Program3

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

# User-defined activation functions
def sigmoid(x):
   return 1 / (1 + np.exp(-x))
def tanh(x):
   return (np.exp(x) - np.exp(-x)) / (np.exp(x) + np.exp(-x))
def relu(x):
   return np.maximum(0, x)

# Generate input values
x = np.linspace(-5, 5, 100)

built_in_sigmoid = 1/(1+np.exp(-x))
built_in_tanh = np.tanh(x)
built_in_relu = np.maximum(0,x)

import matplotlib.pyplot as plt
import numpy as np

# User-defined activation functions
def sigmoid(x):
  """
```

```
    Sigmoid activation function.
    """
    return 1 / (1 + np.exp(-x))

def tanh(x):
    """
    Tanh activation function.
    """
    return (np.exp(x) - np.exp(-x)) / (np.exp(x) + np.exp(-x))

def relu(x):
    """
    ReLU (Rectified Linear Unit) activation function.
    """
    return np.maximum(0, x)

# Built-in activation functions
from scipy.special import expit  # Sigmoid

# Define input range
x = np.linspace(-5, 5, 100)

# Plot user-defined functions
plt.figure(figsize=(10, 6))
```

```
plt.plot(x, sigmoid(x), label='Sigmoid')
plt.plot(x, tanh(x), label='Tanh')
plt.plot(x, relu(x), label='ReLU')

plt.xlabel('Input (x)')
plt.ylabel('Output')
plt.title('User-defined Activation Functions')
plt.legend()
plt.grid(True)
plt.show()
```

program5

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import SGD, Adam

# Generate some random data for demonstration
np.random.seed(0)
X = np.random.rand(1000, 10)
y = np.random.randint(0, 2, 1000)

# Create a simple neural network model
model = Sequential()
model.add(Dense(64, input_dim=10, activation='relu'))
```

```
model.add(Dense(1, activation='sigmoid'))

# Compile the model using SGD optimizer

model.compile(loss='binary_crossentropy', optimizer=SGD(lr=0.01),
metrics=['accuracy'])

# Train the model with SGD optimizer

model.fit(X, y, epochs=10, batch_size=32)

# Compile the model using Adam optimizer

model.compile(loss='binary_crossentropy', optimizer=Adam(lr=0.001),
metrics=['accuracy'])

# Train the model with Adam optimizer

model.fit(X, y, epochs=10, batch_size=32)
```

Program 7

```
import numpy as np
from keras.models import Model,Sequential
from keras.layers import Flatten,Dense,Conv2D,MaxPooling2D,Dropout
from keras.datasets import mnist
from keras.utils import to_categorical
import matplotlib.pyplot as plt

(x_train,y_train),(x_test,y_test) = mnist.load_data()

x_train = x_train.reshape(-1,28,28,1).astype('float32')/255
```

```
x_test = x_test.reshape(-1,28,28,1).astype('float32')/255

y_train = to_categorical(y_train,10)

y_test = to_categorical(y_test,10)


model = Sequential()

model.add(Conv2D(32,kernel_size=5,strides=1,padding='same',activation
='relu',input_shape=(28,28,1)))

model.add(MaxPooling2D(padding='same'))

model.add(Conv2D(64,kernel_size=5,strides=1,padding='same',activation='relu'))

model.add(MaxPooling2D(padding='same'))


model.add(Flatten())

model.add(Dense(1024,activation='relu'))

model.add(Dropout(0.2))

model.add(Dense(10,activation='sigmoid'))


model.compile(optimizer='adam',loss =
'categorical_crossentropy',metrics=['accuracy'])


model.fit(x_train,y_train,epochs=5,batch_size=64,validation_split=0.2)


def plot_filters(layer,layer_index):

   filters,biases = layer.get_weights()

   n_filters = filters.shape[-1]

   fig,axs = plt.subplots(1,n_filters,figsize=(20,5))

   for i in range(n_filters):
```

```
        f = filters[:,:,0,i]
        axs[i].imshow(f,cmap='gray')
        axs[i].axis('off')
    plt.suptitle(f'Filters of layer {layer_index}')
    plt.show()

def plot_feature_maps(image,layer,layer_index):
    feature_extractor = Model(inputs=model.inputs , outputs = layer.output)
    feature_maps = feature_extractor.predict(image[np.newaxis,...])
    n_feature_maps = feature_maps.shape[-1]
    fig,axis= plt.subplots(1,n_feature_maps,figsize=(20,5))
    for i in range(n_feature_maps):
        f= feature_maps[0,:,:,i]
        axis[i].imshow(f,cmap='gray')
        axis[i].axis('off')
    plt.suptitle(f'Feature maps of layer{layer_index}')
    plt.show()

plot_feature_maps(x_test[0],model.layers[0],0)
```

program8

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import SimpleRNN,Dense
from tensorflow.keras.optimizers import Adam

def generate_data(seq_length,num_seqs):
    x = np.random.rand(num_seqs,seq_length,1)
    y = np.random.rand(num_seqs,seq_length)
    return x,y

sequence_length = 10
num_sequences = 1000
x,y = generate_data(sequence_length,num_sequences)

model = Sequential()
model.add(SimpleRNN(50,activation='relu',input_shape=(sequence_length,1)))
model.add(Dense(1))

model.compile(optimizer=Adam(),loss='mse')
model.fit(x,y,epochs=10,batch_size=32)

loss = model.evaluate(x,y)
print(f'Loss:{loss}')

sample_data = np.random.rand(1,sequence_length)
prediction = model.predict(sample_data)
print(f'Prediction : {prediction}')
```

Program 9

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM,Dense
from tensorflow.keras.optimizers import Adam

def generate_data(seq_length,num_seqs):
    x = np.random.rand(num_seqs,seq_length,1)
    y = np.random.rand(num_seqs,seq_length)
    return x,y

sequence_length = 10
num_sequences = 1000
x,y = generate_data(sequence_length,num_sequences)

model = Sequential()
model.add(LSTM(50,activation='tanh',inaput_shape=(sequence_length,1)))
model.add(Dense(1))

model.compile(optimizer=Adam(),loss='mse')
model.fit(x,y,epochs=10,batch_size=32)

loss = model.evaluate(x,y)
```

```
print(f'Loss:{loss}')


sample_data = np.random.rand(1,sequence_length)
prediction = model.predict(sample_data)
print(f'Prediction : {prediction}')
```