Praneesh R V

CB.SC.U4CYS23036

Lab - 3 - Feed Forward Neural Network to classify traffic data

import kagglehub

path = kagglehub.dataset\_download("chethuhn/network-intrusion-dataset")

print("Path to dataset files:", path)

from google.colab import drive

drive.mount('/content/drive')

import os

kaggle\_path = '/root/.cache/kagglehub/datasets/chethuhn/network-intrusion-dataset/versions/1'

for file in os.listdir(kaggle\_path):

print(file)

import os

import shutil

kaggle\_folder = '/root/.cache/kagglehub/datasets/chethuhn/network-intrusion-dataset/versions/1'

drive\_folder = '/content/drive/MyDrive/CICIDS2017\_Parts'

os.makedirs(drive\_folder, exist\_ok=True)

for file in os.listdir(kaggle\_folder):

if file.endswith('.csv'):

shutil.copy(os.path.join(kaggle\_folder, file), os.path.join(drive\_folder, file))

import pandas as pd

import glob

csv\_files = glob.glob('/content/drive/MyDrive/CICIDS2017\_Parts/\*.csv')

df = pd.concat((pd.read\_csv(f) for f in csv\_files), ignore\_index=True)

df.head()

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.model\_selection import train\_test\_split

data = df.dropna()

import numpy as np

from sklearn.preprocessing import StandardScaler

X = df.drop(' Label', axis=1)

y = df[' Label']

X = X[(X >= 0).all(axis=1)]

threshold = 1e8

X = X[(X < threshold).all(axis=1)]

y = y.loc[X.index]

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X\_scaled, y, test\_size=0.2, random\_state=42

)

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout

model = Sequential()

model.add(Dense(128, input\_dim=X\_train.shape[1], activation='relu'))

model.add(Dense(64, activation='relu'))

model.add(Dropout(0.2)) model.add(Dense(32, activation='relu'))

from sklearn.preprocessing import LabelEncoder

from tensorflow.keras.utils import to\_categorical

import numpy as np

import pandas as pd

label\_encoder = LabelEncoder()

label\_encoder.fit(pd.concat([y\_train, y\_test]).unique())

y\_train\_encoded = label\_encoder.transform(y\_train)

y\_test\_encoded = label\_encoder.transform(y\_test)

num\_classes = len(label\_encoder.classes\_)

y\_train\_categorical = to\_categorical(y\_train\_encoded, num\_classes=num\_classes)

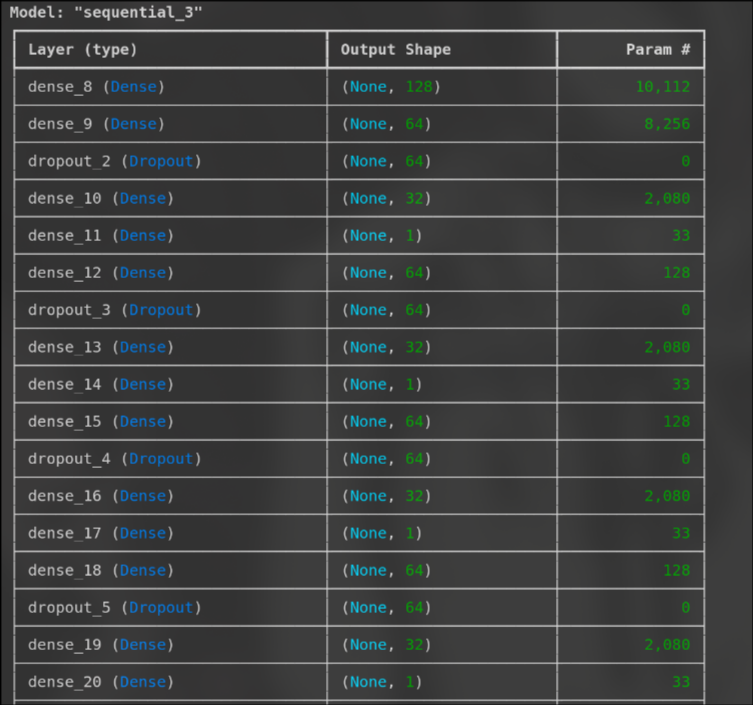
y\_test\_categorical = to\_categorical(y\_test\_encoded, num\_classes=num\_classes)

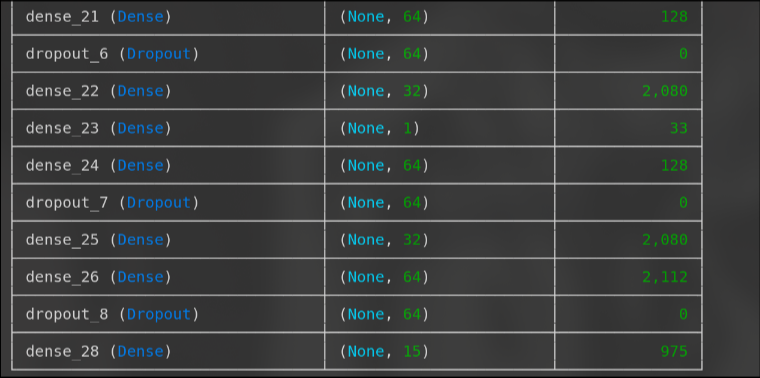
model.pop() model.add(Dense(num\_classes, activation='softmax'))

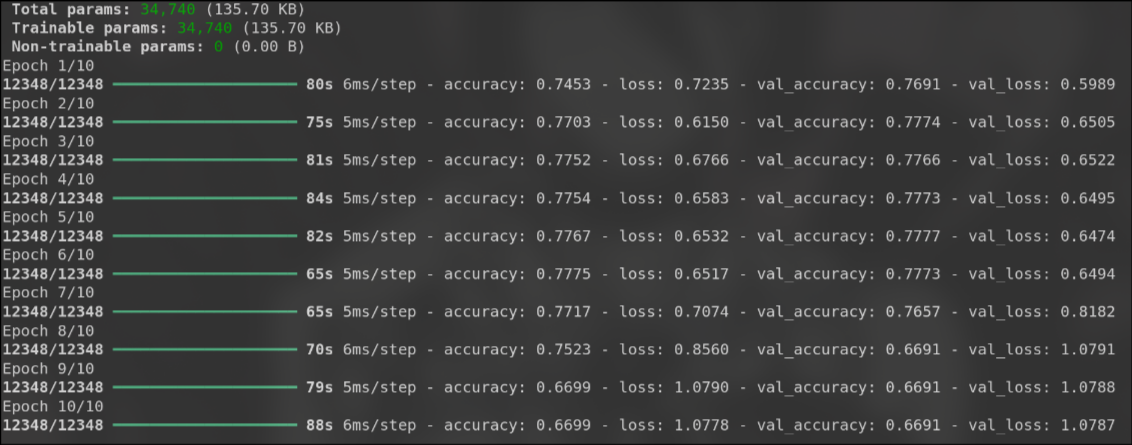
model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

model.summary()

history = model.fit(X\_train, y\_train\_categorical, epochs=10, batch\_size=64, validation\_split=0.2)







# Plot training history

import matplotlib.pyplot as plt

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

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

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend()

plt.show()

