המחלקה להנדסת חשמל ואלקטרוניקה

(31245) מערכות לומדות ולמידה עמוקה

Lab 8 report

פרנסיס עבוד

Prepared for: Dr. Amer Adler

Date: 02/06/2025

Ex.1:

Train the given network (split the trainset to train 97.5% and validation of 2.5%), and test the impact on testing set classification accuracy using the following optimizers and options, plot a graph of training & validation accuracy vs. # of epochs for each task:

A. SGD (no decay, no Momentum, no Nesterov): How many Epochs are needed for test accuracy >88%? Try several learning rates.

B. SGD (with decay, no Momentum, no Nesterov): How many Epochs are needed for test accuracy >88%? Try several learning rates.

C. SGD (no decay, with Momentum, no Nesterov): How many Epochs are needed for test accuracy >88%? Try several learning rates.

D. SGD (no decay, with Momentum, with Nesterov): How many Epochs are needed for test accuracy >88%? Try several learning rates.

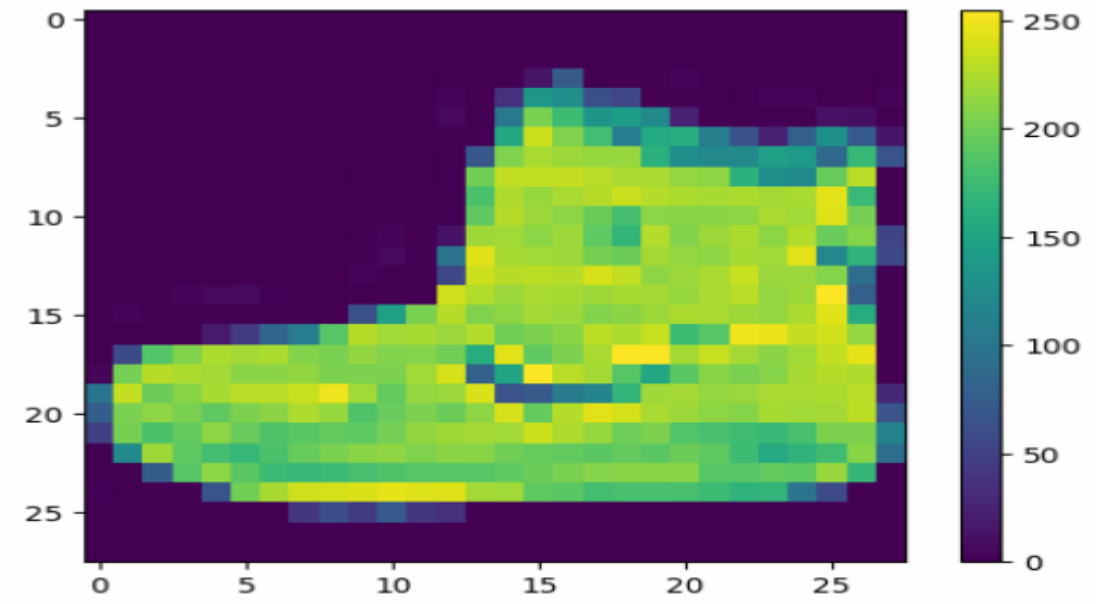
E. AdaGrad: How many Epochs are needed for test accuracy >88%? Try several parameters values.

F. RMSprop: How many Epochs are needed for test accuracy >88%? Try several parameters values.

G. ADAM: How many Epochs are needed for test accuracy >88%? Try several parameters values.


```
# TensorFlow and tf.keras
import tensorflow as tf

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)

fashion_mnist = tf.keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
plt.show()

train_images = train_images / 255.0

test_images = test_images / 255.0

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```

Plots:


0
5
10
15
20
25
0
5
10
15
20
25
250
200
150
100
50
0

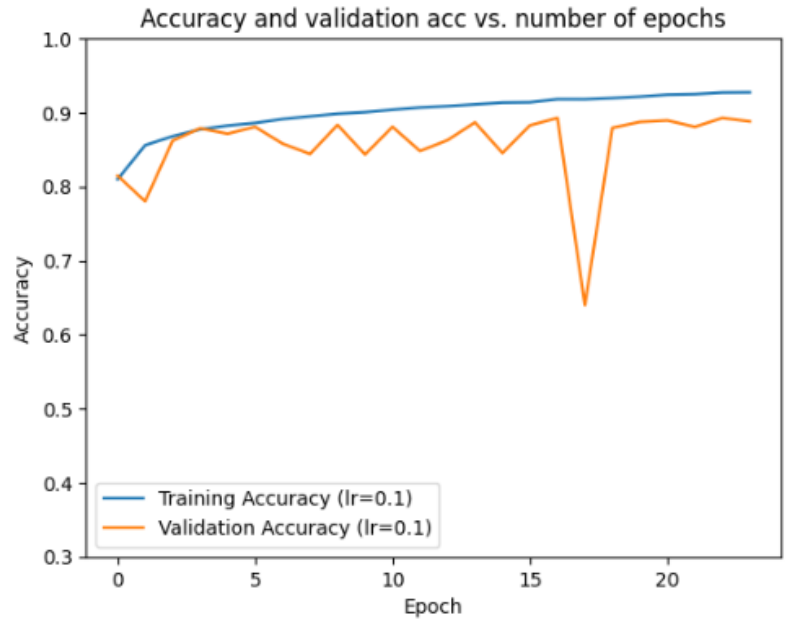

Ankle boot
T-shirt/top
T-shirt/top
Dress
T-shirt/top
Pullover
Sneaker
Pullover
Sandal
Sandal
T-shirt/top
Ankle boot
Sandal
Sandal
Sneaker
Ankle boot
Trouser
T-shirt/top
Shirt
Coat
Dress
Trouser
Coat
Bag
Coat

A.Code:

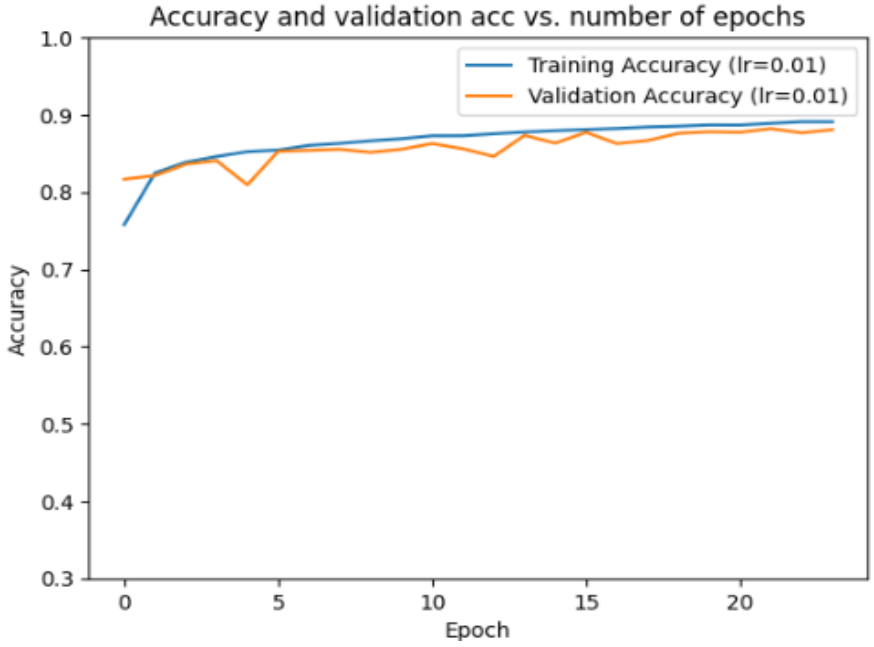
```
learning_rate = [0.1, 0.01, 0.001]
for lr in learning_rate:
    model = tf.keras.Sequential([
        tf.keras.layers.Flatten(input_shape=(28, 28)),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dense(10)
    ])
    model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate = lr),
                  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                  metrics=['accuracy'])

    history = model.fit(train_images, train_labels, epochs=24, validation_split = 0.025 )

    test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

    print('\nTest accuracy:', test_acc)

    probability_model = tf.keras.Sequential([model,
                                             tf.keras.layers.Softmax()])
    predictions = probability_model.predict(test_images)


    plt.plot(history.history['accuracy'], label=f'Training Accuracy (lr={lr})')
    plt.plot(history.history['val_accuracy'], label=f'Validation Accuracy (lr={lr})')

    plt.title("Accuracy and validation acc vs. number of epochs")
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.ylim([0.3,1])
    plt.legend()
    plt.show()
```

A.Output:

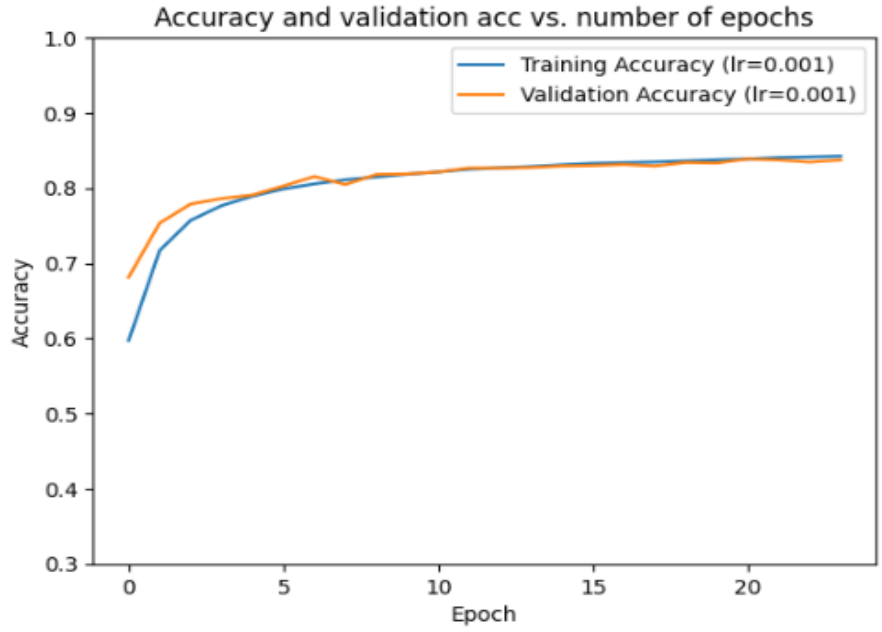
```
Test accuracy: 0.8827000260353088
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```



```
Test accuracy: 0.8711000084877014
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```




```
Test accuracy: 0.830299973487854
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```

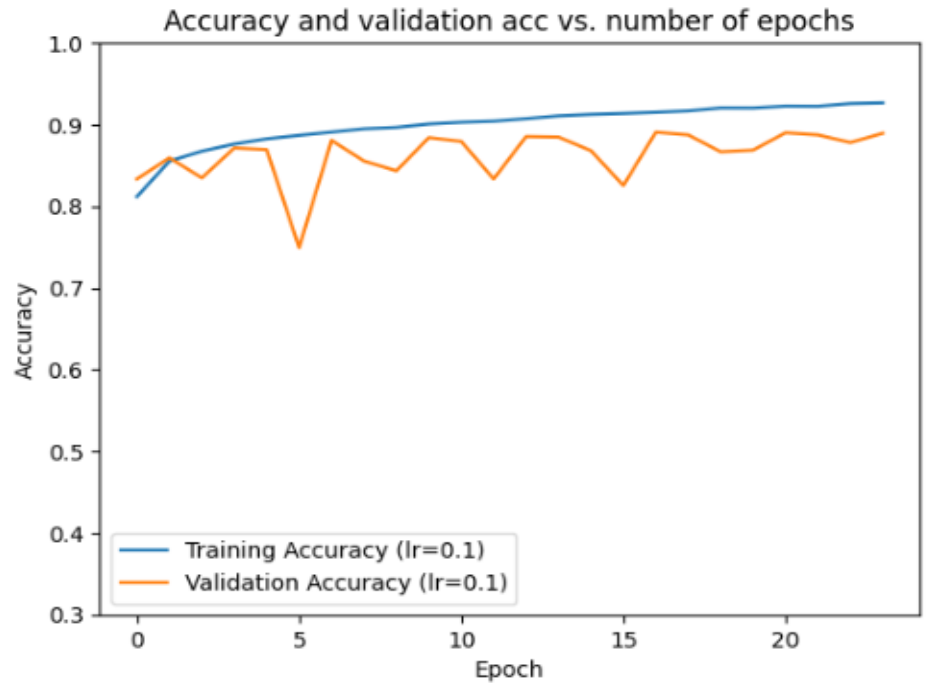


B.Code:

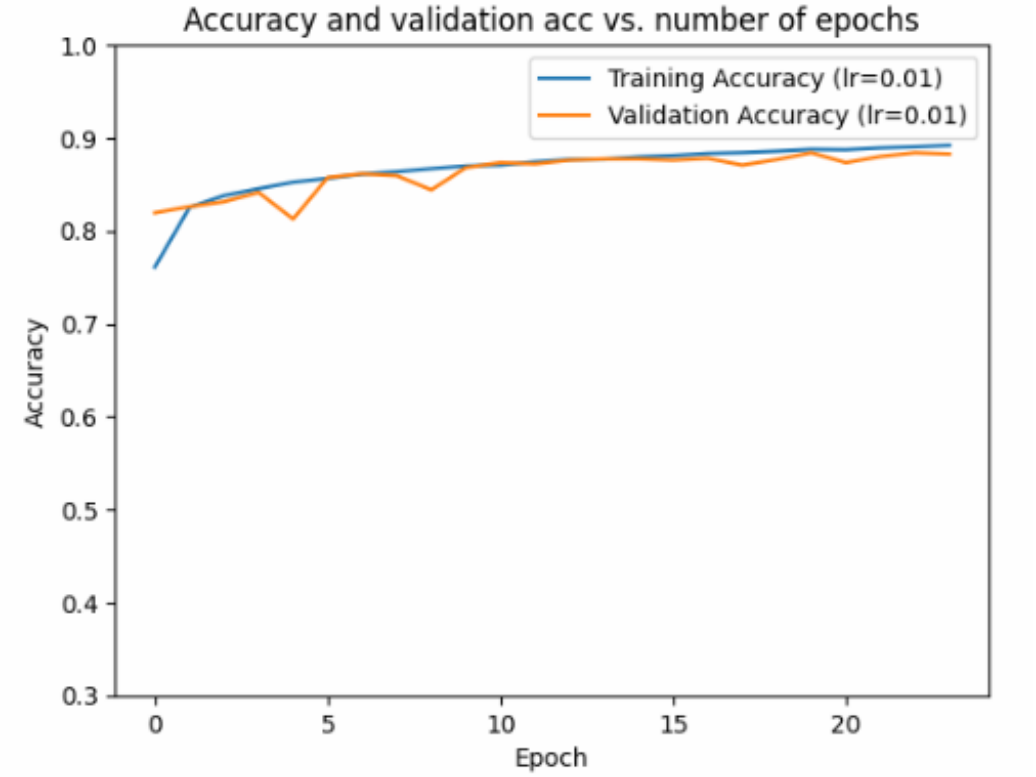
```
learning_rate = [0.1, 0.01, 0.001]
for lr in learning_rate:
    model = tf.keras.Sequential([
        tf.keras.layers.Flatten(input_shape=(28, 28)),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dense(10)
    ])
    model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate = lr , weight_decay= 10e-6),
                  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                  metrics=['accuracy'])

    history = model.fit(train_images, train_labels, epochs=24, validation_split = 0.025 )

    test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

    print('\nTest accuracy:', test_acc)

    probability_model = tf.keras.Sequential([model,
                                             tf.keras.layers.Softmax()])
    predictions = probability_model.predict(test_images)


    plt.plot(history.history['accuracy'], label=f'Training Accuracy (lr={lr})')
    plt.plot(history.history['val_accuracy'], label=f'Validation Accuracy (lr={lr})')

    plt.title("Accuracy and validation acc vs. number of epochs")
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.ylim([0.3,1])
    plt.legend()
    plt.show()
```

B.Output:

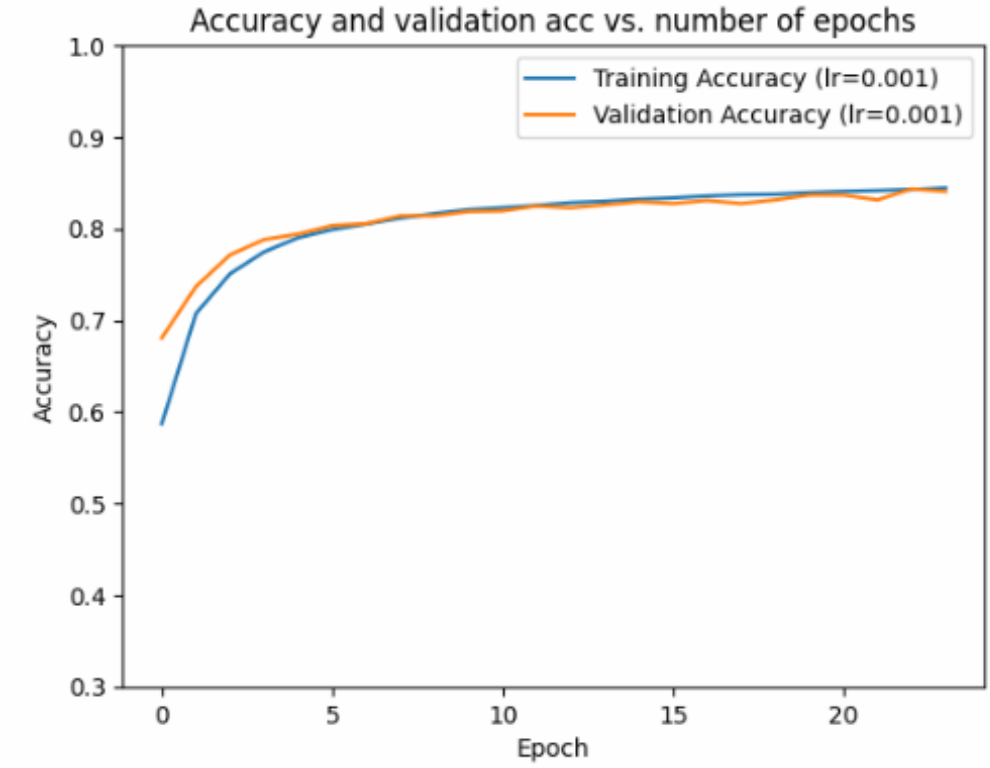
```
Test accuracy: 0.8884000182151794
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
```



```
Test accuracy: 0.8629000186920166
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```




```
Test accuracy: 0.8309999704360962
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```

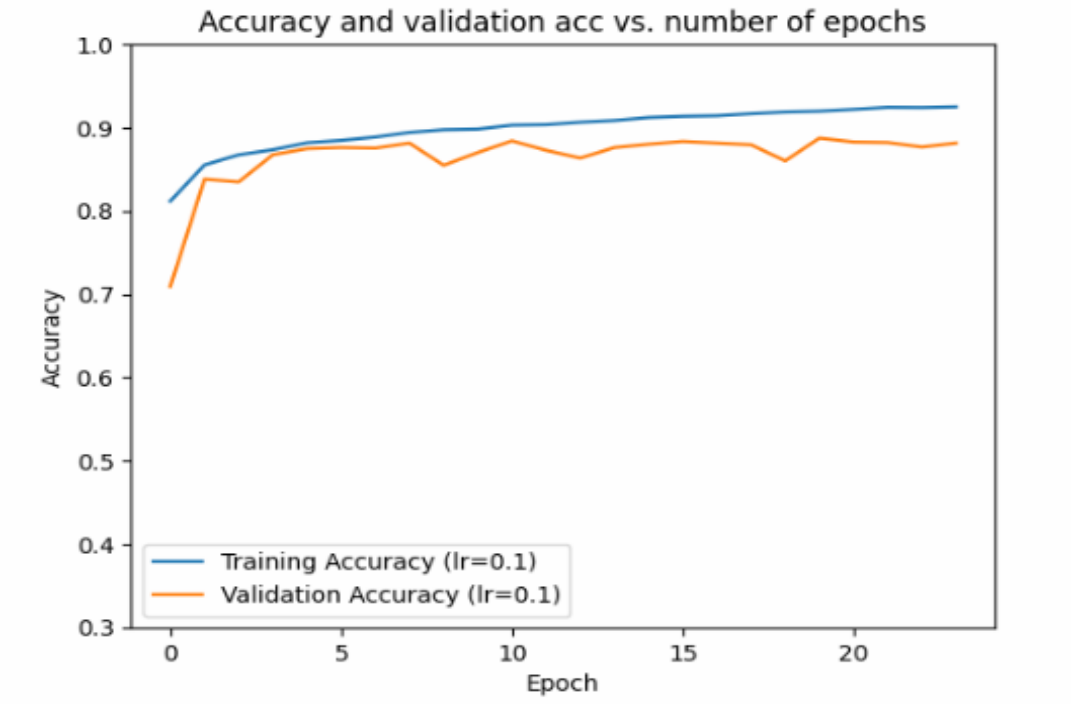


C.Code:

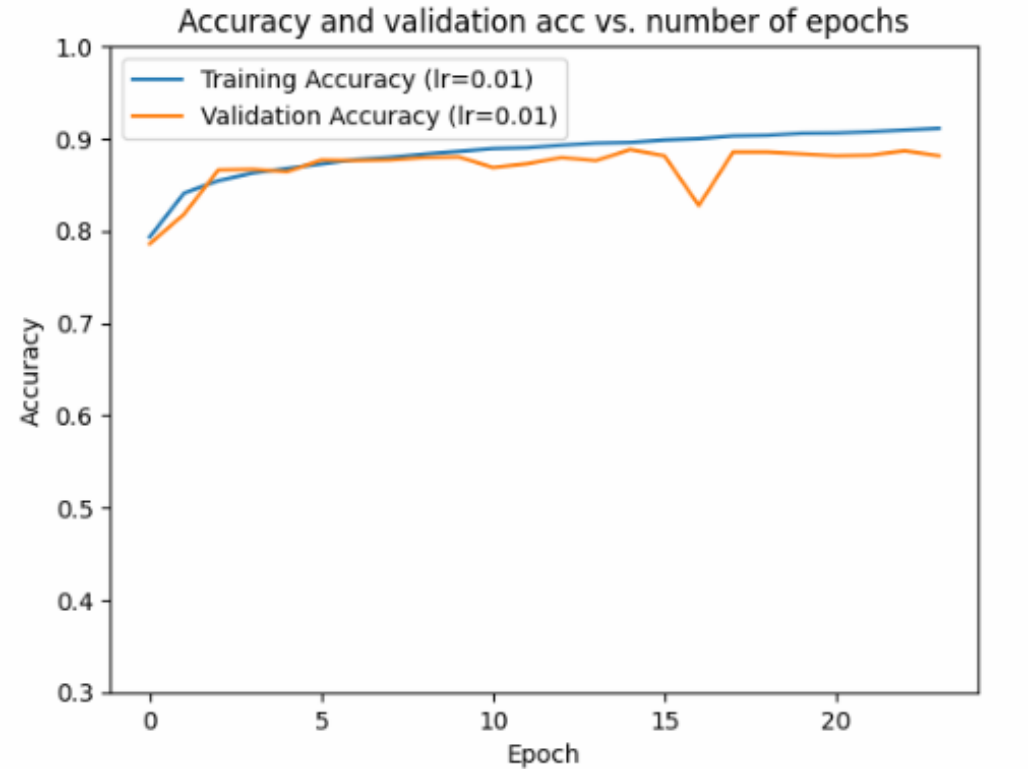
```
learning_rate = [0.1, 0.01, 0.001]
for lr in learning_rate:
    model = tf.keras.Sequential([
        tf.keras.layers.Flatten(input_shape=(28, 28)),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dense(10)
    ])
    model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate = lr , momentum = 0.6),
                  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                  metrics=['accuracy'])

    history = model.fit(train_images, train_labels, epochs=24, validation_split = 0.025 )

    test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

    print('\nTest accuracy:', test_acc)

    probability_model = tf.keras.Sequential([model,
                                             tf.keras.layers.Softmax()])
    predictions = probability_model.predict(test_images)


    plt.plot(history.history['accuracy'], label=f'Training Accuracy (lr={lr})')
    plt.plot(history.history['val_accuracy'], label=f'Validation Accuracy (lr={lr})')

    plt.title("Accuracy and validation acc vs. number of epochs")
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.ylim([0.3,1])
    plt.legend()
    plt.show()
```

C.Output:

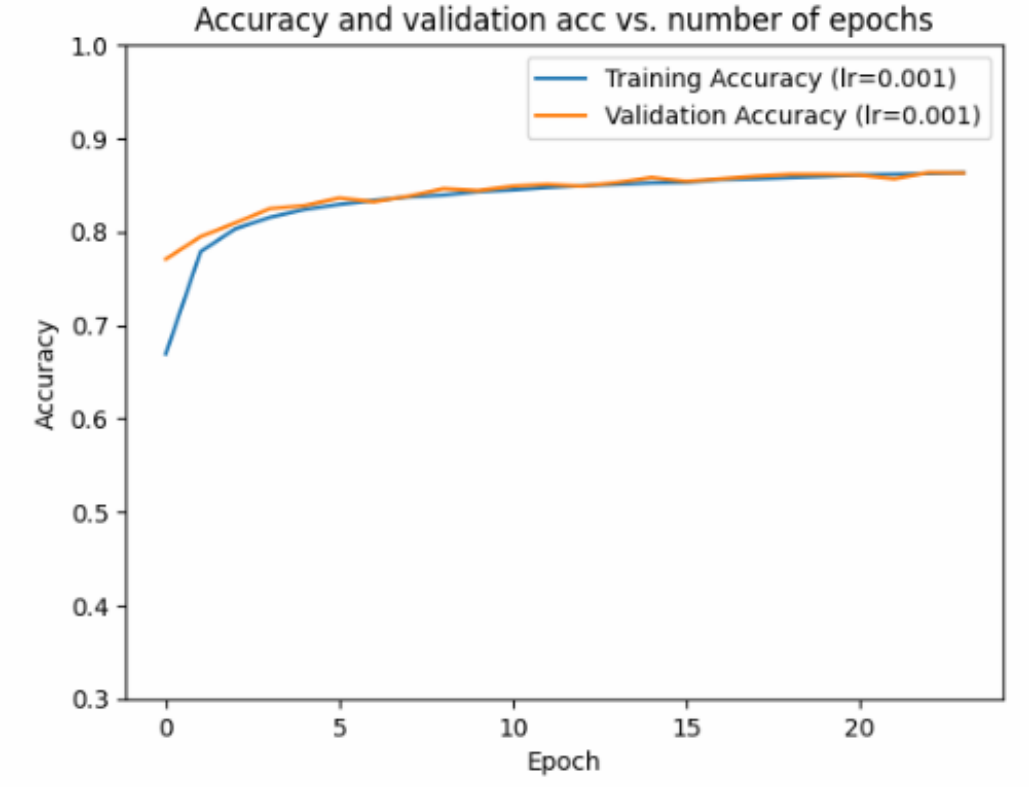
```
Test accuracy: 0.8806999921798706
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```



```
Test accuracy: 0.8712000250816345
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
```




```
Test accuracy: 0.8500999808311462
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```

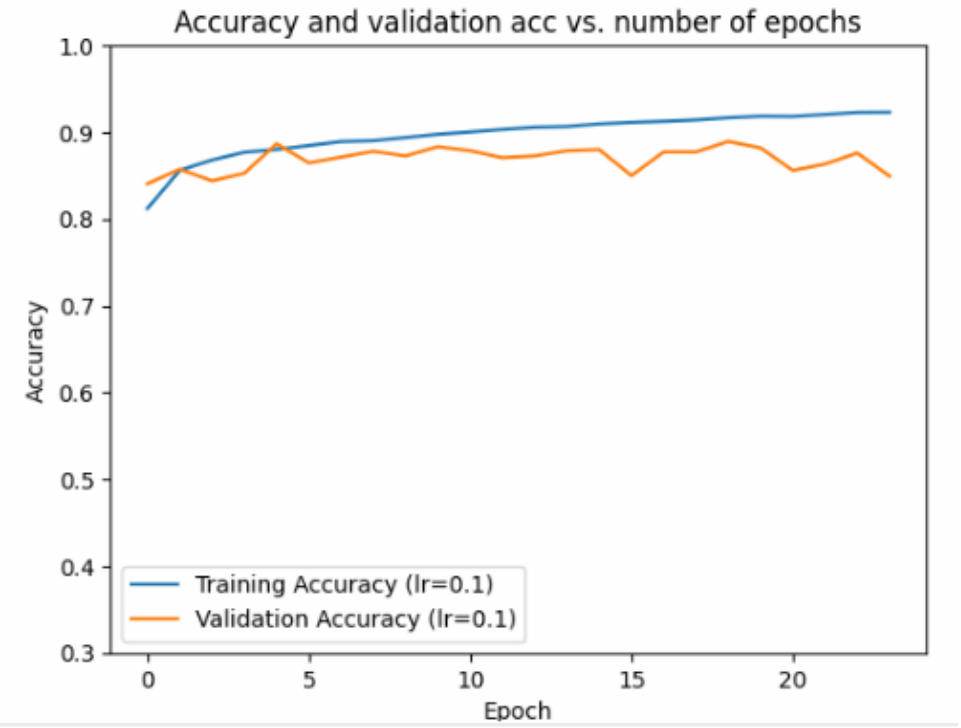


D.Code:

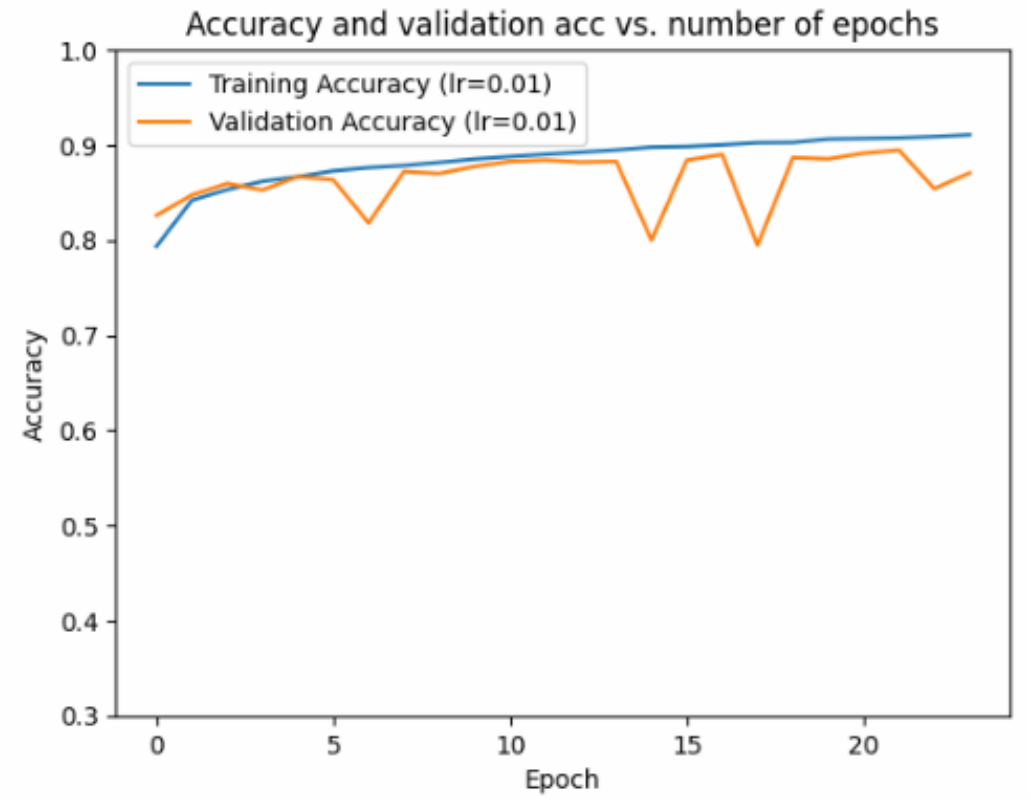
```
learning_rate = [0.1, 0.01, 0.001]
for lr in learning_rate:
    model = tf.keras.Sequential([
        tf.keras.layers.Flatten(input_shape=(28, 28)),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dense(10)
    ])
    model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate = lr , momentum = 0.6, nesterov = True),
                  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                  metrics=['accuracy'])

    history = model.fit(train_images, train_labels, epochs=24, validation_split = 0.025 )

    test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

    print('\nTest accuracy:', test_acc)

    probability_model = tf.keras.Sequential([model,
                                             tf.keras.layers.Softmax()])
    predictions = probability_model.predict(test_images)


    plt.plot(history.history['accuracy'], label=f'Training Accuracy (lr={lr})')
    plt.plot(history.history['val_accuracy'], label=f'Validation Accuracy (lr={lr})')

    plt.title("Accuracy and validation acc vs. number of epochs")
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.ylim([0.3,1])
    plt.legend()
    plt.show()
```

D.Output:

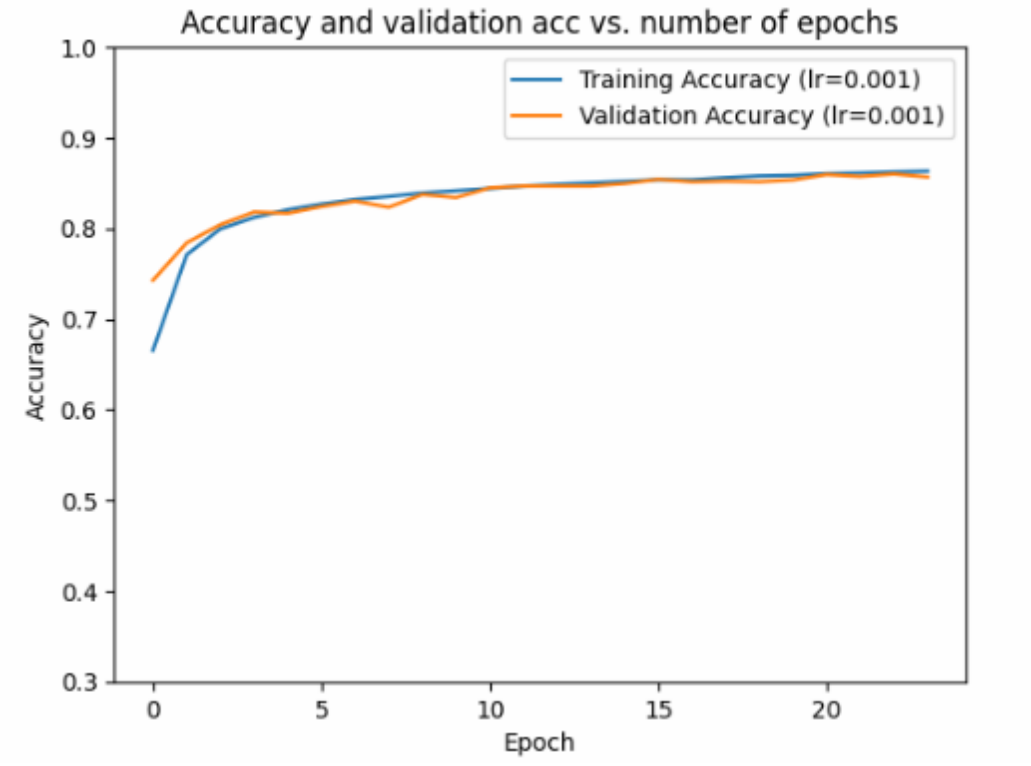
```
Test accuracy: 0.8352000117301941
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```



```
Test accuracy: 0.8615000247955322
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```




```
Test accuracy: 0.8436999917030334
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```

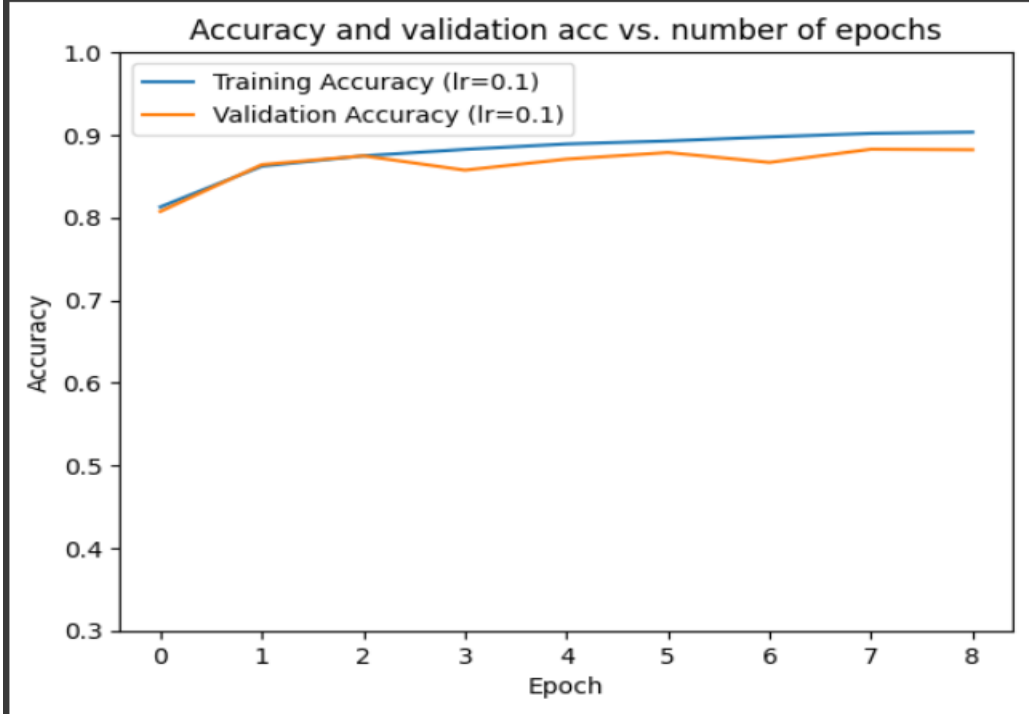


E.Code:

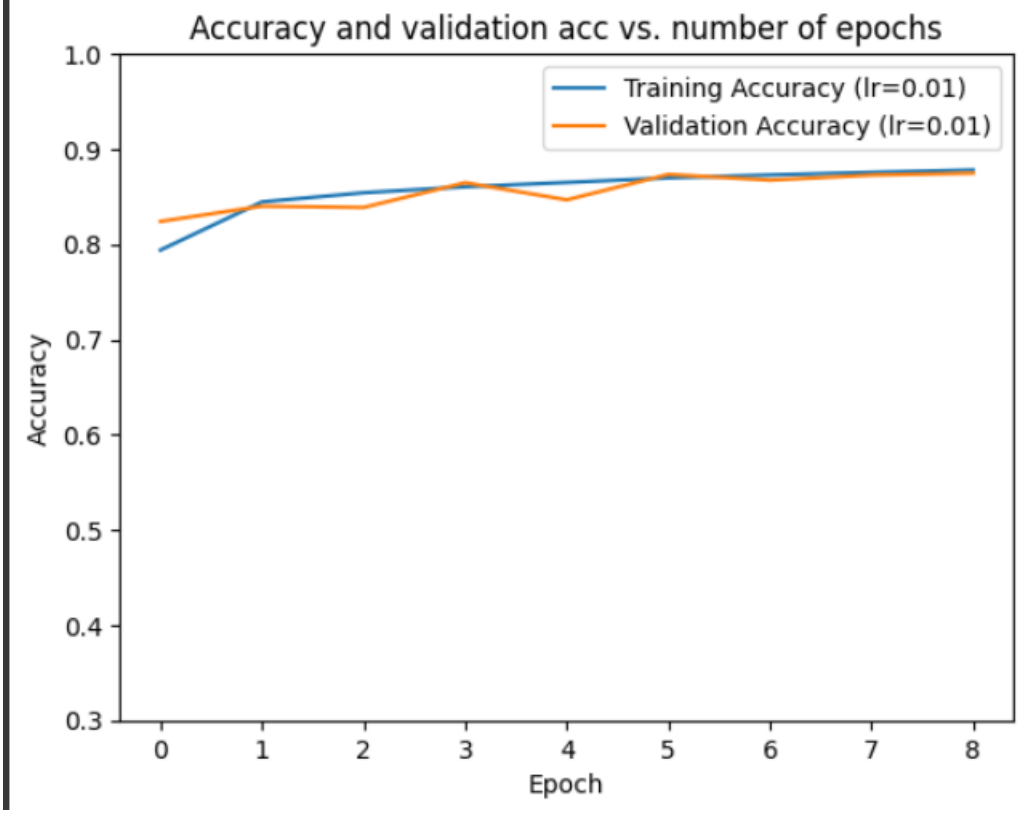
```
learning_rates = [0.1, 0.01, 0.001]
for lr in learning_rates:
     model = tf.keras.Sequential([
     tf.keras.layers.Flatten(input_shape=(28, 28)),
     tf.keras.layers.Dense(128, activation='relu'),
     tf.keras.layers.Dense(10)
     ])
     optimizer1 = tf.keras.optimizers.Adagrad(learning_rate = lr)
     model.compile(optimizer=optimizer1,
               loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
               metrics=['accuracy'])

     history = model.fit(train_images, train_labels, epochs=9, validation_split = 0.025)
     test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

     print('\nTest accuracy:', test_acc)
     probability_model = tf.keras.Sequential([model,
                                              tf.keras.layers.Softmax()])
     predictions = probability_model.predict(test_images)


     plt.plot(history.history['accuracy'], label=f'Training Accuracy (lr={lr})')
     plt.plot(history.history['val_accuracy'], label=f'Validation Accuracy (lr={lr})')

     plt.title("Accuracy and validation acc vs. number of epochs")
     plt.xlabel('Epoch')
     plt.ylabel('Accuracy')
     plt.ylim([0.3,1])
     plt.legend()
     plt.show()
```

E.Output:

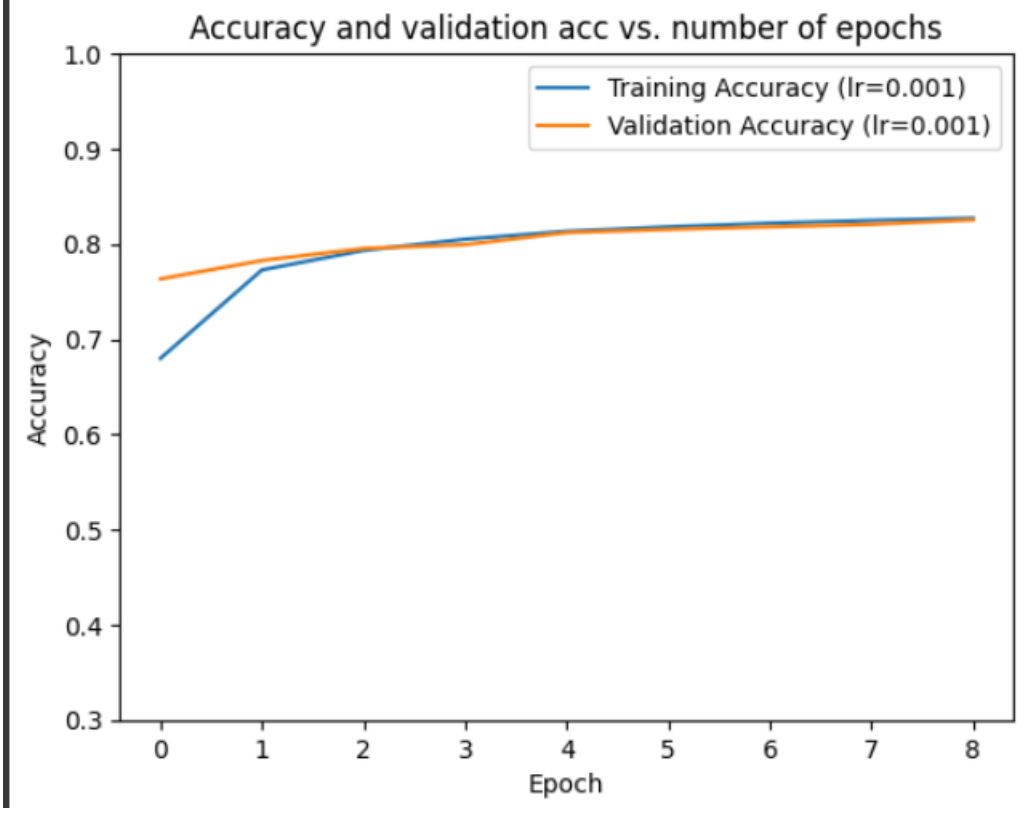
```
Test accuracy: 0.8835999965667725
313/313 [==============================] - 0s 1ms/step
```



```
Test accuracy: 0.8593999743461609
313/313 [==============================] - 1s 2ms/step
```




```
Test accuracy: 0.8144999742507935
313/313 [==============================] - 0s 1ms/step
```

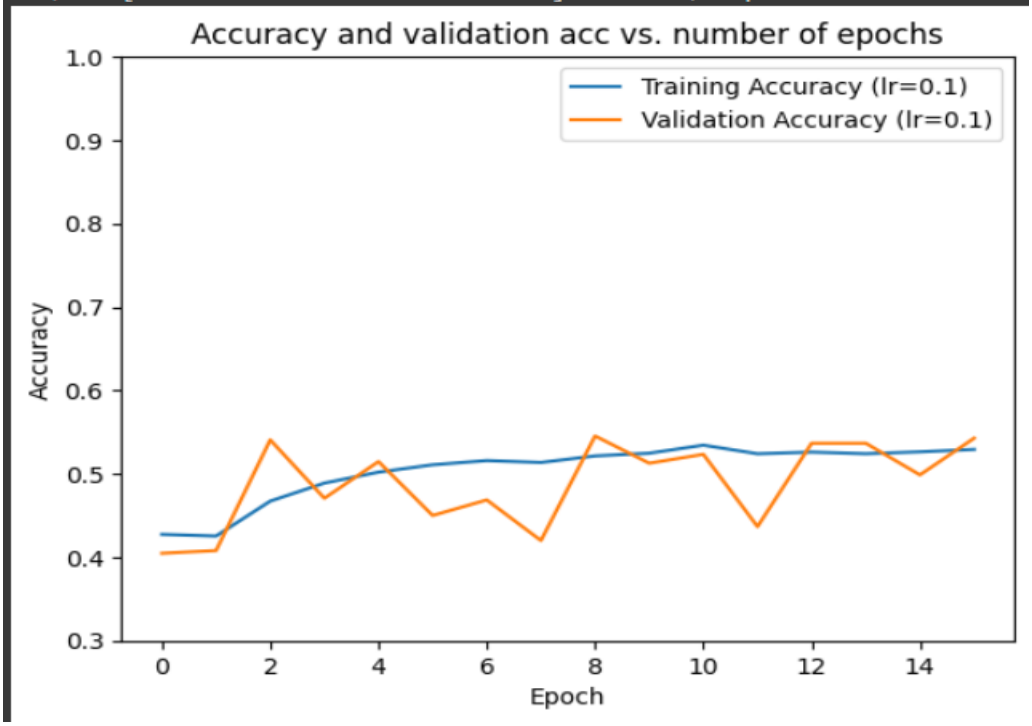


F.Code:

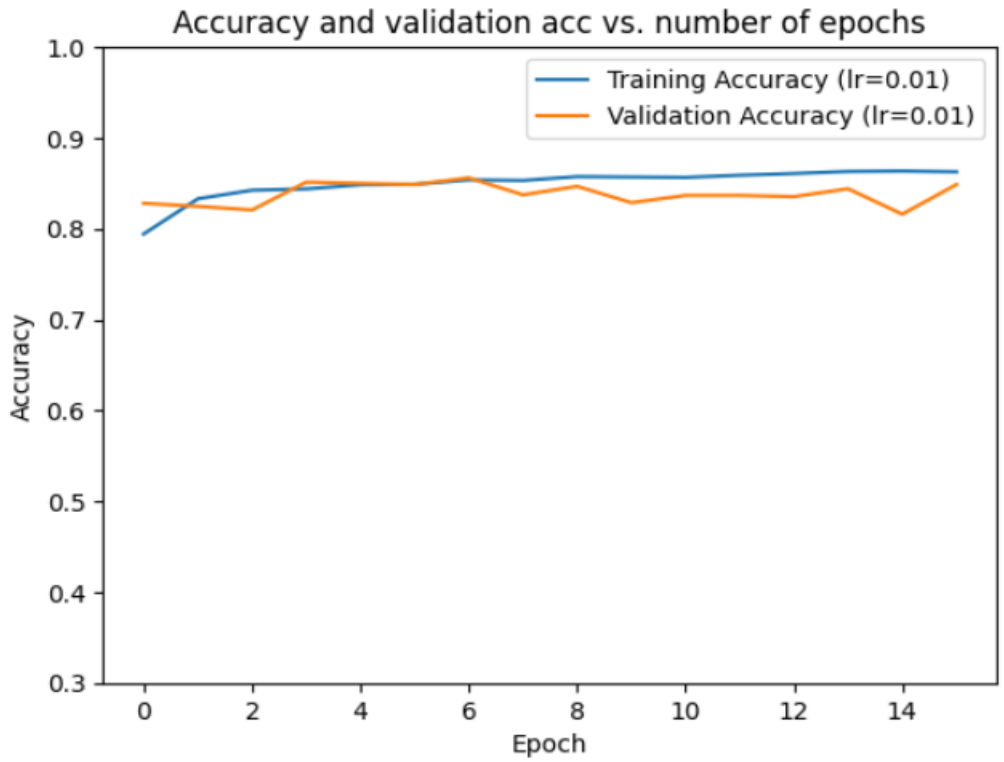
```
learning_rates = [0.1, 0.01, 0.001]
for lr in learning_rates:
     model = tf.keras.Sequential([
     tf.keras.layers.Flatten(input_shape=(28, 28)),
     tf.keras.layers.Dense(128, activation='relu'),
     tf.keras.layers.Dense(10)
     ])
     optimizer1 = tf.keras.optimizers.RMSprop(learning_rate = lr)
     model.compile(optimizer=optimizer1,
               loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
               metrics=['accuracy'])

     history = model.fit(train_images, train_labels, epochs=16, validation_split = 0.025)
     test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

     print('\nTest accuracy:', test_acc)
     probability_model = tf.keras.Sequential([model,
                                              tf.keras.layers.Softmax()])
     predictions = probability_model.predict(test_images)


     plt.plot(history.history['accuracy'], label=f'Training Accuracy (lr={lr})')
     plt.plot(history.history['val_accuracy'], label=f'Validation Accuracy (lr={lr})')

     plt.title("Accuracy and validation acc vs. number of epochs")
     plt.xlabel('Epoch')
     plt.ylabel('Accuracy')
     plt.ylim([0.3,1])
     plt.legend()
     plt.show()
```

F.Output:

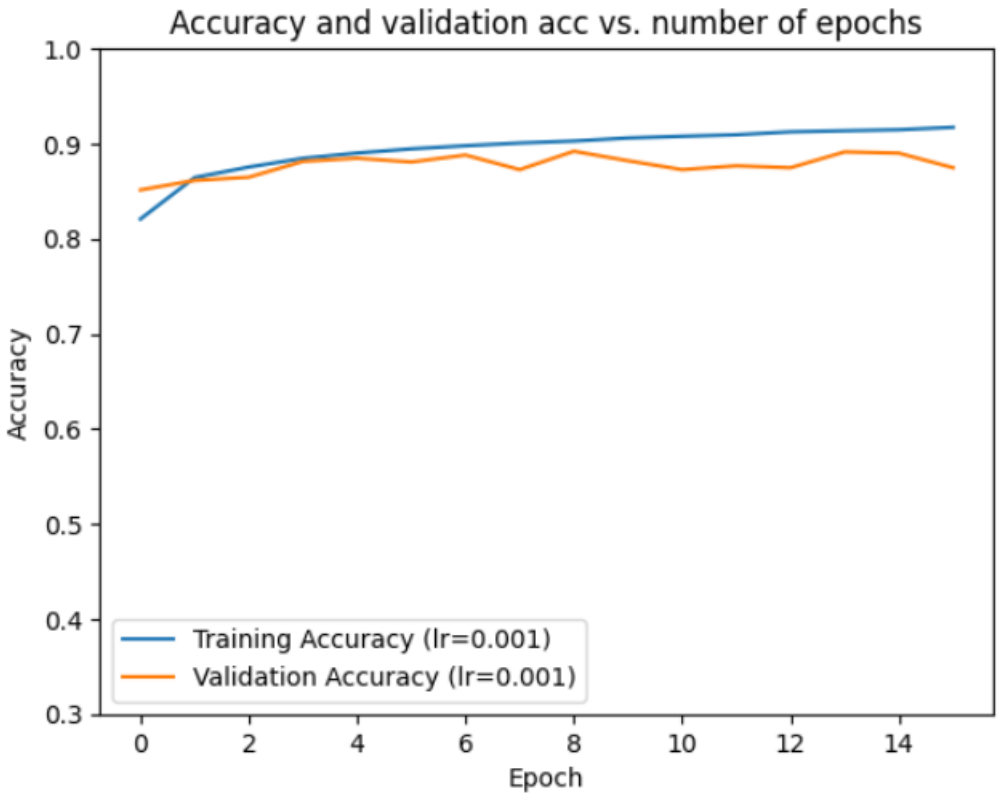
```
Test accuracy: 0.5242000222206116
313/313 [==============================] - 0s 1ms/step
```



```
Test accuracy: 0.840499997138977
313/313 [==============================] - 0s 1ms/step
```




```
Test accuracy: 0.8691999912261963
313/313 [==============================] - 0s 1ms/step
```

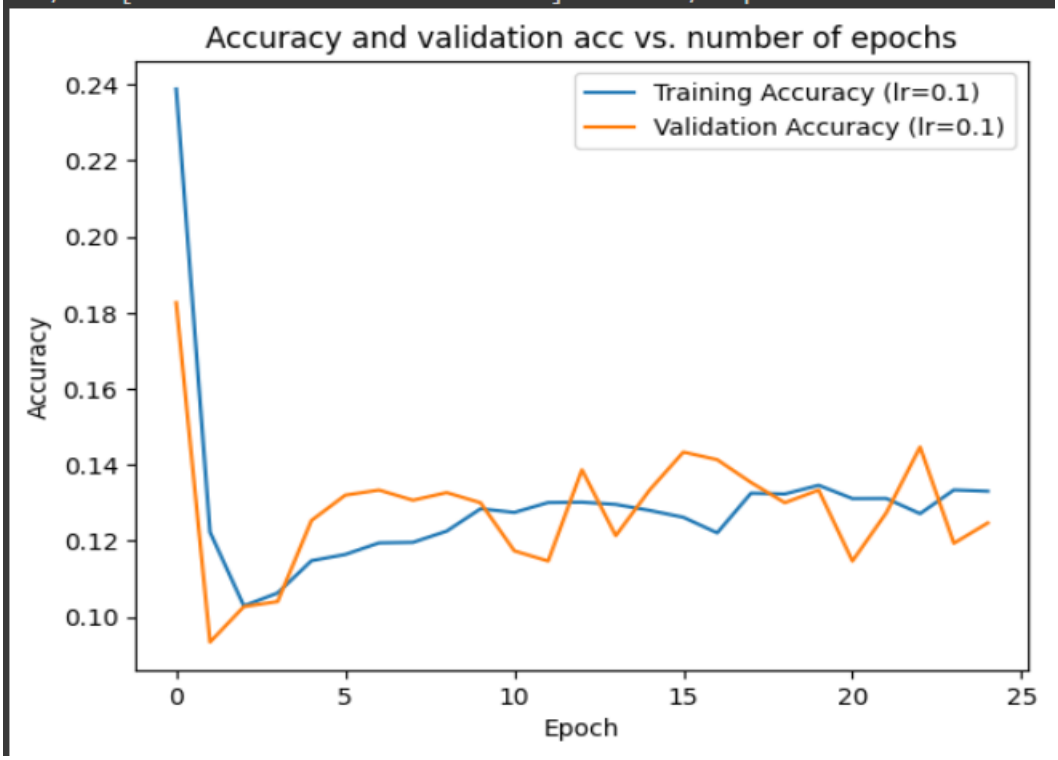


G.Code:

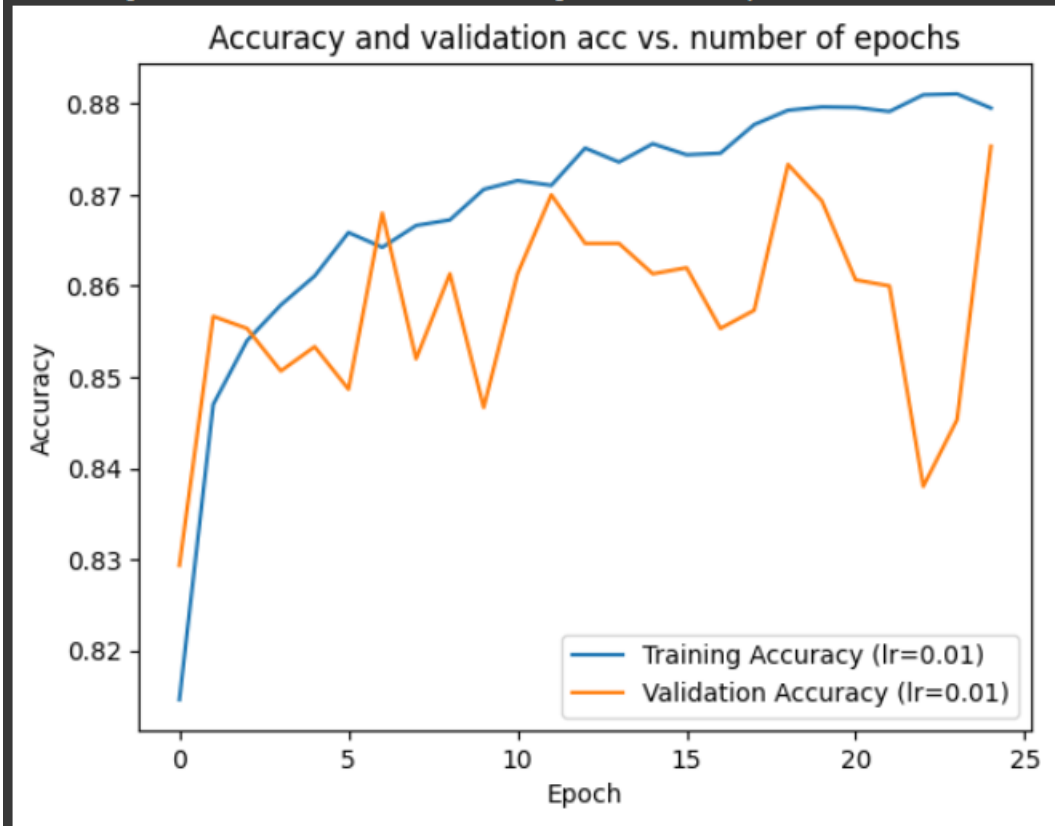
```
learning_rates = [0.1, 0.01, 0.001]
for lr in learning_rates:
     model = tf.keras.Sequential([
     tf.keras.layers.Flatten(input_shape=(28, 28)),
     tf.keras.layers.Dense(128, activation='relu'),
     tf.keras.layers.Dense(10)
     ])
     optimizer1 = tf.keras.optimizers.Adam(learning_rate = lr)
     model.compile(optimizer=optimizer1,
               loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
               metrics=['accuracy'])

     history = model.fit(train_images, train_labels, epochs=25, validation_split = 0.025)
     test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

     print('\nTest accuracy:', test_acc)
     probability_model = tf.keras.Sequential([model,
                                              tf.keras.layers.Softmax()])
     predictions = probability_model.predict(test_images)


     plt.plot(history.history['accuracy'], label=f'Training Accuracy (lr={lr})')
     plt.plot(history.history['val_accuracy'], label=f'Validation Accuracy (lr={lr})')

     plt.title("Accuracy and validation acc vs. number of epochs")
     plt.xlabel('Epoch')
     plt.ylabel('Accuracy')
     plt.legend()
     plt.show()
```

G.Output:

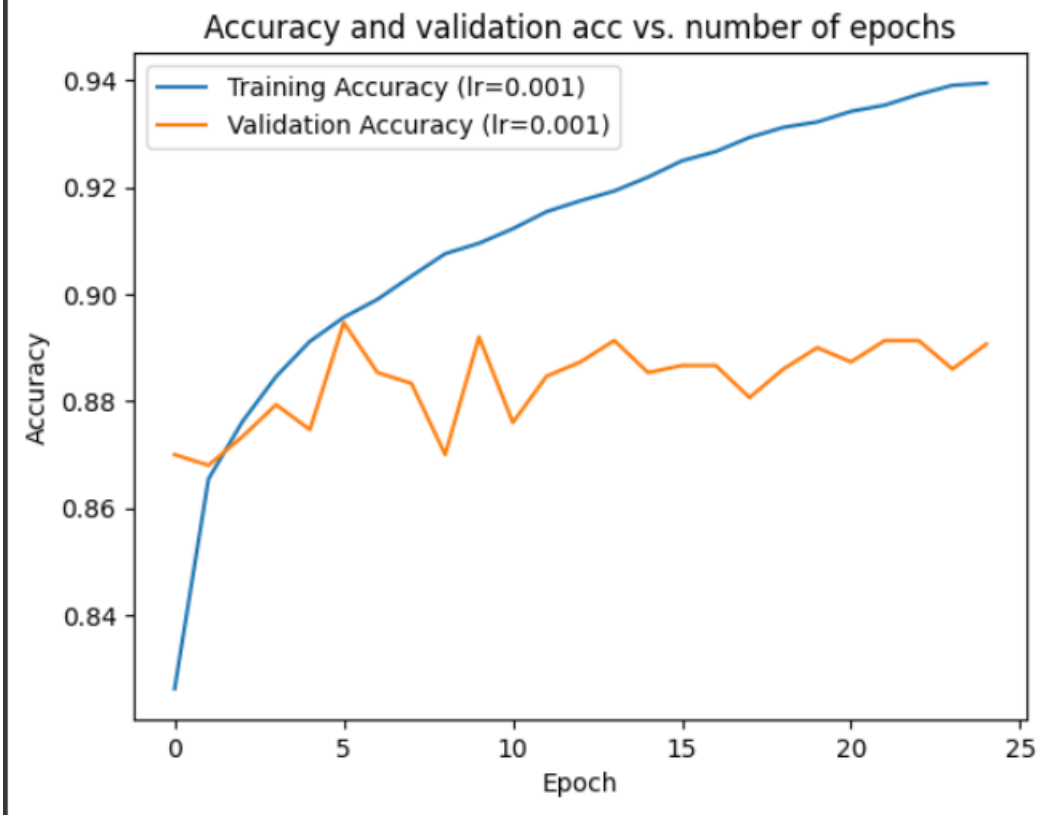
```
Test accuracy: 0.1348000019788742
313/313 [==============================] - 1s 2ms/step
```



```
Test accuracy: 0.8597000241279602
313/313 [==============================] - 0s 1ms/step
```




```
Test accuracy: 0.8881999850273132
313/313 [==============================] - 0s 1ms/step
```

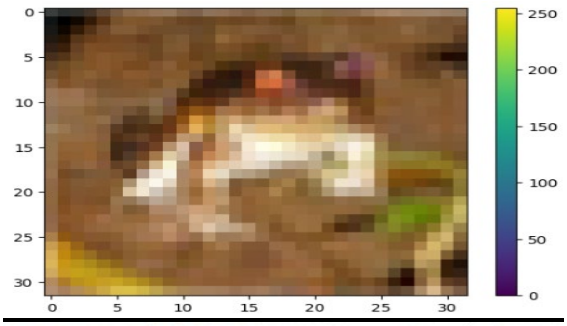


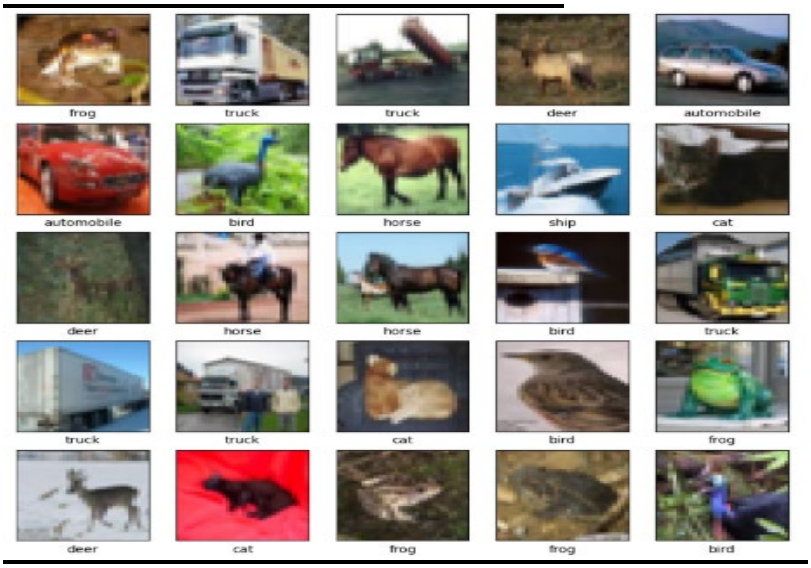
3. Repeat 2, after replacing the MLP network with the CNN network of the CIFAR-10 lab (perform the required changes in the input dimensions)

Code:

```
# TensorFlow and tf.keras
import tensorflow as tf

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)

cifar10 = tf.keras.datasets.cifar10
(train_images, train_labels), (test_images, test_labels) = cifar10.load_data()

class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']


plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
plt.show()

train_images = train_images / 255.0

test_images = test_images / 255.0

train_labels = train_labels.flatten()
test_labels = test_labels.flatten()

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```

Plots:





.Code:

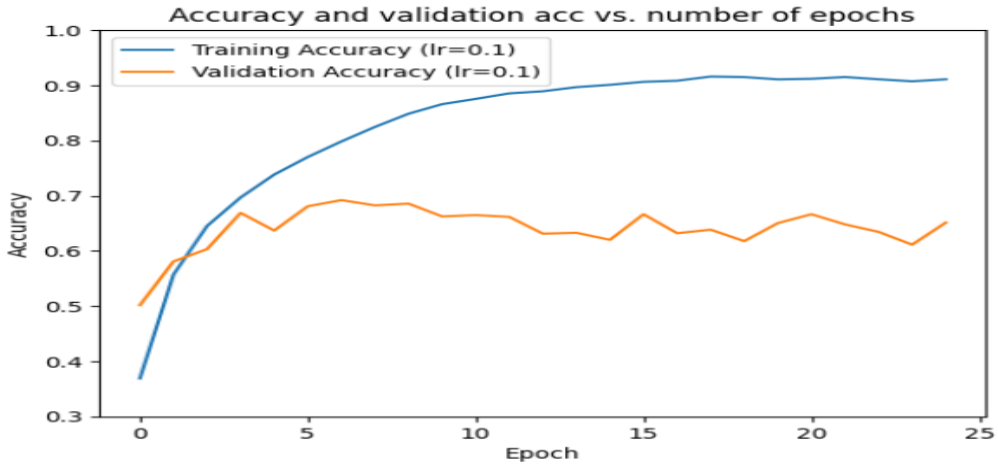
```
# TensorFlow and tf.keras
import tensorflow as tf

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)

cifar10 = tf.keras.datasets.cifar10
(train_images, train_labels), (test_images, test_labels) = cifar10.load_data()

class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']


plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
plt.show()

train_images = train_images / 255.0

test_images = test_images / 255.0

train_labels = train_labels.flatten()
test_labels = test_labels.flatten()

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()

learning_rates = [0.1, 0.01, 0.001]
```

```
# List of optimizer configurations to test
optimizer_configs = [
    # a. SGD (no decay, no Momentum, no Nesterov)
    #{"name": "SGD-basic", "optimizer": lambda lr: tf.keras.optimizers.SGD(learning_rate=lr)},

    # b. SGD (with decay, no Momentum, no Nesterov)
    #{"name": "SGD-decay", "optimizer": lambda lr: tf.keras.optimizers.SGD(learning_rate=lr, weight_decay=10e-6)},

    # c. SGD (no decay, with Momentum, no Nesterov)
    #{"name": "SGD-momentum", "optimizer": lambda lr: tf.keras.optimizers.SGD(learning_rate=lr, momentum=0.6)},

    # d. SGD (no decay, with Momentum, with Nesterov)
    #{"name": "SGD-nesterov", "optimizer": lambda lr: tf.keras.optimizers.SGD(learning_rate=lr, momentum=0.6, nesterov=True)},

    # e. AdaGrad
    #{"name": "AdaGrad", "optimizer": lambda lr: tf.keras.optimizers.Adagrad(learning_rate=lr)},

    # f. RMSprop
    {"name": "RMSprop", "optimizer": lambda lr: tf.keras.optimizers.RMSprop(learning_rate=lr)},

    # g. ADAM
    {"name": "Adam", "optimizer": lambda lr: tf.keras.optimizers.Adam(learning_rate=lr)}
]

# Loop through each optimizer configuration
for config in optimizer_configs:
    print(f"\nTesting {config['name']} optimizer:")

    for lr in learning_rates:
        print(f"\nLearning rate: {lr}")

        # Create CNN model
        model = tf.keras.Sequential([
            tf.keras.layers.Conv2D(32, (3, 3), padding='same', activation='relu', input_shape=(32, 32, 3)),
            tf.keras.layers.MaxPooling2D((2, 2)),
            tf.keras.layers.Conv2D(64, (3, 3), padding='same', activation='relu'),
            tf.keras.layers.MaxPooling2D((2, 2)),
            tf.keras.layers.Conv2D(64, (3, 3), padding='same', activation='relu'),
            tf.keras.layers.Flatten(),
            tf.keras.layers.Dense(64, activation='relu'),
            tf.keras.layers.Dense(10)
        ])

        # Compile with the current optimizer
        optimizer = config["optimizer"](lr)
        model.compile(
            optimizer=optimizer,
            loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
            metrics=['accuracy']
        )
```

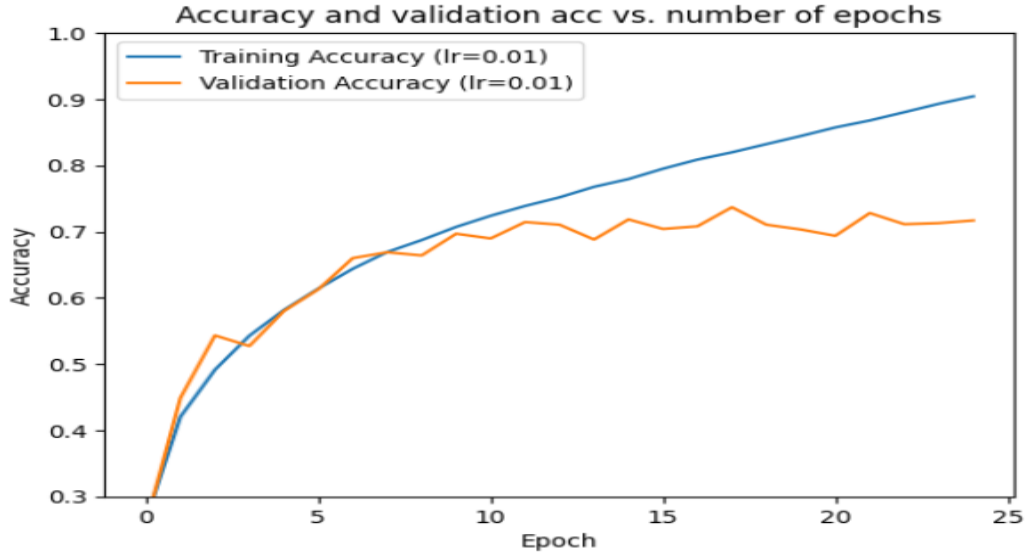
```
        # Train the model
        history = model.fit(
            train_images, train_labels,
            epochs=25,
            validation_split=0.025,
            verbose=1
        )

        # Evaluate
        test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=0)
        print(f'Test accuracy: {test_acc:.4f}')

        # Plot training history
        plt.figure(figsize=(10, 6))
        plt.plot(history.history['accuracy'], label=f'Training Accuracy (lr={lr})')
        plt.plot(history.history['val_accuracy'], label=f'Validation Accuracy (lr={lr})')
        plt.title(f"{config['name']} - Accuracy and validation acc vs. epochs")
        plt.xlabel('Epoch')
        plt.ylabel('Accuracy')
        plt.ylim([0.3, 1])
        plt.legend()
        plt.show()
```

A.Output:

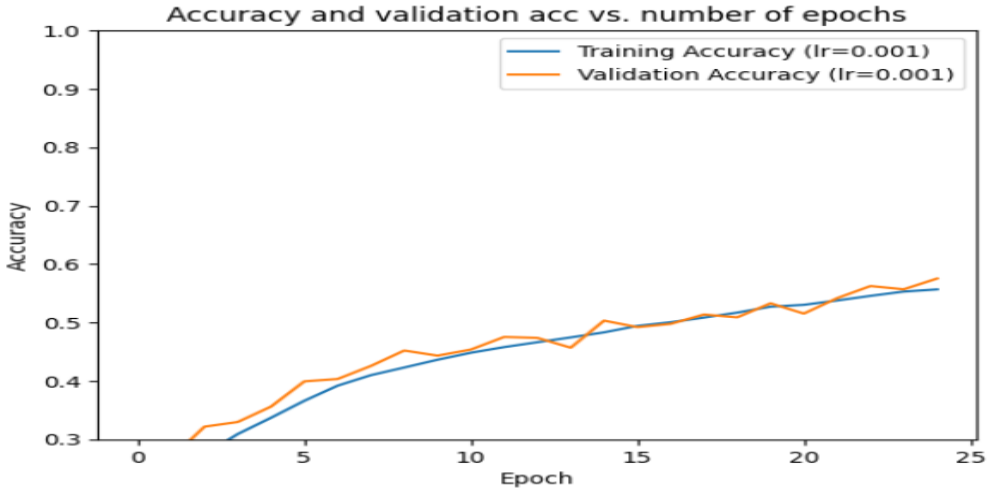
```
Test accuracy: 0.647599995136261
313/313 ━━━━━━━━━━━━━━━━━━━━ 6s 20ms/step
```



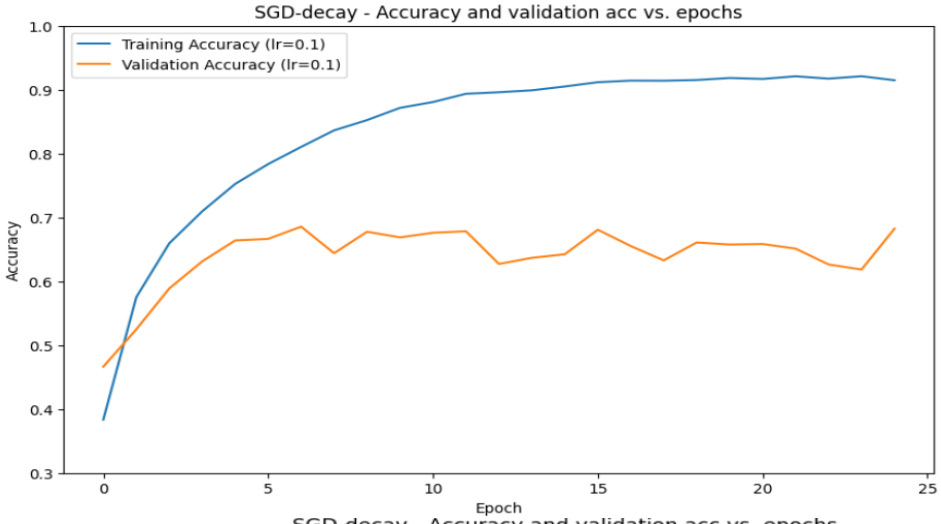

```
Test accuracy: 0.7010999917984009
313/313 ━━━━━━━━━━━━━━━━━━━━ 5s 16ms/step
```

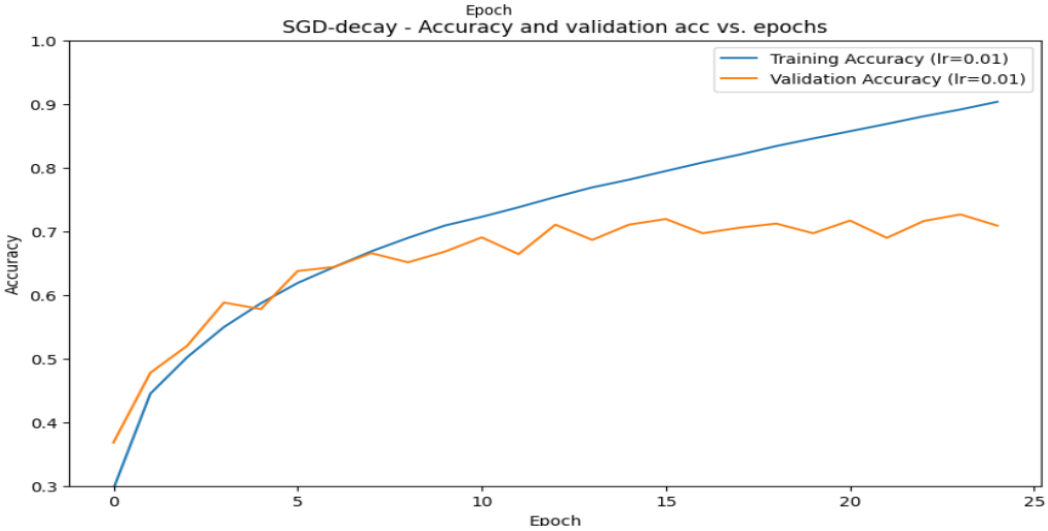



```
Test accuracy: 0.550000011920929
313/313 ━━━━━━━━━━━━━━━━━━━━ 7s 20ms/step
```

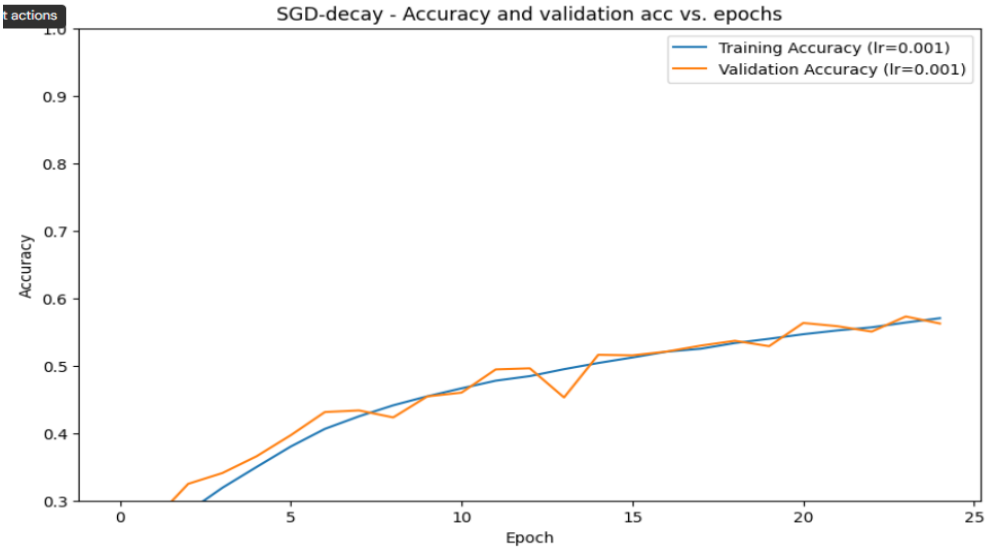


B.Output:



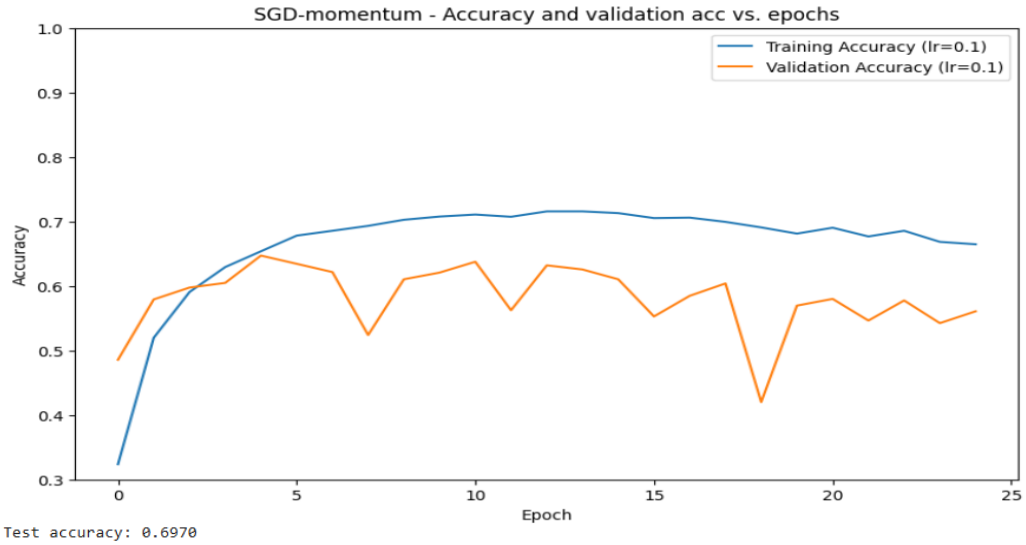


Test accuracy: 0.5566

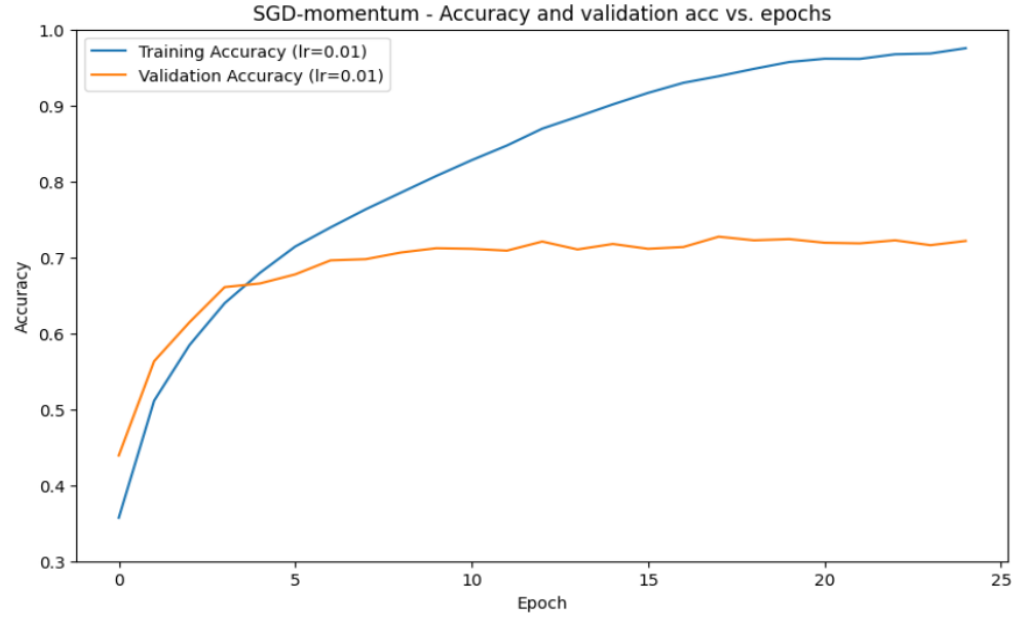


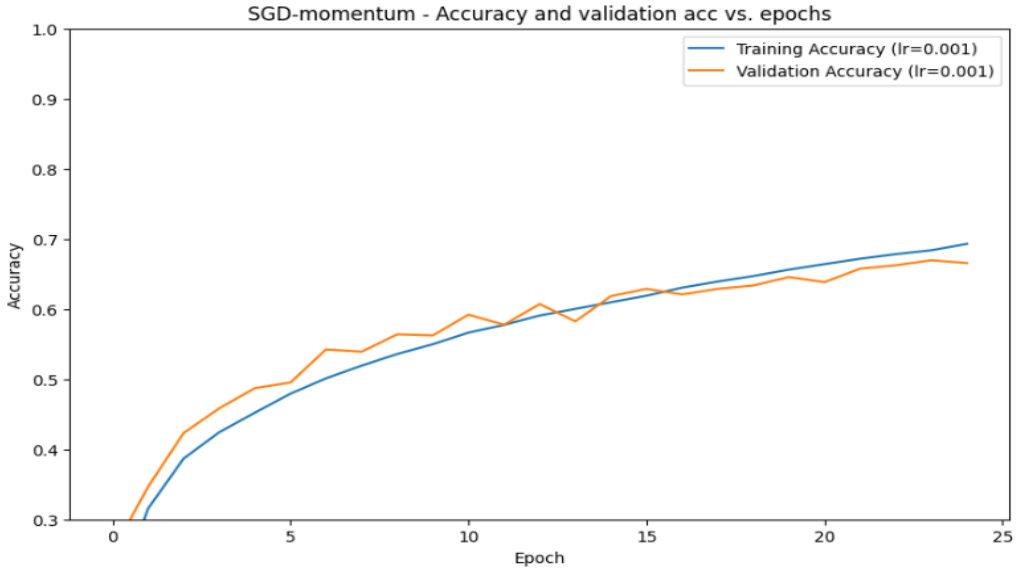
C.Output:

```
Test accuracy: 0.5529
```



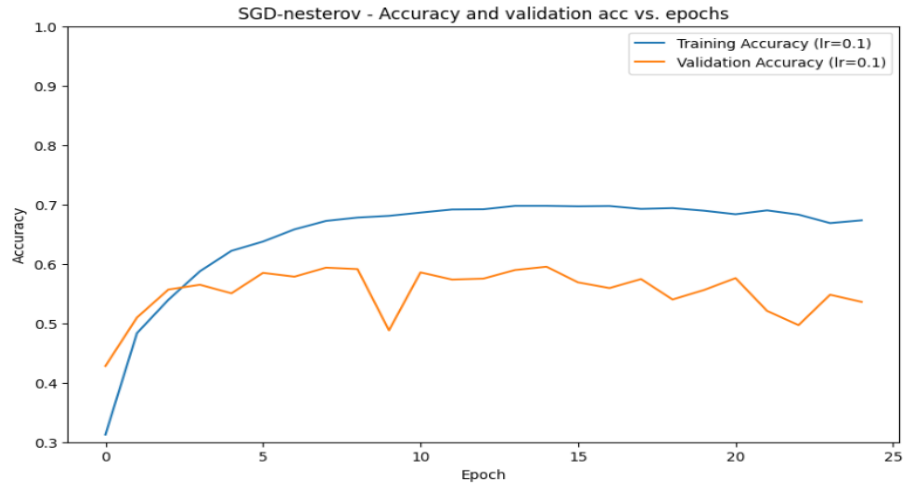

```
Test accuracy: 0.6970
```



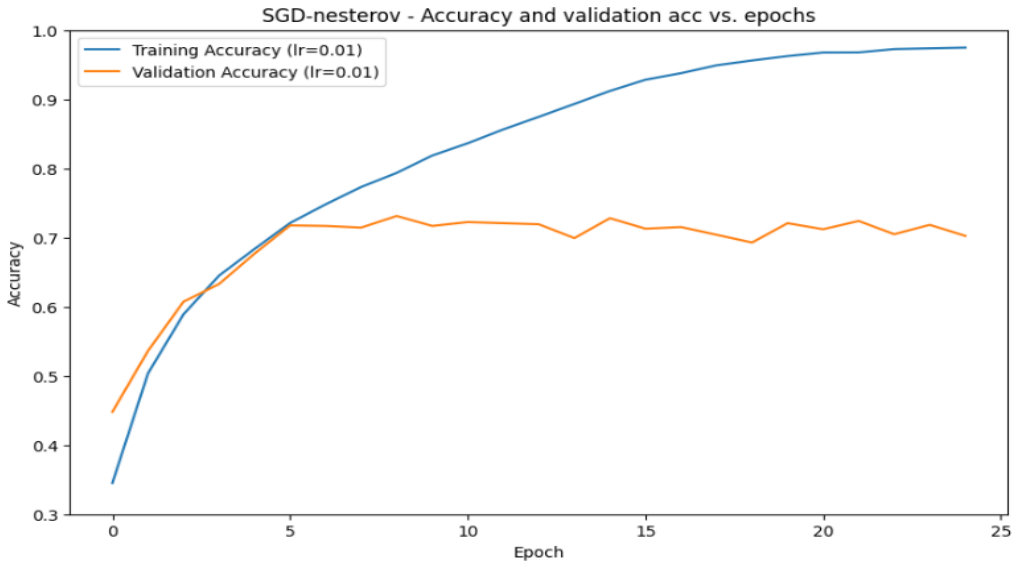


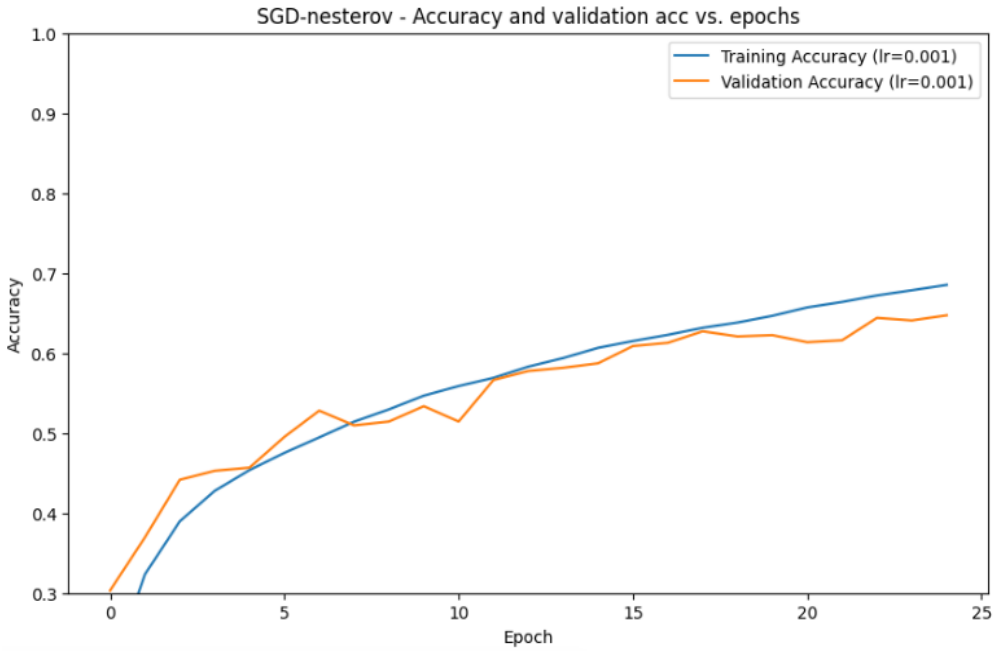
D.Output:

```
Test accuracy: 0.5101
```



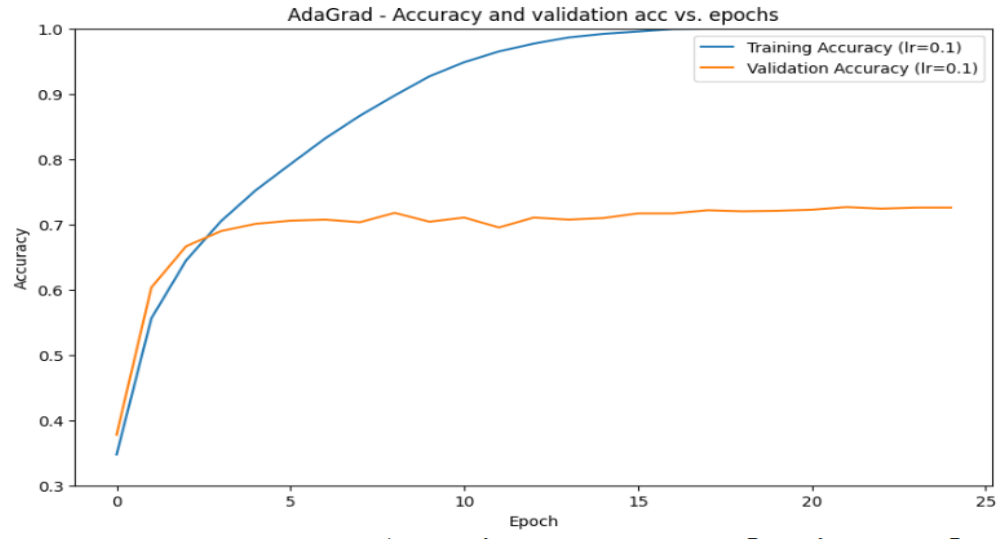

```
Test accuracy: 0.7058
```



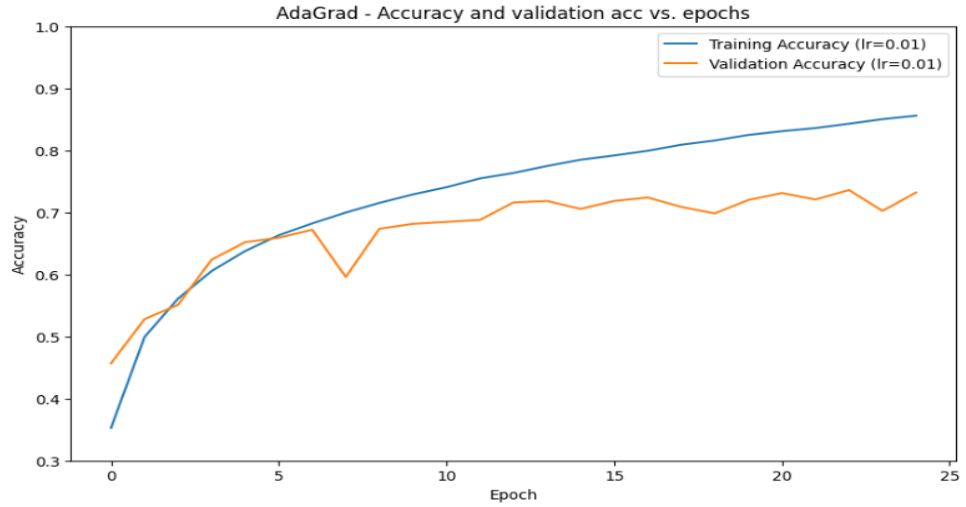


E.Output:

```
Test accuracy: 0.7103
```




```
Test accuracy: 0.7180
```




```
Test accuracy: 0.5447
```