Ministerul Educaţiei, Culturii și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Departamentul Ingineria Software și Automatică

**RAPORT**

Lucrare de laborator Nr.5

Disciplina:Sisteme de inteligență artificială

Tema: Rețele Neuronale de generația II (multistrat)

A efectuat: st.gr. IA-182 ,

Ulmanu Cristian

A verificat : asist.univ. Stratulat Ștefan

Chișinău 2020

**Sarcina:** Rezolvarea unei probleme utilizând Rețele Neurale Multistrat din biblioteca Tensorflow.

**Pași:**

* Studiați exemplul propus la acestă lucrare de laborator.
* Studiați documentația tensorflow și tflearn (https://www.tensorflow.org/tutorials).
* Fiecare dintre studenți își alege un set de date din repozitoriile prezentate în lucrarea de laborator nr. 4 (kaggle, uci, data.world). Puteți lucra și cu setul de date folosit în laboratorul 4.
* Definiți problema.
* Elaborați o rețea neurală multistrat în Tensorflow și încercați să rezolvați problema definită aplicând rețeaua neurală multistrat pe setul dvs. de date.

1. Selectarea setului de date.

Setul de date pentru lucrare reprezintă o colecție de imagini pe care se află diverse indicatoare rutiere. Ele se află pe siteul Kaggle.com și include imaginile indicatoarelor rutiere din Germania https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign.

1. Problema.

Problema care este propusă pentru rezolvarea prezintă: Recunoșterea valorii fiecărui indicator rutier.

1. Studierea bibliotecilor.

TensorFlow este o bibliotecă software deschisă pentru învățarea automată dezvoltată de Google pentru rezolvarea problemelor de construire și formare a unei rețele neuronale în scopul de a găsi și clasifica automat modele, obținând calitatea percepției umane [4] . Este utilizat atât pentru cercetarea, cât și pentru dezvoltarea propriilor produse Google . API-ul principal pentru lucrul cu biblioteca este implementat pentru Python , există și implementări pentru R , C Sharp , C ++ , Haskell , Java , Goși Swift .

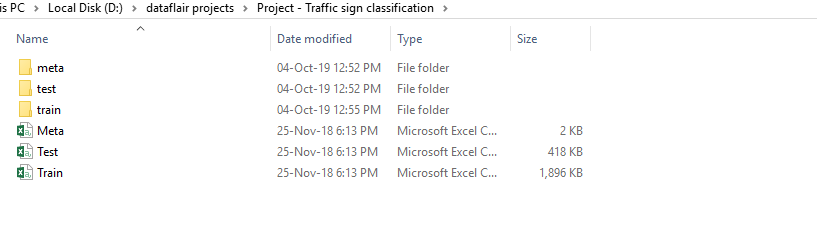
Keras este o bibliotecă de rețea neuronală open source scrisă în Python . Este construit deasupra cadrelor Deeplearning4j , TensorFlow și Theano . Se concentrează pe munca operațională cu rețele de învățare profundă , fiind în același timp concepută pentru a fi compactă, modulară și extensibilă. A fost creat ca parte a eforturilor unui proiect de cercetare Oneiros ( ing. Open-Ended Neuro-of Electronic the Intelligent the Robot the Operating the System ), și autorul său principal și un coleg François Chollet ( fr. François Chollet ), inginer Google .

S-a planificat ca Google să sprijine Keras în biblioteca principală TensorFlow, dar Scholle a selectat Keras ca un supliment separat, deoarece, conform conceptului, Keras este mai mult o interfață decât un sistem de învățare automată end-to-end. Keras oferă un set de abstractizări la nivel înalt, mai intuitiv, care facilitează construirea rețelelor neuronale, indiferent de biblioteca științifică de calcul utilizată ca backend de calcul. Microsoft lucrează la adăugarea de biblioteci CNTK de nivel scăzut la Keras

1. Studierea setului de date.

Dosarul nostru „tren” conține 43 de foldere reprezentând fiecare o clasă diferită. Gama de dosare este de la 0 la 42. Cu ajutorul modulului de operare, vom itera peste toate clasele și adăugăm imagini și etichetele lor respective în lista de date și etichete.

Biblioteca PIL este utilizată pentru a deschide conținutul imaginii într-o matrice.



1. Procesarea setului de date.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import cv2

import tensorflow as tf

from PIL import Image

import os

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

from keras.utils import to\_categorical

from keras.models import Sequential, load\_model

from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout

data = []

labels = []

classes = 43

cur\_path = os.getcwd()

#Retrieving the images and their labels

for i in range(classes):

    path = os.path.join(cur\_path,'train',str(i))

    images = os.listdir(path)

    for a in images:

        try:

            image = Image.open(path + '\\'+ a)

            image = image.resize((30,30))

            image = np.array(image)

            #sim = Image.fromarray(image)

            data.append(image)

            labels.append(i)

        except:

            print("Error loading image")

#Converting lists into numpy arrays

data = np.array(data)

labels = np.array(labels)

* În cele din urmă, am stocat toate imaginile și etichetele acestora în liste (date și etichete).
* Trebuie să convertim lista în matrici numpy pentru a alimenta modelul.
* Forma datelor este (39209, 30, 30, 3) ceea ce înseamnă că există 39.209 imagini cu dimensiunea 30 × 30 pixeli, iar ultimele 3 înseamnă că datele conțin imagini colorate (valoare RGB).
* Cu pachetul sklearn, folosim metoda train\_test\_split () pentru a împărți datele de formare și testare.
* Din pachetul keras.utils, folosim metoda to\_categorical pentru a converti etichetele prezente în y\_train și t\_test într-o codare one-hot.
* print(data.shape, labels.shape)
* #Splitting training and testing dataset
* X\_train, X\_test, y\_train, y\_test = train\_test\_split(data, labels, test\_size=0.2, random\_state=42)
* print(X\_train.shape, X\_test.shape, y\_train.shape, y\_test.shape)
* #Converting the labels into one hot encoding
* y\_train = to\_categorical(y\_train, 43)
* y\_test = to\_categorical(y\_test, 43)

Pentru a clasifica imaginile în categoriile lor respective, vom construi un model CNN ( Convolutional Neural Network ). CNN este cel mai bun în scopul clasificării imaginilor.

Arhitectura modelului nostru este:

* 2 strat Conv2D (filtru = 32, kernel\_size = (5,5), activare = ”relu”)
* Stratul MaxPool2D (pool\_size = (2,2))
* Stratul de abandon (rata = 0,25)
* 2 strat Conv2D (filtru = 64, kernel\_size = (3,3), activare = ”relu”)
* Stratul MaxPool2D (pool\_size = (2,2))
* Stratul de abandon (rata = 0,25)
* Aplatizați stratul pentru a strânge straturile într-o singură dimensiune
* Strat dens conectat complet (256 noduri, activare = ”relu”)
* Stratul de abandon (rata = 0,5)
* Strat dens (43 noduri, activare = ”softmax”)
* Compilăm modelul cu optimizatorul Adam, care funcționează bine, iar pierderea este „categorical\_crossentropy”, deoarece avem mai multe clase de clasificat.

#Building the model

model = Sequential()

model.add(Conv2D(filters=32, kernel\_size=(5,5), activation='relu', input\_shape=X\_train.shape[1:]))

model.add(Conv2D(filters=32, kernel\_size=(5,5), activation='relu'))

model.add(MaxPool2D(pool\_size=(2, 2)))

model.add(Dropout(rate=0.25))

model.add(Conv2D(filters=64, kernel\_size=(3, 3), activation='relu'))

model.add(Conv2D(filters=64, kernel\_size=(3, 3), activation='relu'))

model.add(MaxPool2D(pool\_size=(2, 2)))

model.add(Dropout(rate=0.25))

model.add(Flatten())

model.add(Dense(256, activation='relu'))

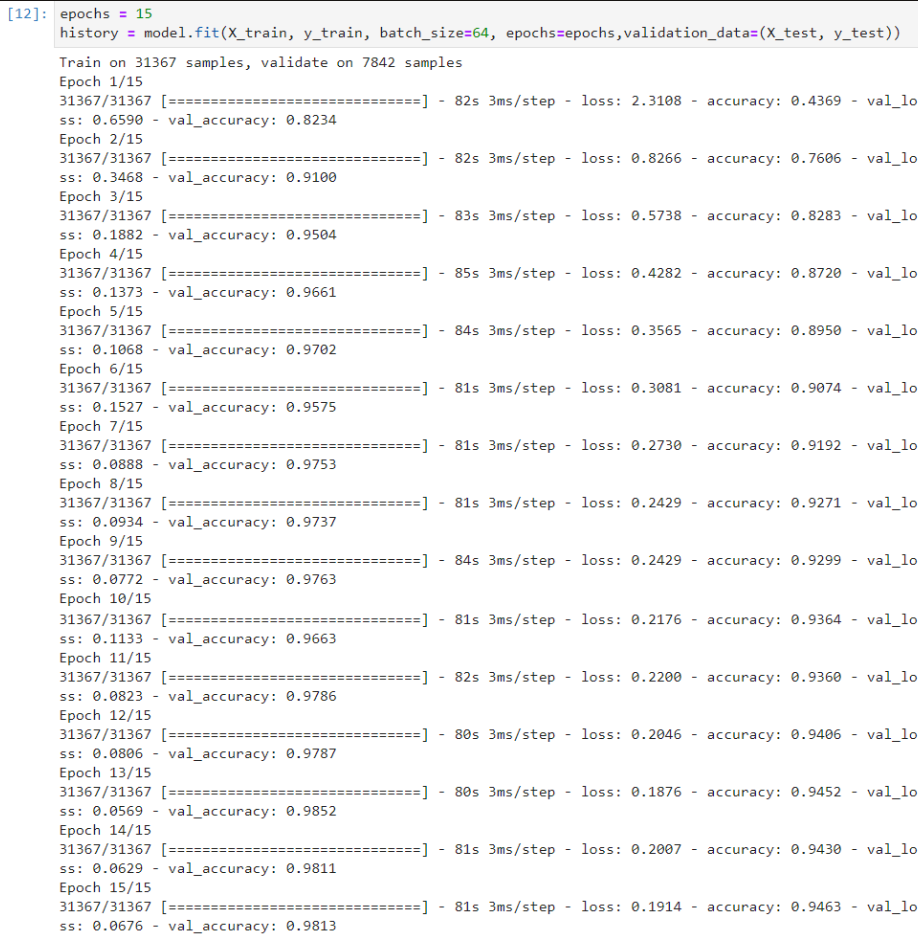
model.add(Dropout(rate=0.5))

model.add(Dense(43, activation='softmax'))

#Compilation of the model

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

După construirea arhitecturii modelului, instruim modelul folosind model.fit (). Am încercat cu mărimea lotului 32 și 64. Modelul nostru a funcționat mai bine cu dimensiunea lotului 64. Și după 15 epoci, precizia a fost stabilă.



Modelul nostru a obținut o precizie de 95% asupra setului de date de antrenament. Cu matplotlib, trasăm graficul pentru precizie și pierdere.

plotting graphs for accuracy

plt.figure(0)

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

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

plt.title('Accuracy')

plt.xlabel('epochs')

plt.ylabel('accuracy')

plt.legend()

plt.show()

plt.figure(1)

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

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

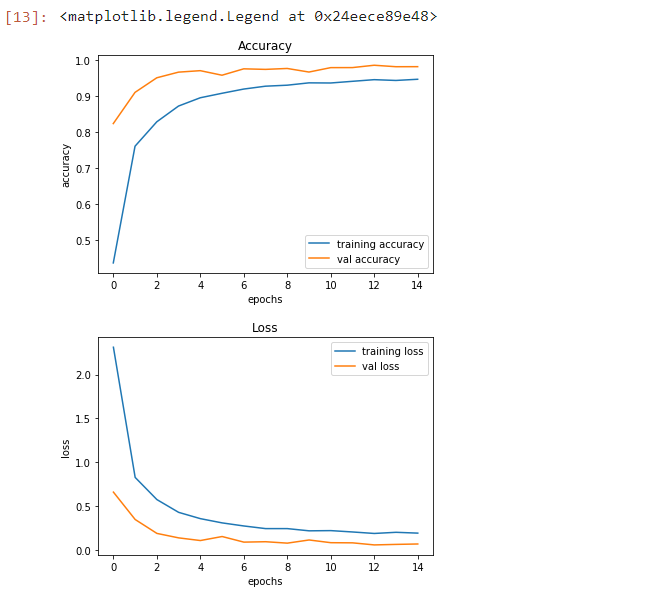
plt.title('Loss')

plt.xlabel('epochs')

plt.ylabel('loss')

plt.legend()

plt.show()



Setul nostru de date conține un folder de testare și, într-un fișier test.csv, avem detalii legate de calea imaginii și etichetele clasei respective. Extragem calea imaginii și etichetele folosind panda. Apoi, pentru a prezice modelul, trebuie să ne redimensionăm imaginile la 30 × 30 pixeli și să realizăm o matrice numpy care conține toate datele imaginii. Din sklearn.metrics, am importat accurate\_score și am observat modul în care modelul nostru a prezis etichetele reale. Am obținut o precizie de 95% în acest model.

#testing accuracy on test dataset

from sklearn.metrics import accuracy\_score

y\_test = pd.read\_csv('Test.csv')

labels = y\_test["ClassId"].values

imgs = y\_test["Path"].values

data=[]

for img in imgs:

    image = Image.open(img)

    image = image.resize((30,30))

    data.append(np.array(image))

X\_test=np.array(data)

pred = model.predict\_classes(X\_test)

#Accuracy with the test data

from sklearn.metrics import accuracy\_score

print(accuracy\_score(labels, pred))

**Concluzii:**

Utilizat în principal de producătorii de rețele sociale, telecomunicații și telefoane; recunoașterea feței, căutarea imaginilor, detectarea mișcărilor, viziunea automată și clusterizarea foto pot fi utilizate și în industria auto, aviației și a asistenței medicale. Recunoașterea imaginilor este concepută pentru a identifica și identifica persoanele și persoanele din imagini Obiectează și înțelege conținutul și contextul acestuia. Algoritmul de recunoaștere a obiectelor TensorFlow poate clasifica și recunoaște orice obiect dintr-o imagine mai mare. Este utilizat în mod obișnuit în aplicațiile de inginerie pentru a identifica forme în scopuri de modelare (construiți spațiu 3D din imagini 2D) și rețele sociale pentru etichetarea fotografiilor (Facebook's Deep Face). De exemplu, analizând mii de fotografii cu copaci, tehnologia poate învăța să recunoască copaci care nu au mai fost văzuți până acum.

Recunoașterea imaginilor începe să se extindă și în industria sănătății, unde algoritmii TensorFlow pot procesa mai multe informații și detecta mai multe tipare decât omologii lor umani. Computerele sunt acum capabile să analizeze scanările și să identifice mai multe boli decât oamenii.