Recunoașterea automată a acordurilor muzicale acustice
Proiect practic

Linca Răzvan Cosmin

Universitatea Babeș-Bolyai
Facultatea de Matematică și Informatică

2020
10 Mai

Mediu de lucru

Limbajul de programare folosit pentru realizarea proiectului practic este Python. Acesta este unul din limbajele favorite în rândul dezvoltatorilor, fiind potrivit pentru multe tipuri de aplicații. În mod particular însă, este considerat ca fiind cea mai bună alegere pentru proiectele care implică inteligența artificială. Acest lucru este datorat faptului că limbajul conține foarte multe biblioteci și framework-uri externe, care facilitează procesul de codare și economisește timpul de dezvoltare.

Învățarea automată și învățarea profundă sunt foarte bine tratate, prin numeroase librării printre acestea se numără NumPy, folosit pentru calcul științific, SciPy pentru calcule avansate și scikit-learning pentru minerirea datelor și analiza datelor, fiind printre cele mai populare biblioteci, lucrând alături de framework-uri externe puternice precum TensorFlow, Keras, CNTK sau Apache Spark.

Mediul de dezvoltare ales pentru dezvoltare a fost editorul PyCharm. PyCharm este un IDE cu caracteristici complete pentru limbajul Python, bazat pe puternica platformă Java IntelliJ Idea de la JetBrans. Nu numai că PyCharm acceptă o serie de funcții la nivel enterprise, dar este, de asemenea, o plăcere de a lucra cu el și în modul community [5]. Pentru mulți dezvoltatori în domeniul inteligenței artificiale, PyCharm este următorul pas după mediile de învățare precum notebook-urile iPython sau Google Cobab și este o alegere populară pentru dezvoltarea aplicațiilor full-stack.

Pachete și librării externe

Construcția întregului sistem a urmat o serie de etape clare, care au condus în final la construirea unui model neuronal convoluțional și punerea acestuia în funcțiune. Toate acestea ar fi fost realizate mult mai greu, dacă nu ar fi existat librăriile externe utilizate. Printre ele se numără:

- LibROSA: librărie Python pentru analiză audio. Furnizează pachetele necesare construirii unui sistem de obținere a informațiilor muzicale [6];
- TensorFlow: platformă open source de tip "end-to-end" destinată învățării automate. TensorFlow este un sistem bogat pentru gestionarea tuturor aspectelor unui sistem de învățare automată. API-urile TensorFlow sunt aranjate ierarhic, cu API-uri de nivel înalt construite pe API-uri de nivel scăzut [8]. Cercetătorii în domeniu folosesc API-uri de nivel scăzut pentru a crea și a explora noi algoritmi de învățare;
- Keras: este un API de nivel înalt, orientat pe rețele neuronale, scris în Python, fiind capabil să ruleze pe TensorFlow, CNTK sau Theano [8]. Acesta a fost dezvoltat cu focus pe realizarea experimentelor într-un mod cât mai simplu și rapid. Se recomandă Keras pentru următoarele:
  - crearea unor prototipuri simple, extrem de rapid;
  - suport pentru rețele neuronale convoluționale și pentru rețele recurente, cât și combinații între cele două;
  - rulare pe CPU sau GPU, întrucât eficiența este aceeași.
- NumPy: este biblioteca principală pentru calcul științific în Python. Oferă un obiect de tip matriceal multidimensional de înaltă performanță și instrumente de lucru cu acest tip de tablouri, printe care o colecție mare de funcții matematice [9];
- Pandas: este un pachet Python care oferă structuri de date rapide și flexibile, concepute pentru a face lucrul cu datele structurate (tabulare, multidimensionale, potențial eterogene) cât mai ușor și intuitiv [10].

Rezultate experimentale

Setul de date

Setul de date utilizat pentru antrenarea și evaluarea modelului convoluțional creat este format din combinarea a 3 seturi de date pentru chitară, disponibile online, asupra cărora s-au aplicat o serie de algoritmi de augmentare. Setul de date este astfel compus din:

- 7398 de acorduri individuale, grupate în 16 fișiere audio de tip wav, aparținând Institutului de semantică muzicală, Fraunhofer, din cadrul Universității tehnice Ilmenau, Germania. Setul de date se numește IDMT-SMT-CHORDS;
- 6580 de înregistrări audio, fiecare având în compoziție un singur acord, aparținând laboratorului de cercetări muzicale al Universității din New York, SUA, numit GuitarSet[11];
- 200 de fișiere audio pentru 10 tipuri de acorduri, fiind colectat de grupul de cercetare Motefiore al Universității din Liège, Belgia.

După aplicarea algoritmilor de augmentare asupra setului de date complet, s-a obținut un total de 59.472 de înregistrări audio de tip wav, grupate în cele 24+1 clase(24 de acorduri cu etichete corespunzătoare cunoscute, și o etichetă pentru orice alt acord care nu se încadrează).

Cele 24 de acorduri recunoscute de către sistem, împreună cu valorile numerice asociate sunt următoarele(perechi de forma acord muzical-valoare asociată): A-0, A#-1, A#m-2, Am-3, B-4, Bm-5, C-6, C#-7, C#m-8, Cm-9, D-10, D#-11, D#m-12, Dm-13, E-14, Em-15, F-16, F#-17, F#m-18, Fm-19, G-20, G#-21, G#m-22, Gm-23. Pentru orice alt acord care nu se încadrează în această înșiruire s-a creat perechea N-24 (acord necunoscut N, cu valoarea asociată de 24). Distribuția datelor pe clase este se poate observa în Figura 1.

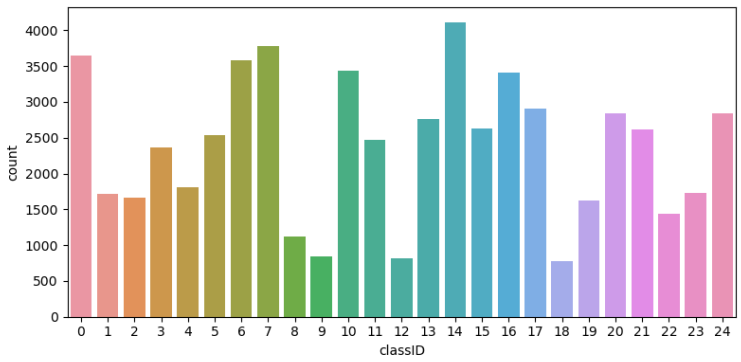

Figura 1: Distribuția datelor pe clase, prin specificarea numărului de fișiere audio disponibile pentru fiecare clasă asociată acordurilor muzicale.

Augmentarea datelor

Metodele de augmentare a datelor au rolul de a crește în mod artificial dimensiunea setului de date, generând multe variante realiste ale fiecărei instanțe set. Această procedură reduce posibilitatea de overfitting.

Instanțele generate trebuie să fie cât mai realiste, astfel încât nimeni să nu fie capabil să diferențieze între varianta originală și cea augmentată(caz ideal, în realitate vor exista aspecte care le vor diferenția) [4]. Asupra acestui set de date s-au aplicat două metode de augmentare diferite, anume:

1. Modicarea vitezei de redare, pentru fiecare instanță în parte, prin două metode:
    - Incrementarea vitezei cu 1.25% față de original;
    - Decrementarea vitezei cu 0.75% față de original.
2. Injectarea unui zgomot de fundal, pentru fiecare fișier audio, prin adăugarea unei valori generate random în vectorul de date specific.

Normalizarea datelor

Normalizarea este procesul de redimimensionare a datelor din intervalul inițial într-un alt interval prestabilit, cel mai frecvent în intervalul $[0, 1]$. Algoritmul de normalizare folosit asupra setului de date este normalizarea $L - \infty$.

Normalizarea $L - \infty$ este o strategie de normalizare prin care toate valorile unui vector de caracteristici sunt aduse în intervalul $[0, 1]$, prin împărțirea fiecărei valori la maximul acelui vector [2]. LibROSA oferă acest tip de normalizare ca fiind prestabilită pentru obținerea chromagramei, atât prin STFT, cât și prin tranformata Q constantă. De asemenea, librăria oferă prestabilită valoarea pentru marginea inferioară acceptată, numită *threshold*. Acest parametru are valoare 0, astfel orice valoarea sub 0 din vectorii caracteristici asociați chromagramei va fi ignorată și setată cu valoarea 0.

Construirea, antrenarea și evaluarea modelului convoluțional

Pentru construirea modelului convoluțional s-a folosit o secvență de straturi specializate, așezate într-o ordine bine definită pentru a spori capacitatea de învățare. Straturile utilizate sunt:

- Sequencial: Cel mai simplu model este definit folosind clasa Sequencial. Clasa definește o stivă liniară de straturi;
- ConvBlock: Definește un strat de convoluție 2D;

- BatchNormalization: Normalizarea lotului sau *batch normalization*, este o tehnică de normalizarea a activării nodurilor din straturile intermediare ale unei rețele neuronale profunde. Tendința sa de a îmbunătăți precizia și de a accelera antrenamentul, au determinat ca această tehnică să fie preferată și integrată în numeroase rețele de învățare profundă [1].

- MaxPooling: Reprezintă un strat clasic de agregare prin metoda determinării maximului din regiunea acoperită de filtru;

- Dropout: Abandonarea, cunoscută drept *dropout*, este o metodă de regularizare a rețelelor neuronale. Este o tehnică prin care anumiți neuroni selectați în mod aleator sunt ignorați în timpul antrenamentului [4]. Rezultă o rețea capabililă să generalizeze mai bine orice fel input și este mai puțin probabil ca rețeaua să se specializeze doar pentru un anumit set de date *(avoiding overfitting)*;

- Dense: Este un strat clasic de rețea neuronală complet conectată. Fiecare nod de intrare este conectat la fiecare nod de ieșire.

Astfel, cunoscând rolul fiecărui strat în parte, împreună cu funcțiile de activare ReLU și Softmax, arhitectura rețelei convoluționale construite este prezentată în Figura 2.

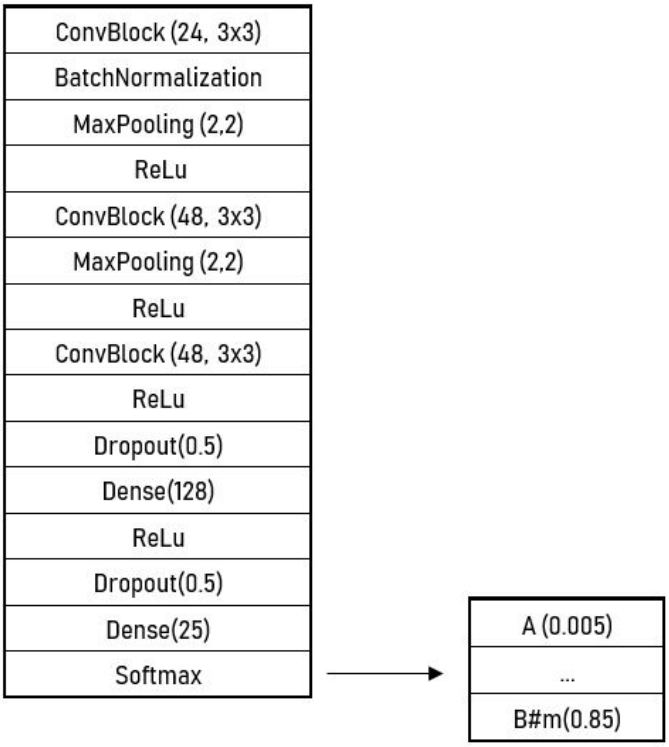

Figura 2: Arhitectura rețelei convoluționale construite. Numărul filtrelor și dimensiunea nucleului de convoluție sunt indicate între paranteze, pentru fiecare ConvBlock. Funcția de activare softmax este folosită în ultimul strat dens.

Parametrii experimentali

Cu setul de date procesat în urma extragerii trăsăturilor muzicale pentru fiecare fișier audio și modelul convoluțional construit, mai rămân de stabilit valorile unor parametrii înainte de startul procesului de învățare.

În primul rând, pentru antrenare, respectiv testare, setul de date va fi distribuit în mod aleator astfel: 80% din numărul total de instanțe vor fi utilizate pentru antrenare, în timp ce restul de 20% vor fi folosite pentru testarea modelului convoluțional.

Pentru determinarea erorii sau a valorii pentru *loss*, s-a folosit o strategie bazată pe *entropie încrucișată*. Entropia încrucișată compară predicția modelului cu valoarea etichetei corespunzătoare. Valoarea ei scade, pe măsură ce predicția devine din ce în ce mai precisă. Ea devine zero dacă predicția este perfectă [7]. Astfel, entropia încrucișată este o funcție de pierdere pentru a forma un model de clasificare.

Numărul de epoci setat este de 25. Această valoare determină de câte ori sistemul va repeta procesul de antrenare asupra datelor corespunzătoare, acesta învățând de la o etapă la alta, devenind tot mai precis, cu o eroare din ce în ce mai mică.

În ceea ce privește algoritmul de optimizare, s-a ales algoritmul de optimizare Adam, cu dimensiunea unui bloc de 64 *(batch_size)*. Adam este o metodă de adaptare a ratei de învățare, ceea ce înseamnă că rata de învățare este calculată individual pentru diferiți parametri [4].

Întrucât librăria utilizată pentru construirea modelului este Keras, se acceptă setarea ca parametru a unei instanțe pentru monitorizarea rezultatelor. Astfel, asupra modelului s-a aplicat o instanță de tip *EarlyStopping* care are rolul de a opri antrenamentul în momentul în care una valorile monitorizare a încetat să se îmbunătățească [3].

Cu toate aceste valori cunoscute, înțelese și setate corespunzător pentru rețeaua neuronală convoluțională, antrenarea poate începe cu succes. Metoda de antrenare utilizată în proiect este afișată în secțiune de cod de mai jos.

```
    import keras
    import logging
    import os

    def train(self, X_train, Y_train, X_test, Y_test):
        logger.info("Start training model")
        self.__model.compile(
            optimizer="Adam",
            loss="categorical_crossentropy",
            metrics=['accuracy', precision, recall, f1measure])

        # TensorBoard Logging
        log_dir = os.path.join(AUDIO_CONSTANTS.LOGGING_PATH, datetime.
                                            datetime.now().strftime("%Y
```

```
                                        %m%d-%H%M%S"))
        tensorboard_callback = keras.callbacks
                                    .TensorBoard(log_dir=log_dir,
                                                 histogram_freq=1)

        self.__history = self.__model.fit(
            x=X_train,
            y=Y_train,
            epochs=25,
            batch_size=64,
            validation_data=(X_test, Y_test),
            callbacks=[tensorboard_callback,
                       EarlyStopping(monitor='val_loss',
                                     verbose=1, patience=2)])
        logger.info("Training completed")
```

După finalizarea antrenării, s-a realizat evaluarea modelului neuronal convoluțional pentru a calcula valorile metricilor la antrenare, dar cel mai important la testare, asupra unor date noi, care nu au mai fost "văzute" de către sistem. Valorile lor au fost salvate prin sistemul de logging al proiectului, urmând a fi preluate din fișierul de logări pentru concluzii. Secțiunea de cod care realizează evalurea este afișată mai jos.

```
    def evaluate(self, X_train, Y_train, X_test, Y_test):
        self.score_train = self.model.evaluate(
            x=X_train,
            y=Y_train)

        self.score_test = self.model.evaluate(
            x=X_test,
            y=Y_test)

        logger.info(f'Train loss: {self.score_train[0]}')
        logger.info(f'Train accuracy: {self.score_train[1]}')
        logger.info(f'Train precision: {self.score_train[2]}')
        logger.info(f'Train recall: {self.score_train[3]}')
        logger.info(f'Train f1-score: {self.score_train[4]}')

        logger.info(f'Test loss: {self.score_test[0]}')
        logger.info(f'Test accuracy: {self.score_test[1]}')
        logger.info(f'Test precision: {self.score_test[2]}')
        logger.info(f'Test recall: {self.score_test[3]}')
        logger.info(f'Test f1-score: {self.score_test[4]}')
```

Rezultate

Metricile utilizate au fost acuratețe, precizie, recall și scorul F. Pentru a defini mai clar modul de calcul al acestor măsurători, se vor introduce următoarele variabile:

- True Positives (TP): Numărul de cazuri cu predicția DA, și valoarea reală DA;
- True Negatives (TN): Numărul de cazuri cu predicția NU, și valoarea reală NU;
- False Positives (FP): Numărul de cazuri cu predicția DA, și valoarea reală NU;
- False Negatives (FN): Numărul de cazuri cu predicția NU, și valoarea reală DA.

Metricile alese au valori relevate doar în cazul în care există un număr echilibrat de probe pentru fiecare clasă în parte. De exemplu, se consideră existența a 98% probe din clasa A și 2% eșantioane din clasa B. Modelul obținut poate obține cu ușurință acuratețea de 98% prezicând corect fiecare probă aparținând clasei A. Valoarea nu este însă una reală, modelul nefiind pregătit să recunoască instanțe ale clasei B.

În cazul setului de date utilizat această problemă nu există, fișierele audio fiind distribuite în mod echilibrat în cele 25 de clase. Cunoscând măsurătorile auxiliare și proprietatea de echilibru în distribuția datelor, metricile utilizate pot fi folosite cu succes. Acestea se definesc prin formulele de mai jos:

$$Acuratete(A) = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Recall(R) = \frac{TP}{TP + FN}$$

$$Precizie(P) = \frac{TP}{TP + FP}$$

$$Scor - F = \frac{2 \cdot R \cdot P}{R + P}$$

Aceste metrici au fost aplicate la compilarea modelului neuronal convoluțional. Rezultatele obținute sunt prezentate în Tabelul 5.1.

Etapă	Acuratețe	Precizie	Recall	Scor-F
Antrenare	97.23%	98.38%	95.85%	97.05%
Testare	89.75%	93.66%	86.67%	89.98%

Tabelul 1: Rezultate obținute, în urma celor două etape: antrenare și testare.

Analiză

Pentru a observa evoluția proceselor de antrenare și testare, și impactul întregului proces asupra metricilor de calcul, s-au realizat două grafice pentru a evidenția scăderea loss-ului și creșterea acurateții. Astfel, Figura 3 prezintă valorile pe care le-a avut eroarea pe parcursul celor 25 de epoci. Se observă o scădere relativ exponențială, valorile loss-ului la antrenare, respectiv la testare fiind apropiate. Figura 4 prezintă valorile pentru acuratețe, fiind evidențiat un trend ascendent, valorile fiind din nou apropiate spre ultimele epoci.

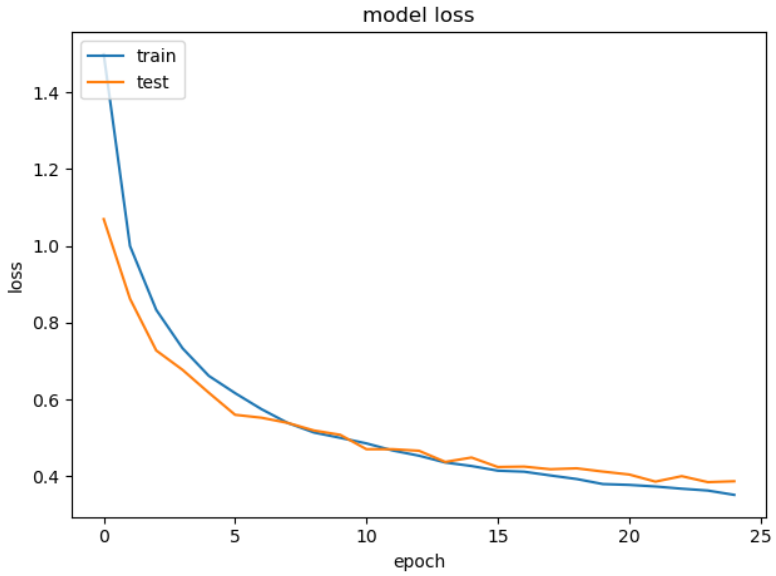

Figura 3: Evoluția valorii pentru loss, la antrenare și testare.

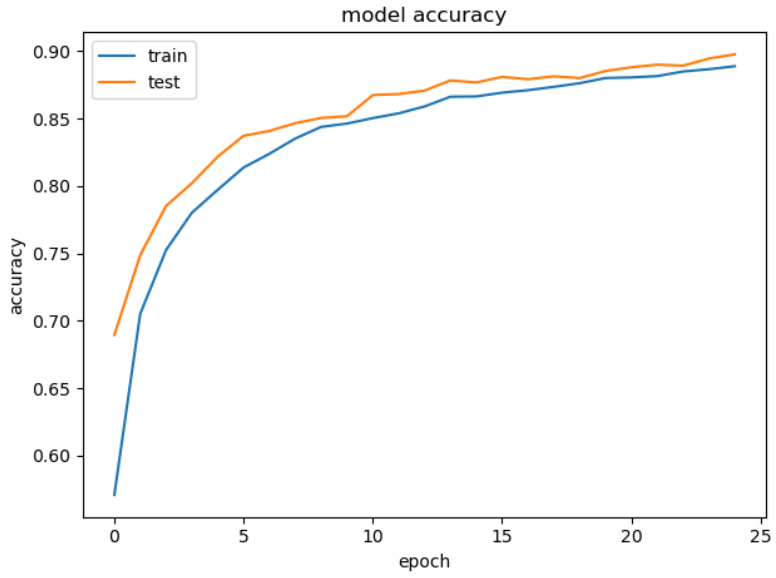

Figura 4: Evoluția valorii pentru acuratețe, la antrenare și testare.

Bibliografie

[1] Johan Bjorck et al. "Understanding Batch Normalization". In: *Cornell University* (2018).

[2] Alessandro Bonvini. "Automatic chord recognition using Deep Learning techniques". In: *Journal: POLITECNICO DI MILANO Facoltà di Ingegneria dell'Informazione Corso di Laurea Magistrale in Ingegneria e Design del suono Dipartimento di Elettronica e Informazione* (2012-2013).

[3] *EarlyStopping*. https://keras.io/api/callbacks/early_stopping/. Accesat în 2020-05-10.

[4] Aurélien Géron. *Hands-on Machine Learning with Scikit-Learn, Keras and TensorFlow*. O'Reilly Media, ISBN-13: 9781492032649, 2019.

[5] *Guide to PyCharm for Windows and MacOS*. https://kite.com/blog/python/pycharm/. Accesat în 2020-04-26.

[6] *LibROSA*. https://librosa.github.io/librosa/index.html. Accesat în 2020-04-06.

[7] *Loss Functions*. https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html. Accesat în 2020-05-10.

[8] *Machine Learning Crash Course*. https://developers.google.com/machine-learning/crash-course. Accesat în 2020-05-02.

[9] *NumPy*. https://numpy.org/. Accesat în 2020-05-02.

[10] *Pandas 1.0.3*. https://pypi.org/project/pandas/. Accesat în 2020-05-02.

[11] J. Pauwels et al. "Guitarset, A Dataset for Guitar Transcription". In: *19th International Society for Music Information Retrieval Conference, Paris, France* (2018).