Documentatie proiect IA

Pentru acest proiect am folosit un model CNN (Convolutional neural network), implementat prin libraria Keras (din backend TensorFlow). Mai exact, am folosit arhitectura Sequential pentru acest model, prin care layerele network-ului neural sunt inlantuite.

CNN este un tip de retele neurale adanci, prin care, la fiecare layer (sau strat convolutional) extrage trasaturi, informatii din datele de input, care apoi devin mai complexe, odata cu parcurgerea retelei.

Motivatia alegerii acestui model este pentru ca este metoda standard in prelucrarea imaginilor sau in aplicatii de Computer Vision. Prin aceste layere convolutionale putem extrage date importante, precum linii, unghiuri, care se pot transforma in patternuri mai complexe odata cu adancirea straturilor convolutionale.

De asemenea, am incercat sa aplic si un KNN algoritm, dar am primit o rata foarte mica, iar timpul necesar determinarii acestei rate a fost foarte mare (IDE-ul meu a fost notebook-ul integrat al website-ului Kaggle).

Pentru citirea/scrierea datelor, am folosit modulele numpy, os si pandas, precum si functii specifice API-ului oferit de Keras, din keras.preprocessing.image. Aceste functii (load_img si img_to_array) m-au ajutat sa preprocesez datele, pentru a le transforma intr-un array de dimensiuni 3d: 32x32x3.

Dupa ce am incarcat aceste date, am eliminat ultimele date aditionale pentru fiecare imagine. Pentru fiecare imagine, am realizat un reshape si le-am incarcat intr-un np array, unde apoi am normalizat datele. Datele au fost normalizate prin transformare in float, iar apoi prin impartirea la 255, realizam normalizarea prin reducerea efectului de diferenta din cauza iluminarii. De asemenea, datele din cadrul algoritmului CNN converg mai rapid in intervalul [0,1], decat in [0,255]

Urmatorul pas este sa tranform labelurile pentru categorii intr-un vector binar “hot vector”, care va pastra in format binar datele, de ex, pentru o poza din categoria 6, as avea vectorul (0,0,0,0,0,1,0,0,0)

Pentru arhitectura modelului, am 2 zone principale:

1. Prima, este formata din straturile de tip Convolution si Pooling
2. A doua, formata din straturi Fully Connected.

Layer-ul de tip Convolution (Conv2D) presupune alegerea unui set de filtre/ detector de trasaturi ale imaginii (eu am ales 8 la primul, si 16 la al doilea) Fiecare filtru transforma o parte a imaginii (adica o matrice, determinata de dimensiunea kernelului, pe care l-am ales 5x5 si apoi 3x3). Matricea kernelului este aplicata pe intreaga imagine, astfel transformam imaginea, intr-un fel. Prin Conv2D, izolam niste informatii, care mai apoi fi generalizate, devenind mai complexe, odata cu parcurgerea retelei.

De asemenea, am folosit un strat de activare ReLU (Rectified Linear Activated Function), ce reprezinta functia de activare a stratului, adaugand non-linearitate stratului (din moment ce avem de aface cu imagini, imaginile sunt nonlineare).

Al doilea strat CNN este cel de Pooling (MaxPool2D), care functioneaza ca un filtru down-sampling (reduce numarul de parametrii) ce se uita la pixelii vecini si alege valoarea maxima. Acesta este folosit pentru a reduce costul computational, si pentru a reduce overfitting. Face detectia de informatii invarianta la orientare si scara.

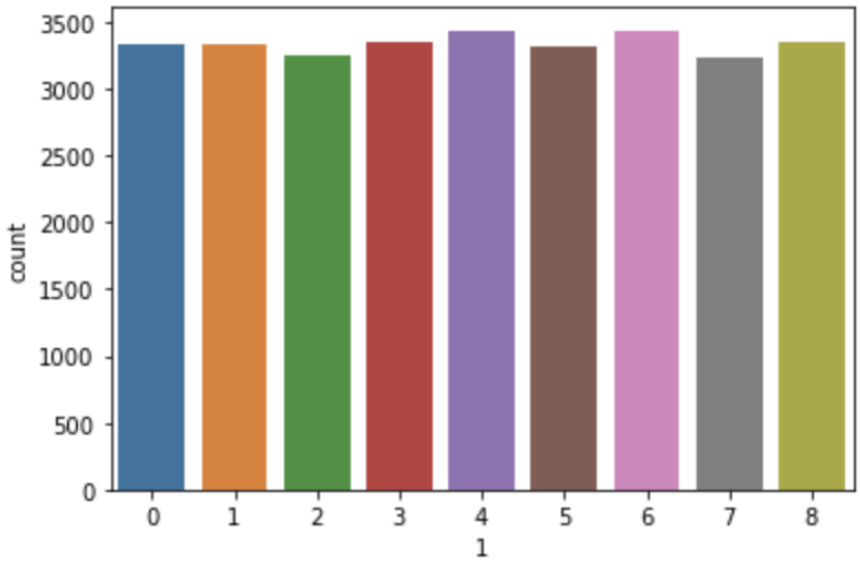
Combinarea celor 2 straturi, de Convolution si Pooling, CNN poate combina informatii locale pentru a determina informatii globale.

De asemenea, folosim Dropout, careia ii trimitem ca un parametru rata la care reteaua ar trebui sa ignore aleatoriu o proportie de noduri din strat, pentru fiecare set de date. Astfel, o proportie din retea este ignorata si forteaza reteaua sa invete trasaturi intr-o metoda distribuita. Este o metoda buna pentru a reduce overfitting.

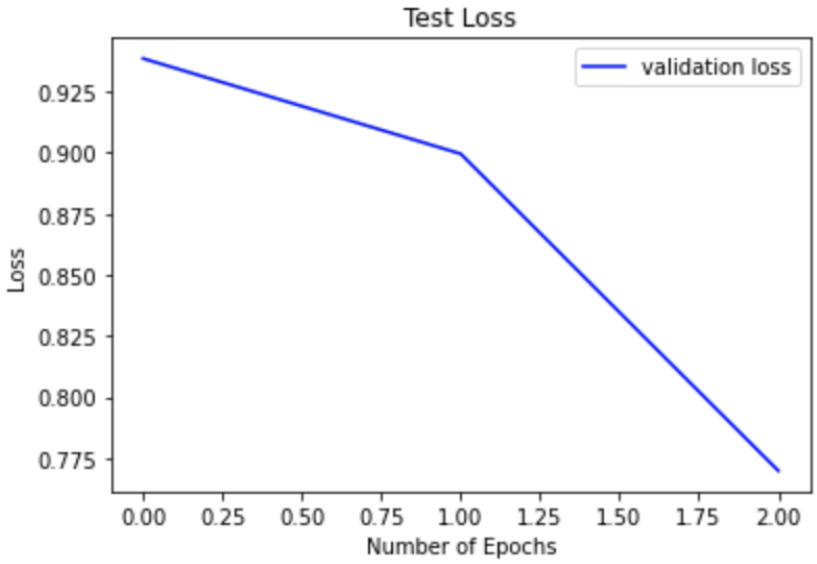
Inainte de a putea folosi hartile de legaturi, pe care le-am determinat prin folosirea celor 2 straturi mentionate anterior, trebuie sa le transformam intr-un vector unu-dimensional. Acest pas este realizat prin stratul Flatten, prin care combina toate informatiile descoperite in celelalte straturi convolutionale.

In final, am folosit 2 straturi dens conectate, care functioneaza ca un classifier si returneaza distributia claselor

Vizualizarea distributiei categoriilor datelor de training, initiale:



Plot pentru loss function dupa 3 epoci



Matricea de confuzie pentru split data