Group Project - ResNet 1D CNN

of Numerical Analysis for Machine Learning Course
(NAML)
held by
Miglio Edie
Regazzoni Francesco

Group 5

Donati Riccardo
`10669618`

Borsatto Andrea
`10628989`

Academic year 2021/2022

POLITECNICO
MILANO 1863

Contents

1 Problem Specification

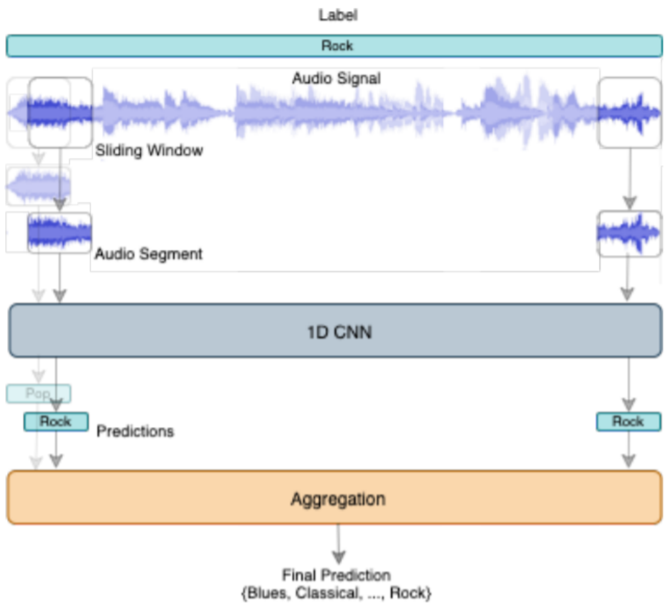
The problem was about genre classification of a song, using an Artificial Neural Network. In recent years music genre recognition has become a popular problem because various app that offer streaming music or other services related to music tend to give a personalized experience, where the application can suggest songs similar to the ones listened by the user, or where songs can be automatically tagged by genre and grouped together.
In our case, for the problem we were given a dataset of 1000 songs with a length of 30 seconds belonging to 10 different musical genres (reggae, rock, country, disco, hip hop, classical, metal, blues, jazz and pop). The Neural Network, given a song, has to tag it with the correct musical genre among the ones mentioned.

2 Approach

Our approach followed the one used in the paper *1D CNN Architectures for Music Genre Classification* by Safaa Allamy and Alessandro L. Koerich.[1] This approach uses a **1D Residual Convolutional Neural Network** for the model of the ANN and it uses a sliding window approach for the sampling of the songs.

The 1D residual CNN uses 1D Convolutional Layers and Residual Layers alternated with MaxPooling Layers to extract features and learn directly from the raw audio signal. The sliding windows approach is used for the sampling: the audio signal given in input is split into overlapping segments of fixed length (5 seconds); every segment is fed to the Neural Network that returns a prediction for that part of the song. The prediction of the genre of the entire song is done aggregating the predictions of all the overlapping segment.

The approach can be summarized with this image:



3 Dataset

We used the GTzan, this dataset was used for the well known paper in genre classification *Musical genre classification of audio signals* by G. Tzanetakis and P. Cook[4] and consists in 1000 different songs with a duration of 30 seconds. The songs are distributed evenly among the 10 musical genres: reggae, rock, country, disco, hip hop, classical, metal, blues, jazz and pop.
We decided to split the dataset in 6 partitions, in order to be able to handle the whole dataset; in each partition we have the same amount of song for each genre. Each song has a frequency rate of 22’050 Hz that results in a total of 661’500 input elements for a song of 30 seconds. We then splitted each song in 21 segments of 5 seconds with an overlapping of the 75%.
We ended up with a final input of 110’250 elements.
We had to truncate to 20 segments some songs because they were slightly shorter than 30 seconds in order to have the correct input for the neural network.

3.1 Problems of the dataset

As we can read from the article *An Analysis of the GTZAN Music Genre Dataset* of Bob L. Sturm[5], we find in GTZAN:

- 7.2% of the excerpts come from the same recording (including 5% duplicated exactly);

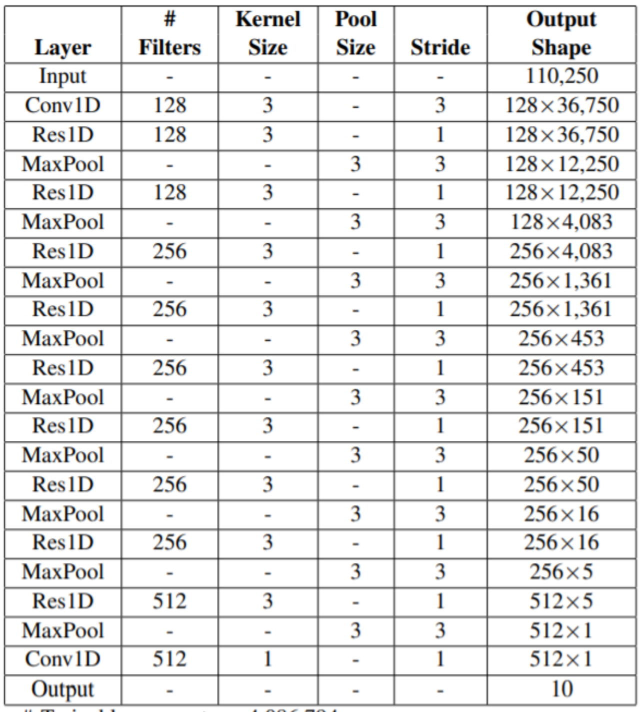
- 10.6% of the dataset is mislabeled;
- Distortion significantly degrades only one excerpt.

Downstream of these considerations we can proceed with the development of a solution.

4 Architecture

4.1 Convolutional Neural Network (CNN)

The ANN is a 1D Residual Convolutional Neural Network where the structure follows the one illustrated in the following table:



Layer	# Filters	Kernel Size	Pool Size	Stride	Output Shape
Input	-	-	-	-	110,250
Conv1D	128	3	-	3	128×36,750
Res1D	128	3	-	1	128×36,750
MaxPool	-	-	3	3	128×12,250
Res1D	128	3	-	1	128×12,250
MaxPool	-	-	3	3	128×4,083
Res1D	256	3	-	1	256×4,083
MaxPool	-	-	3	3	256×1,361
Res1D	256	3	-	1	256×1,361
MaxPool	-	-	3	3	256×453
Res1D	256	3	-	1	256×453
MaxPool	-	-	3	3	256×151
Res1D	256	3	-	1	256×151
MaxPool	-	-	3	3	256×50
Res1D	256	3	-	1	256×50
MaxPool	-	-	3	3	256×16
Res1D	256	3	-	1	256×16
MaxPool	-	-	3	3	256×5
Res1D	512	3	-	1	512×5
MaxPool	-	-	3	3	512×1
Conv1D	512	1	-	1	512×1
Output	-	-	-	-	10

Trainable parameters: 4,086,794.

The network has an input layer with 110'250 neurons, obtained sampling the signal of the song at 22'050 Hz for 5 seconds (the length of the sliding window).

After the Input Layer a first Convolutional1D layer is applied and after it the model is composed by a sequence of Residual1D Layers alternated with Max Pooling Layers with the parameters showed in the table above.

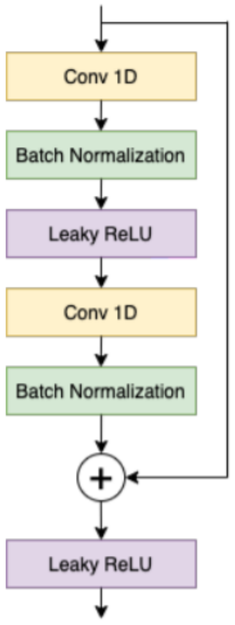
The Residual Layer is used to improve the learning ability of deep neural networks; the Convolutional Layers are used to extract and recognize features in the dataset while MaxPooling Layers are used to generalize the ability of the network to recognize those features.

After all the Residual Layers and MaxPooling Layers the network has the last Convolutional1D Layer before the output.

The Output Layer has 10 neurons that represent the 10 different music genres considered in the problem; the neurons have a "Softmax" activation function that is the most used function for the output layer in multi class classification problems and that converts the results of the network into the probability of membership for each class.

4.2 ResNet

The Residual1D Layer has the structure showed in the following picture:



The Residual Layer is used to overcome the problem of deep neural networks, where increasing too much the number of layers tends to degrade the accuracy and makes the network unable to learn simple functions.

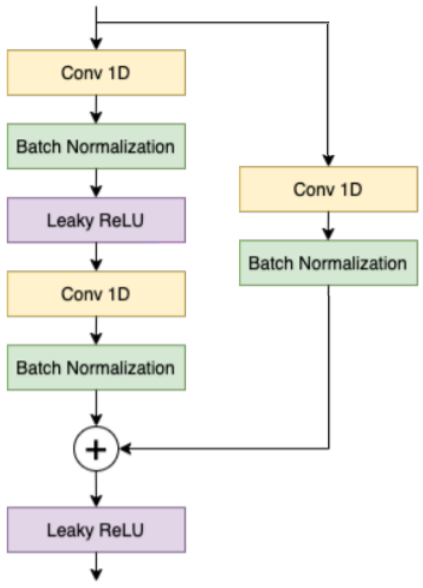
The Residual Block solves this by skipping some layers and adding the input of the block to their result; in our case the skipped layers were two Convolutional Layers, two Batch Normalizations (used to keep the mean of the data close to zero and the standard deviation of the data close to one) and a Leaky Relu. At this point the input of the Residual Block is added to the result of the last Batch Normalization and finally is applied a Leaky Relu to add a non-linearity before the application of the MaxPooling Layer.

5 Choices

We tried to follow the paper[1] as much as possible but we had to make some design choices due to lack of information and memory limitations.

5.1 Decisions

- We used the Adam optimizer with sparse categorical cross entropy loss and sparse categorical cross entropy accuracy.
- We implemented the early stopping with patience equals to 5.
- Some songs are slightly shorter than others of a few seconds so we decided to truncate the last segment of these songs to have only segments with the correct length of 5 seconds, as required also by the network.
- Shortcut in Residual Block: in the residual layer we have situations in which the input and the result of the second batch normalization have different shapes because the Convolutional Layers change the number of filters with respect to the input; so the two values can't be summed together. In these cases we decided to apply a Conv1D Layer and a Batch Normalization also to the input to give it the same shape of the other value so that we can sum them correctly.



5.2 Limitations

We ran into RAM problems right away because Google Colab provides only a limited memory (12GB) and our dataset is pretty heavy: the number of samples is about 21000 and for each sample we have 110250 features.

- This resulted in the impossibility to handle all the data together so we decided to split the data in 6 partitions and handle them separately, 3 partitions at a time: 1 for training, 1 for validation, 1 for testing.
- We had to reduce also the batch size of the learning phase from 80 to 32 (in the first training phase) because we could not allocate large enough tensors due to the memory limitations.
- We tried an alternative training phase with Gradients Accumulation in order to being able to simulate a batch size of 80.

6 Training Phases

We followed the paper[1] for the training phase, i.e. we used 3 partitions at a time: batch 1 for training, batch 2 for validation, and batch 3 for testing. Next, we rotate the partitions in a way that, at the end, all dataset is covered for training.

```
[0 train - 1 valid - 2 test ] -> [1 train - 2 valid - 3 test]
-> ... -> [5 train - 0 valid - 1 test]
```

We trained 3 different models in order to being able to compare the results.

6.1 First Training, Normal

We trained our model for 100 epochs with batch size of 32 and early termination. After predicting the music genre for each segment of the test set, we aggregated the predictions for all 21 segments belonging to the same song with a the mean of them.

We obtained an accuracy between the 40% and the 50%.

6.1.1 Considerations

We experienced an overfitting of the data because we noticed that while the training accuracy kept increasing, the validation accuracy stopped growing towards 60%.

This is probably caused by:

- Complexity of the model (over 4 millions of parameters)
- GTZan dataset quality
- Absencee of regularization
- Batch size

We wanted to stick with the architecture defined by the paper[1] so we decided to keep the architecture fixed and try something else.

1. Use L^2 regularization
2. Use Gradients Accumulations

6.2 Second Training, L^2 Regularization

In the second Training we used the L^2 Regularization for the weights in the Convolutional layers of the Residual Block.

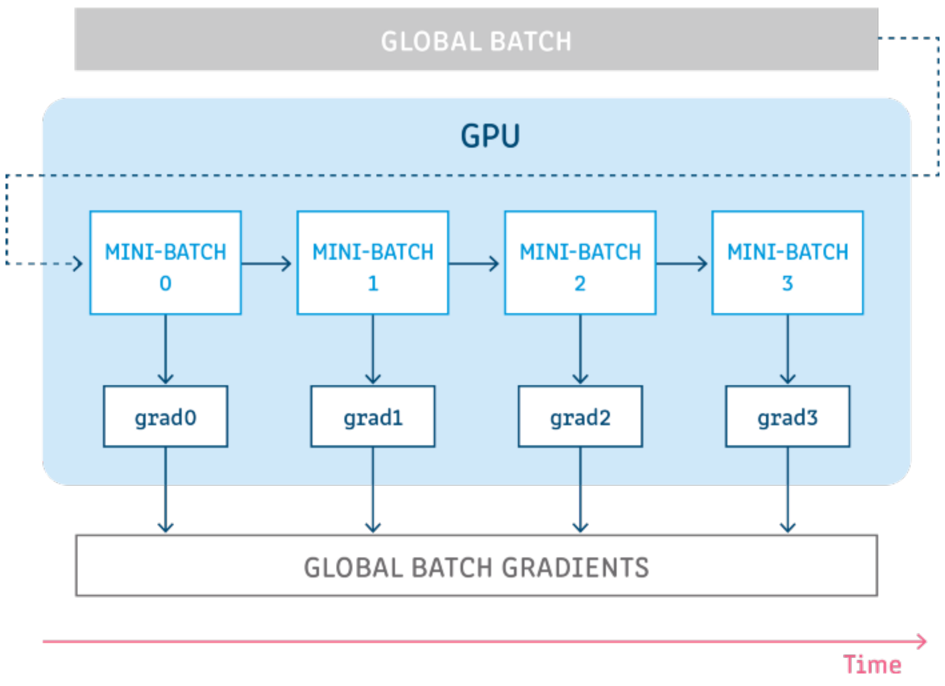
$$Loss = Loss(y - \hat{y}) + \lambda[\frac{1}{N}\sum_{i=1}^{N} w_i^2]$$

We used a $\lambda = 0.01, 0.001, 0.0001$. We trained for 100 epochs with batch size of 32 and early termination.
In the end we achieved comparable results with $\lambda = 0.001$ but the overfitting was still here.

6.3 Third Training, Gradient Accumulation

From the article of Raz Rotemberg[2].

What is Gradient Accumulation?
Gradient accumulation is a mechanism to split the batch of samples — used for training a neural network — into several mini-batches of samples that will be run sequentially.
This is particularly useful for problem of batch size being limited by GPU memory.



$$accumulated = \sum_{i=1}^{N} grad_i$$

Weights update:

$$w_t = w_t - \eta(\sum_{i=1}^{N} grad_i)$$

We realized a Custom Optimizer based on Adam with Gradient Accumulation[3] with Steps = 8 and we used a mini-batch of 10 in order to obtain a global batch of 80. We executed for 100 epochs and early termination.
We also had to change the code because custom optimizers are no longer supported in *tf.keras*.

7 Results

We report in this table the average accuracy of the three models after the training.

Model	Segments	Aggregation
Normal	52.52%	55.35%
L^2	35.88%	38.32%
Gradients Accumulation	37.31%	37.92%

7.1 Possible Improvements

In order to try an improvement in the final accuracy we could try:

- Data augmentation;
- Other model regularization such us Dropouts layers;
- Modifying the structure of the model.

8 Conclusion

Unluckily we couldn't improve our accuracy with the Grandient Accumulation and the regularization but we achieved a nice percentage with the normal model. Probably with more time to train and more resources we could find better hyperparameters in order to obtain a better convergence. However we are quite satisfied by the result that we got.

References & Bibliography

[1] 1D CNN Architectures for Music Genre Classification, Safaa Allamy and Alessandro L. Koerich, 15 May 2021.

[2] What is Gradient Accumulation in Deep Learning?, Raz Rotenberg, Jan 22, 2020, https://towardsdatascience.com/what-is-gradient-accumulation-in-deep-learning-ec034122cfa.

[3] https://stackoverflow.com/a/56946898 ·

[4] Musical genre classification of audio signals by G. Tzanetakis and P. Cook in IEEE Transactions on Audio and Speech Processing 2002.

[5] An analysis of the GTZAN music genre dataset, by Bob L. Sturm Department of Architecture, Design and Media Technology, 02 November 2012. Aalborg University Copenhagen