import numpy as np

import matplotlib.pyplot as plt

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

from keras.optimizers import Adam

from keras.callbacks import ReduceLROnPlateau

# Load MNIST dataset

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

# Preprocess data

x\_train = x\_train.reshape(x\_train.shape[0], 28, 28, 1).astype('float32') / 255

x\_test = x\_test.reshape(x\_test.shape[0], 28, 28, 1).astype('float32') / 255

y\_train = np.eye(10)[y\_train]

y\_test = np.eye(10)[y\_test]

# Define CNN model

def create\_model(learning\_rate=0.001, optimizer='adam'):

model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input\_shape=(28, 28, 1)))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

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

model.compile(loss='categorical\_crossentropy',

optimizer=optimizer,

metrics=['accuracy'])

return model

# Create model

model = create\_model()

# Callbacks

reduce\_lr = ReduceLROnPlateau(monitor='val\_loss', factor=0.2, patience=3, min\_lr=0.0001)

# Train the model

history = model.fit(x\_train, y\_train, validation\_data=(x\_test, y\_test), epochs=10, batch\_size=128, callbacks=[reduce\_lr])

# Evaluate model

test\_loss, test\_acc = model.evaluate(x\_test, y\_test)

print('Test accuracy:', test\_acc)

# Plot training history

plt.plot(history.history['accuracy'], label='accuracy')

plt.plot(history.history['val\_accuracy'], label='val\_accuracy')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend()

plt.show()

import numpy as np

from keras.datasets import imdb

from keras.models import Sequential

from keras.layers import Embedding, SimpleRNN, Dense

from keras.preprocessing.sequence import pad\_sequences

from keras.optimizers import Adam

from keras.callbacks import ReduceLROnPlateau

from sklearn.model\_selection import GridSearchCV

# Load IMDb movie reviews dataset

(x\_train, y\_train), (x\_test, y\_test) = imdb.load\_data(num\_words=10000)

# Preprocess the text data

maxlen = 200

x\_train = pad\_sequences(x\_train, maxlen=maxlen)

x\_test = pad\_sequences(x\_test, maxlen=maxlen)

# Define RNN model

def create\_model(learning\_rate=0.001, optimizer='adam'):

model = Sequential()

model.add(Embedding(10000, 32))

model.add(SimpleRNN(32))

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

model.compile(optimizer=optimizer, loss='binary\_crossentropy', metrics=['accuracy'])

return model

# Create model

model = create\_model()

# Callbacks

reduce\_lr = ReduceLROnPlateau(monitor='val\_loss', factor=0.2, patience=3, min\_lr=0.0001)

# Train the model

history = model.fit(x\_train, y\_train, epochs=10, batch\_size=128, validation\_data=(x\_test, y\_test), callbacks=[reduce\_lr])

# Evaluate model

test\_loss, test\_acc = model.evaluate(x\_test, y\_test)

print('Test accuracy:', test\_acc)

# Plot training history

import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'], label='accuracy')

plt.plot(history.history['val\_accuracy'], label='val\_accuracy')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend()

plt.show()