Universitatea Tehnică „Gheorghe Asachi”, Iaşi
Facultatea de Automatică şi Calculatoare

Raport II

Clasificarea pieselor muzicale în genuri folosind rețele neuronale de tip DeepLearning

Profesor coordonator:
Șef lucrări Dr. Marius Gavrilescu

Student:
Muraru Ștefan

Scopul proiectului

Extragerea sunetelor, în special a pieselor muzicale, din baze de date este un domeniu în continuă creştere, iar seturile mari de date reprezintă o provocare în procesele de căutare, preluare şi organizare a muzicii.

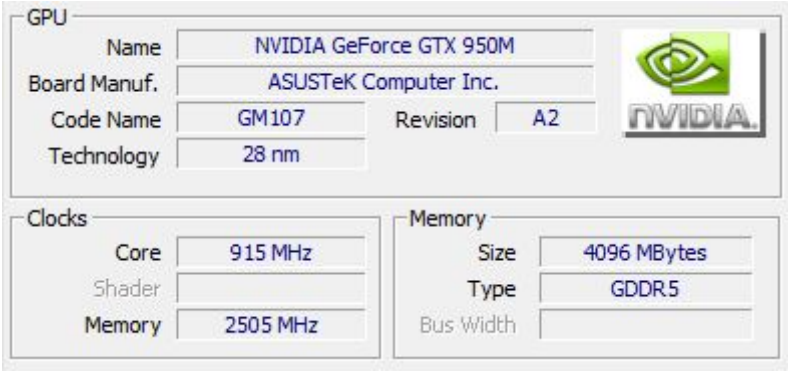
În acest sens, un sistem de clasificare a pieselor muzicale în genuri poate fi extrem de folositor în etapa de organizare a datelor muzicale în clase în funcţie de anumite trăsături ale acestora. Totodată, un astfel de sistem ar veni şi în ajutorul persoanelor ce deţin colecţii mari de piese muzicale şi îşi doresc o mai bună organizare a acestora, dar nu dispun de timpul necesar clasificării manuale a acestora.

Lucrarea îşi propune găsirea unui model şi antrenarea unei reţele neuronale care să permită o acurateţe cât mai mare în procesul de clasificare a melodiilor.

Proiectarea aplicației

Componente hardware folosite

- Placă video Nvidia GTX 950M folosită pentru antrenarea reţelei neuronale ce are ca scop clasificarea pieselor muzicale în genuri.



Componente software folosite

- Anaconda (sistem de management a dependinţelor folosit în cazul de faţă pentru crearea unui mediu de lucru cu toate pachetele de mai jos)
- Python v3.6 (limbajul de programare folosit)
- Tensorflow GPU-version v1.8 (bibliotecă Python scrisa de Google prin intermediul căreia se crează, antrenează şi testează modelul reţelei neuronale)
- Keras (API de nivel înalt ce funcţionează ca wrapper peste Tensorflow, reducând complexitatea codului utilizat în scopul acestui proiect)

- Numpy (bibliotecă Python ce facilitează operaţiile cu matrici de una sau mai multe dimensiuni)
- scikit-learn (bibliotecă Python ce implementează mai mulţi algoritmi folosiţi în domeniul învăţării automate, utilizată în cadrul acestui proiect pentru calcularea acurateţii după fiecare perioadă de antrenare, a matricei de confuzie cât şi pentru divizarea setului de date la începutul antrenării)
- matplotlib (bibliotecă Python folosită pentru afişarea graficului evoluţiei acurateţii odată cu trecerea epocilor de antrenare)

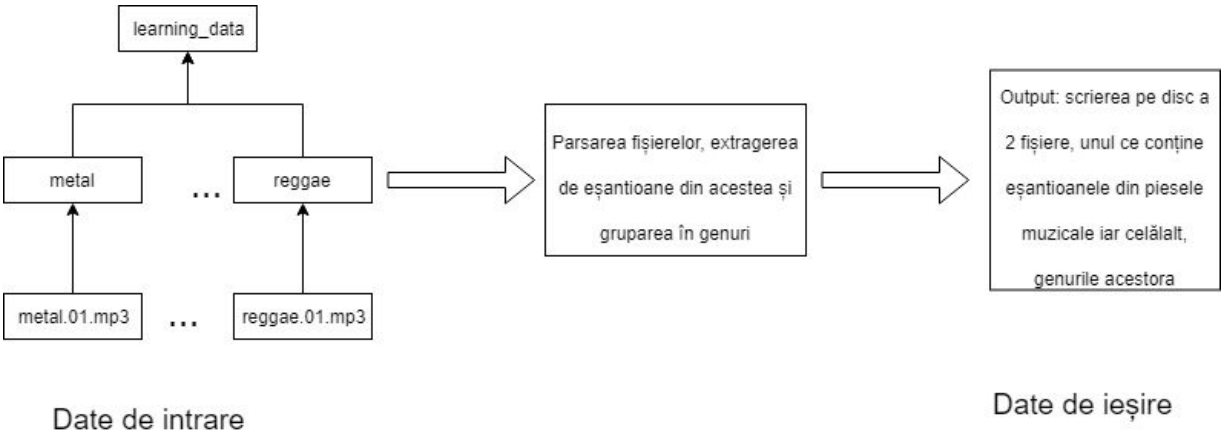
```
import os
import gc
import numpy
import keras

import matplotlib.pyplot as pyplot
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split
```

Rezultate intermediare obținute

În urma studiului asupra lucrărilor deja existente s-a ajuns la concluzia că o reţea neuronală convoluţională combinată cu spectrograma Mel va aduce rezultatele cele mai promiţătoare. Astfel, modelul reţelei neuronale ce oferă acurateţea maximă în momentul de faţă este compus din 3 straturi convoluţionale ascunse (Conv1D) la ieşirea cărora sunt aplicate operaţii de MaxPooling şi/sau MeanPooling ce au ca scop reducerea dimensiunii datelor de intrare, urmând 2 straturi ce aplică funcţia de activare „relu”, iar la sfârşit un strat ce aplică funcţia de activare „softmax”.

Aplicaţia este constituită din trei module, după cum urmează:

- **createAudioModels.py** - script ce, la rulare, va încărca pe rând fiecare din fişierele audio din directorul „learning_data”, va parcurge structura de directoare ce reprezintă genurile pieselor muzicale şi va extrage din toate acestea eşantioane ce vor fi folosite în procesul de antrenare al modelului reţelei neuronale pe care în cele din urmă le va concatena, având ca ieşire 2 fişiere, „models\songs_data.npy” şi „models\genres_data.npy”. Totodată, trebuie menţionat că s-a încercat o utilizare cât mai eficientă a resurselor hardware, şi anume a procesorului, procesarea pieselor muzicale date ca intrări făcându-se în mod paralel pe fiecare nucleu disponibil al acestuia.



- **train.py** - modul ce conţine crearea propriu-zisă a modelului reţelei neuronale, încărcarea datelor extrase la pasul anterior, antrenarea acesteia într-un număr dat de perioade şi la final, generarea unei vizualizări asupra evoluţiei acurateţii modelului.

```
for execPeriod in range(ExecTimes):
  # Split the soundwave dateset into training and test
  inputs_train, inputs_test, outputs_train, outputs_test = train_test_split(···

  # Split the training dataset into training and validation
  inputs_train, inputs_validation, outputs_train, outputs_validation = train_test_split(···

  # Split the train, test and validation datasets in sizes of 128x128
  inputs_validation, outputs_validation = split_songs(···
  inputs_test, outputs_test = split_songs(···
  inputs_train, outputs_train = split_songs(···

  model = TrainingModels.CNNMelspec(input_shape)

  print("Training dataset shape: {0}".format(inputs_train.shape))
  print("Validation dataset shape: {0}".format(inputs_validation.shape))
  print("Test dataset shape: {0}\n".format(inputs_test.shape))
  print("CNN size: %s\n" % model.count_params( ))

  # optimizer used for compiling the model
  sgd = keras.optimizers.SGD(lr = 0.001, momentum = 0.9, decay = 1e-5, nesterov = True)

  # Compile the model using the SGD optimizer
  model.compile(···
  )

  # Callback used to stop epoch training if the loss is small enough
  stop_callback = keras.callbacks.EarlyStopping(···
  )

  # Fit the model
  history = model.fit(···

  # Start training and then validate the given number of epochs in one execution period
  score = model.evaluate(···
  )
  score_validation = model.evaluate(···
  )
```

Figura de mai sus reprezintă o vizualizare asupra paşilor executaţi în procesul de antrenare. Se pot observa etapele de ajustare şi împărţire a setului de date de intrare, de compilare a modelului şi de antrenare propriu-zisă.

- **test.py** - scriptul folosit pentru testarea modelului antrenat cu piese muzicale aleatorii, altele decât cele date ca intrare pasului anterior, aparţinând sau nu genurilor muzicale pe care a fost antrenată reţeaua. Scriptul are ca scop prelevarea de eşantioane din piesa muzicală dată, evaluarea acestora pe baza modelului antrenat şi afişarea rezultatelor.

```
Model loaded!
5/5 [==============================] - 2s 465ms/step
[0.8 0.  0.2 0. ]
```

În cazurile favorabile, modelul va reuşi să clasifice corect piesa muzicală în genul corespunzător, matricea de ieşire având ponderea cu valoarea cea mai mare pe indexul

genului prezis. Așa cum se poate observa în poza de mai sus, modelul clasifică corect eșantioane din piesa rock-metal dată ca intrare, genul acesta fiind pe prima poziție.
În cazurile defavorabile în care forma de undă a piesei muzicale diferă suficient de mult de cea a datelor de antrenare ale genului respectiv, vor apărea erori de clasificare. În imaginea de mai jos, modelul este testat cu o piesă muzicală aparținând baladelor rock, aceasta fiind incorect clasificată ca muzică clasică.

```
Model loaded!
5/5 [==============================] - 1s 256ms/step
[0.1 0.1 0.1 0.7]
```

Dificultăți întâmpinate

Printre dificultățile întâmpinate până în momentul de față se numără:

- Găsirea unui set concret de caracteristici utile în clasificarea corectă a fiecărui gen muzical,
- Obținerea de piese aparținând mai multor genuri, din cauza drepturilor de autor,
- Durata mare de procesare pentru extragerea eșantioanelor și dimensiunile mari ale setului de date de intrare,
- Volumul de memorie RAM necesar în majoritatea etapelor de procesare și antrenare, datorat dimensiunilor setului de date de antrenare,
- Capacitatea de procesare redusă a plăcii video de pe laptopul personal și nu în ultimul rând,
- Dificultăți în instalarea corectă și completă a unui mediu de lucru (tensorflow-gpu).

Următorii pași

- Încercarea de testare și optimizare a rețelei pentru un număr mai mare de genuri muzicale concomitente,
- Crearea unei interfețe pentru facilitarea testării modelului.