#CNN ALGORITHM

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential

from keras.layers import Dense, Conv1D, MaxPooling1D, Flatten

from keras.optimizers import Adam

from keras.callbacks import EarlyStopping

from sklearn.metrics import mean\_squared\_error

data = pd.read\_csv("C:/Users/Amit Nagari/Desktop/rainfall/weatherAUS.csv")

data = data[['Date', 'Rainfall']]

data['Date'] = pd.to\_datetime(data['Date'])

data.set\_index('Date', inplace=True)

data.fillna(0, inplace=True)

train\_data = data[:'2015']

test\_data = data['2016':]

scaler = MinMaxScaler(feature\_range=(0, 1))

train\_data\_norm = scaler.fit\_transform(train\_data)

test\_data\_norm = scaler.transform(test\_data)

def create\_sequences(data, lookback=30):

X, y = [], []

for i in range(len(data)-lookback):

X.append(data[i:i+lookback])

y.append(data[i+lookback])

return np.array(X), np.array(y)

lookback = 30

X\_train, y\_train = create\_sequences(train\_data\_norm, lookback)

X\_test, y\_test = create\_sequences(test\_data\_norm, lookback)

model = Sequential()

model.add(Conv1D(filters=64, kernel\_size=2, activation='relu', input\_shape=(X\_train.shape[1], 1)))

model.add(MaxPooling1D(pool\_size=2))

model.add(Flatten())

model.add(Dense(units=50, activation='relu'))

model.add(Dense(units=1))

model.compile(optimizer=Adam(learning\_rate=0.001), loss='mean\_squared\_error')

early\_stop = EarlyStopping(monitor='val\_loss', patience=5, verbose=1)

history = model.fit(X\_train, y\_train, epochs=100, batch\_size=32, validation\_data=(X\_test, y\_test), callbacks=[early\_stop])

train\_pred = model.predict(X\_train)

test\_pred = model.predict(X\_test)

train\_rmse = np.sqrt(mean\_squared\_error(y\_train, train\_pred))

test\_rmse = np.sqrt(mean\_squared\_error(y\_test, test\_pred))

print('Train RMSE:', train\_rmse)

print('Test RMSE:', test\_rmse)

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

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

plt.legend()

plt.show()

Epoch 1/100

3715/3715 [==============================] - 13s 3ms/step - loss: 5.0577e-04 - val\_loss: 4.9260e-04

Epoch 2/100

3715/3715 [==============================] - 11s 3ms/step - loss: 5.0103e-04 - val\_loss: 4.9920e-04

Epoch 3/100

3715/3715 [==============================] - 11s 3ms/step - loss: 4.9861e-04 - val\_loss: 4.9350e-04

Epoch 4/100

3715/3715 [==============================] - 11s 3ms/step - loss: 4.9855e-04 - val\_loss: 4.9094e-04

Epoch 5/100

3715/3715 [==============================] - 11s 3ms/step - loss: 4.9789e-04 - val\_loss: 4.9197e-04

Epoch 6/100

3715/3715 [==============================] - 11s 3ms/step - loss: 4.9624e-04 - val\_loss: 4.9125e-04

Epoch 7/100

3715/3715 [==============================] - 12s 3ms/step - loss: 4.9621e-04 - val\_loss: 4.9370e-04

Epoch 8/100

3715/3715 [==============================] - 10s 3ms/step - loss: 4.9567e-04 - val\_loss: 4.9271e-04

Epoch 9/100

3715/3715 [==============================] - 11s 3ms/step - loss: 4.9532e-04 - val\_loss: 5.0207e-04

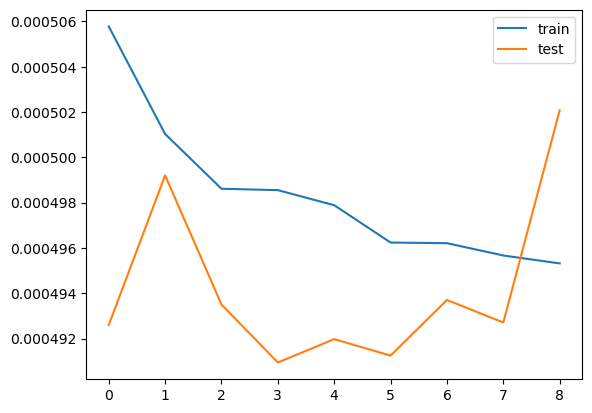
Epoch 9: early stopping

3715/3715 [==============================] - 7s 2ms/step

829/829 [==============================] - 1s 2ms/step

Train RMSE: 0.022370779291632545

Test RMSE: 0.022407006674675236



#LSTM ALGORITHM

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential

from keras.layers import Dense, LSTM, Dropout

from keras.optimizers import Adam

from keras.callbacks import EarlyStopping

from sklearn.metrics import mean\_squared\_error

data = pd.read\_csv("C:/Users/Amit Nagari/Desktop/rainfall/weatherAUS.csv")

data = data[['Date', 'Rainfall']]

data['Date'] = pd.to\_datetime(data['Date'])

data.set\_index('Date', inplace=True)

data.fillna(0, inplace=True)

train\_data = data[:'2015']

test\_data = data['2016':]

scaler = MinMaxScaler(feature\_range=(0, 1))

train\_data\_norm = scaler.fit\_transform(train\_data)

test\_data\_norm = scaler.transform(test\_data)

def create\_sequences(data, lookback=30):

X, y = [], []

for i in range(len(data)-lookback):

X.append(data[i:i+lookback])

y.append(data[i+lookback])

return np.array(X), np.array(y)

lookback = 30

X\_train, y\_train = create\_sequences(train\_data\_norm, lookback)

X\_test, y\_test = create\_sequences(test\_data\_norm, lookback)

model = Sequential()

model.add(LSTM(units=32, return\_sequences=True, input\_shape=(X\_train.shape[1], 1)))

model.add(Dropout(0.2))

model.add(LSTM(units=32))

model.add(Dropout(0.2))

model.add(Dense(units=1))

model.compile(optimizer=Adam(learning\_rate=0.001), loss='mean\_squared\_error')

early\_stop = EarlyStopping(monitor='val\_loss', patience=5, verbose=1)

history = model.fit(X\_train, y\_train, epochs=100, batch\_size=32, validation\_data=(X\_test, y\_test), callbacks=[early\_stop])

train\_pred = model.predict(X\_train)

test\_pred = model.predict(X\_test)

train\_rmse = np.sqrt(mean\_squared\_error(y\_train, train\_pred))

test\_rmse = np.sqrt(mean\_squared\_error(y\_test, test\_pred))

print('Train RMSE:', train\_rmse)

print('Test RMSE:', test\_rmse)

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

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

plt.legend()

plt.show()

#RNNs ALGORITHM

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential

from keras.layers import Dense, SimpleRNN, Dropout

from keras.optimizers import Adam

from keras.callbacks import EarlyStopping

from sklearn.metrics import mean\_squared\_error

data = pd.read\_csv("C:/Users/Amit Nagari/Desktop/rainfall/weatherAUS.csv")

data = data[['Date', 'Rainfall']]

data['Date'] = pd.to\_datetime(data['Date'])

data.set\_index('Date', inplace=True)

data.fillna(0, inplace=True)

train\_data = data[:'2015']

test\_data = data['2016':]

scaler = MinMaxScaler(feature\_range=(0, 1))

train\_data\_norm = scaler.fit\_transform(train\_data)

test\_data\_norm = scaler.transform(test\_data)

def create\_sequences(data, lookback=30):

X, y = [], []

for i in range(len(data)-lookback):

X.append(data[i:i+lookback])

y.append(data[i+lookback])

return np.array(X), np.array(y)

lookback = 30

X\_train, y\_train = create\_sequences(train\_data\_norm, lookback)

X\_test, y\_test = create\_sequences(test\_data\_norm, lookback)

model = Sequential()

model.add(SimpleRNN(units=32, return\_sequences=True, input\_shape=(X\_train.shape[1], 1)))

model.add(Dropout(0.2))

model.add(SimpleRNN(units=32))

model.add(Dropout(0.2))

model.add(Dense(units=1))

model.compile(optimizer=Adam(learning\_rate=0.001), loss='mean\_squared\_error')

early\_stop = EarlyStopping(monitor='val\_loss', patience=5, verbose=1)

history = model.fit(X\_train, y\_train, epochs=100, batch\_size=32, validation\_data=(X\_test, y\_test), callbacks=[early\_stop])

train\_pred = model.predict(X\_train)

test\_pred = model.predict(X\_test)

train\_rmse = np.sqrt(mean\_squared\_error(y\_train, train\_pred))

test\_rmse = np.sqrt(mean\_squared\_error(y\_test, test\_pred))

print('Train RMSE:', train\_rmse)

print('Test RMSE:', test\_rmse)

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

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

plt.legend()

plt.show()

Epoch 1/100

3715/3715 [==============================] - 76s 20ms/step - loss: 4.8047e-04 - val\_loss: 4.6773e-04

Epoch 2/100

3715/3715 [==============================] - 69s 19ms/step - loss: 4.7008e-04 - val\_loss: 4.7664e-04

Epoch 3/100

3715/3715 [==============================] - 73s 20ms/step - loss: 4.6932e-04 - val\_loss: 4.6451e-04

Epoch 4/100

3715/3715 [==============================] - 73s 20ms/step - loss: 4.6948e-04 - val\_loss: 4.8256e-04

Epoch 5/100

3715/3715 [==============================] - 74s 20ms/step - loss: 4.6885e-04 - val\_loss: 4.6458e-04

Epoch 6/100

3715/3715 [==============================] - 73s 20ms/step - loss: 4.6794e-04 - val\_loss: 4.6117e-04

Epoch 7/100

3715/3715 [==============================] - 75s 20ms/step - loss: 4.6627e-04 - val\_loss: 4.6674e-04

Epoch 8/100

3715/3715 [==============================] - 76s 20ms/step - loss: 4.6735e-04 - val\_loss: 4.6187e-04

Epoch 9/100

3715/3715 [==============================] - 73s 20ms/step - loss: 4.6648e-04 - val\_loss: 4.6194e-04

Epoch 10/100

3715/3715 [==============================] - 77s 21ms/step - loss: 4.6618e-04 - val\_loss: 4.6137e-04

Epoch 11/100

3715/3715 [==============================] - 75s 20ms/step - loss: 4.6588e-04 - val\_loss: 4.6345e-04

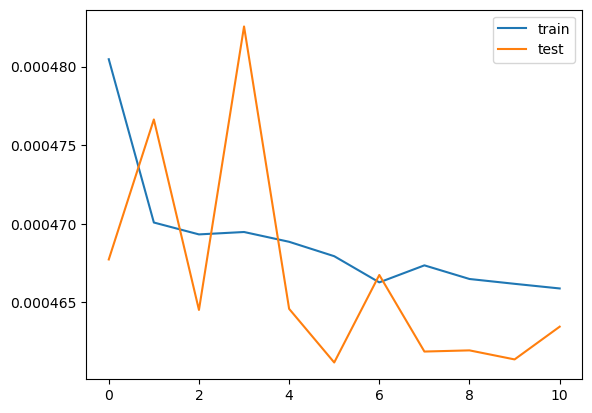
Epoch 11: early stopping

3715/3715 [==============================] - 27s 7ms/step

829/829 [==============================] - 6s 7ms/step

Train RMSE: 0.021447470215571376

Test RMSE: 0.021527892419209122



#RNN ALGORITHM

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential

from keras.layers import Dense, SimpleRNN, Dropout

from keras.optimizers import Adam

from keras.callbacks import EarlyStopping

from sklearn.metrics import mean\_squared\_error

data = pd.read\_csv("C:/Users/Amit Nagari/Desktop/rainfall/weatherAUS.csv")

data = data[['Date', 'Rainfall']]

data['Date'] = pd.to\_datetime(data['Date'])

data.set\_index('Date', inplace=True)

data.fillna(0, inplace=True)

train\_data = data[:'2015']

test\_data = data['2016':]

scaler = MinMaxScaler(feature\_range=(0, 1))

train\_data\_norm = scaler.fit\_transform(train\_data)

test\_data\_norm = scaler.transform(test\_data)

def create\_sequences(data, lookback=30):

X, y = [], []

for i in range(len(data)-lookback):

X.append(data[i:i+lookback])

y.append(data[i+lookback])

return np.array(X), np.array(y)

lookback = 30

X\_train, y\_train = create\_sequences(train\_data\_norm, lookback)

X\_test, y\_test = create\_sequences(test\_data\_norm, lookback)

model = Sequential()

model.add(SimpleRNN(units=32, return\_sequences=True, input\_shape=(X\_train.shape[1], 1)))

model.add(Dropout(0.2))

model.add(SimpleRNN(units=32))

model.add(Dropout(0.2))

model.add(Dense(units=1))

model.compile(optimizer=Adam(learning\_rate=0.001), loss='mean\_squared\_error')

early\_stop = EarlyStopping(monitor='val\_loss', patience=5, verbose=1)

history = model.fit(X\_train, y\_train, epochs=100, batch\_size=32, validation\_data=(X\_test, y\_test), callbacks=[early\_stop])

train\_pred = model.predict(X\_train)

test\_pred = model.predict(X\_test)

train\_rmse = np.sqrt(mean\_squared\_error(y\_train, train\_pred))

test\_rmse = np.sqrt(mean\_squared\_error(y\_test, test\_pred))

print('Train RMSE:', train\_rmse)

print('Test RMSE:', test\_rmse)

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

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

plt.legend()

plt.show()

Epoch 1/100

3715/3715 [==============================] - 37s 10ms/step - loss: 6.2780e-04 - val\_loss: 4.7714e-04

Epoch 2/100

3715/3715 [==============================] - 35s 9ms/step - loss: 4.7958e-04 - val\_loss: 4.6974e-04

Epoch 3/100

3715/3715 [==============================] - 35s 9ms/step - loss: 4.7525e-04 - val\_loss: 4.7058e-04

Epoch 4/100

3715/3715 [==============================] - 35s 9ms/step - loss: 4.7459e-04 - val\_loss: 4.7239e-04

Epoch 5/100

3715/3715 [==============================] - 35s 9ms/step - loss: 4.7631e-04 - val\_loss: 4.6533e-04

Epoch 6/100

3715/3715 [==============================] - 33s 9ms/step - loss: 4.7425e-04 - val\_loss: 4.6910e-04

Epoch 7/100

3715/3715 [==============================] - 33s 9ms/step - loss: 4.7304e-04 - val\_loss: 4.7133e-04

Epoch 8/100

3715/3715 [==============================] - 33s 9ms/step - loss: 4.7207e-04 - val\_loss: 4.6728e-04

Epoch 9/100

3715/3715 [==============================] - 33s 9ms/step - loss: 4.7218e-04 - val\_loss: 4.8090e-04

Epoch 10/100

3715/3715 [==============================] - 32s 9ms/step - loss: 4.7289e-04 - val\_loss: 4.6768e-04

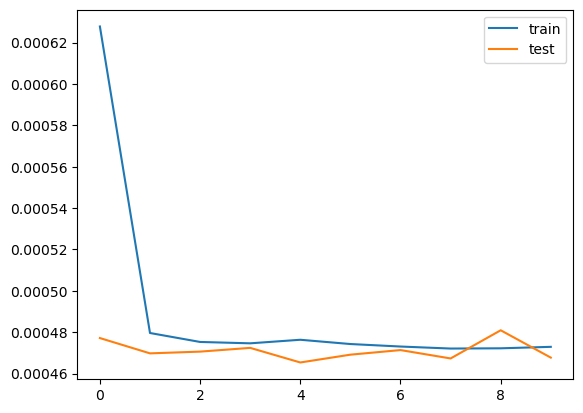
Epoch 10: early stopping

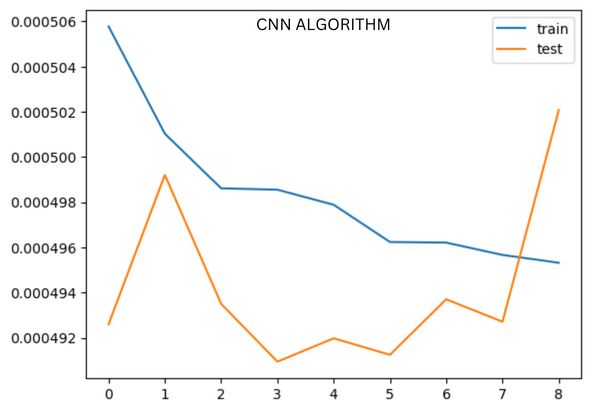
3715/3715 [==============================] - 12s 3ms/step

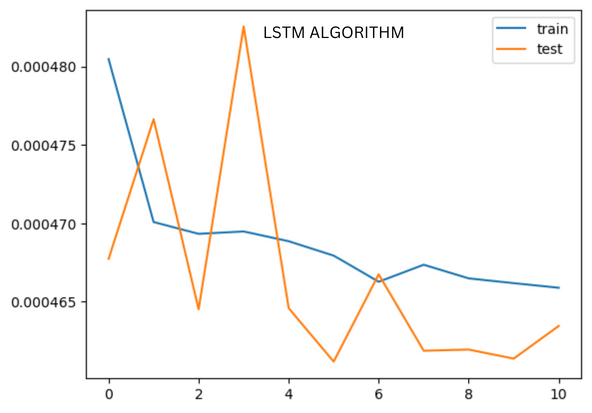
829/829 [==============================] - 3s 3ms/step

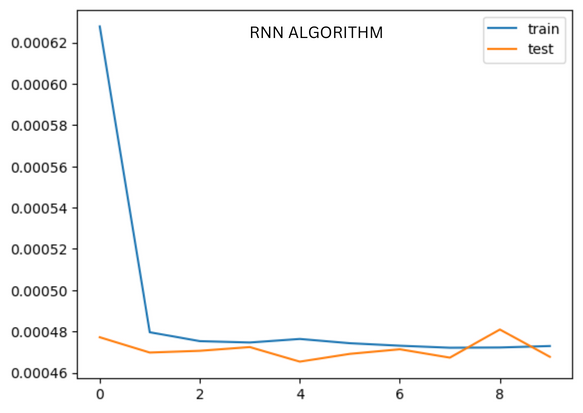
Train RMSE: 0.021579829639850705

Test RMSE: 0.0216259043828627









#CNN Test RMSE:

CNN\_RMSE=0.022407006674675236

#LSTM Test RMSE :

LSTM\_RMSE=0.021527892419209122

#RNN Test RMSE:

RNN\_RMSE=0.0216259043828627

# Determine which algorithm has the lowest RMSE score

if CNN\_RMSE < LSTM\_RMSE and CNN\_RMSE < RNN\_RMSE:

print("CNN algorithm has more accuracy for given dataset")

elif LSTM\_RMSE < CNN\_RMSE and LSTM\_RMSE < RNN\_RMSE:

print("LSTM algorithm has more accuracy for given dataset")

else:

print("RNN algorithm has more accuracy for given dataset")

#output

-🡪LSTM algorithm has more accuracy for given dataset

import matplotlib.pyplot as plt

# Assign the RMSE scores for each algorithm

CNN\_RMSE = 0.022407006674675236

LSTM\_RMSE =0.021527892419209122

RNN\_RMSE = 0.0216259043828627

# Create a bar chart of the RMSE scores

labels = ['CNN', 'LSTM', 'RNN']

values = [CNN\_RMSE, LSTM\_RMSE, RNN\_RMSE]

plt.bar(labels, values)

# Add labels and title to the chart

plt.xlabel('Algorithm')

plt.ylabel('RMSE Score')

plt.title('Comparison of RMSE Scores for Weather Prediction Algorithms')

# Display the chart

plt.show()

