NAME: VEENA T.G.S ROLL NO:225229145 SUB: PDL LAB 12

```
#1.Import
import datetime
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D
```

```
#2.paramaters
now = datetime.datetime.now
batch_size = 128
num_classes = 5
epochs = 5
img_rows, img_cols = 28, 28
filters = 32
pool_size = 2
kernel_size = 3
```

```
#3.MNIST dataset
#data, shuffed and split between train and test sets
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
#create 2 datasets : one with digits below 5 and one with 5 and above
x_train_lt5 = x_train[y_train < 5]
y_train_lt5 = y_train[y_train < 5]
x_test_lt5 = x_test[y_test < 5]
y_test_lt5 = y_test[y_test < 5]

x_train_gte5 = x_train[y_train >= 5]
y_train_gte5 = y_train[y_train >= 5] - 5
x_test_gte5 = x_test[y_test >= 5]
y_test_gte5 = y_test[y_test >= 5] - 5
```

```
#4.Feature layers
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten

# Define the list of feature layers
feature_layers = [
    Conv2D(32, (3, 3), activation='relu', padding='valid', input_shape=(img_rows, img_cols, 1)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.5),
    Flatten()
]
```

```
#5.Classification layers
from keras.layers import Dense, Dropout

# Define the list of classification layers
classification_layers = [
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
]
```

```
#6. Sequential layers
from keras.models import Sequential

# Initialize a Sequential model
model = Sequential()

# Add the feature layers
for layer in feature_layers:
    model.add(layer)

# Add the classification layers
for layer in classification_layers:
    model.add(layer)

# Print the model summary
```

```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_2 (Conv2D)           (None, 26, 26, 32)        320       
                                                                 
 conv2d_3 (Conv2D)           (None, 24, 24, 64)        18496     
                                                                 
 max_pooling2d_1 (MaxPoolin  (None, 12, 12, 64)        0         
 g2D)                                                            
                                                                 
 dropout_2 (Dropout)         (None, 12, 12, 64)        0         
                                                                 
 flatten_1 (Flatten)         (None, 9216)              0         
                                                                 
 dense_2 (Dense)             (None, 128)               1179776   
                                                                 
 dropout_3 (Dropout)         (None, 128)               0         
                                                                 
 dense_3 (Dense)             (None, 5)                 645       
                                                                 
=================================================================
Total params: 1199237 (4.57 MB)
Trainable params: 1199237 (4.57 MB)
Non-trainable params: 0 (0.00 Byte)
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
#7.function train model()
import numpy as np
from datetime import datetime
from tensorflow.keras.utils import to_categorical

def train_model(model, train_data, test_data, num_classes):
    # Unpack the train and test data
    x_train, y_train = train_data
    x_test, y_test = test_data

    # 1) Reshape X and y values of train and test data
    x_train = x_train.reshape(x_train.shape[0], x_train.shape[1], x_train.shape[2], 1)
    x_test = x_test.reshape(x_test.shape[0], x_test.shape[1], x_test.shape[2], 1)
    y_train = y_train.reshape(-1, 1)
    y_test = y_test.reshape(-1, 1)

    # 2) Ensure data type is float32
    x_train = x_train.astype('float32')
    x_test = x_test.astype('float32')

    # 3) Normalize by dividing into 255
    x_train /= 255
    x_test /= 255

    # 4) Print shape of train and test data
    print("Train data shape:", x_train.shape)
    print("Test data shape:", x_test.shape)

    # 5) Convert class vector of y train and y test values into binary class matrices
    y_train = to_categorical(y_train, num_classes)
    y_test = to_categorical(y_test, num_classes)

    # 6) Compile the model
    model.compile(loss='categorical_crossentropy',
                  optimizer='adadelta',
                  metrics=['accuracy'])

    # 7) Fit the model and print training time
    start_time = datetime.now()
    model.fit(x_train, y_train, batch_size=128, epochs=5, verbose=2, validation_split=0.2)
    end_time = datetime.now()
    print("Training time:", end_time - start_time)

    # 8) Evaluate the model on test data and print loss and accuracy
    score = model.evaluate(x_test, y_test, verbose=0)
    print("Test loss:", score[0])
    print("Test accuracy:", score[1])

    return model
```

```
#8.Training on 5 to 9

# Define the number of classes
num_classes = 5  # Since you are working with digits 5, 6, 7, 8, and 9

# Train the model on the specified data
trained_model = train_model(model, (x_train_gte5, y_train_gte5), (x_test_lt5, y_test_lt5), num_classes)
```

```
Train data shape: (29404, 28, 28, 1)
Test data shape: (5139, 28, 28, 1)
Epoch 1/5
184/184 - 62s - loss: 1.6039 - accuracy: 0.2217 - val_loss: 1.5909 - val_accuracy: 0.3566 - 62s/epoch - 339ms/step
Epoch 2/5
184/184 - 60s - loss: 1.5897 - accuracy: 0.2690 - val_loss: 1.5741 - val_accuracy: 0.4198 - 60s/epoch - 324ms/step
Epoch 3/5
184/184 - 59s - loss: 1.5749 - accuracy: 0.3194 - val_loss: 1.5565 - val_accuracy: 0.4870 - 59s/epoch - 322ms/step
Epoch 4/5
184/184 - 59s - loss: 1.5594 - accuracy: 0.3679 - val_loss: 1.5373 - val_accuracy: 0.5649 - 59s/epoch - 320ms/step
Epoch 5/5
184/184 - 59s - loss: 1.5422 - accuracy: 0.4178 - val_loss: 1.5163 - val_accuracy: 0.6373 - 59s/epoch - 322ms/step
Training time: 0:05:23.058946
Test loss: 1.5936849117279053
Test accuracy: 0.30433937907218933
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
    l.trainable = False
```

```
#10.summary
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_2 (Conv2D)           (None, 26, 26, 32)        320       
                                                                 
 conv2d_3 (Conv2D)           (None, 24, 24, 64)        18496     
                                                                 
 max_pooling2d_1 (MaxPoolin  (None, 12, 12, 64)        0         
 g2D)                                                            
                                                                 
 dropout_2 (Dropout)         (None, 12, 12, 64)        0         
                                                                 
 flatten_1 (Flatten)         (None, 9216)              0         
                                                                 
 dense_2 (Dense)             (None, 128)               1179776   
                                                                 
 dropout_3 (Dropout)         (None, 128)               0         
                                                                 
 dense_3 (Dense)             (None, 5)                 645       
                                                                 
=================================================================
Total params: 1199237 (4.57 MB)
Trainable params: 1180421 (4.50 MB)
Non-trainable params: 18816 (73.50 KB)
_________________________________________________________________
```

```
#11.Training for digits
# Train the model on the specified data (digits 0-4)
trained_model = train_model(model, (x_train_lt5, y_train_lt5), (x_test_lt5, y_test_lt5), num_classes)
```

```
Train data shape: (30596, 28, 28, 1)
Test data shape: (5139, 28, 28, 1)
Epoch 1/5
192/192 - 25s - loss: 1.5886 - accuracy: 0.2835 - val_loss: 1.5655 - val_accuracy: 0.4462 - 25s/epoch - 130ms/step
Epoch 2/5
192/192 - 24s - loss: 1.5645 - accuracy: 0.3371 - val_loss: 1.5389 - val_accuracy: 0.5127 - 24s/epoch - 123ms/step
Epoch 3/5
192/192 - 25s - loss: 1.5429 - accuracy: 0.3854 - val_loss: 1.5123 - val_accuracy: 0.5475 - 25s/epoch - 133ms/step
Epoch 4/5
192/192 - 24s - loss: 1.5201 - accuracy: 0.4344 - val_loss: 1.4858 - val_accuracy: 0.5908 - 24s/epoch - 126ms/step
Epoch 5/5
192/192 - 24s - loss: 1.4962 - accuracy: 0.4745 - val_loss: 1.4595 - val_accuracy: 0.6430 - 24s/epoch - 127ms/step
Training time: 0:02:02.906822
Test loss: 1.4635987281799316
Test accuracy: 0.6402024030685425
```

```
#12.Reversing the training process
# Step 1: Train the initial model on digits 0-4
initial_model = train_model(model, (x_train_lt5, y_train_lt5), (x_test_lt5, y_test_lt5), num_classes)

# Step 2: Freeze the layers of the initial model (except classification layers)
for layer in initial_model.layers[:-len(classification_layers)]:
    layer.trainable = False

# Step 3: Replace the last classification layers with new layers for digits 5-9
# Define new classification layers for digits 5-9 (you can customize this architecture)
new_classification_layers = [
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(5, activation='softmax')  # 5 classes for digits 5-9
]

# Create a new model by combining the feature layers from the initial model
# and the new classification layers for digits 5-9
new_model = Sequential(feature_layers + new_classification_layers)

# Step 4: Unfreeze the last layers and fine-tune the model on digits 5-9
for layer in new_model.layers[:-len(new_classification_layers)]:
    layer.trainable = True

# Compile the new model with an appropriate optimizer, loss, and metrics
new_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
est_gte5, y_test_gte5), num_classes)
```

```
Train data shape: (30596, 28, 28, 1)
Test data shape: (5139, 28, 28, 1)
Epoch 1/5
192/192 - 26s - loss: 1.4741 - accuracy: 0.5174 - val_loss: 1.4341 - val_accuracy: 0.6887 - 26s/epoch - 136ms/step
Epoch 2/5
192/192 - 26s - loss: 1.4498 - accuracy: 0.5501 - val_loss: 1.4085 - val_accuracy: 0.7487 - 26s/epoch - 134ms/step
Epoch 3/5
192/192 - 25s - loss: 1.4260 - accuracy: 0.5940 - val_loss: 1.3827 - val_accuracy: 0.7948 - 25s/epoch - 133ms/step
Epoch 4/5
192/192 - 27s - loss: 1.4033 - accuracy: 0.6273 - val_loss: 1.3570 - val_accuracy: 0.8288 - 27s/epoch - 138ms/step
Epoch 5/5
192/192 - 26s - loss: 1.3792 - accuracy: 0.6566 - val_loss: 1.3312 - val_accuracy: 0.8469 - 26s/epoch - 134ms/step
Training time: 0:02:09.623846
Test loss: 1.337362289428711
Test accuracy: 0.8388791680335999
Train data shape: (29404, 28, 28, 1)
Test data shape: (4861, 28, 28, 1)
Epoch 1/5
184/184 - 71s - loss: 1.6065 - accuracy: 0.2362 - val_loss: 1.5946 - val_accuracy: 0.3176 - 71s/epoch - 387ms/step
Epoch 2/5
184/184 - 65s - loss: 1.5919 - accuracy: 0.2838 - val_loss: 1.5765 - val_accuracy: 0.4370 - 65s/epoch - 355ms/step
Epoch 3/5
184/184 - 66s - loss: 1.5758 - accuracy: 0.3383 - val_loss: 1.5566 - val_accuracy: 0.5574 - 66s/epoch - 356ms/step
Epoch 4/5
184/184 - 67s - loss: 1.5594 - accuracy: 0.3908 - val_loss: 1.5348 - val_accuracy: 0.6434 - 67s/epoch - 364ms/step
Epoch 5/5
184/184 - 60s - loss: 1.5395 - accuracy: 0.4467 - val_loss: 1.5112 - val_accuracy: 0.7058 - 60s/epoch - 325ms/step
Training time: 0:06:23.291915
Test loss: 1.5120892524719238
Test accuracy: 0.6817527413368225
```

✓ 8m 43s completed at 10:58 AM

Automatic saving failed. This file was updated remotely or in another tab. Show diff