Classicaly Punk

Music Genre Classification using Convolutional Neural Network

Introduction

Importance of Data Science for music recommandation system

Data collection and exploration

For this project we use a dataset with 1000 audio files. The dataset contain 10 differents genres and each genre have 100 samples.

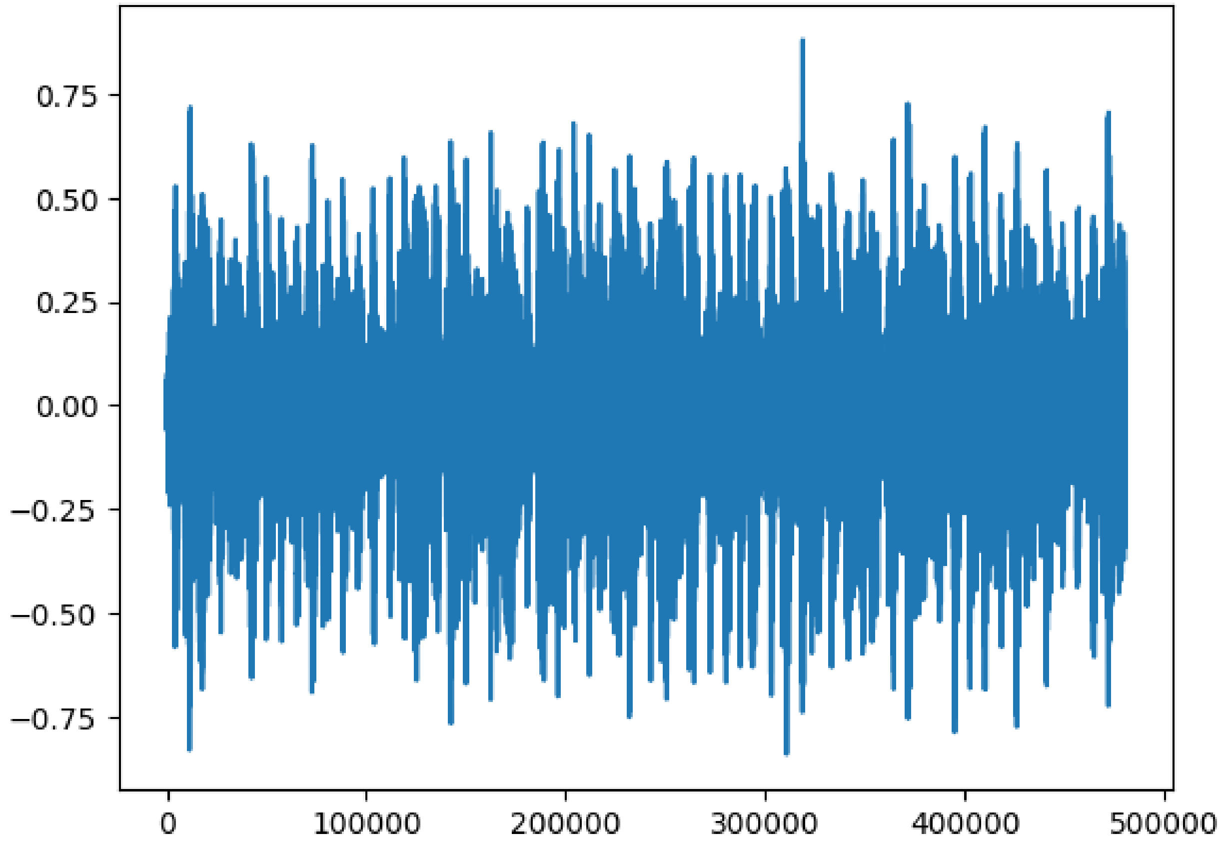
We create a tensorflow dataset using the path to each of the 1000 audio files :

```
Processing genre: reggae, Path: /content/drive/MyDrive/Qwasar/Classically Punk/Data_classically_punk_music_genres/genres/reggae
Processing genre: disco, Path: /content/drive/MyDrive/Qwasar/Classically Punk/Data_classically_punk_music_genres/genres/disco
Processing genre: jazz, Path: /content/drive/MyDrive/Qwasar/Classically Punk/Data_classically_punk_music_genres/genres/jazz
Processing genre: rock, Path: /content/drive/MyDrive/Qwasar/Classically Punk/Data_classically_punk_music_genres/genres/rock
Processing genre: hiphop, Path: /content/drive/MyDrive/Qwasar/Classically Punk/Data_classically_punk_music_genres/genres/hiphop
Processing genre: pop, Path: /content/drive/MyDrive/Qwasar/Classically Punk/Data_classically_punk_music_genres/genres/pop
Processing genre: metal, Path: /content/drive/MyDrive/Qwasar/Classically Punk/Data_classically_punk_music_genres/genres/metal
Processing genre: country, Path: /content/drive/MyDrive/Qwasar/Classically Punk/Data_classically_punk_music_genres/genres/country
Processing genre: classical, Path: /content/drive/MyDrive/Qwasar/Classically Punk/Data_classically_punk_music_genres/genres/classical
Processing genre: blues, Path: /content/drive/MyDrive/Qwasar/Classically Punk/Data_classically_punk_music_genres/genres/blues
(<tf.Tensor: shape=(), dtype=string, numpy=b'/content/drive/MyDrive/Qwasar/Classically Punk/Data_classically_punk_music_genres/genres/reggae/reggae.00006.wav'>, <tf.Tensor: shape=(
(<tf.Tensor: shape=(), dtype=string, numpy=b'/content/drive/MyDrive/Qwasar/Classically Punk/Data_classically_punk_music_genres/genres/reggae/reggae.00010.wav'>, <tf.Tensor: shape=(
(<tf.Tensor: shape=(), dtype=string, numpy=b'/content/drive/MyDrive/Qwasar/Classically Punk/Data_classically_punk_music_genres/genres/reggae/reggae.00002.wav'>, <tf.Tensor: shape=(
(<tf.Tensor: shape=(), dtype=string, numpy=b'/content/drive/MyDrive/Qwasar/Classically Punk/Data_classically_punk_music_genres/genres/reggae/reggae.00003.wav'>, <tf.Tensor: shape=(
(<tf.Tensor: shape=(), dtype=string, numpy=b'/content/drive/MyDrive/Qwasar/Classically Punk/Data_classically_punk_music_genres/genres/reggae/reggae.00001.wav'>, <tf.Tensor: shape=(
```

Data collection and exploration

For this project we use a dataset with 1000 audio files. The dataset contain 10 differents genres and each genre have 100 samples.

We read a sample of audio file using librosa library , and then plot the wav format the audio file.



Date Preprocessing

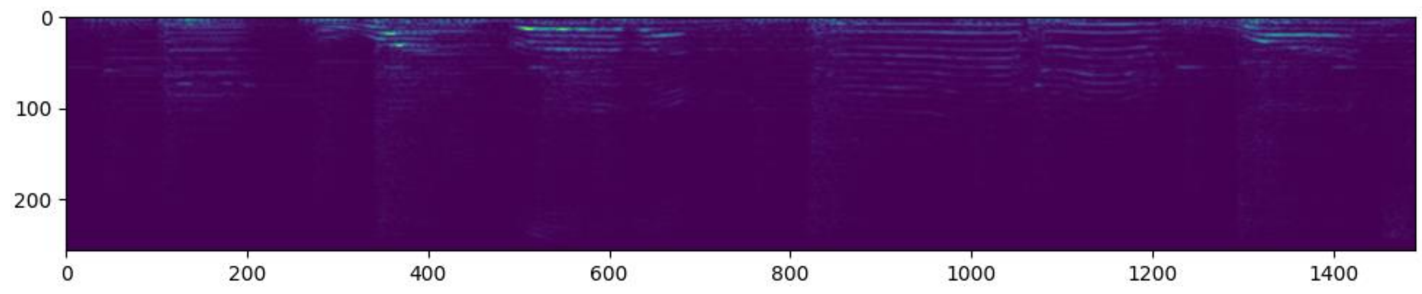
Audio file to Image

Converting Audio file to Spectrogram

Each of our dataset audio file were giving to this function that convert them to a Spectrogram.

The spectrogram is an image representation of our audio that can be passed to our CNN model for classification

```
def process_file(file_path, label, duration=3):
    def load_and_process(file_path):
        # Convert the file path tensor to a string
        file_path = file_path.numpy().decode('utf-8')

        # Load the audio file to wave format
        wav = load_wav_mono(file_path, duration)

        # Convert to spectrogram
        spectrogram = tf.signal.stft(wav, frame_length=320, frame_step=32)
        spectrogram = tf.abs(spectrogram)
        spectrogram = tf.expand_dims(spectrogram, axis=2)

        return spectrogram

    # Apply tf.py_function
    spectrogram = tf.py_function(load_and_process, [file_path], tf.float32)
    spectrogram.set_shape([None, None, 1])  # Adjust the shape if necessary

    return spectrogram, label

def tf_process_file(file_path, label):
    spectrogram, label = tf.py_function(process_file, [file_path, label], [tf.float32, tf.int32])
    spectrogram.set_shape([None, None, 1])  # Adjust the shape if necessary
    label.set_shape([])
    return spectrogram, label
```



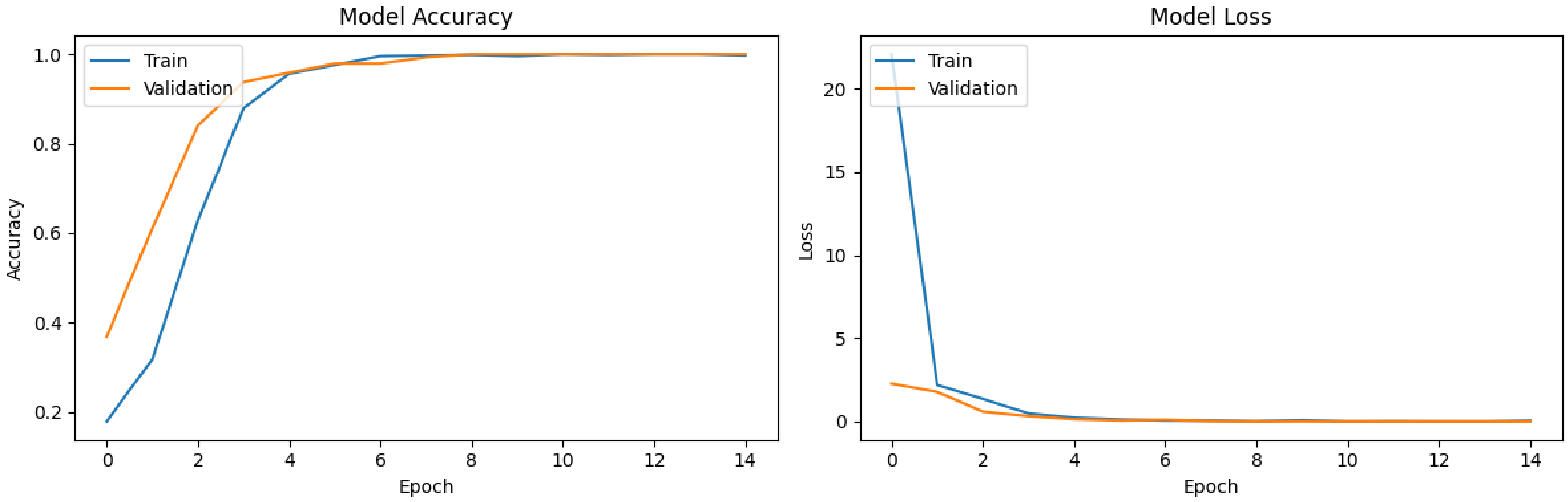
Machine Learning Model

Convolutional Neural Network

CNN

Model Performance

Plot of our Model traning and validation metrics: Accuracy and Loss



- **Model Accuracy**

The accuracy curve informes us about how well our model predict the right class correctly. In short it gave us an idea on our model learning performance over time. The most higher our accurracy goes the most performant our model is.

In fact, both of our traning accuracy and validation goes higher as the traning time increase. They respectively stared at 20% and 40% and reach 100% around the 10th epochs.

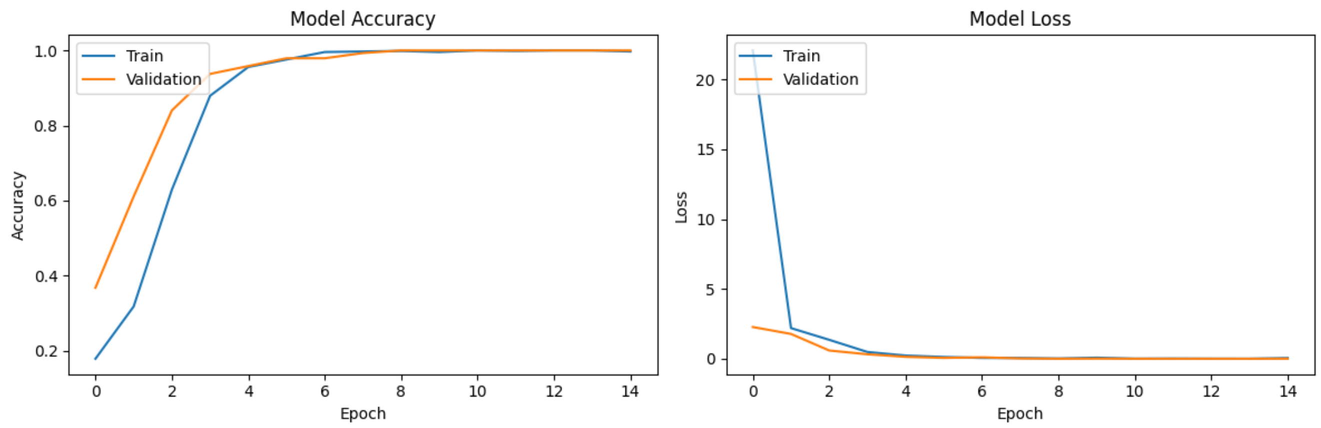
- **Model Loss**

This metrics informes us about how our model makes error during traning and validation period. The more this value is close to zero the better our model is.

We notice that our train and validation Loss curve starts higher and decrease progressivly and get more and more closer to zero which means our model learn correctly and improve himself over epochs resulting of a lower loss.

Model Performance

Plot of our Model traning and validation metrics: Accuracy and Loss



Model Evaluation on never seen data

```
[ ] # Evaluate the model
    test_loss, test_acc = model.evaluate(test)
    print(f'Test accuracy: {test_acc}')

    10/10 [==============================] - 1s 119ms/step - loss: 0.0090 - accuracy: 1.0000
    Test accuracy: 1.0
```

- **Model Accuracy**

The accuracy curve informes us about how well our model predict the right class correctly. In short it gave us an idea on our model learning performance over time. The most higher our accurracy goes the most performant our model is.

In fact, both of our traning accuracy and validation goes higher as the traning time increase. They respectively stared at 20% and 40% and reach 100% around the 10th epochs.

- **Model Loss**

This metrics informes us about how our model makes error during traning and validation period. The more this value is close to zero the better our model is.

We notice that our train and validation Loss curve starts higher and decrease progressivly and get more and more closer to zero which means our model learn correctly and improve himself over epochs resulting of a lower loss.

- **Model Evaluation**

The above interpretation are confirmed by the model Evalution.

On new data never seen by the model before we get 100% accuracy un audio genre prediction. This stand as a proof that our model generalize well on data

COMMUNICATION

Conclusion

- Overall, we perform Music Genre Classification by developping a Deep learning model able to identify pattern in audi

- With 98+% of accuracy and also very close to 0,   our model is good at predicting the music genres.

On top of that 100% of accuracy is observed during evaluation on data that the model never see before.

To wrap-up , our model is very performant but can be finetune when it comes to deal with more large and complex dataset.