Music genre recognition

Lenart Rupnik
Faculty of Computer and Information Science
E-mail: lr95263@student.uni-lj.si

***Abstract*—With the continuous development of information technology, data transmission bandwidth and speed also increase. Therefore retrieving favourite music quickly and effectively has become a key research direction at present. Genre is one of the most mentioned music labels. Music detection by genre has become the mainstream method of the music information search. It is also an important part of the music service platform to recommend music. Thus, music genre recognition has attracted much attention and has become the mainstream direction of our research. This article presents a music genre recognition based on models with K-nearest neighbours, Long short-term memory and Convolutional neural networks. GTZAN dataset is used to extract features based on Mel-Frequency Cepstral Coefficients and create models to classify music genres. Experimental results show that the proposed algorithms can efficiently extract music features and make general classification.**

***Keywords*—Mel-Frequency Cepstral Coefficients, Convolutional neural network (CNN), Recurrent neural network (RNN), Long short-term memory.**

I. Introduction

Distinguishing between music genres is a trivial task for human beings. Most of us can roughly classify music genre after a few seconds of music, although there are 41 major genres and more than 300 subgenres [1]. Music genres are categories that have arisen through a complex interplay of cultures, artists, and market forces to characterize similarities between compositions and organize music collections. Yet the boundaries between genres still remain fuzzy, making the problem of music genre recognition a non-trivial task [2]. While usefulness of genres has been debated, it remains widely used. This general classification helps listeners find similar music and discuss its properties. The purpose of this project was to train a model which will be able to automatically classify music into music genres.

II. Related work

There are many options for methods when tackling automatic music genre recognitions. At first, most models were based on typical machine learning classifiers like K-nearest neighbours (kNN), Gaussian mixture models (GMM) and Support vector machines (SVMs). Lately these methods were replaced by neural networks, specifically Long short-term memory (LSTM) and Convolutional neural networks (CNNs). We'll discuss a few proposed implementations.

In article [3] Wu et al. proposes music genre classification with weighted kNN algorithm. Author's algorithm makes improvements in two aspects of the traditional kNN algorithm, which can effectively solve the problem that traditional KNN algorithm ignores: the degree of correlation between attributes and categories in the classification process. Another problem it solves is that it only considers the number of the nearest samples. In their work, authors achieve precision of classification up to 94%.

Chaitanya Kakarla et al. [4] developed a model with Recurrent neural networks (RNNs) that help resolve complex temporal features of the audio signal and identifying music genres. Authors's model was able to achieve 84% accuracy based on GTZAN library.

In Caifeng Liu et al. [4] authors worked on a model for music recognition based on Convolutional neural networks (CNNs). In this article, authors treat audio features as an image based problem. The Proposed CNN architecture also takes the long contextual information into considerations, which transfers more suitable information for the decision-making layer.

III. DATA

To train our model, we'll use a widely known dataset called GTZAN . Although GTZAN library originates in year 2004 we decided to use it, since it is referenced in all major papers. That way it is easier to compare and evaluate the results. We collected it from *kaggle* [5] in zip format that contained 10 folders and corresponding files. In this dataset, there are 1000 audio tracks, each 30 seconds long. Tracks are classified in 10 genres, where each genre is represented with 100 tracks. All the tracks are 22050 Hz Mono 16-bit audio files in .wav format. Beside audio files, this dataset also contains pre-prepared collection of features extracted from this dataset in .csv format. These features not only include Mel Frequency Cepstral Coefficients (MFFCs) but also features like harmony, zero crossing rate, tempo etc. All together, there are 58 features, out of which 20 of them are directly connected with MFCCs. We used the latter to compare LSTM models trained on our feature extraction algorithm with the one trained on already given features.

IV. Feature extraction

Raw audio is tough to manage, because it includes too many knowledge points (22050 per second). Training a model with that much information would be time-consuming and our models would have a hard time finding relevant information, so we need some sort of compression. In general, audio features gathered from audio signal are classified into 3 categories high-level, mid-level, and low-level audio features.

- High-level features are related to music lyrics like chords, rhythm, melody, etc.
- Mid-level features include beat level attributes, pitch-like fluctuation patterns, and MFCCs.
- Low-level features include energy, a zero-crossing rate.

From here we'll focus on MFCCs since they are proven [6] to give the best results with genre recognition. In sound processing, the Mel Frequency Ceptsrum is a representation of the short-term power spectrum of a sound, based on a Mel scaled spectrogram. Mel Frequency Cepstral Coefficients of an audio signal are a small set of features which concisely describe the overall shape of the spectral envelope.

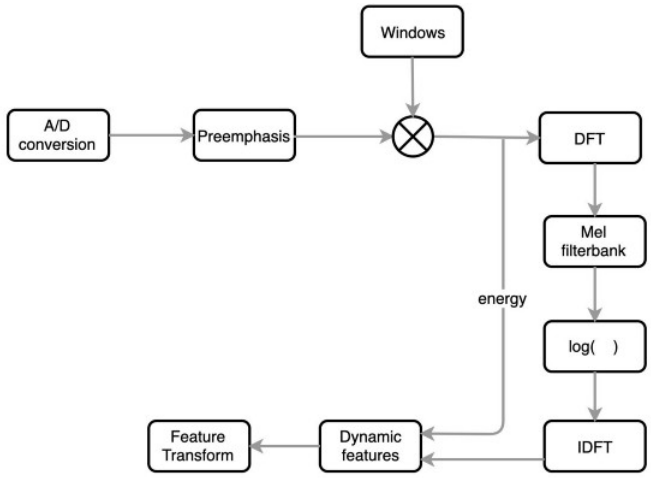

Fig. 1. MFCC extraction diagram. Source [7]

The whole purpose of the MFCC is to adapt the audio signal feature to the human hearing system. Since we are focusing on genre recognition, it is important that our model receives audio as we hear it, so it classifies easier [8]. Humans can hear frequencies below 1000 Hz as a linear scale and frequencies above as a logarithmic scale. For this reason, MFCC is prevalently used in speech recognition systems [8]. MFCCs are obtained by averaging spectrogram values over Mel frequency bands which improves stability, but also removes information. For this reason, MFCCs are effective at describing a signal over a short time window.

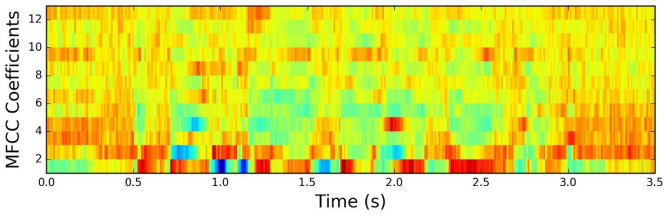

Fig. 2. Mfcc 13 coefficient spectrum visualization. Source [9].

To help us extract MFCC features, we used python library *librosa* with already implemented MFCC extraction function. When extracting, we used parameters shown in Table I.

TABLE I
PARAMETERS USED FOR FEATURE EXTRACTION.

Number of MFCs to return	13, 40
Sample rate	22050
Number of samples between successful frames	512
Length of FFT window	2048

V. PROPOSED MODELS

In our research, we used and compared three different machine learning models to perform music genre classification.

A. K-nearest neighbours

Although being one of the conventional methods, kNN is still a popular algorithm because of its simplicity and efficiency. It is considered as non-parametric supervised learning method. The base concept of kNN is the assumption of similarity between the new case/data and available cases.

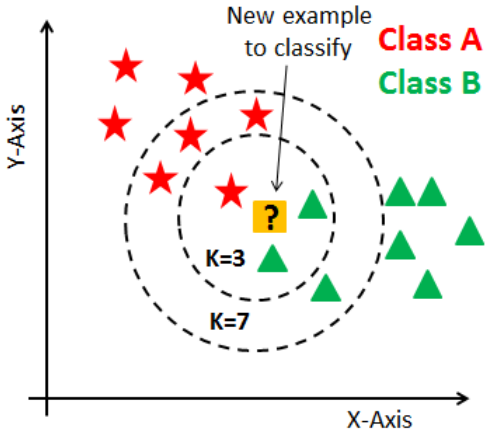

Fig. 3. K-Nearest Neighbors. Source [10].

KNN algorithm stores all the available data and classifies a new data point based on the similarity, usually represented as distances. It is also called a lazy learner algorithm because it does not learn from the training set immediately. Instead it stores the dataset and at the time of classification it performs an action on the dataset. The kNN algorithm, when implemented in music genre classification, looks at similar songs and assumes that they belong to the same category because they seem to be near each other. A simple 2D representation of kNN model is shown in Fig. 3.

B. LSTM neural network

Long short-term memory (LSTM) is a part of Recurrent neural networks. Recurrent neural networks are dynamic systems; they have an internal state at each timestep of classification. Unlike standard feedforward neural networks, LSTM has feedback connections. These feedback connections enable RNNs to propagate data from earlies events to current processing steps. As such a recurrent neural network can

process not only single data points, but also entire sequences of data [11].

The name of LSTM refers to the analogy that a standard RNN has both ”long-term memory” and ”shot-term memory”. The connection weights in the networks change once per episode of training, analogous to how physiological changes in synaptic strengths store long-term memories in our brain. The activation pattern in the network change once per timestep is analogous to how the moment-to-moment change in electric firing pattern in the brain stores short term memories. The LSTM architecture aims to provide a short-term memory of RNN that can last thousands of timesteps, therefore ”long short-term memory” [12]. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals, and three gates regulate the flow of information into and out of the cell.

In a standard RNN, a input feature vector sequence is represented as x = (x1, x2, ..., xT), then the hidden vector sequence h = (h1, h2, ..., hT) from t = 1 to T are computed using following equations:

$$h_t = \sigma(W^{xh}x_t + W^{hh}h_{t-1} + b^h) \tag{1}$$

Where $\sigma(\cdot)$ is activation function, the superscript ‘x’, ‘h’ and in ‘W’ and ‘b’ represent the input layer, the hidden layer and the output layer respectively. For example, W^{xh} is the weight matrix connecting the input layer and the hidden layer, W^{hh} is the weight matrix connecting different hidden layers, b^h is the hidden bias vector, by is the output bias vector [13].

It is widely recognized in RNN that the activation function may lead to the solution of vanishing gradient problem. To solve this problem, the hidden notes are replaced by a set of cells, which are called the LSTM memory blocks. Each LSTM memory cell contains three gates: input gate, output gate and forget gate. The forget gate is shown to be essential for problems with continual or very long input strings [14]. In place of equations (1), the LSTM RNN is implemented according to the following equations [13], [14].

$$c_t = f_t \odot c_{t-1} + i_t \odot g(Q^{ex}x_t + W^{em}m_{t-1} + b^e) \tag{2}$$

$$m_t = o_t \odot h(c_t) \tag{3}$$

$$y_t = \phi(W^{ym}m_t + b^y) \tag{4}$$

Where $\odot$ is the element-wise product operator, $g(\cdot)$ and $h(\cdot)$ are the input and cell output activation functions, $\phi(\cdot)$ and $(\cdot)$are other activation functions, c_t is the cell state vector, W is a weight matrix, b is bias vector and the superscript i, f, o, c, x, y represent input gate, forget gate, output gate, cell state, input layer and output layer respectively [13].

Fig 4 is a simplified visual representation of LSTM cell and its states.

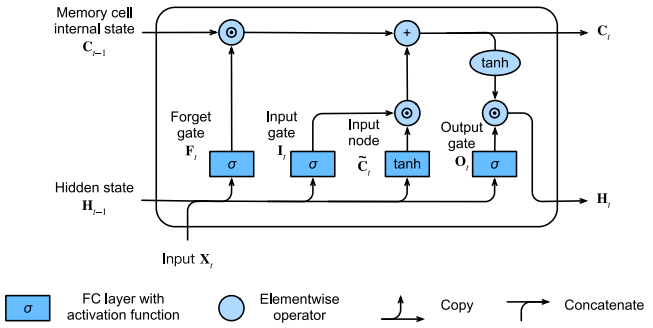

Fig. 4. LSTM architecture diagram. Source [15]

C. Convolutional neural network

To reiterate from the Neural Networks Learn Hub article, neural networks are a subset of machine learning, and they are at the heart of deep learning algorithms. They are comprised of node layers, containing an input layer, one or more hidden layers, and an output layer. Each node connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network. [16].

Convolutional neural networks are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs which is due to usage of convolutional layers. The convolutional layer is the core building block of a CNN, and it is where the majority of computation occurs. It requires a few components, which are input data, a filter, and a feature map [17].

Between sequences of multiple convolutional layers, the feature maps are commonly down-sampled using pooling operations. The most typical pooling operation is max-pooling. The idea of max-pooling (instead of for instance average pooling) is that strong activations (eg. edge or line features) are conserved within the network and not averaged out. A typical max-pooling operation with 2-by-2 filter size and a stride of 2 reduces the size of input feature may be a factor of 4, whereas the output cells contain the maximum value of 4 input cells within the 2-by-2 filter [16].

VI. Methodology

Before training our models we applied a previously described feature extraction method to our GTZAN dataset. Since GTZAN dataset includes only 100 audio files for every music genre, we decided to split every audio signal into 10 sections (3 s in length) so that we got a bigger dataset which is crucial when training neural networks. When splitting signals into smaller sections, we also improved MFCC information integrity since longer MFCCs lose some of its important data. The same structure is also used in already extracted features from *kaggle* dataset.

A. Neural network parameters optimization

When training our neural networks models, we had to perform hypertunning of our parameters. To tackle this problem, we used Tensorflow's integrated package called *keras.tunner*. As of the method to optimize parameters, we decided to go with Hyperband Concept. This technique tries to remove one of the problems in random search of hyperparameters. Problem is as follows: Random search may pick some values which are very obviously bad and will do full training and evaluation on it, which is wasteful. Hyperband provides one way to solve this problem. Hyperband Solution: Randomly sample all the combinations of hyperparameter and now instead of running full training and evaluation on it, train the model for few epochs (less than max_epochs) with these combinations and select the best candidates based on the results on these few epochs. It does this iteratively and finally runs full training and evaluation on the final chosen candidates. The number of iterations done depends on parameter 'hyperband_iterations' and number of epochs in each iteration are less than 'max_epochs' [18].

B. Used neural network models

After optimizing all neural networks, we decided to use models with parameters presented in below tables.

TABLE II
LSTM MODEL WITH ITS LAYERS AND THEIR PARAMETERS.

Layer	Num. units	Activation
LSTM	130	tanh
Dropout	0,15	/
LSTM	299	tanh
Droupout	0,15	/
Dense	299	relu
Dense	10	softmax

TABLE III
CNN MODEL WITH ITS LAYERS AND THEIR PARAMETERS

Layer	Num. units	Activation
Conv2D	64	relu
MaxPooling2D	2	/
Conv2D	256	relu
MaxPooling2D	2	/
Dropout	0,3	/
Conv2D	256	relu
MaxPooling2D	2	/
Dropout	0,3	
Dense	512	relu
Dense	10	softmax

We essentially built 2 neural network models. One to make classification with LSTMs - Table II and one for CNNs - Table III.

To tackle any overfitting problems, we introduced Dropout layers in our model. Dropout is a regularization method that approximates training many neural networks with different architectures in parallel. During training, some number of layer outputs are randomly ignored or "dropped out." This has the effect of making the layer look-like and be treated-like a layer with a different number of nodes and connectivity to the prior layer. In effect, each update to a layer during training is performed with a different "view" of the configured layer [19].

There were also same parameters that were common for both models. In both cases, we used Adam optimizer for backpropagation and binary cross entropy to calculate loss in every step.

VII. RESULTS

We trained each model on a previously described feature set. Before training the model, we split the data to train and test set in ratio 75:25. Then we trained the model on the whole training set. After we finished with optimizing parameters, we tested our models with test set and plotted the results. To make results comparable, we used the same split random seed for all models. Results are presented in Table IV and in confusion matrixes, Fig. 5, Fig. 6, Fig. 7. Note that when classifying with kNN model, we didn't additionally slice our data therefore we had less classifications.

TABLE IV
CLASSIFICATION RESULTS

Data	kNN	LSTM	CNN
13 MFCCs	0,723	0,821	0,899
40 MFCCs	0,721	0,859	0,915
kaggle extracted features	/	0,948	/

Looking at the results from Table IV we can see that out of our models, CNN model performed the best. The worst classification were done with kNN model. This is partially due to unoptimised methods used for classifications in kNN. Since our goal was to focus on neural networks, we didn't fully optimize our kNN model. If we would like to slightly improve our results with kNN model we could try different methods to calculate the distances, try a different number of neighbours or play with different weight distributions or filter our input data to remove unimportant parts. One thing that can still be seen from our results is that when using kNN model, number of Mel-Frequency Cepstral Coefficients doesn't play an important role in final results or in our case, it actually slightly worsens the results.

If we compare the results of both neural networks, we see that CNN model made better classification than LSTM model. Both are still quite successful. Part of the reason for this is that MFFCs are in a way a visual representation of an audio signal. Hence, using convolutional layers is by default a better model for this type of classification. If we take a look at LSTM model where we also compared our extracted features with the ones from *kaggle* dataset, we see that the model performed much better with the *kaggle* features. This gives us the idea that it's hard to classify genres based only on Mel-Frequency Cepstral Coefficients and that one should also account features like melody, rhythm etc.

Additionally, to try our model in real time application and demonstrate results from our model, we tried to classify additional full length unknown music that is not part of our dataset. For this example, we downloaded the whole audio signal (4min and 30s) of Still D.R.E song by artist Dr. Dre.

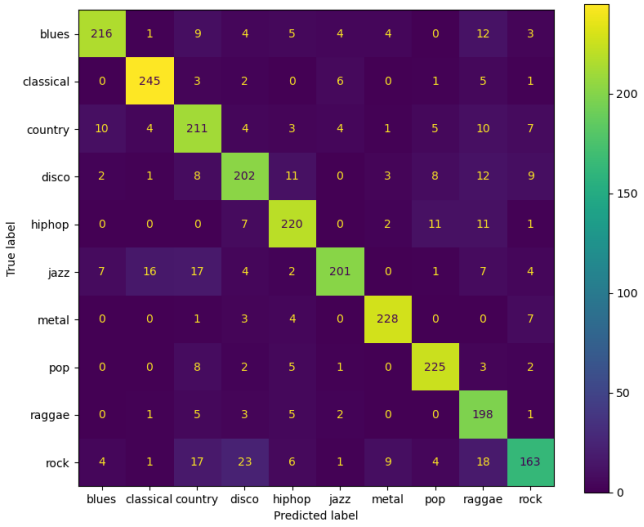

Fig. 5. Confusion matrix for LSTM mode classification.

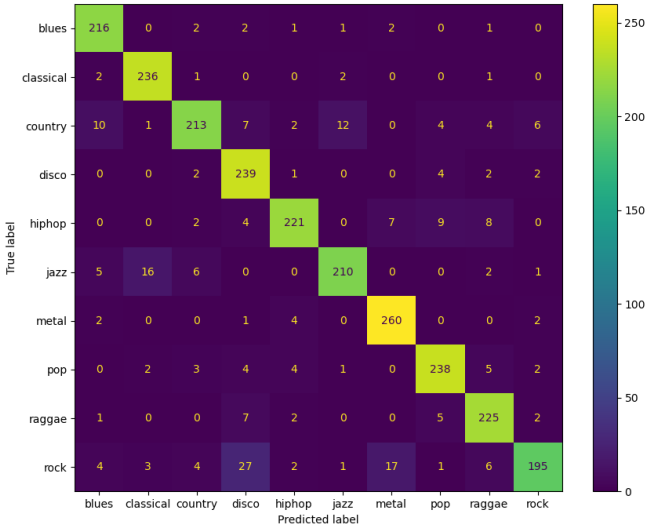

Fig. 6. Confusion matrix for CNN model classification.

If we wanted to use it in our model, we had to split it into 3s long sequences. After that, we loaded our model and made classification. Since we focused on neural networks, we made classification with LSTM and CNN model. A bar plot of classified genres for every segment of sliced audio file is shown in Fig. 8.

As we can see, most sequences were predicted in correct genre. There are some fluctuations. This is expected since our model is not perfect and only based on MFFCs but more importantly if we know that we sequenced music into short

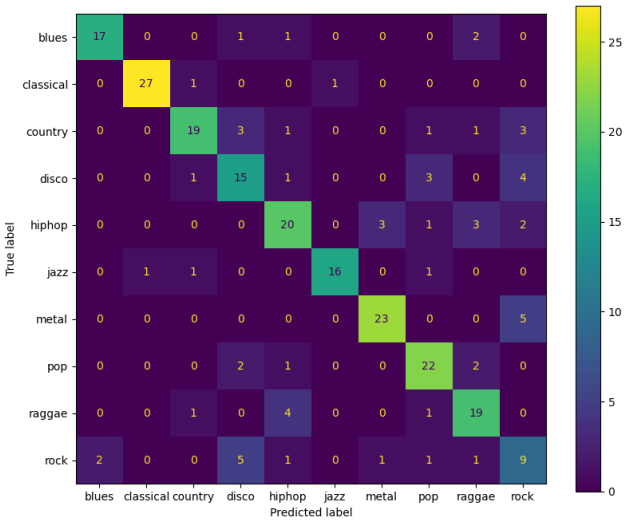

Fig. 7. Confusion matrix for kNN model classification.

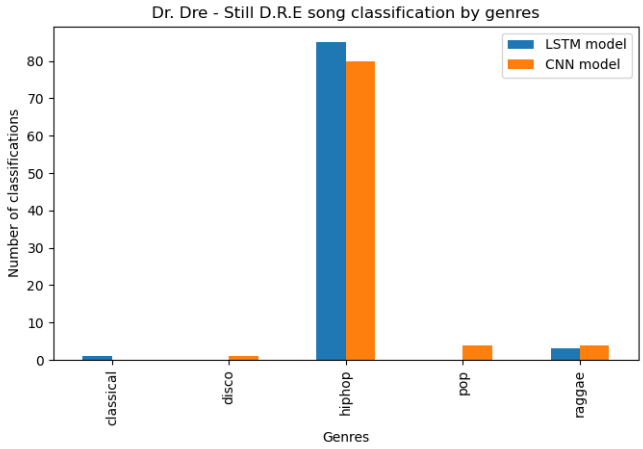

Fig. 8. Representation of genre distribution with CNN model.

fragments about 3s long there may be a short fragment that is actually closer to other genre than to general one. In that way, we know that our method can't produce the most optimal results.

VIII. Conclusion

In our work, we tried to classify music genres based on Mel-Frequency Cepstral Coefficients. We managed to create an efficient feature extraction algorithm, which we used to get features and train the models. The best results were obtained with Convolutional Neural Networks, which performed slightly better than Recurrent Neural Networks and kNN model. Changing the number of MFCC Coefficients improves our models slightly.

We achieved the best accuracies around 90% which is close

to what some other papers proposed with their models. Our model could be improved and further optimized.

REFERENCES

[1] K. Negus, *Music genres and corporate cultures*. Routledge, 2013.

[2] N. Scaringella, G. Zoia, and D. Mlynek, "Automatic genre classification of music content: a survey," *IEEE Signal Processing Magazine*, vol. 23, no. 2, pp. 133–141, 2006.

[3] M. Wu and X. Liu, "A double weighted knn algorithm and its application in the music genre classification," in *2019 6th International Conference on Dependable Systems and Their Applications (DSA)*, 2020, pp. 335–340.

[4] C. Kakarla, V. Eshwarappa, L. Babu Saheer, and M. Maktabdar Oghaz, "Recurrent neural networks for music genre classification," in *Artificial Intelligence XXXIX*, M. Bramer and F. Stahl, Eds. Cham: Springer International Publishing, 2022, pp. 267–279.

[5] A. Olteanu, "Gtzan dataset - music genre classification," Mar 2020. [Online]. Available: https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification

[6] G. Jawaherlalnehru, S. Jothilakshmi, T. Nadu, and T. Nadu, "Music genre classification using deep neural networks," *International Journal of Scientific Research in Science, Engineering and Technology*, vol. 4, no. 4, p. 935, 2018.

[7] "Mfcc technique for speech recognition," https://www.analyticsvidhya.com/blog/2021/06/mfcc-technique-for-speech-recognition/, accessed: 2022-29-12.

[8] A. Elbir, H. Bilal Çam, M. Emre Iyican, B. Öztürk, and N. Aydin, "Music genre classification and recommendation by using machine learning techniques," in *2018 Innovations in Intelligent Systems and Applications Conference (ASYU)*, 2018, pp. 1–5.

[9] "The guide to mfcc," https://medium.com/prathena/the-dummys-guide-to-mfcc-aceab2450fd, accessed: 2022-29-12.

[10] "k-nearest-neighbors," https://www.datacamp.com/tutorial/k-nearest-neighbor-classification-scikit-learn, accessed: 2022-29-12.

[11] R. C. Staudemeyer and E. R. Morris, "Understanding LSTM - a tutorial into long short-term memory recurrent neural networks," *CoRR*, vol. abs/1909.09586, 2019. [Online]. Available: http://arxiv.org/abs/1909.09586

[12] "Long short-term memory," https://en.wikipedia.org/wiki/Long_short-termmemory, accessed: 2022-30-12.

[13] J. Dai, S. Liang, W. Xue, C. Ni, and W. Liu, "Long short-term memory recurrent neural network based segment features for music genre classification," in *2016 10th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, 2016, pp. 1–5.

[14] F. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: continual prediction with lstm," in *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, vol. 2, 1999, pp. 850–855 vol.2.

[15] "Long short-term memory," https://d2l.ai/chapter_recurrent-modern/lstm.htmlfig-lstm-3, accessed: 2022-30-12.

[16] T. Kattenborn, J. Leitloff, F. Schiefer, and S. Hinz, "Review on convolutional neural networks (cnn) in vegetation remote sensing," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 173, pp. 24–49, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271620303488

[17] K. Palanisamy, D. Singhania, and A. Yao, "Rethinking CNN models for audio classification," *CoRR*, vol. abs/2007.11154, 2020. [Online]. Available: https://arxiv.org/abs/2007.11154

[18] N. H. Awad, N. Mallik, and F. Hutter, "DEHB: evolutionary hyberband for scalable, robust and efficient hyperparameter optimization," *CoRR*, vol. abs/2105.09821, 2021. [Online]. Available: https://arxiv.org/abs/2105.09821

[19] x. liang, L. Wu, J. Li, Y. Wang, Q. Meng, T. Qin, W. Chen, M. Zhang, and T.-Y. Liu, "R-drop: Regularized dropout for neural networks," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 10 890–10 905. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/5a66b9200f29ac3fa0ae244cc2a51b39-Paper.pdf