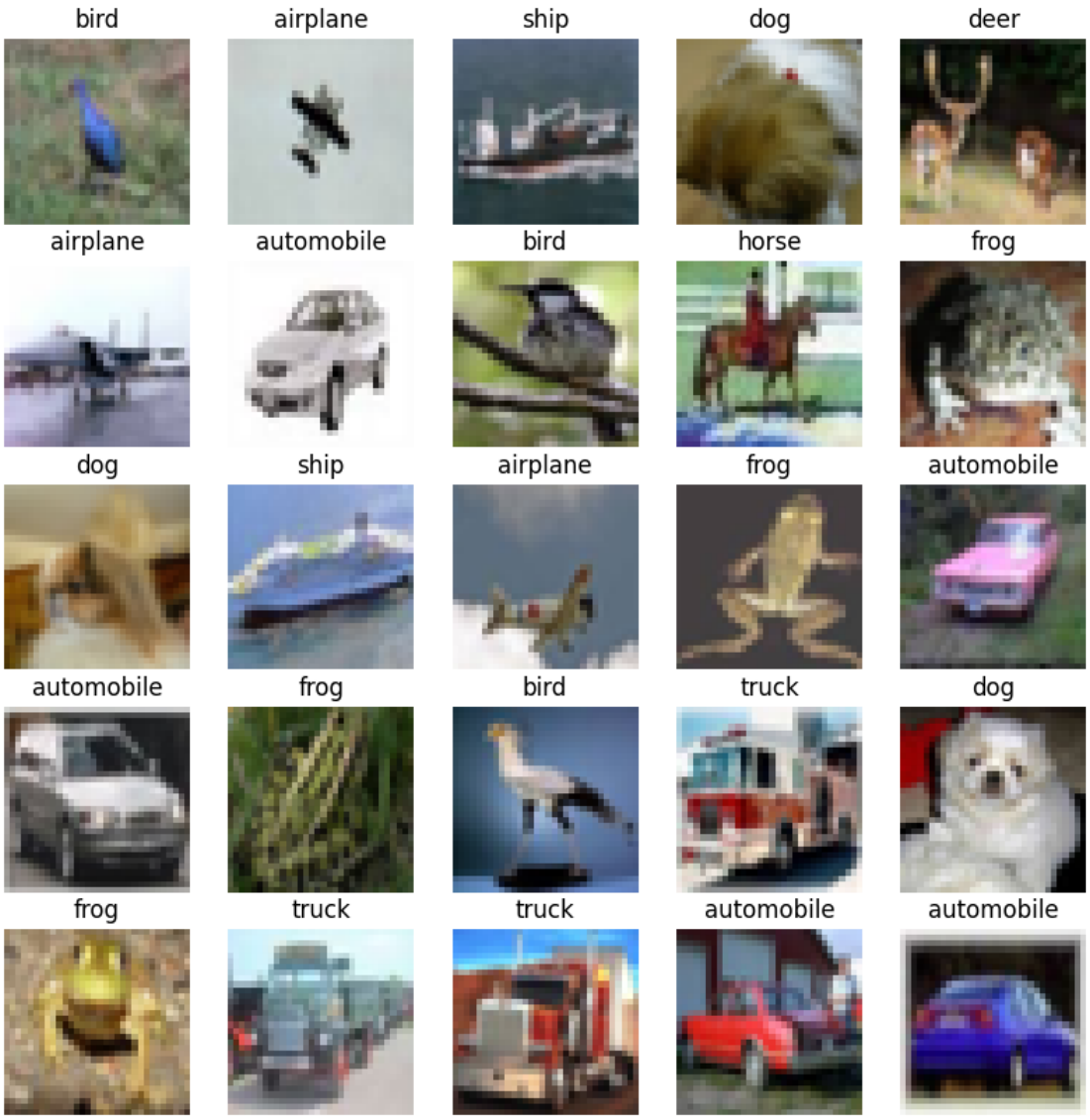
```
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to
./data/cifar-10-python.tar.gz

100%|██████████| 170498071/170498071 [00:02<00:00, 78124261.63it/s]

Extracting ./data/cifar-10-python.tar.gz to ./data
```

```
# List all classes in the dataset
classes = trainset.classes
print("The Different classes of the images in the CIFAR-10 Dataset
are: ")
print(classes)
```

```
The Different classes of the images in the CIFAR-10 Dataset are:
['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog',
'horse', 'ship', 'truck']
```

```
# Select random images from the dataset
num_images = 25
indices = np.random.randint(0, len(trainset), size=num_images)
images = [trainset[i][0] for i in indices]
class_names = [trainset[i][1] for i in indices]
```

```
# Create 2-D array of images
rows = 5
cols = 5
fig, axs = plt.subplots(rows, cols, figsize=(10, 10))
for i in range(rows):
    for j in range(cols):
        axs[i,j].imshow(np.transpose(images[i*rows+j], (1, 2, 0)))
        axs[i,j].axis('off')
        axs[i,j].set_title(trainset.classes[class_names[i*rows+j]])
plt.show()
```

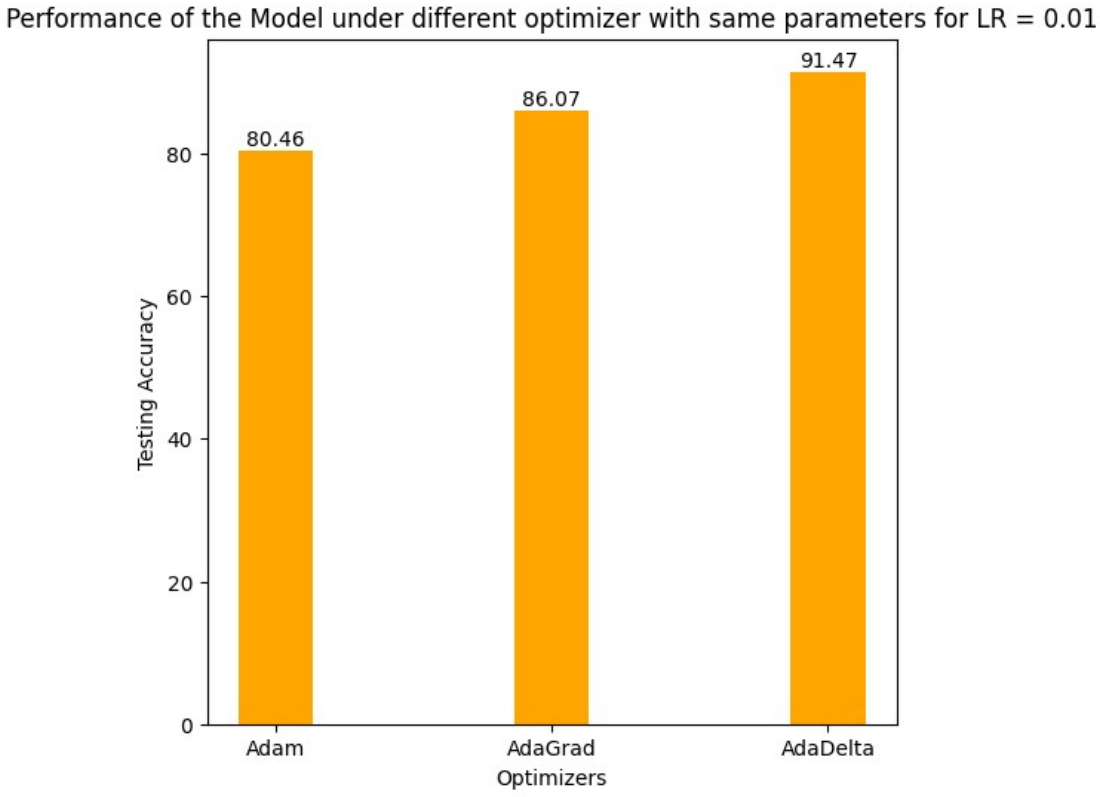


#Visualisation of Performance Comparisons

#Learning Rate: 0.01

```
import matplotlib.pyplot as plt

# creating the dataset
data = {'Adam':80.46, 'AdaGrad':86.07, 'AdaDelta':91.47}
optimizer = list(data.keys())
accuracy = list(data.values())

fig = plt.figure(figsize = (6, 6))
```

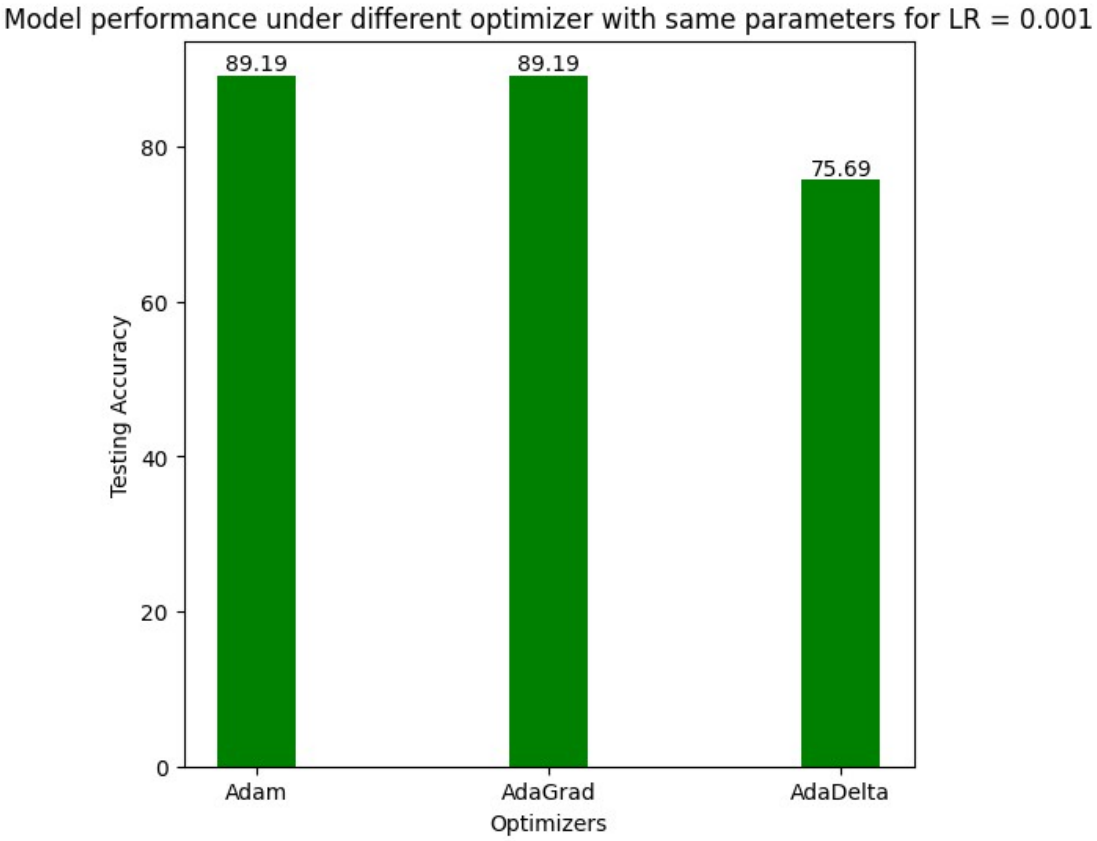
```
# creating the bar plot
plt.bar(optimizer, accuracy, color ='orange',
        width = 0.27)

# Add value labels to the top of each bar
for i, v in enumerate(accuracy):
    plt.text(i, v, str(v), ha='center', va='bottom')

plt.xlabel("Optimizers")
plt.ylabel("Testing Accuracy")
plt.title("Performance of the Model under different optimizer with
same parameters for LR = 0.01")
plt.show()
```



#Learning Rate: 0.001

```
import matplotlib.pyplot as plt

# creating the dataset
data = {'Adam':89.19, 'AdaGrad':89.19, 'AdaDelta':75.69}
optimizer = list(data.keys())
accuracy = list(data.values())

fig = plt.figure(figsize = (6, 6))

# creating the bar plot
```

```
plt.bar(optimizer, accuracy, color ='green',
        width = 0.27)

# Add value labels to the top of each bar
for i, v in enumerate(accuracy):
    plt.text(i, v, str(v), ha='center', va='bottom')

plt.xlabel("Optimizers")
plt.ylabel("Testing Accuracy")
plt.title("Model performance under different optimizer with same
parameters for LR = 0.001")
plt.show()
```



##AdaDelta performed the best(Highest Accuracy) among the three optimizers we used So, we ran AdaDelta thrice for 3 different Learning Rates

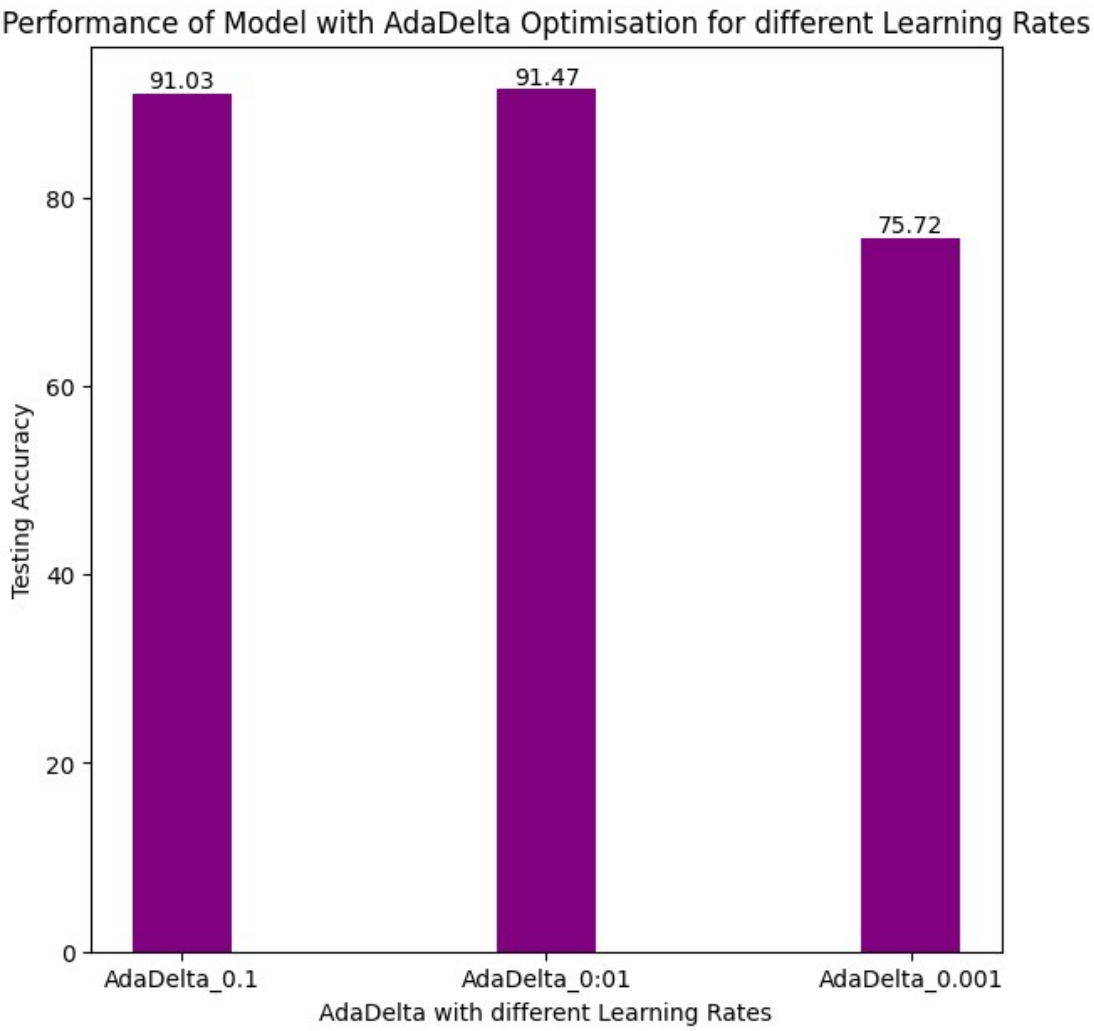
#Learning Rate 1 = 0.1

#Learning Rate 2 = 0.01

#Learning Rate 3 = 0.001

Below is the visualisation to show how changing LR for a particular optimiser affected the accuracy.

```
import matplotlib.pyplot as plt

# creating the dataset
data = {'AdaDelta_0.1':91.03, 'AdaDelta_0:01':91.47,
'AdaDelta_0.001':75.72}
optimizer = list(data.keys())
accuracy = list(data.values())

fig = plt.figure(figsize = (7, 7))

# creating the bar plot
plt.bar(optimizer, accuracy, color ='purple',
        width = 0.27)
# Add value labels to the top of each bar
for i, v in enumerate(accuracy):
    plt.text(i, v, str(v), ha='center', va='bottom')

plt.xlabel("AdaDelta with different Learning Rates")
plt.ylabel("Testing Accuracy")
plt.title("Performance of Model with AdaDelta Optimisation for
different Learning Rates")
plt.show()
```


Performance of Model with AdaDelta Optimisation for different Learning Rates
91.03
91.47
75.72
80
60
40
20
0
Testing Accuracy
AdaDelta_0.1
AdaDelta_0:01
AdaDelta_0.001
AdaDelta with different Learning Rates