POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Music genre classification based on res-gated CNN and attention mechanism

Laurea Magistrale in Artificial Intelligence

Author: Enrico Targhini, Kevin Vargu, Alessandro De Luca

Academic year: 2023-2024

1. Project Overview

The need to correctly classify music genres in the era of digital streaming music has become of paramount importance, traditional music genre annotation relies on manual tagging and leads to issues of cost, expertise and accuracy, even when classification is performed via user-generated tags, this may reduce cost but may introduce labeling inconsistencies. Automated methods are thus needed to overcome the problem, but even among them standard machine learning methods, that are feature based, struggle with large volumes of music data. In the paper [5] authors propose a neural network called a *RGLU-former*, which combines the strengths of CCNs for feature extraction and those of the transformer which excels at capturing global patterns. Their novel architecture uses a modified 1D res-gated CNN to handle the local audio features and adapts the Transformer decoder to aggregate global information. The combination allows for superior classification performance by leveraging both local and global audio dependencies. The aim of this project is to replicate the learning process described in the paper using all the expertise gained during the Numerical Analysis for Machine Learning course and compare the results obtained with those reported in the paper.

2. Dataset Creation

The first step of the project was to produce the dataset as described in the paper. The chosen dataset to start from is **GTZAN** dataset [4] which consists of 1000 30-second-long audio tracks stored at 22,050 Hz, taken from **10 different musical genres** (Blues, Classical, Country, Disco, Hip Hop, Jazz, Metal, Pop, Reggae and Rock), each one made of 100 tracks. To perform all the steps needed to produce a working dataset we used the librosa python library [3] which is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems. As described in the paper, the audio tracks needed to be transformed in 217x334 size images in order to be fed to the **RGLUformer**. First, due to redundant information on the polyphonic channels, the audio files were converted to monophonic and down sampled at 16,000 Hz, then the **Mel spectrogram** is obtained via a Fourier Transform with a window length of 512 and hop size of 256. In order to have more data to train the model on, as described in the paper, each track was split in 5 second samples each one overlapping by 50% with the following sample, resulting in a total of 11 5-second clips for each 30-second audio file present in the dataset, for a total of 11000 samples in the whole dataset. While processing the dataset though, we noticed that audio file number 54 of the jazz

category was corrupted, our choice was to simply not consider this audio file, as a lack of 11 sample in one class doesn't really make the dataset unbalanced. The total thus became:

Table 1: Number of samples per music genre and their percentages

Music Genre	Number of Samples	Percentage (%)
Blues	1100	10.01
Classical	1100	10.01
Country	1100	10.01
Disco	1100	10.01
Hip Hop	1100	10.01
Jazz	1089	9.91
Metal	1100	10.01
Pop	1100	10.01
Reggae	1100	10.01
Rock	1100	10.01
Total	**10989**	**100.00**

3. Architecture of the model

We'll now focus on the structure of the RGLU-former, seeing every single block separately and elaborating on the rationale behind the coding decisions implemented.

3.1. RGLU Blocks

The first part of the network is particular kind of convolution neural network, called **RGLU**. A gated linear unit, inspired by Dauphin et al. [1], it is made up of three fundamental stages: convolution, activation and multiplication. At first, the unit performs two different 1D-convolution over the input, in order to keep different weights for each convolution. Since the input is composed by time series, the use of **1D-convolutions** over the 2D-convolution helps to focuses more on capturing the data features in a particular direction. The receptive field of 1D-convolution covers all frequency ranges of the spectrogram and convolves only in the time dimension, which can capture various musical elements that appeared in the spectrogram. In contrast, the convolution kernel of 2D-convolution convolves in both the time dimension and the frequency dimension, which is not interpretable for the sound signal.

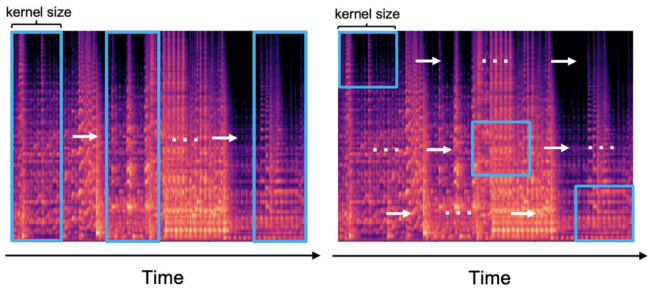

Figure 1: Difference between 1D and 2D convolution

In the second step, one of the convolutions' output passes through a sigmoid as activation block, and then the output of the convolution is element-wise multiplied with the output of the sigmoid, applied to the output of the other convolution. The multiplication is performed in order to keep only values activated by the sigmoid, and discard low and less interesting values.
Since the network slowly degrades after adding too many layers, Kaiming He et al. [2] proposed to add a residual network in order to keep relevant features from previous iteration and avoid possible overfitting problems. This network simply adds the input of each GLU to it's output, obtaining the formula of the RGLU:

$$Y = X + \text{Conv1D}_1(X) \otimes \sigma(\text{Conv1D}_2(X)) \quad (1)$$

Since the output of the GLU block might have a different size from the input X due to a decreasing number of filters, we decided to insert the residual network only in RGLU where the input and output shapes are the same.
The RGLU is then used to form an **RGLU block**, composed by two consecutive RGLU (which might have different number of kernels) and a final Max Pooling 1D layer. The max-pooling layer reduces the feature map's size, which helps reduce the computational effort and prevent overfitting. To summarize, the convolution network of the classifier is composed by:

- One 1D convolution to reduce the size of the input.
- Five RGLU blocks, composed by:
  - Two Res-net Gated Linear Units with increasing kernels' number, all with kernel size = 3 and stride = 1.
  - A Max-Pooling 1D layer with poolsize = 2 and stride = 2.

3.2. Encoder

Since we're dealing, as we said, with musical audio, we need a recursive mechanism able to extract abstract features of the spectrogram without losing temporal information within the music. To do so, this model relies on a classical **transformer block** as encoder. Starting from an input of size 256x10, The encoder first adds positional encoding to preserve the significance of the positions within the sequence. Then the encoder implements a **multi-head self-attention** mechanism with 8 heads, which allows the model to capture the abstract features that significantly impact the audio category. The abstract features of all moments will be aggregated into an overall feature vector, which can then be passed to the subsequent network to complete the music genre classification task. The output of the attention, as usual transformers, is passed through a feed-forward layer. At first, we thought about flattening the data and apply a feed-forward transformation over them. However, this procedure is not correct since the feed-forward part of the transformer is designed to apply transformations to each position in the sequence independently, which is why dense layers are used. Flattening the input would lose the positional information and the structure of the sequence, which is crucial for the transformer's operation. The feed-forward layer was so composed by a dense transformation over the feature with dimension = 2048, and then reduced back to the output dimension of 10. We decided to opt for 2048 as feed-forward dimension because it's the standard dimension used for transformers blocks. In addition, in order to reduce overfitting's effect, we added some dropout layers in the encoder with dropout rate = 0,1.

3.3. Decoder

Since we study the task of music genre classification, we do not use the usual decoder structure. Instead, we use a **global-pooling feature aggregation layer** and several fully-connected layers stacked together. The global-pooling aggregation first performs separately 1D-GlobalAvgPooling and 1D-GlobalMaxPooling on the encoder's output. The results are stacked together to form a 1D feature vector, which contains both the overall features and the most significant features of the feature map. The vector then passes through two feed-forward layers, with respectively 200 and 100 units, adding also a dropout layer with dropout rate = 0,2 to reduce the overfitting phenomenon. Finally, the vector passes through a final layer with 10 units as the number of classes, using softmax to perform the classification.

3.4. Overall structure

In the following picture, we can see the overall architecture of the network.

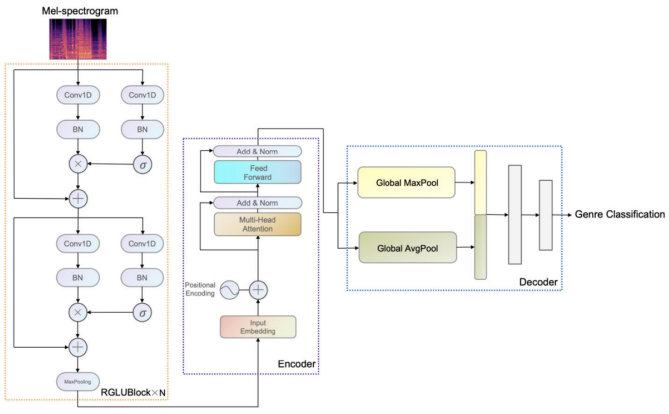

Figure 2: Architecture of the model

4. Training and Testing

The training procedure, we used a slightly improved version of **10-fold cross validation**. For each fold, The dataset was divided into training, validation and test sets with a ratio 8:1:1, and this division was performed assuring also that the same ratio is preserved for each class. We decided to use this form of K-fold for two main reasons: first, since the dataset is not composed by a huge amount of samples, we force to have equal distribution of classes to train the model over a sufficient amount of data for each class. In addition, we noticed from similar works [6] that this procedure is common for this type of problem. Each fold was trained with 500 epochs and with early stopping fixed at 80 in order to reduce overfitting. As loss function, we decided to opt for the **categorical cross-entropy**, which helps to compare the softmax output with the real label when classifying multiple classes. Lastly, we decided to use **Adam** as the learning rate update procedure, which is very efficient and less sensitive to hyper-parameters. Once the model is trained,

we tested it over the test set of its fold and we stored the result obtained. The procedure was repeated for the 10 folds and the final experimental results were averaged over the ten experiments.

5. Results and Comparison

5.1. Results

The **RGLUformer** architecture demonstrated impressive performance in the task of music genre classification. Using 10-fold cross-validation, the model achieved a mean accuracy of 88.70%, with a precision of 88.76%, recall of 88.70%, and an F1-score of 88.64%. These results are highly competitive and nearly match the performance reported in the original paper.

Metric	Value
Mean Accuracy	0.8870
Mean Precision	0.8876
Mean Recall	0.8870
Mean F1-Score	0.8864

Table 2: Mean Accuracy, Precision, Recall, and F1-Score across all 10 folds.

Overall, the model proved to be effective, with very consistent performance across all the evaluation metrics. These metrics suggest that the architecture is well-balanced in terms of precision and recall, demonstrating the model's ability to correctly identify music genres while minimizing false classifications.
Furthermore, the graph below shows the performance of the model across the 10 individual folds. The fluctuations indicate some variability, with folds 4 and 10 showing lower performance. However, in most cases, the accuracy, precision, recall, and F1-score remain above 0.88, and for some folds, such as fold 3, the model achieved accuracy and other metrics close to 0.94.

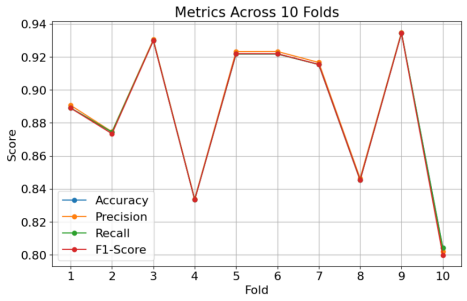

Figure 3: Metrics across 10 folds: Accuracy, Precision, Recall, and F1-Score.

Despite the fluctuations, the overall results remain very strong and confirm that the model generalizes well across different splits of the dataset.
While our results do not fully match the original paper, these small differences could be explained by variations in dataset preprocessing, hyperparameter choices, or training strategies. Nonetheless, the achieved results indicate that the model is highly reliable and effective for music genre classification.

5.2. Comparison between RGLU-LSTM and RGLUformer

Both the **RGLU-LSTM** and **RGLUformer** architectures demonstrated similar performance in the task of music genre classification. Using 10-fold cross-validation, the **RGLU-LSTM** achieved a mean accuracy of 90.37%, with a precision of 90.48%, recall of 90.37%, and an F1-score of 90.35%. These results are highly comparable to those obtained by the **RGLUformer**, with both models exhibiting accuracy close to or above 90%.

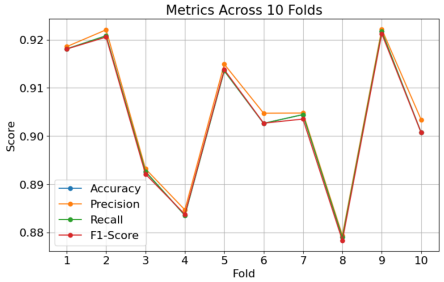

Figure 4: Metrics Across 10 Folds for the RGLU-LSTM model: Accuracy, Precision, Recall, and F1-Score.

The performance of the **RGLU-LSTM** across the 10 folds remained consistent, with accuracy

generally staying above 89% in most folds. However, there were some fluctuations, as seen in folds 3, 4, and 8, where the accuracy and other metrics dipped slightly below 90%. On the other hand, folds 1 and 9 achieved accuracy as high as 92%, indicating the robustness of the model in capturing patterns in the data across different cross-validation splits.

Despite their similar performance, the **RGLU-LSTM** offers an advantage in terms of ease of use, especially in scenarios like this one. While the **RGLUformer**, with its Transformer-based architecture, is a powerful tool for capturing both local and global patterns through self-attention, it typically requires more careful hyperparameter tuning and optimization. Transformers are sensitive to various factors, such as learning rate schedules, dropout rates, and the dimensionality of the feed-forward layers, which makes achieving optimal performance more challenging.

In contrast, the **RGLU-LSTM** architecture is more straightforward to implement and requires less tuning to perform well. LSTM layers are well-established for processing sequential data, making them particularly effective in tasks like music genre classification, where temporal dependencies are crucial. Furthermore, LSTMs tend to be more robust with default or minimal hyperparameter adjustments, making them easier to use, especially in scenarios where extensive tuning is impractical or time-consuming.

Metric	Value
Mean Accuracy	0.9037
Mean Precision	0.9048
Mean Recall	0.9037
Mean F1-Score	0.9035

Table 3: Mean Accuracy, Precision, Recall, and F1-Score of the RGLUformer model.

5.3. Implementation of Voting-Based Model Evaluation

In our effort to improve the performance of the model and handle sequential data more effectively, we implemented a **voting-based prediction** mechanism. This approach aimed to enhance model robustness by aggregating predictions over segments belonging to the same input audio. The voting mechanism was able to smooth out fluctuations in the predictions for individual segments, which theoretically leads to improved consistency in classification results.

The idea was to group **11 consecutive segments** from the dataset for each sample in the same fold, and then perform 10-fold cross-validation as before. For the testing, instead of making predictions for individual segments, we implemented a voting mechanism where predictions from 11 consecutive segments were aggregated through **max voting** to produce the final prediction for each sample.

The implementation followed these steps:

1. **Data Preparation**: For each fold, we generated the training, validation, and test sets by including each of the 11 fragments of audio in the same fold.
2. **Voting Mechanism**: We performed max voting over the fragments of a single sample. The class with the highest votes was then selected as the final prediction for that sample.
3. **Evaluation**: The final predictions were compared against the true class labels using common evaluation metrics.

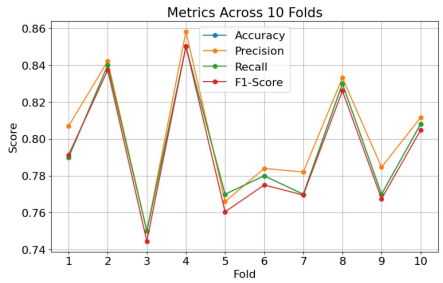

Figure 5: Metrics Across 10 Folds for the Voting-Based Model: Accuracy, Precision, Recall, and F1-Score.

After implementing this voting-based strategy, the model achieved a mean accuracy of 79.18%, a precision of 80.77%, a recall of 79.18%, and an F1-score of 79.12%. While these results are slightly lower compared to the other models, the voting mechanism did provide some improvements in terms of reducing noise and increasing the stability of predictions by aggregating information from multiple segments.

The slightly lower performance could be due to the nature of the voting mechanism itself, which may dilute some of the finer nuances that individual segments contribute to the overall prediction. By averaging over multiple segments, the

model might miss out on important details that could have otherwise influenced the classification decision.
Despite these results, the voting-based approach still showed value by aggregating information and smoothing the predictions across multiple segments, offering an alternative method to deal with noisy or inconsistent data. However, its performance in this specific case suggests that more refined tuning or modifications to the voting process might be needed to fully unlock its potential in music genre classification.

Metric	Value
Mean Accuracy	0.7918
Mean Precision	0.8077
Mean Recall	0.7918
Mean F1-Score	0.7912

Table 4: Mean Accuracy, Precision, Recall, and F1-Score of the Voting-Based Model.

6. Conclusions

The goal of this project was to replicate and evaluate the performance of the **RGLUformer** architecture, as presented in the referenced paper, for the task of music genre classification. Our approach combined Res-Gated CNNs (RGLU) for local feature extraction with the Transformer encoder to capture global dependencies within the audio data. In addition to this, we explored the performance of an **RGLU-LSTM** architecture, comparing the use of Transformer-based and LSTM-based models in this context.

6.1. Key Findings

The **RGLUformer** performed admirably, achieving a mean accuracy of 88.70% across 10-fold cross-validation, along with strong precision, recall, and F1-scores. These results demonstrate the effectiveness of the model in combining local and global feature extraction, showcasing the power of its architecture for complex tasks like music genre classification.
In parallel, the **RGLU-LSTM** model also showed strong performance, with a mean accuracy of 90.37%, similar to the RGLUformer. Importantly, the comparison highlights that both models are highly competitive, with their respective strengths making them suitable for different types of tasks within music genre classification.

6.2. Possible Reasons for Lower Performance

Although the models demonstrated solid performance, several factors could explain why the results were slightly lower than those reported in the original paper:
1. Dataset Preprocessing: One key factor could be differences in dataset preprocessing. Although we followed the same procedures described in the paper, such as converting audio tracks to Mel spectrograms and splitting them into overlapping 5-second segments, small variations in preprocessing parameters (e.g., window size, overlap percentage, or audio normalization techniques) might have affected the quality of the input features.
2. Model Hyperparameters: Another possible explanation is that the hyperparameters used in our model, such as the Transformer encoder's feed-forward layer dimensions, might not have been fully optimized. While we tried to replicate the architecture as closely as possible, minor differences in the model configuration could have contributed to the performance gap.
3. Training Strategy: The paper may have used a more refined training strategy. For instance, a larger batch size or a longer training period. We used early stopping to prevent overfitting, but it is possible that the original authors had a more aggressive data augmentation strategy or trained for longer epochs, leading to improved results.
4. Computational Resources: Finally, the computational resources available for our training process might have also influenced the results. The paper's authors may have had access to more powerful hardware, allowing them to use larger models or more extensive hyperparameter searches. In contrast, resource constraints in our case may have limited our ability to explore a broader range of configurations.

6.3. Final Remarks

In conclusion, both the **RGLUformer** and **RGLU-LSTM** architectures demonstrated strong capabilities for music genre classification. The RGLUformer, with its Transformer-based architecture, remains a powerful choice for tasks requiring both local and global feature extrac-

tion. On the other hand, the RGLU-LSTM proved to be an effective alternative, leveraging LSTM's strengths in modeling sequential data. This project highlights the value of combining both local and global feature extraction techniques for automated music genre classification. Whether through Transformers or LSTMs, the results indicate that advanced deep learning architectures are well-suited to this complex task, providing robust and reliable performance.

References

[1] Auli M Grangier D Dauphin YN, Fan A. Language modeling with gated convolutional networks. *International conference on machine learning. PMLR, pp 933–941*, 2017.

[2] Ren S Sun J He K, Zhang X. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778*, 2016.

[3] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8, 2015.

[4] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[5] Changjiang Xie, Huazhu Song, Hao Zhu, Kaituo Mi, Zhouhan Li, Yi Zhang, Jiawen Cheng, Honglin Zhou, Renjie Li, and Haofeng Cai. Music genre classification based on res-gated cnn and attention mechanism. *Multimedia Tools and Applications*, 83:1–16, 07 2023.

[6] Xu X Xing X Zhang W, Lei W. Improved music genre classification with convolutional neural networks. *Interspeech, pp 3304–3308*, 2016.