DA_3

September 15, 2025

Name:	Tufan Kundu
Registration no:	24MDT0184
Course Name:	Deep Learning Lab
Course Code:	PMDS603P
Digital Assessment:	3

0.1 Question1. Use the MNIST dataset and do necessary pre-processing, and split the data into training, validation, and testing sets. Create a new ANN model with appropriate hidden layers and output layer neurons. Choose appropriate activation functions. Choose the error function appropriately and use SGD as the optimizer. Include early stopping technique in your model and run the model for 500 epochs and report the Performance.

0.1.1 Importing the necessary libraries

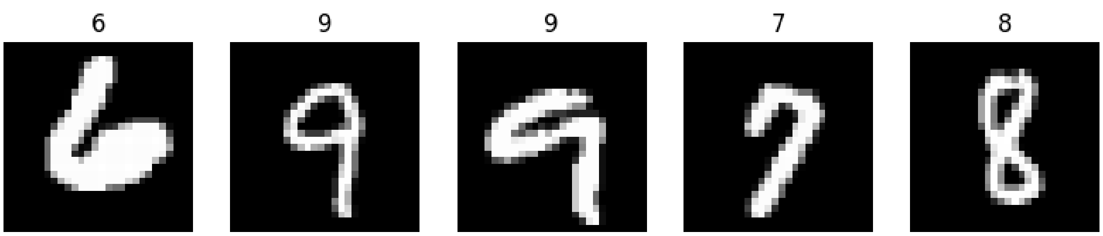
[38]:
```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense, Flatten, Dropout
from keras.optimizers import SGD
from keras.callbacks import EarlyStopping
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import numpy as np

import warnings
warnings.filterwarnings('ignore')
```

0.1.2 Loading the MNIST Dataset

[39]:
```
from keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = x_train/255.0
x_test = x_test/255.0
```

0.1.3 Displaying Random Images

[40]:
```
class_names = [str(i) for i in range(10)]
indices = np.random.choice(len(x_train), size = 5 , replace = False)

plt.figure(figsize = (2*5,3))
for i, idx in enumerate(indices):
    ax = plt.subplot(1,5,i+1)
    img = x_train[idx]
    plt.imshow(img, cmap = 'gray')
    label = class_names[y_train[idx]]
    plt.title(label)
    plt.axis('off')
plt.show()
```



[41]:
```
x_train, x_val, y_train, y_val = train_test_split(
    x_train, y_train,
    test_size=0.2, random_state=42, stratify=y_train
)

print(f"X_train shape: {x_train.shape}")
print(f"X_valid shape: {x_val.shape}")
print(f"X_test shape: {x_test.shape}")


print(f"y_train shape: {y_train.shape}")
print(f"y_valid shape: {y_val.shape}")
print(f"y_test shape: {y_test.shape}")
```

```
X_train shape: (48000, 28, 28)
X_valid shape: (12000, 28, 28)
X_test shape: (10000, 28, 28)
y_train shape: (48000,)
y_valid shape: (12000,)
y_test shape: (10000,)
```

0.1.4 Building the ANN Model

[42]:
```
model = Sequential()
model.add(Flatten(input_shape = (28,28)))
model.add(Dense(512, activation = 'relu'))
model.add(Dropout(0.5))
model.add(Dense(256, activation = 'relu'))
model.add(Dropout(0.3))
model.add(Dense(128, activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(10, activation = 'softmax'))
model.summary()
```

Model: "sequential_9"

Layer (type)	Output Shape	Param #
flatten_9 (Flatten)	(None, 784)	0
dense_36 (Dense)	(None, 512)	401,920
dropout_27 (Dropout)	(None, 512)	0
dense_37 (Dense)	(None, 256)	131,328
dropout_28 (Dropout)	(None, 256)	0
dense_38 (Dense)	(None, 128)	32,896
dropout_29 (Dropout)	(None, 128)	0
dense_39 (Dense)	(None, 10)	1,290

Total params: 567,434 (2.16 MB)

Trainable params: 567,434 (2.16 MB)

Non-trainable params: 0 (0.00 B)

[43]:
```
sgd = SGD(learning_rate= 0.01, momentum=0.9)
model.compile(optimizer = sgd, loss = 'sparse_categorical_crossentropy',␣
↪metrics = ['accuracy'])
```

[44]:
```
estop = EarlyStopping(monitor = 'val_loss', min_delta= 1e-4, patience= 5,␣
↪verbose = 1, restore_best_weights=True)
history = model.fit(x_train,y_train,batch_size=128, epochs = 500, verbose = 1,␣
↪validation_data=(x_val,y_val), callbacks=[estop])
```

```
Epoch 1/500
375/375              3s 6ms/step -
accuracy: 0.5860 - loss: 1.2496 - val_accuracy: 0.9233 - val_loss: 0.2554
Epoch 2/500
375/375              2s 6ms/step -
accuracy: 0.8955 - loss: 0.3440 - val_accuracy: 0.9447 - val_loss: 0.1780
Epoch 3/500
375/375              2s 6ms/step -
accuracy: 0.9253 - loss: 0.2481 - val_accuracy: 0.9555 - val_loss: 0.1487
Epoch 4/500
375/375              2s 6ms/step -
accuracy: 0.9389 - loss: 0.2023 - val_accuracy: 0.9625 - val_loss: 0.1249
Epoch 5/500
375/375              2s 6ms/step -
accuracy: 0.9488 - loss: 0.1700 - val_accuracy: 0.9670 - val_loss: 0.1131
Epoch 6/500
375/375              2s 6ms/step -
accuracy: 0.9540 - loss: 0.1538 - val_accuracy: 0.9707 - val_loss: 0.1025
Epoch 7/500
375/375              2s 6ms/step -
accuracy: 0.9582 - loss: 0.1383 - val_accuracy: 0.9699 - val_loss: 0.1021
Epoch 8/500
375/375              2s 6ms/step -
accuracy: 0.9599 - loss: 0.1319 - val_accuracy: 0.9734 - val_loss: 0.0913
Epoch 9/500
375/375              2s 6ms/step -
accuracy: 0.9651 - loss: 0.1150 - val_accuracy: 0.9741 - val_loss: 0.0900
Epoch 10/500
375/375              3s 7ms/step -
accuracy: 0.9662 - loss: 0.1120 - val_accuracy: 0.9757 - val_loss: 0.0849
Epoch 11/500
375/375              3s 7ms/step -
accuracy: 0.9700 - loss: 0.1007 - val_accuracy: 0.9768 - val_loss: 0.0815
Epoch 12/500
375/375              3s 7ms/step -
accuracy: 0.9709 - loss: 0.0953 - val_accuracy: 0.9755 - val_loss: 0.0851
Epoch 13/500
375/375              3s 7ms/step -
accuracy: 0.9729 - loss: 0.0859 - val_accuracy: 0.9783 - val_loss: 0.0781
Epoch 14/500
375/375              3s 7ms/step -
accuracy: 0.9745 - loss: 0.0814 - val_accuracy: 0.9792 - val_loss: 0.0751
Epoch 15/500
```

```
375/375              3s 7ms/step -
accuracy: 0.9742 - loss: 0.0817 - val_accuracy: 0.9789 - val_loss: 0.0751
Epoch 16/500
375/375              3s 8ms/step -
accuracy: 0.9764 - loss: 0.0755 - val_accuracy: 0.9801 - val_loss: 0.0745
Epoch 17/500
375/375              3s 8ms/step -
accuracy: 0.9776 - loss: 0.0748 - val_accuracy: 0.9791 - val_loss: 0.0740
Epoch 18/500
375/375              3s 9ms/step -
accuracy: 0.9775 - loss: 0.0723 - val_accuracy: 0.9803 - val_loss: 0.0742
Epoch 19/500
375/375              4s 10ms/step -
accuracy: 0.9783 - loss: 0.0672 - val_accuracy: 0.9798 - val_loss: 0.0748
Epoch 20/500
375/375              3s 7ms/step -
accuracy: 0.9804 - loss: 0.0633 - val_accuracy: 0.9803 - val_loss: 0.0740
Epoch 21/500
375/375              3s 7ms/step -
accuracy: 0.9804 - loss: 0.0623 - val_accuracy: 0.9810 - val_loss: 0.0722
Epoch 22/500
375/375              3s 7ms/step -
accuracy: 0.9829 - loss: 0.0543 - val_accuracy: 0.9814 - val_loss: 0.0695
Epoch 23/500
375/375              3s 7ms/step -
accuracy: 0.9826 - loss: 0.0557 - val_accuracy: 0.9811 - val_loss: 0.0713
Epoch 24/500
375/375              3s 8ms/step -
accuracy: 0.9836 - loss: 0.0539 - val_accuracy: 0.9803 - val_loss: 0.0698
Epoch 25/500
375/375              3s 8ms/step -
accuracy: 0.9837 - loss: 0.0511 - val_accuracy: 0.9810 - val_loss: 0.0721
Epoch 26/500
375/375              3s 7ms/step -
accuracy: 0.9860 - loss: 0.0452 - val_accuracy: 0.9816 - val_loss: 0.0695
Epoch 27/500
375/375              3s 7ms/step -
accuracy: 0.9838 - loss: 0.0498 - val_accuracy: 0.9817 - val_loss: 0.0736
Epoch 27: early stopping
Restoring model weights from the end of the best epoch: 22.
```

[45]:

```
loss, val_accuracy = model.evaluate(x_test,y_test)
```

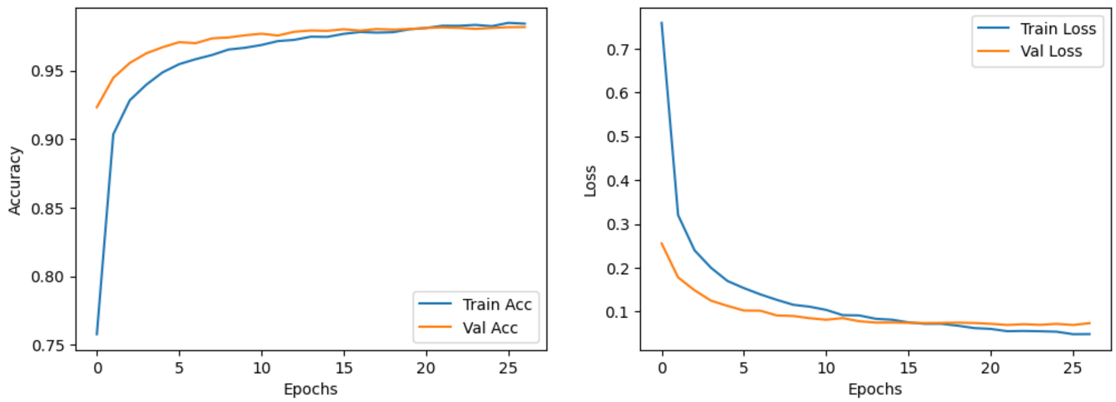
```
313/313              1s 2ms/step -
accuracy: 0.9762 - loss: 0.0817
```

[46]:

```
print(f"Test Accuracy with ANN model: {val_accuracy*100:.4f}%")
```

```
Test Accuracy with ANN model: 98.0700%
```

```
[47]: plt.figure(figsize=(12,4))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Train Acc')
plt.plot(history.history['val_accuracy'], label='Val Acc')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.subplot(1,2,2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.xlabel("Epochs")
plt.ylabel('Loss')
plt.legend()
plt.show()
```



0.1.5 Question 2: Now refit the model with three learning rate schedulers, linear, polynomialdecay and exponentialdecay and report the answers. The fitting should be done with early stopping on.

```
[48]: def build_model():
    model = Sequential()
    model.add(Flatten(input_shape = (28,28)))
    model.add(Dense(512, activation = 'relu'))
    model.add(Dropout(0.5))
    model.add(Dense(256, activation = 'relu'))
    model.add(Dropout(0.3))
    model.add(Dense(128, activation = 'relu'))
    model.add(Dropout(0.2))
    model.add(Dense(10, activation = 'softmax'))
    return model
```

[49]:
```
import keras.optimizers

initial_lr = 0.01

## Linear decay
linear_decay = keras.optimizers.schedules.PolynomialDecay(
    initial_learning_rate= initial_lr,
    decay_steps= 10000,
    end_learning_rate=0.0,
    power = 1.0
)

## Polynomial Decay
polynomial_decay = keras.optimizers.schedules.PolynomialDecay(
    initial_learning_rate=initial_lr,
    decay_steps=10000,
    end_learning_rate=0.0001,
    power = 2.0
)

## Exponential Decay
exponential_decay = keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate=initial_lr,
    decay_steps=1000,
    decay_rate=0.9
)

schedulers = {
    "Linear Decay": linear_decay,
    "Polynomial Decay": polynomial_decay,
    "Exponential Decay": exponential_decay
}
```

[50]:
```
results = {}
for name, schedule in schedulers.items():
    sgd = SGD(learning_rate=schedule, momentum=0.9)
    print(f"\nTraining with {name} learning rate schedular:\n")
    model = build_model()
    model.compile(optimizer=sgd, loss = 'sparse_categorical_crossentropy',␣
 ↪metrics = ['accuracy'])

    history = model.fit(x_train,y_train, validation_data=(x_val,y_val),␣
 ↪batch_size=128, epochs = 500, callbacks=[estop], verbose = 1)
    test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)
    print(f"{name} - Test Accuracy: {test_acc*100:.4f}%, Test Loss: {test_loss:.
 ↪4f}")
```

```
    # Store results
    results[name] = {
        "history": history.history,
        "test_acc": test_acc,
        "test_loss": test_loss
    }
```

```
Training with Linear Decay learning rate schedular:

Epoch 1/500
375/375              3s 7ms/step -
accuracy: 0.5811 - loss: 1.2293 - val_accuracy: 0.9234 - val_loss: 0.2504
Epoch 2/500
375/375              3s 7ms/step -
accuracy: 0.8992 - loss: 0.3363 - val_accuracy: 0.9457 - val_loss: 0.1758
Epoch 3/500
375/375              3s 7ms/step -
accuracy: 0.9253 - loss: 0.2468 - val_accuracy: 0.9551 - val_loss: 0.1467
Epoch 4/500
375/375              3s 7ms/step -
accuracy: 0.9383 - loss: 0.2063 - val_accuracy: 0.9619 - val_loss: 0.1266
Epoch 5/500
375/375              3s 7ms/step -
accuracy: 0.9471 - loss: 0.1791 - val_accuracy: 0.9657 - val_loss: 0.1171
Epoch 6/500
375/375              3s 8ms/step -
accuracy: 0.9522 - loss: 0.1595 - val_accuracy: 0.9703 - val_loss: 0.1044
Epoch 7/500
375/375              3s 7ms/step -
accuracy: 0.9584 - loss: 0.1340 - val_accuracy: 0.9722 - val_loss: 0.0980
Epoch 8/500
375/375              3s 7ms/step -
accuracy: 0.9620 - loss: 0.1286 - val_accuracy: 0.9721 - val_loss: 0.0954
Epoch 9/500
375/375              3s 7ms/step -
accuracy: 0.9664 - loss: 0.1109 - val_accuracy: 0.9739 - val_loss: 0.0913
Epoch 10/500
375/375              3s 7ms/step -
accuracy: 0.9656 - loss: 0.1142 - val_accuracy: 0.9747 - val_loss: 0.0883
Epoch 11/500
375/375              3s 7ms/step -
accuracy: 0.9675 - loss: 0.1061 - val_accuracy: 0.9768 - val_loss: 0.0846
Epoch 12/500
375/375              3s 7ms/step -
accuracy: 0.9700 - loss: 0.0993 - val_accuracy: 0.9785 - val_loss: 0.0829
Epoch 13/500
375/375              3s 7ms/step -
```

```
accuracy: 0.9713 - loss: 0.0942 - val_accuracy: 0.9784 - val_loss: 0.0807
Epoch 14/500
375/375              3s 7ms/step -
accuracy: 0.9720 - loss: 0.0895 - val_accuracy: 0.9769 - val_loss: 0.0820
Epoch 15/500
375/375              3s 7ms/step -
accuracy: 0.9735 - loss: 0.0886 - val_accuracy: 0.9777 - val_loss: 0.0799
Epoch 16/500
375/375              3s 7ms/step -
accuracy: 0.9751 - loss: 0.0807 - val_accuracy: 0.9794 - val_loss: 0.0768
Epoch 17/500
375/375              3s 7ms/step -
accuracy: 0.9750 - loss: 0.0797 - val_accuracy: 0.9798 - val_loss: 0.0761
Epoch 18/500
375/375              3s 7ms/step -
accuracy: 0.9762 - loss: 0.0726 - val_accuracy: 0.9792 - val_loss: 0.0759
Epoch 19/500
375/375              3s 7ms/step -
accuracy: 0.9777 - loss: 0.0767 - val_accuracy: 0.9801 - val_loss: 0.0761
Epoch 20/500
375/375              3s 7ms/step -
accuracy: 0.9784 - loss: 0.0722 - val_accuracy: 0.9803 - val_loss: 0.0756
Epoch 21/500
375/375              3s 7ms/step -
accuracy: 0.9776 - loss: 0.0738 - val_accuracy: 0.9804 - val_loss: 0.0754
Epoch 22/500
375/375              3s 7ms/step -
accuracy: 0.9798 - loss: 0.0660 - val_accuracy: 0.9808 - val_loss: 0.0745
Epoch 23/500
375/375              3s 7ms/step -
accuracy: 0.9810 - loss: 0.0631 - val_accuracy: 0.9809 - val_loss: 0.0748
Epoch 24/500
375/375              3s 7ms/step -
accuracy: 0.9788 - loss: 0.0666 - val_accuracy: 0.9812 - val_loss: 0.0740
Epoch 25/500
375/375              3s 7ms/step -
accuracy: 0.9815 - loss: 0.0599 - val_accuracy: 0.9813 - val_loss: 0.0740
Epoch 26/500
375/375              3s 7ms/step -
accuracy: 0.9813 - loss: 0.0630 - val_accuracy: 0.9814 - val_loss: 0.0739
Epoch 27/500
375/375              3s 7ms/step -
accuracy: 0.9814 - loss: 0.0601 - val_accuracy: 0.9816 - val_loss: 0.0738
Epoch 28/500
375/375              3s 7ms/step -
accuracy: 0.9801 - loss: 0.0611 - val_accuracy: 0.9816 - val_loss: 0.0738
Epoch 29/500
375/375              3s 7ms/step -
```

```
accuracy: 0.9809 - loss: 0.0612 - val_accuracy: 0.9816 - val_loss: 0.0738
Epoch 30/500
375/375            3s 7ms/step -
accuracy: 0.9804 - loss: 0.0612 - val_accuracy: 0.9816 - val_loss: 0.0738
Epoch 31/500
375/375            3s 7ms/step -
accuracy: 0.9805 - loss: 0.0636 - val_accuracy: 0.9816 - val_loss: 0.0738
Epoch 32/500
375/375            3s 7ms/step -
accuracy: 0.9819 - loss: 0.0603 - val_accuracy: 0.9816 - val_loss: 0.0738
Epoch 32: early stopping
Restoring model weights from the end of the best epoch: 27.
Linear Decay - Test Accuracy: 98.0200%, Test Loss: 0.0670

Training with Polynomial Decay learning rate schedular:

Epoch 1/500
375/375            3s 7ms/step -
accuracy: 0.5726 - loss: 1.2735 - val_accuracy: 0.9208 - val_loss: 0.2617
Epoch 2/500
375/375            3s 7ms/step -
accuracy: 0.8925 - loss: 0.3502 - val_accuracy: 0.9428 - val_loss: 0.1863
Epoch 3/500
375/375            3s 8ms/step -
accuracy: 0.9234 - loss: 0.2588 - val_accuracy: 0.9534 - val_loss: 0.1523
Epoch 4/500
375/375            3s 7ms/step -
accuracy: 0.9338 - loss: 0.2138 - val_accuracy: 0.9595 - val_loss: 0.1323
Epoch 5/500
375/375            3s 7ms/step -
accuracy: 0.9455 - loss: 0.1875 - val_accuracy: 0.9646 - val_loss: 0.1191
Epoch 6/500
375/375            3s 7ms/step -
accuracy: 0.9502 - loss: 0.1661 - val_accuracy: 0.9653 - val_loss: 0.1143
Epoch 7/500
375/375            3s 7ms/step -
accuracy: 0.9546 - loss: 0.1517 - val_accuracy: 0.9692 - val_loss: 0.1052
Epoch 8/500
375/375            3s 7ms/step -
accuracy: 0.9570 - loss: 0.1411 - val_accuracy: 0.9704 - val_loss: 0.1003
Epoch 9/500
375/375            3s 7ms/step -
accuracy: 0.9620 - loss: 0.1267 - val_accuracy: 0.9712 - val_loss: 0.0983
Epoch 10/500
375/375            3s 7ms/step -
accuracy: 0.9633 - loss: 0.1232 - val_accuracy: 0.9722 - val_loss: 0.0958
Epoch 11/500
375/375            3s 7ms/step -
```

```
accuracy: 0.9638 - loss: 0.1166 - val_accuracy: 0.9739 - val_loss: 0.0921
Epoch 12/500
375/375            3s 7ms/step -
accuracy: 0.9653 - loss: 0.1145 - val_accuracy: 0.9737 - val_loss: 0.0904
Epoch 13/500
375/375            3s 7ms/step -
accuracy: 0.9685 - loss: 0.1056 - val_accuracy: 0.9752 - val_loss: 0.0886
Epoch 14/500
375/375            3s 7ms/step -
accuracy: 0.9686 - loss: 0.1071 - val_accuracy: 0.9756 - val_loss: 0.0864
Epoch 15/500
375/375            3s 7ms/step -
accuracy: 0.9685 - loss: 0.1014 - val_accuracy: 0.9753 - val_loss: 0.0870
Epoch 16/500
375/375            3s 7ms/step -
accuracy: 0.9706 - loss: 0.0972 - val_accuracy: 0.9761 - val_loss: 0.0854
Epoch 17/500
375/375            3s 7ms/step -
accuracy: 0.9724 - loss: 0.0963 - val_accuracy: 0.9758 - val_loss: 0.0850
Epoch 18/500
375/375            3s 7ms/step -
accuracy: 0.9713 - loss: 0.0943 - val_accuracy: 0.9761 - val_loss: 0.0845
Epoch 19/500
375/375            3s 7ms/step -
accuracy: 0.9704 - loss: 0.0950 - val_accuracy: 0.9759 - val_loss: 0.0842
Epoch 20/500
375/375            3s 7ms/step -
accuracy: 0.9728 - loss: 0.0908 - val_accuracy: 0.9762 - val_loss: 0.0837
Epoch 21/500
375/375            3s 7ms/step -
accuracy: 0.9733 - loss: 0.0882 - val_accuracy: 0.9766 - val_loss: 0.0831
Epoch 22/500
375/375            3s 7ms/step -
accuracy: 0.9735 - loss: 0.0893 - val_accuracy: 0.9766 - val_loss: 0.0831
Epoch 23/500
375/375            3s 7ms/step -
accuracy: 0.9741 - loss: 0.0860 - val_accuracy: 0.9769 - val_loss: 0.0831
Epoch 24/500
375/375            3s 7ms/step -
accuracy: 0.9727 - loss: 0.0910 - val_accuracy: 0.9770 - val_loss: 0.0828
Epoch 25/500
375/375            3s 8ms/step -
accuracy: 0.9752 - loss: 0.0863 - val_accuracy: 0.9770 - val_loss: 0.0827
Epoch 26/500
375/375            3s 7ms/step -
accuracy: 0.9730 - loss: 0.0864 - val_accuracy: 0.9770 - val_loss: 0.0826
Epoch 27/500
375/375            3s 7ms/step -
```

```
accuracy: 0.9725 - loss: 0.0869 - val_accuracy: 0.9771 - val_loss: 0.0826
Epoch 28/500
375/375              3s 7ms/step -
accuracy: 0.9727 - loss: 0.0907 - val_accuracy: 0.9772 - val_loss: 0.0826
Epoch 29/500
375/375              3s 7ms/step -
accuracy: 0.9727 - loss: 0.0871 - val_accuracy: 0.9771 - val_loss: 0.0825
Epoch 30/500
375/375              3s 7ms/step -
accuracy: 0.9746 - loss: 0.0814 - val_accuracy: 0.9771 - val_loss: 0.0825
Epoch 31/500
375/375              3s 7ms/step -
accuracy: 0.9735 - loss: 0.0866 - val_accuracy: 0.9770 - val_loss: 0.0824
Epoch 32/500
375/375              3s 7ms/step -
accuracy: 0.9737 - loss: 0.0844 - val_accuracy: 0.9770 - val_loss: 0.0824
Epoch 33/500
375/375              3s 7ms/step -
accuracy: 0.9742 - loss: 0.0840 - val_accuracy: 0.9770 - val_loss: 0.0824
Epoch 34/500
375/375              3s 7ms/step -
accuracy: 0.9759 - loss: 0.0845 - val_accuracy: 0.9769 - val_loss: 0.0824
Epoch 35/500
375/375              3s 7ms/step -
accuracy: 0.9756 - loss: 0.0806 - val_accuracy: 0.9770 - val_loss: 0.0823
Epoch 36/500
375/375              3s 7ms/step -
accuracy: 0.9737 - loss: 0.0869 - val_accuracy: 0.9770 - val_loss: 0.0822
Epoch 37/500
375/375              3s 8ms/step -
accuracy: 0.9749 - loss: 0.0829 - val_accuracy: 0.9770 - val_loss: 0.0823
Epoch 38/500
375/375              3s 7ms/step -
accuracy: 0.9707 - loss: 0.0904 - val_accuracy: 0.9769 - val_loss: 0.0822
Epoch 39/500
375/375              3s 8ms/step -
accuracy: 0.9737 - loss: 0.0853 - val_accuracy: 0.9769 - val_loss: 0.0821
Epoch 40/500
375/375              3s 7ms/step -
accuracy: 0.9722 - loss: 0.0859 - val_accuracy: 0.9769 - val_loss: 0.0821
Epoch 41/500
375/375              3s 8ms/step -
accuracy: 0.9739 - loss: 0.0842 - val_accuracy: 0.9770 - val_loss: 0.0820
Epoch 42/500
375/375              3s 8ms/step -
accuracy: 0.9740 - loss: 0.0854 - val_accuracy: 0.9772 - val_loss: 0.0819
Epoch 43/500
375/375              3s 8ms/step -
```

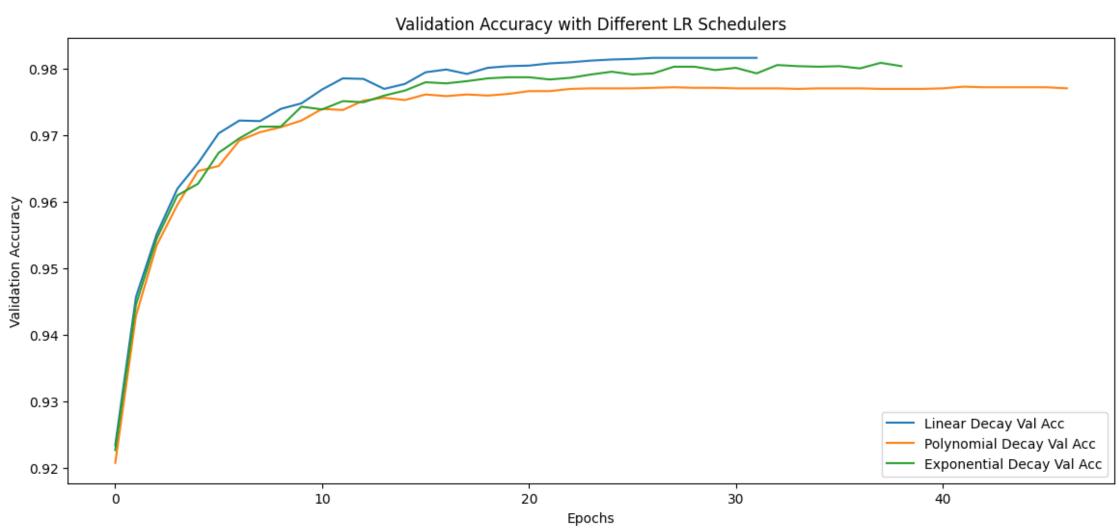
```
accuracy: 0.9736 - loss: 0.0861 - val_accuracy: 0.9772 - val_loss: 0.0820
Epoch 44/500
375/375             3s 7ms/step -
accuracy: 0.9732 - loss: 0.0876 - val_accuracy: 0.9772 - val_loss: 0.0819
Epoch 45/500
375/375             3s 8ms/step -
accuracy: 0.9724 - loss: 0.0887 - val_accuracy: 0.9772 - val_loss: 0.0820
Epoch 46/500
375/375             3s 7ms/step -
accuracy: 0.9735 - loss: 0.0855 - val_accuracy: 0.9772 - val_loss: 0.0821
Epoch 47/500
375/375             3s 8ms/step -
accuracy: 0.9734 - loss: 0.0835 - val_accuracy: 0.9770 - val_loss: 0.0820
Epoch 47: early stopping
Restoring model weights from the end of the best epoch: 42.
Polynomial Decay - Test Accuracy: 97.6400%, Test Loss: 0.0770

Training with Exponential Decay learning rate schedular:

Epoch 1/500
375/375             4s 8ms/step -
accuracy: 0.5878 - loss: 1.2296 - val_accuracy: 0.9227 - val_loss: 0.2570
Epoch 2/500
375/375             3s 8ms/step -
accuracy: 0.8972 - loss: 0.3423 - val_accuracy: 0.9445 - val_loss: 0.1753
Epoch 3/500
375/375             3s 7ms/step -
accuracy: 0.9231 - loss: 0.2518 - val_accuracy: 0.9544 - val_loss: 0.1486
Epoch 4/500
375/375             3s 8ms/step -
accuracy: 0.9385 - loss: 0.2098 - val_accuracy: 0.9609 - val_loss: 0.1307
Epoch 5/500
375/375             3s 7ms/step -
accuracy: 0.9468 - loss: 0.1741 - val_accuracy: 0.9627 - val_loss: 0.1203
Epoch 6/500
375/375             3s 8ms/step -
accuracy: 0.9504 - loss: 0.1632 - val_accuracy: 0.9673 - val_loss: 0.1070
Epoch 7/500
375/375             3s 8ms/step -
accuracy: 0.9569 - loss: 0.1420 - val_accuracy: 0.9695 - val_loss: 0.1030
Epoch 8/500
375/375             3s 8ms/step -
accuracy: 0.9593 - loss: 0.1339 - val_accuracy: 0.9712 - val_loss: 0.0987
Epoch 9/500
375/375             3s 8ms/step -
accuracy: 0.9660 - loss: 0.1180 - val_accuracy: 0.9712 - val_loss: 0.0958
Epoch 10/500
375/375             3s 9ms/step -
```

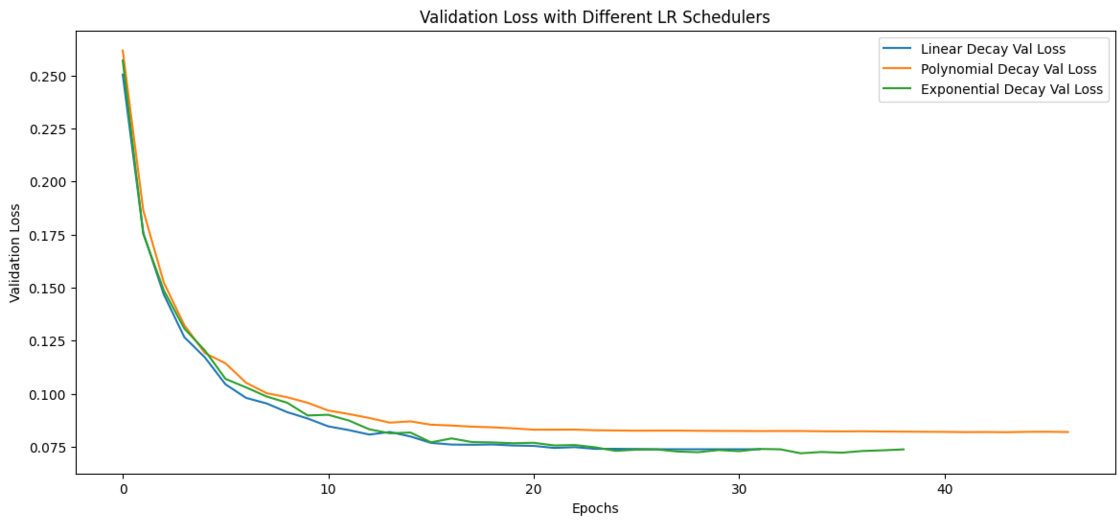
```
accuracy: 0.9681 - loss: 0.1067 - val_accuracy: 0.9743 - val_loss: 0.0898
Epoch 11/500
375/375            3s 8ms/step -
accuracy: 0.9672 - loss: 0.1050 - val_accuracy: 0.9738 - val_loss: 0.0901
Epoch 12/500
375/375            3s 8ms/step -
accuracy: 0.9708 - loss: 0.0956 - val_accuracy: 0.9751 - val_loss: 0.0874
Epoch 13/500
375/375            3s 8ms/step -
accuracy: 0.9703 - loss: 0.0973 - val_accuracy: 0.9749 - val_loss: 0.0832
Epoch 14/500
375/375            3s 8ms/step -
accuracy: 0.9701 - loss: 0.0938 - val_accuracy: 0.9759 - val_loss: 0.0814
Epoch 15/500
375/375            3s 7ms/step -
accuracy: 0.9720 - loss: 0.0888 - val_accuracy: 0.9767 - val_loss: 0.0817
Epoch 16/500
375/375            3s 8ms/step -
accuracy: 0.9748 - loss: 0.0820 - val_accuracy: 0.9779 - val_loss: 0.0771
Epoch 17/500
375/375            3s 8ms/step -
accuracy: 0.9759 - loss: 0.0791 - val_accuracy: 0.9778 - val_loss: 0.0789
Epoch 18/500
375/375            3s 8ms/step -
accuracy: 0.9766 - loss: 0.0781 - val_accuracy: 0.9781 - val_loss: 0.0772
Epoch 19/500
375/375            3s 8ms/step -
accuracy: 0.9792 - loss: 0.0681 - val_accuracy: 0.9785 - val_loss: 0.0770
Epoch 20/500
375/375            3s 8ms/step -
accuracy: 0.9777 - loss: 0.0699 - val_accuracy: 0.9787 - val_loss: 0.0767
Epoch 21/500
375/375            3s 8ms/step -
accuracy: 0.9786 - loss: 0.0696 - val_accuracy: 0.9787 - val_loss: 0.0769
Epoch 22/500
375/375            3s 8ms/step -
accuracy: 0.9794 - loss: 0.0651 - val_accuracy: 0.9783 - val_loss: 0.0756
Epoch 23/500
375/375            3s 8ms/step -
accuracy: 0.9805 - loss: 0.0642 - val_accuracy: 0.9786 - val_loss: 0.0758
Epoch 24/500
375/375            3s 8ms/step -
accuracy: 0.9817 - loss: 0.0606 - val_accuracy: 0.9791 - val_loss: 0.0747
Epoch 25/500
375/375            3s 9ms/step -
accuracy: 0.9815 - loss: 0.0600 - val_accuracy: 0.9795 - val_loss: 0.0731
Epoch 26/500
375/375            3s 8ms/step -
```

```
accuracy: 0.9815 - loss: 0.0587 - val_accuracy: 0.9791 - val_loss: 0.0737
Epoch 27/500
375/375            3s 9ms/step -
accuracy: 0.9822 - loss: 0.0547 - val_accuracy: 0.9793 - val_loss: 0.0737
Epoch 28/500
375/375            3s 8ms/step -
accuracy: 0.9820 - loss: 0.0570 - val_accuracy: 0.9803 - val_loss: 0.0728
Epoch 29/500
375/375            3s 8ms/step -
accuracy: 0.9832 - loss: 0.0537 - val_accuracy: 0.9803 - val_loss: 0.0724
Epoch 30/500
375/375            3s 8ms/step -
accuracy: 0.9841 - loss: 0.0523 - val_accuracy: 0.9797 - val_loss: 0.0735
Epoch 31/500
375/375            3s 8ms/step -
accuracy: 0.9837 - loss: 0.0515 - val_accuracy: 0.9801 - val_loss: 0.0729
Epoch 32/500
375/375            3s 8ms/step -
accuracy: 0.9838 - loss: 0.0526 - val_accuracy: 0.9793 - val_loss: 0.0740
Epoch 33/500
375/375            3s 8ms/step -
accuracy: 0.9852 - loss: 0.0485 - val_accuracy: 0.9805 - val_loss: 0.0738
Epoch 34/500
375/375            3s 8ms/step -
accuracy: 0.9847 - loss: 0.0504 - val_accuracy: 0.9803 - val_loss: 0.0719
Epoch 35/500
375/375            3s 9ms/step -
accuracy: 0.9847 - loss: 0.0465 - val_accuracy: 0.9803 - val_loss: 0.0726
Epoch 36/500
375/375            3s 8ms/step -
accuracy: 0.9852 - loss: 0.0476 - val_accuracy: 0.9803 - val_loss: 0.0722
Epoch 37/500
375/375            3s 8ms/step -
accuracy: 0.9862 - loss: 0.0455 - val_accuracy: 0.9800 - val_loss: 0.0730
Epoch 38/500
375/375            3s 8ms/step -
accuracy: 0.9854 - loss: 0.0464 - val_accuracy: 0.9808 - val_loss: 0.0733
Epoch 39/500
375/375            3s 8ms/step -
accuracy: 0.9864 - loss: 0.0421 - val_accuracy: 0.9803 - val_loss: 0.0738
Epoch 39: early stopping
Restoring model weights from the end of the best epoch: 34.
Exponential Decay - Test Accuracy: 98.1600%, Test Loss: 0.0633
```

0.1.6 Plotting loss and accuracy curves

[51]:
```
plt.figure(figsize=(14,6))

# Accuracy comparison
for name, res in results.items():
    plt.plot(res["history"]["val_accuracy"], label=f"{name} Val Acc")

plt.xlabel("Epochs")
plt.ylabel("Validation Accuracy")
plt.title("Validation Accuracy with Different LR Schedulers")
plt.legend()
plt.show()

plt.figure(figsize=(14,6))

# Loss comparison
for name, res in results.items():
    plt.plot(res["history"]["val_loss"], label=f"{name} Val Loss")

plt.xlabel("Epochs")
plt.ylabel("Validation Loss")
plt.title("Validation Loss with Different LR Schedulers")
plt.legend()
plt.show()
```





[52]:
```
for name, res in results.items():
    print(f"\n{name}:")
    print(f"  Final Test Accuracy = {res['test_acc']*100:.4f}%")
    print(f"  Final Test Loss     = {res['test_loss']:.4f}")
```

```
Linear Decay:
  Final Test Accuracy = 98.0200%
  Final Test Loss     = 0.0670

Polynomial Decay:
  Final Test Accuracy = 97.6400%
  Final Test Loss     = 0.0770

Exponential Decay:
  Final Test Accuracy = 98.1600%
  Final Test Loss     = 0.0633
```

0.2 Question 3: Optimizer Comparison

0.2.1 Report the best optimizer that would result in the best performance for the above models. Try at least three to four optimizers (e.g., SGD with momentum, RMSprop, Adam). Train the same model architecture with each optimizer. Compare their performances. Report which optimizer gives the best results

[53]:
```
def build_model():
    model = Sequential()
    model.add(Flatten(input_shape=(28,28)))
    model.add(Dense(512, activation='relu'))
    model.add(Dropout(0.5))
```

```
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.3))
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(10, activation='softmax'))
    return model
```

[54]:
```
from keras.optimizers import SGD, Adam, RMSprop,Adamax
optimizers = {
    "SGD": SGD(learning_rate=0.01, momentum=0.9),
    "RMSprop": RMSprop(learning_rate=0.001),
    "Adam": Adam(learning_rate=0.001),
    "Adamax": Adamax(learning_rate=0.002)
}

estop = EarlyStopping(monitor = 'val_loss', min_delta= 1e-4, patience= 5,␣
 ↪verbose = 1, restore_best_weights=True)
```

[55]:
```
results = {}

for name, optimizer in optimizers.items():
    print(f"\nTraining with {name} optimizer:\n")

    model = build_model()
    model.compile(optimizer=optimizer,
                  loss = 'sparse_categorical_crossentropy',
                  metrics = ['accuracy'])
    history = model.fit(
        x_train, y_train,
        validation_data=(x_val, y_val),
        batch_size=128,
        epochs=500,
        callbacks=[estop],
        verbose=1
    )
    # Evaluating on test set
    test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)
    print(f"{name} - Test Accuracy: {test_acc*100:.4f}%, Test Loss: {test_loss:.
 ↪4f}")

    results[name] = {
        "history": history.history,
        "test_acc": test_acc,
        "test_loss": test_loss
    }
```

```
Training with SGD optimizer:
```

```
Epoch 1/500
375/375            3s 7ms/step -
accuracy: 0.5820 - loss: 1.2470 - val_accuracy: 0.9237 - val_loss: 0.2473
Epoch 2/500
375/375            2s 7ms/step -
accuracy: 0.8986 - loss: 0.3402 - val_accuracy: 0.9463 - val_loss: 0.1743
Epoch 3/500
375/375            3s 7ms/step -
accuracy: 0.9257 - loss: 0.2489 - val_accuracy: 0.9560 - val_loss: 0.1439
Epoch 4/500
375/375            3s 7ms/step -
accuracy: 0.9385 - loss: 0.2056 - val_accuracy: 0.9617 - val_loss: 0.1243
Epoch 5/500
375/375            2s 6ms/step -
accuracy: 0.9465 - loss: 0.1781 - val_accuracy: 0.9647 - val_loss: 0.1170
Epoch 6/500
375/375            3s 7ms/step -
accuracy: 0.9536 - loss: 0.1527 - val_accuracy: 0.9693 - val_loss: 0.1026
Epoch 7/500
375/375            3s 7ms/step -
accuracy: 0.9600 - loss: 0.1366 - val_accuracy: 0.9712 - val_loss: 0.1001
Epoch 8/500
375/375            3s 7ms/step -
accuracy: 0.9620 - loss: 0.1259 - val_accuracy: 0.9736 - val_loss: 0.0910
Epoch 9/500
375/375            2s 7ms/step -
accuracy: 0.9637 - loss: 0.1175 - val_accuracy: 0.9737 - val_loss: 0.0884
Epoch 10/500
375/375            3s 7ms/step -
accuracy: 0.9648 - loss: 0.1152 - val_accuracy: 0.9756 - val_loss: 0.0857
Epoch 11/500
375/375            3s 8ms/step -
accuracy: 0.9697 - loss: 0.0987 - val_accuracy: 0.9758 - val_loss: 0.0806
Epoch 12/500
375/375            3s 7ms/step -
accuracy: 0.9706 - loss: 0.0984 - val_accuracy: 0.9755 - val_loss: 0.0858
Epoch 13/500
375/375            3s 7ms/step -
accuracy: 0.9732 - loss: 0.0891 - val_accuracy: 0.9777 - val_loss: 0.0785
Epoch 14/500
375/375            3s 7ms/step -
accuracy: 0.9754 - loss: 0.0826 - val_accuracy: 0.9787 - val_loss: 0.0788
Epoch 15/500
375/375            3s 7ms/step -
accuracy: 0.9752 - loss: 0.0798 - val_accuracy: 0.9778 - val_loss: 0.0791
Epoch 16/500
375/375            3s 7ms/step -
```

```
accuracy: 0.9755 - loss: 0.0787 - val_accuracy: 0.9784 - val_loss: 0.0756
Epoch 17/500
375/375            3s 8ms/step -
accuracy: 0.9766 - loss: 0.0728 - val_accuracy: 0.9799 - val_loss: 0.0755
Epoch 18/500
375/375            3s 7ms/step -
accuracy: 0.9793 - loss: 0.0698 - val_accuracy: 0.9797 - val_loss: 0.0717
Epoch 19/500
375/375            3s 8ms/step -
accuracy: 0.9793 - loss: 0.0666 - val_accuracy: 0.9791 - val_loss: 0.0771
Epoch 20/500
375/375            3s 7ms/step -
accuracy: 0.9798 - loss: 0.0632 - val_accuracy: 0.9794 - val_loss: 0.0724
Epoch 21/500
375/375            3s 7ms/step -
accuracy: 0.9812 - loss: 0.0603 - val_accuracy: 0.9804 - val_loss: 0.0726
Epoch 22/500
375/375            2s 7ms/step -
accuracy: 0.9800 - loss: 0.0613 - val_accuracy: 0.9806 - val_loss: 0.0715
Epoch 23/500
375/375            3s 7ms/step -
accuracy: 0.9819 - loss: 0.0580 - val_accuracy: 0.9797 - val_loss: 0.0739
Epoch 24/500
375/375            3s 8ms/step -
accuracy: 0.9832 - loss: 0.0528 - val_accuracy: 0.9805 - val_loss: 0.0760
Epoch 25/500
375/375            3s 8ms/step -
accuracy: 0.9833 - loss: 0.0527 - val_accuracy: 0.9798 - val_loss: 0.0737
Epoch 26/500
375/375            3s 7ms/step -
accuracy: 0.9846 - loss: 0.0495 - val_accuracy: 0.9799 - val_loss: 0.0753
Epoch 27/500
375/375            3s 7ms/step -
accuracy: 0.9843 - loss: 0.0496 - val_accuracy: 0.9807 - val_loss: 0.0721
Epoch 27: early stopping
Restoring model weights from the end of the best epoch: 22.
SGD - Test Accuracy: 98.1900%, Test Loss: 0.0638

Training with RMSprop optimizer:

Epoch 1/500
375/375            4s 9ms/step -
accuracy: 0.7769 - loss: 0.6982 - val_accuracy: 0.9516 - val_loss: 0.1627
Epoch 2/500
375/375            3s 9ms/step -
accuracy: 0.9400 - loss: 0.2034 - val_accuracy: 0.9666 - val_loss: 0.1122
Epoch 3/500
375/375            3s 8ms/step -
```

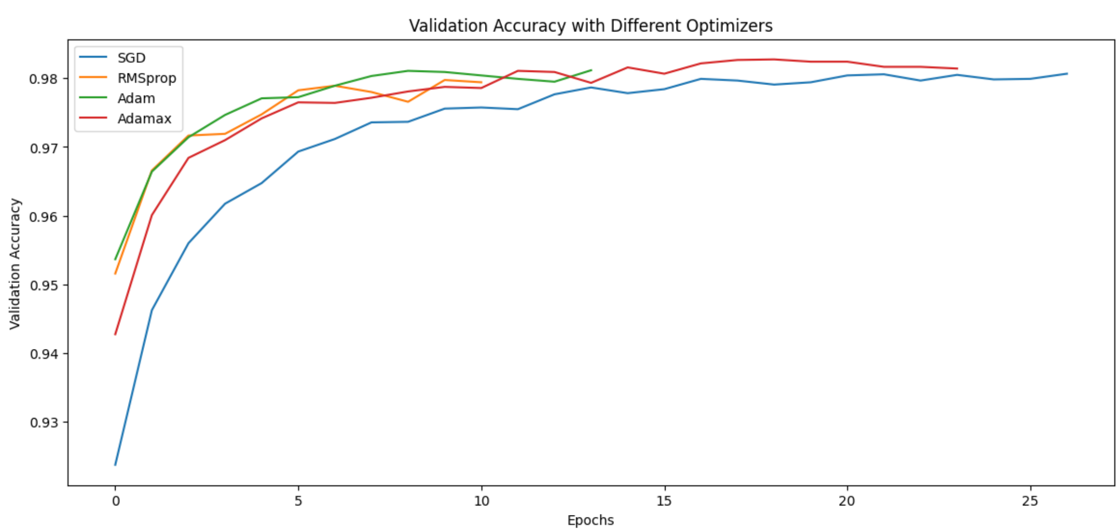
```
accuracy: 0.9534 - loss: 0.1557 - val_accuracy: 0.9717 - val_loss: 0.1035
Epoch 4/500
375/375            3s 8ms/step -
accuracy: 0.9612 - loss: 0.1275 - val_accuracy: 0.9719 - val_loss: 0.1004
Epoch 5/500
375/375            3s 8ms/step -
accuracy: 0.9663 - loss: 0.1158 - val_accuracy: 0.9747 - val_loss: 0.0919
Epoch 6/500
375/375            3s 8ms/step -
accuracy: 0.9707 - loss: 0.1004 - val_accuracy: 0.9783 - val_loss: 0.0824
Epoch 7/500
375/375            3s 9ms/step -
accuracy: 0.9714 - loss: 0.0934 - val_accuracy: 0.9789 - val_loss: 0.0837
Epoch 8/500
375/375            3s 8ms/step -
accuracy: 0.9742 - loss: 0.0911 - val_accuracy: 0.9780 - val_loss: 0.0829
Epoch 9/500
375/375            3s 8ms/step -
accuracy: 0.9773 - loss: 0.0777 - val_accuracy: 0.9766 - val_loss: 0.0935
Epoch 10/500
375/375            3s 8ms/step -
accuracy: 0.9766 - loss: 0.0793 - val_accuracy: 0.9797 - val_loss: 0.0841
Epoch 11/500
375/375            3s 8ms/step -
accuracy: 0.9782 - loss: 0.0741 - val_accuracy: 0.9794 - val_loss: 0.0841
Epoch 11: early stopping
Restoring model weights from the end of the best epoch: 6.
RMSprop - Test Accuracy: 97.8600%, Test Loss: 0.0766

Training with Adam optimizer:

Epoch 1/500
375/375            4s 9ms/step -
accuracy: 0.7487 - loss: 0.7641 - val_accuracy: 0.9537 - val_loss: 0.1502
Epoch 2/500
375/375            3s 9ms/step -
accuracy: 0.9392 - loss: 0.2043 - val_accuracy: 0.9664 - val_loss: 0.1108
Epoch 3/500
375/375            3s 9ms/step -
accuracy: 0.9536 - loss: 0.1570 - val_accuracy: 0.9714 - val_loss: 0.1003
Epoch 4/500
375/375            3s 8ms/step -
accuracy: 0.9619 - loss: 0.1261 - val_accuracy: 0.9747 - val_loss: 0.0907
Epoch 5/500
375/375            3s 9ms/step -
accuracy: 0.9674 - loss: 0.1077 - val_accuracy: 0.9771 - val_loss: 0.0819
Epoch 6/500
375/375            3s 8ms/step -
```

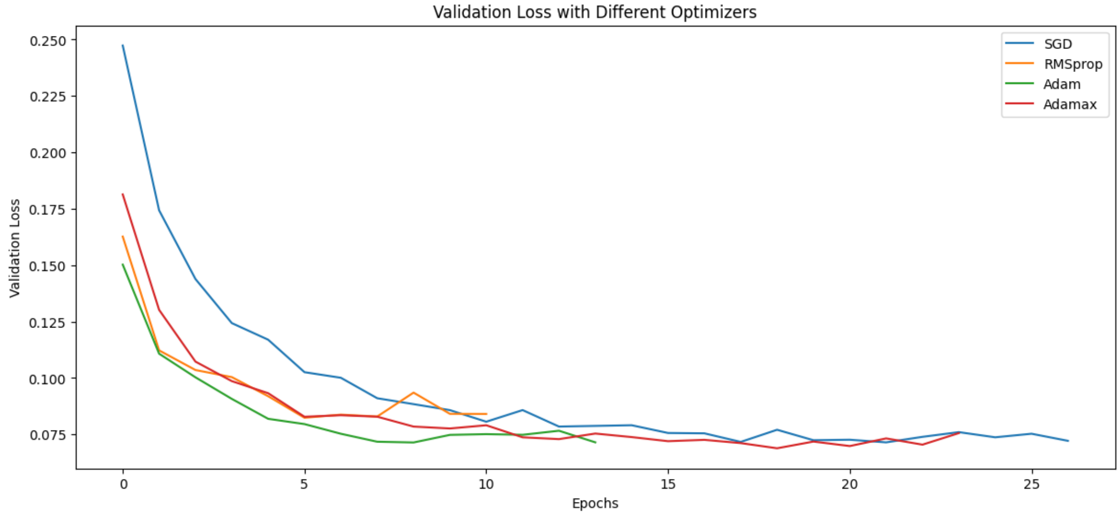
```
accuracy: 0.9695 - loss: 0.1004 - val_accuracy: 0.9772 - val_loss: 0.0796
Epoch 7/500
375/375            3s 9ms/step -
accuracy: 0.9722 - loss: 0.0909 - val_accuracy: 0.9789 - val_loss: 0.0753
Epoch 8/500
375/375            3s 8ms/step -
accuracy: 0.9748 - loss: 0.0800 - val_accuracy: 0.9803 - val_loss: 0.0717
Epoch 9/500
375/375            3s 8ms/step -
accuracy: 0.9760 - loss: 0.0789 - val_accuracy: 0.9811 - val_loss: 0.0714
Epoch 10/500
375/375            3s 9ms/step -
accuracy: 0.9781 - loss: 0.0735 - val_accuracy: 0.9809 - val_loss: 0.0748
Epoch 11/500
375/375            4s 10ms/step -
accuracy: 0.9796 - loss: 0.0658 - val_accuracy: 0.9804 - val_loss: 0.0751
Epoch 12/500
375/375            3s 9ms/step -
accuracy: 0.9809 - loss: 0.0637 - val_accuracy: 0.9799 - val_loss: 0.0748
Epoch 13/500
375/375            3s 8ms/step -
accuracy: 0.9815 - loss: 0.0609 - val_accuracy: 0.9795 - val_loss: 0.0766
Epoch 14/500
375/375            3s 8ms/step -
accuracy: 0.9808 - loss: 0.0585 - val_accuracy: 0.9812 - val_loss: 0.0715
Epoch 14: early stopping
Restoring model weights from the end of the best epoch: 9.
Adam - Test Accuracy: 97.9700%, Test Loss: 0.0671

Training with Adamax optimizer:

Epoch 1/500
375/375            5s 10ms/step -
accuracy: 0.7543 - loss: 0.7553 - val_accuracy: 0.9427 - val_loss: 0.1813
Epoch 2/500
375/375            3s 9ms/step -
accuracy: 0.9311 - loss: 0.2269 - val_accuracy: 0.9601 - val_loss: 0.1302
Epoch 3/500
375/375            3s 9ms/step -
accuracy: 0.9504 - loss: 0.1704 - val_accuracy: 0.9684 - val_loss: 0.1073
Epoch 4/500
375/375            3s 9ms/step -
accuracy: 0.9587 - loss: 0.1397 - val_accuracy: 0.9710 - val_loss: 0.0986
Epoch 5/500
375/375            3s 9ms/step -
accuracy: 0.9640 - loss: 0.1200 - val_accuracy: 0.9742 - val_loss: 0.0933
Epoch 6/500
375/375            4s 9ms/step -
```

```
accuracy: 0.9672 - loss: 0.1025 - val_accuracy: 0.9765 - val_loss: 0.0828
Epoch 7/500
375/375              3s 9ms/step -
accuracy: 0.9707 - loss: 0.0938 - val_accuracy: 0.9764 - val_loss: 0.0835
Epoch 8/500
375/375              3s 9ms/step -
accuracy: 0.9736 - loss: 0.0821 - val_accuracy: 0.9772 - val_loss: 0.0828
Epoch 9/500
375/375              3s 9ms/step -
accuracy: 0.9750 - loss: 0.0798 - val_accuracy: 0.9781 - val_loss: 0.0785
Epoch 10/500
375/375              3s 9ms/step -
accuracy: 0.9767 - loss: 0.0736 - val_accuracy: 0.9787 - val_loss: 0.0776
Epoch 11/500
375/375              3s 9ms/step -
accuracy: 0.9794 - loss: 0.0674 - val_accuracy: 0.9786 - val_loss: 0.0790
Epoch 12/500
375/375              3s 9ms/step -
accuracy: 0.9795 - loss: 0.0653 - val_accuracy: 0.9811 - val_loss: 0.0737
Epoch 13/500
375/375              3s 9ms/step -
accuracy: 0.9812 - loss: 0.0587 - val_accuracy: 0.9809 - val_loss: 0.0729
Epoch 14/500
375/375              3s 9ms/step -
accuracy: 0.9840 - loss: 0.0534 - val_accuracy: 0.9793 - val_loss: 0.0754
Epoch 15/500
375/375              4s 9ms/step -
accuracy: 0.9824 - loss: 0.0540 - val_accuracy: 0.9816 - val_loss: 0.0738
Epoch 16/500
375/375              3s 9ms/step -
accuracy: 0.9837 - loss: 0.0489 - val_accuracy: 0.9807 - val_loss: 0.0720
Epoch 17/500
375/375              3s 9ms/step -
accuracy: 0.9849 - loss: 0.0463 - val_accuracy: 0.9822 - val_loss: 0.0726
Epoch 18/500
375/375              3s 9ms/step -
accuracy: 0.9879 - loss: 0.0395 - val_accuracy: 0.9827 - val_loss: 0.0711
Epoch 19/500
375/375              3s 9ms/step -
accuracy: 0.9858 - loss: 0.0421 - val_accuracy: 0.9827 - val_loss: 0.0688
Epoch 20/500
375/375              3s 9ms/step -
accuracy: 0.9879 - loss: 0.0392 - val_accuracy: 0.9824 - val_loss: 0.0718
Epoch 21/500
375/375              3s 9ms/step -
accuracy: 0.9872 - loss: 0.0384 - val_accuracy: 0.9824 - val_loss: 0.0698
Epoch 22/500
375/375              3s 9ms/step -
```

```
accuracy: 0.9869 - loss: 0.0408 - val_accuracy: 0.9817 - val_loss: 0.0732
Epoch 23/500
375/375              3s 9ms/step -
accuracy: 0.9885 - loss: 0.0338 - val_accuracy: 0.9817 - val_loss: 0.0705
Epoch 24/500
375/375              4s 10ms/step -
accuracy: 0.9900 - loss: 0.0300 - val_accuracy: 0.9814 - val_loss: 0.0756
Epoch 24: early stopping
Restoring model weights from the end of the best epoch: 19.
Adamax - Test Accuracy: 98.1300%, Test Loss: 0.0676
```

[56]:
```python
plt.figure(figsize=(14,6))
for name, res in results.items():
    plt.plot(res["history"]["val_accuracy"], label=f"{name}")
plt.xlabel("Epochs")
plt.ylabel("Validation Accuracy")
plt.title("Validation Accuracy with Different Optimizers")
plt.legend()
plt.show()

plt.figure(figsize=(14,6))
for name, res in results.items():
    plt.plot(res["history"]["val_loss"], label=f"{name}")
plt.xlabel("Epochs")
plt.ylabel("Validation Loss")
plt.title("Validation Loss with Different Optimizers")
plt.legend()
plt.show()
```





[57]:
```python
for name, res in results.items():
    print(f"\n{name}:")
    print(f"\tFinal Test Accuracy = {res['test_acc']*100:.4f}%")
    print(f"\tFinal Test Loss     = {res['test_loss']:.4f}")
```

```
SGD:
        Final Test Accuracy = 98.1900%
        Final Test Loss     = 0.0638

RMSprop:
        Final Test Accuracy = 97.8600%
        Final Test Loss     = 0.0766

Adam:
        Final Test Accuracy = 97.9700%
        Final Test Loss     = 0.0671

Adamax:
        Final Test Accuracy = 98.1300%
        Final Test Loss     = 0.0676
```

- Adamax Optimizer gives the best test accuracy