Documentație Proiect Kaggle

Introducere

Raportul acesta documentează abordările de învățare automată utilizate pentru competiția Kaggle, concentrându-se pe două modele: o rețea neuronală convoluțională(CNN) și o retea neuronala feedforward(FNN). Raportul include descrieri ale seturilor de caracteristici alese, preprocesarea datelor, reprezentarea caracteristicilor, alegerea hiperparametrilor și rezultatele modelelor.

Preprocesarea și Augmentarea Datelor

Încărcarea Datelor

Setul de date constă în imagini și etichetele lor corespunzătoare, furnizate în fișiere CSV. A fost creată o clasă de pentru prelucrarea datelor personalizată, `DataCustom`, care pregătește datele pentru load și antrenare. Această clasă mosteneste clasa `DataSet` din PyTorch, avand metoda `__getitem__` suprascrisa, pentru a putea folosi mai tarziu DataLoader.

Augmentarea și Transformarea Datelor

Augmentarea datelor a fost realizată folosind modulul `transforms` din `torchvision`. Transformările aplicate imaginilor includ redimensionarea, intoarcerea pe orizontală la întâmplare și normalizarea(de medie 0.5 și deviatie standard 1). Aceste augumentări ajută la creșterea diversității datelor de train.

Vizualizarea Datelor

Înainte de a începe sa fac primul model, mi-am plotat câteva imagini cu ajutorul bibliotecii matplotlib, pentru a mă putea familiariza cu imaginile și clasele asociate acesteia si pentru a-mi putea face o idee generala asupra intregului proiect

Model 1: Rețea Neuronală Convoluțională(CNN)

Arhitectura Modelului

Modelul ReteaConv l-am alcătuit din patru straturi convoluționale(conv), 2 straturi conectate în întregime(sct) fiecare urmat de o funcție de activare ReLU și un strat de max-pooling. După straturile convoluționale, sunt utilizate două straturi complet conectate pentru a realiza clasificarea finală.

Straturile folosesc un kernel de dimensiune 3x3, un pas (stride) de 1 și padding de 1 și sunt urmate de o operație de pooling pentru a reduce dimensiunea spațială. Ele au următoarea structură:

- Stratul 1: 3 canale de intrare și 32 de ieșire
- Stratul 2: 32 canale de intrare și 64 ieșire
- Stratul 3: 64 canale de intrare și 128 de ieșire
- Stratul 4: 128 de canale de ieșire și 256 de intrare

După aceste 4 straturi sunt 2 straturi conectate complet:

- Stratul 1: 256 * 5 *5 neuroni de intrare și 512 neuroni de ieșire
- Stratul 2: 512 neuroni de intrare și 3 de ieșire

Alegerea Hiperparametrilor

Pentru modelul CNN, au fost aleși următorii hiperparametri:

- **Rata de învățare:** 0.001
- **Funcția de pierdere:** CrossEntropyLoss
- **Optimizator:** Adam
- **Numărul de epoci:** 15
- **Dimensiunea batch-ului:** 32

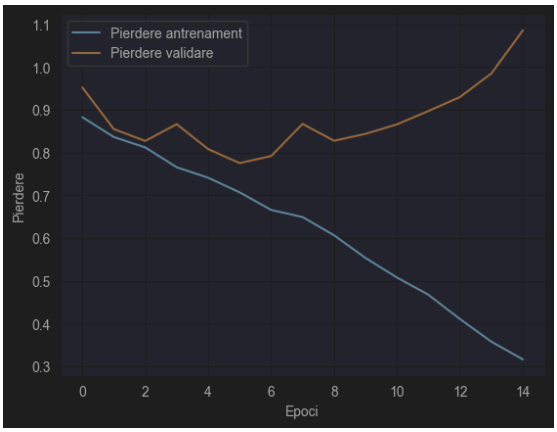
Aceste valori au fost selectate pe baza experimentelor pe care le-am făcut înainte și a celor invatate la curs, care spun că acești hiperparametri sunt eficienți pentru antrenarea rețelelor neuronale care prelucreaza poze.

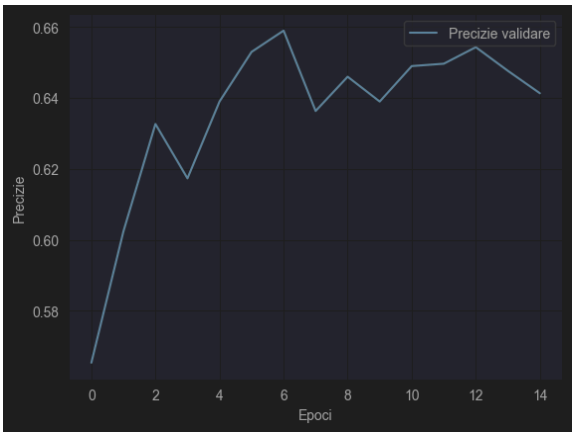
Antrenare și Validare

Modelul a fost antrenat folosind optimizatorul Adam și funcția de pierdere de entropie încrucișată. Procesul de antrenare a inclus validarea performanței modelului pe un set de validare separat după fiecare epocă.

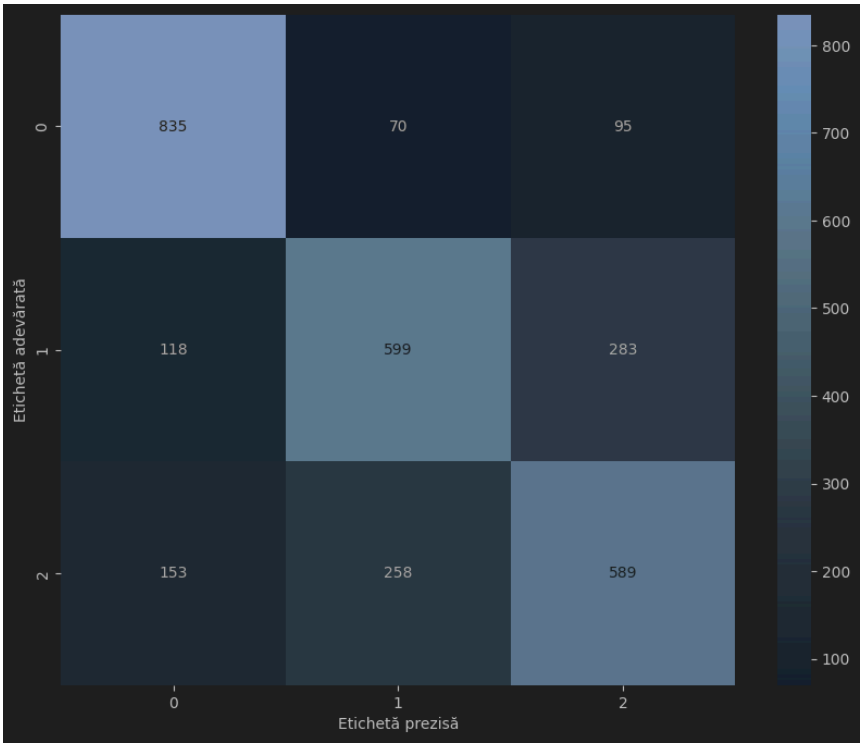
Rezultate

Cu acest model am obtinut o acuratețe de 0.66 pe datele de validare, după ce am antrenat modelul timp de 15 epoci. După cele 15 epoci acuratețea pe datele de validare scădea, iar modelul începea să facă overfit. Graficele evolutiei loss-ului și al acuratetii sunt următoarele





Matrice de Confuzie



Model 2: Alt tip Rețea Neuronală Convoluțională(CNN)

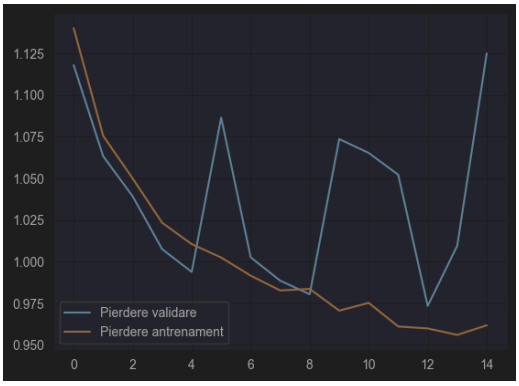
Arhitectura Modelului

Acest model este similar cu cel de mai sus, însă am încercat sa complic mai mult transformarea și straturile rețelei. Astfel, pe lângă pooling am adăugat și un strat de batch normalization pentru fiecare strat convolutional. Cat despre transformare, am modificat aleatoriu luminozitatea, contrastul, saturația imaginilor, am folosit RandomAffine pentru a roti și translata imaginile si RandomResizedCrop pentru a tăia o porțiune aleatorie a imaginii. De asemenea, în loc de 2 straturi conectate complet, am folosit 3. În rest, hiperparametrii și clasa `DataCustom` au rămas aceleași.

Rezultate

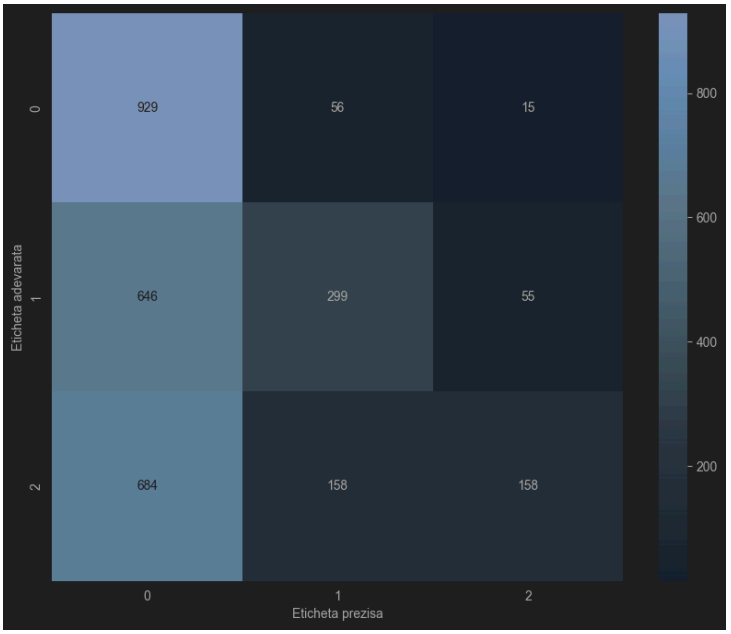
Din păcate, acest model nu a reușit să-l întrecă pe primul, obținând o acuratețe maximă pe datele de validare de 0.64 :(. Modelul a fost antrenat de la 15, pana la 100 de epoci.

Am re-antrenat modelul pentru 15 epoci, ca să pot face niște plotări:



Matricea de confuzie

Tot pentru cele 15 epoci, am obținut următoarea matrice de confuzie…



Model 3:Rețea Neuronală FeedForward(FNN)

Arhitectura Modelului

Modelul ReteaFF este un model de rețea neuronala feedforward, care seamănă cu cel studiat la curs. Am ales sa folosesc 4 straturi complet conectate, acestea avand 3 * 80 * 80, 1024, 2048 și 2048 de neuroni. Funcția de activare și optimizatorul sunt cele folosite și la celelalte modele.

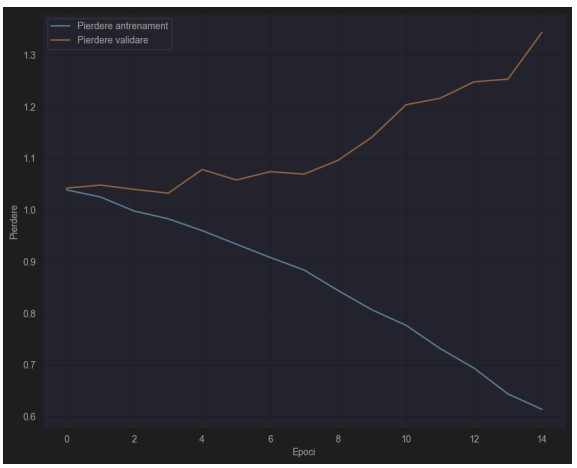
Alegerea Hiperparametrilor

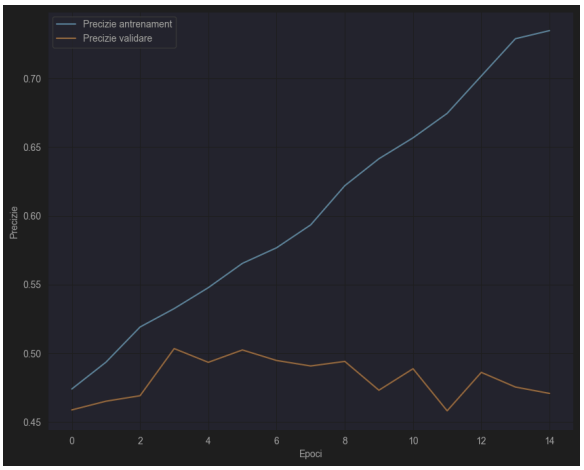
Pentru modelul RFF, au fost aleși următorii hiperparametri:

- **Rata de învățare:** 0.001
- **Funcția de pierdere:** CrossEntropyLoss
- **Optimizator:** Adam
- **Numărul de epoci:** 15
- **Dimensiunea batch-ului:** 32

Rezultate

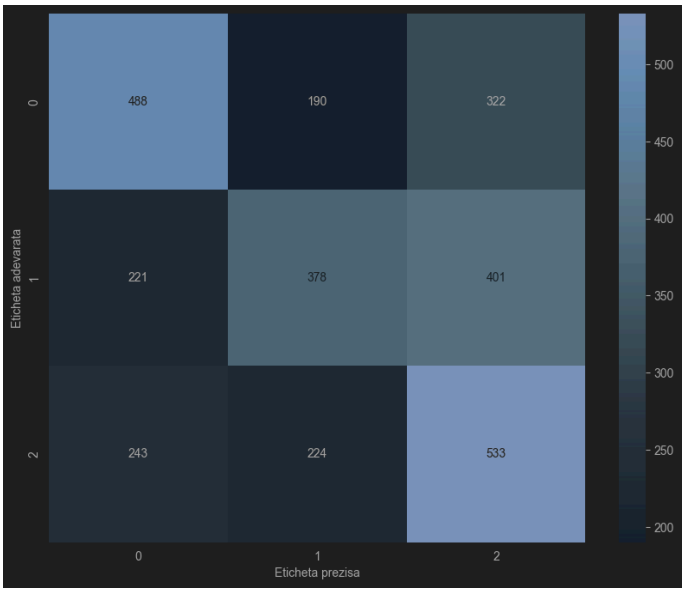
Din păcate, acest model a fost cel mai slab dintre toate cele incercate, obtinând un maxim de acuratețe de 0.54 pe datele de validare. Graficele evolutiei functiei de pierdere si al acuratetii sunt urmatoarele:





Matrice de confuzie

Matricea de confuzie pe care am obtinut-o este urmatoarea:



Încercarea numărul 2

Am încercat sa cresc numărul de straturi complet comentate și numărul de neuroni al acestora, folosind astfel 5 straturi și mărind numărul de neuroni la 8192 pe unul dintre aceste straturi. Acest lucru s-a dovedit a fi însă inutil, întrucât nu am obținut o performanță mai bune pentru datele de validare. Graficele obținute pentru acest model sunt următoarele: