Raport proiect

1. Preprocesarea datelor

Datele pentru clasificare sunt imagini de dimensiuni (80x80x3) sau (80X80x1) de aceea pentru a aveea dimeniunea input-ului constant replicam imaginile cu un canal de 3 ori.
Pentru modelele knn, svm si nn am redimensionat datele intr-o dimensiune (19200x1)
Normalizarea este facuta impartind fiecare pixel la 255, astfel valorile noastre vor fi cuprinse intre 0 si 1.

2. Vizualizarea datelor

Am scris o functie pentru plotarea imaginilor cu acelasi label pentru a observa similaritatile intre imaginile din aceeasi clasa, dar nu am putut identifica nici o corelatie intre ele.

3. Model knn

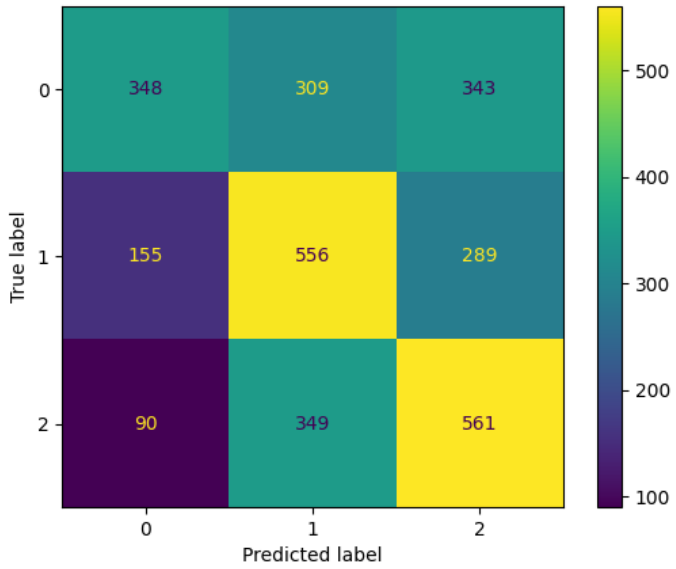
Prima incercare pentru clasificarea imaginilor a fost folosind un model knn si antrenarea lui cu diferiti hyperparametri: numarul vecinilor si algoritmul folsit.

Nr_vecini\algoritm	Auto	BallTree	KDTree	Brute
13	0.42	0.42	0.42	0.42
21	0.44	0.44	0.44	0.44
51	0.45	0.45	0.45	0.45

Algoritmmul nu prea influenteaza invatarea deci cautam cel mai bun hyperparametru prin numarul de vecini

Nr vecini	91	131	161	221	241	261
Acuratetea	0.47	0.48	0.49	0.49	0.49	0.48

Matricea de confuzie pentru un model knn cu parametrul k_neighbours = 161

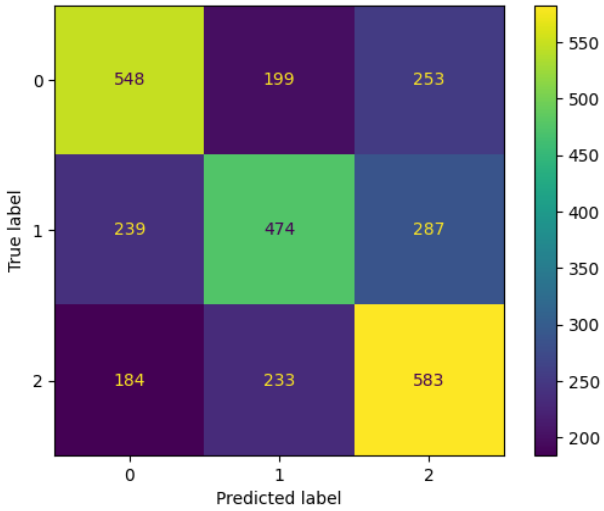


4. Model SVM

Pentru acest model l-am antrenat cu diferiti hypeparametri pentru kernel si paramentrul de regularizare C

C\Kernel	rbf	poly	linear
0.1	0.52	0.47	0.39
1	0.54	0.43	0.37
10	0.49	0.41	0.37
100	0.48	0.4	0.36

Matricea de confuzie pentru un model SVM cu parametrii kernel = rbf si c = 1

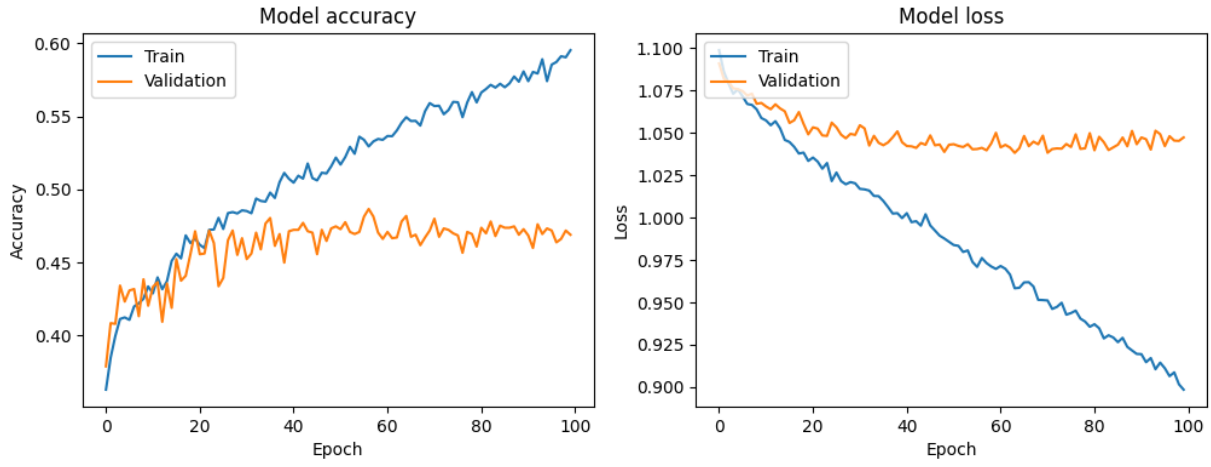


5. Model NN

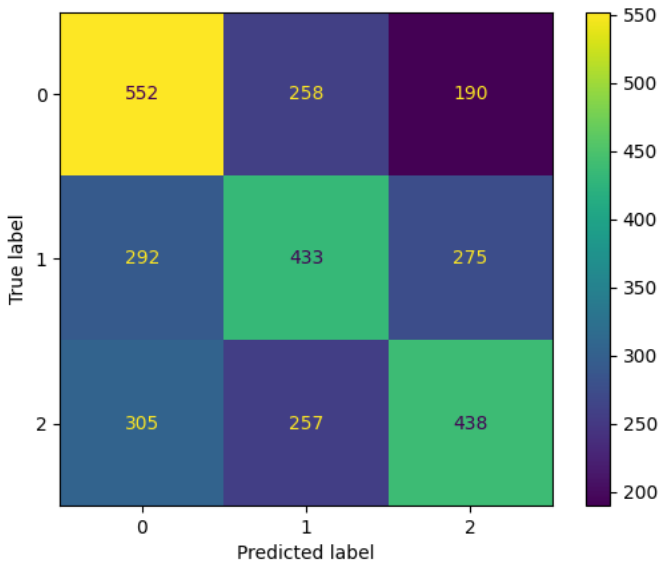
Am creat o retea neuronala cu 3 layere dense folosind libraria tenserflow, antrenand-o cu diferite rate de invatare pe un interval de 25-100 epoci

Learning rate	0.01	0.001	0.00001	0.000001
Acuratete	0.33	0.34	0.48	0.45

Acuratetea si pierderea modelului cu rata de invatare 0.00001



Matricea de confuzie:



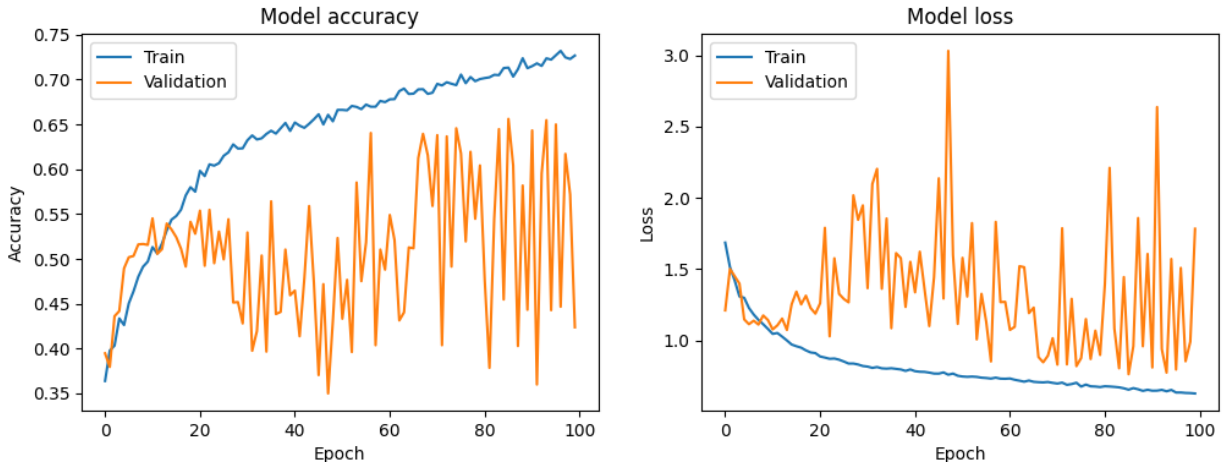
6. Model CNN

Pentru primul modelul convolutional am adaugat 3 layere convolutionale cu functia de activare relu, fiecare fiind urmat de un layer de normalizare, pooling pentru a micsora dimensiunea spatiului si layer de dropout pentru a evita overfitting-ul.

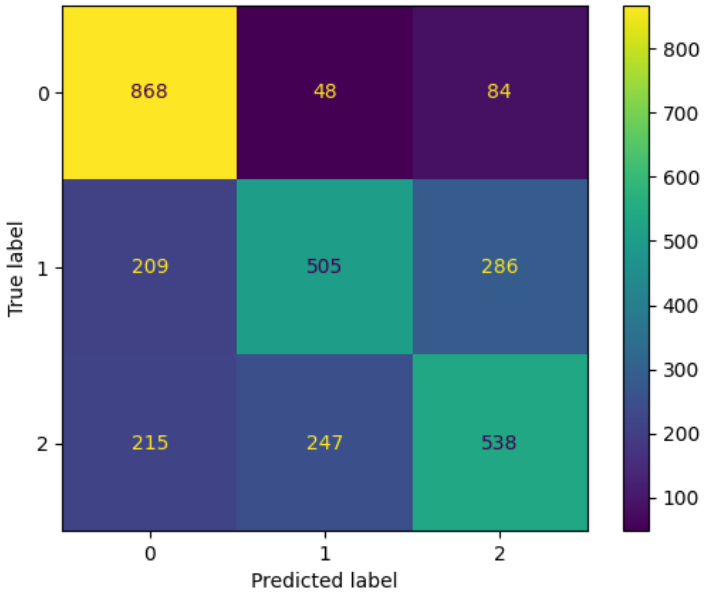
Pentru a transforma datele intrun vector 1D am folosit un layer Flatten urmat de 3 layere dense. Ultimul avand 3 neuroni si functia de activare softmax pentru a putea clasifica imaginile in 3 clase.

Acest model nu are o rata de invatare stabila, astfel acuratetea pe datele de validare fiind intre 0.5 si 0.65. Pentru a salva modelul in epoca cu acuratetea maxima pe datele de validare am folosit functia de callback model_checkpoint

Graficul de acuratete si pierdere:



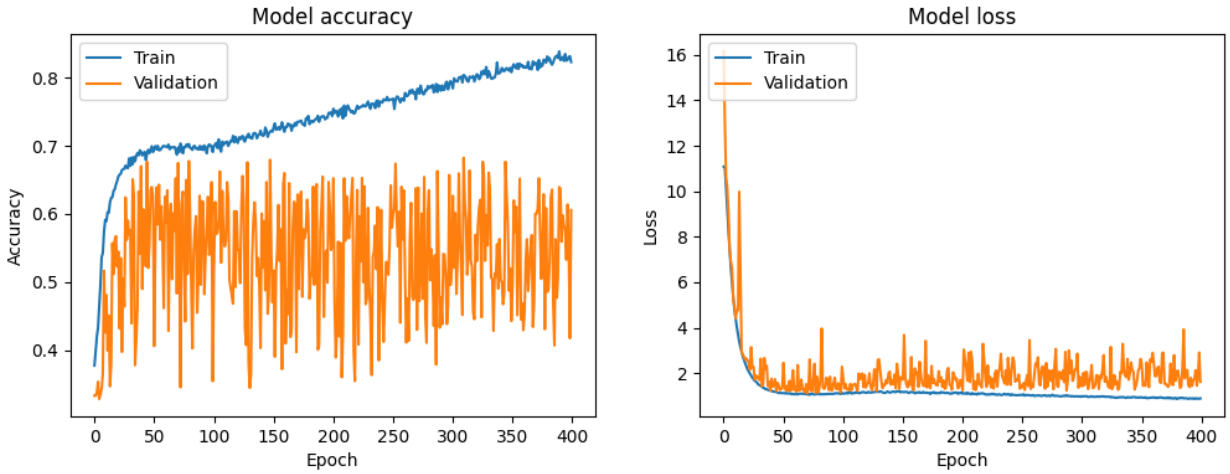
Matricea de confuzie:



Pentru urmatorul model am crescut complexitatea adaugand pana la 8 layere convolutionale si 4 layere dense, care a fost antrenat timp de 400

de epoci si cu o rata de invatare 0.0005, care e destul de mare deoarece modelul nu mentine o invatare constanta, dar ajunge la o acuratete mai mare decat cu o rata de invatare mai mica

Graficul de acuratete si pierdere:



Matricea de confuzie pentru 2 cele mai bune modele generate: