International Journal of Engineering Business and Social Science
Vol. 1 No. 04, April 2023, pages: 257-268
e-ISSN: 2980-4108, p-ISSN: 2980-4272
https://ijebss.ph/index.php/ijebss

A Study on Music Genre Classification using Machine Learning

Partha Ghosh[1], Soham Mahapatra[2], Subhadeep Jana[3], Ritesh Kr. Jha[4]
1,2,3,4 Department of Computer Science and Engineering
Government College of Engineering and Ceramic Technology, Kolkata, India
Email: parth_ghos@rediffmail.com

Submitted: 08-03-2023 Revised: 12-03-2023, Publication: 03-04-2023

Keywords
Artificial Intelligence, Machine Learning, Music Genre

Abstract
Artificial Intelligence (AI) and Machine Learning can be cited as one of the greatest technological advancements in this century. They are revolutionizing the fields of computing, finance, healthcare, agriculture, music, space and tourism. Powerful models have achieved excellent performance on a myriad of complex learning tasks. One such subset of AI is audio analysis. It entails music information retrieval, music generation and music classification. Music data is one the most abstruse type of source data present, mainly because it is a tough work to extract meaningful correlating features from it. Hence a myriad of algorithms ranging from classical to hybrid neural networks have been tried on music data for a getting a good accuracy. This paper studies the various methods that can be used for music genre classification and compares between them. The accuracies we obtained on a small sample of the Free Music Archive (FMA) dataset were: 46% using Support Vector Classifier (SVC), 40% using Logistic Regression, 67% using Artificial Neural Network (ANN), 77% using Convolutional Neural Networks (CNN), 90% using Convolution-Recurrent Neural Network (CRNN), 88% using Parallel Convolution-Recurrent Neural Network (PCRNN), 73% without using Ensemble technique and 85% using Ensemble technique of Ada Boost. We defined SVC as our baseline model which had 46% accuracy, and we defined the succeeding models to achieve accuracy greater than that. ANN gave us 67% on the test dataset, which was surpassed by CNN at 77%. We noticed image based features worked better at classifying the labels than normal extracted features from the audio. A combination of CNN and RNN worked the best for the dataset, with a series CRNN model giving the best accuracy. Succeeding that, we tried to fit an ensemble model onto our dataset and analyzed its workings. This paper presents a comprehensive study of the various methods that can be used for music genre classification, with a focus on some parallel models and ensembling techniques.

1. Intoduction

Music is the art of combining various vocal and instrumental sounds to produce a melody or rhythm. Music can be classified into a broad range of genres, some of the most popular ones being classical, folk, rock, pop, electronic, hip-hop etc. Over the years, the idea and perception of music has evolved, which has given rise to further genres and sub-genres. The way people consume music has also developed with the advancement of technology.

Music genre classification, a subset of music information retrieval, is a challenging and progressive task in the AI domain. It basically involves using machine learning concepts and algorithms to recognize the genre of a particular music audio file, the style or category of the music. For example the algorithm tries to differentiate between a rock music file and a classical music file based on the features of the audio. Classifying the genre of a piece of music automatically has manifold uses in this modern world. The applications are plenty. It can be used in an audio streaming platform or app (e.g. Soundcloud, Spotify, Gaana) for categorization and music recommendation, which then can be used to curate playlists based on the genre. The algorithm can be simply released as a product and can be used as a music identification app (e.g. Shazam). It can also be used in smart bots like Alexa, Google Assistant, Siri present in our smartphones to enhance the music listening experience of the user.

In this paper we have conducted music genre classification using various machine learning algorithms and compared the accuracy attained by the discrete algorithms. We also focus on ensemble techniques for music genre classification and whether it is more efficient than classical algorithms. The paper is structured as follows: - related work discusses the past work done on the topic of music genre classification, followed by dataset and data preprocessing, followed by models which lists the machine learning models we have used to achieve our task, followed by results and discussions rounding up the results we achieved, followed by future scope and conclusion, ending with references.

There has been quite a bit of work on the subject of music genre classification using convolutional neural networks, recurrent neural networks, combination of both and even feed forward networks.

K. Choi et al [1] presented a convolutional recurrent model for recognizing genre, moods, instruments and era from the Million Songs dataset. They used a 2D convolution model followed by recurrent layers and fully connected layers to perform the classification task. Our CRNN model described is similar but we used 1D convolution layers instead of 2D (Choi et al., 2017).

P. Kozakowski & B. Michalak [2] from DeepSound used a 1D convolution model followed by a time distributed dense layer on GTZAN dataset. We got the idea for 1D convolution layers from them, but found that the RNN layers after 1D CNN performed better for our dataset.

L. Feng et al [3] paralleled CNN and RNN blocks to allow the RNN layer to work on the raw spectrograms instead of the output from the CNN. Our parallel CNN-RNN model was heavily influenced by this paper and our final architecture is similar to theirs with some modifications since our dataset size is much smaller (Feng et al., 2017).

N. Pelchat et al [4] reviews some of the machine learning techniques utilized in the domain. They made use of spectrograms generated from time slices of songs as input the neural networks (Pelchat & Gelowitz, 2020).

T. Lidy et al [5] presents an approach using parallel CNN architectures to classify the genre of music files. They worked on the MIREX 2016 Train/Test Classification Tasks for Genre, Mood and Composer detection dataset (Lidy & Schindler, 2016).

K. K. Chang et al [6] introduces an approach of music genre classification using compressive sampling (CS). They use a CS based classifier which uses both short term and long term features of the audio file (Chang et al., 2010).

I. Y. Jeoung et al [7] describes a framework which learns the temporal features from audio using deep neural networks and uses them for music genre classification (Jeong & Lee, 2016).

P. Annesi et al [8] used a Support Vector Machine to design an automatic classifier of music genres. They used certain conventional features and engineered some new ones like beat and chorus for enhanced accuracy (Annesi et al., 2007).

Y. M. D. Chathuranga et al [9] also proposed an ensemble approach using frequency domain, temporal domain and cepstral domain features. They used a Support Vector Machine for the base learner and used AdaBoost technique for classification (Chathuranga & Jayaratne, 2013).

S. Jothilakshmi et al [10] explored the idea of using Gaussian Mixture Model (GMM) and k-Nearest Neighbour (kNN) classifier for the task. They specifically worked on Indian genres like Hindustani, Carnatic, Ghazal, Folk and Indian Western.de (Jothilakshmi & Kathiresan, 2012).

2. Materials and Methods

Data is the most important ingredient for any machine learning task. The accuracy of the model is directly proportional to the quality of data. There are many open source music datasets present over the Internet, some of which are listed below: -

1. GTZAN

GTZAN dataset comprises of total 1000 audio files of 30 second each, equally divided into 10 genres – blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, rock. It is available for download on kaggle (link: https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genre-classification).

2. FMA

Free Music Archive (FMA) dataset is divided into 4 subsets: -
I) FMA Small – 8000 audio files of 30s each equally divided into 8 genres.
II) FMA Medium – 25000 audio files of 30s each unequally divided into 16 genres.
III) FMA Large – 106574 audio files of 30s each unequally divided into 161 genres.
IV) FMA Full – 106574 full length audio files unequally divided into 161 genres.
We made use of the FMA Small dataset; the 8 genres are - electronic, experimental, folk, hip-hop, instrumental, international, pop and rock. It is available for download on github (link: https://github.com/mdeff/fma).

3. Million Song

T. Bertin-Mahieux et al [11] introduced the Million Song dataset which is a vast collection of audio features and metadata for a million contemporary popular music tracks. It is available for download on its website (link: http://millionsongdataset.com/).
The FMA Small is a digital audio dataset in which the audio files are already preprocessed. So we did not perform any explicit noise removal operations on the audio files. We made use of 480 audio files of 30 seconds each. Taking more audio files was computationally very expensive. We converted the mp3 to wav, as wav files are easier to use and process (Bertin-Mahieux et al., 2011). We use **librosa [12]**, a python package for music and audio analysis to extract some basic features from the audio files and append them into a dataframe. The features extracted are:-
1. Zero crossing rate: The zero crossing rate is the rate of sign-changes along a signal, i.e. the rate at which the signal changes from positive to zero to negative or from negative to zero to positive.
2. Chroma STFT: We first compute a chromagram from the waveform. Then we extract the normalized energy for each chroma bin at each frame.
3. Spectral centroid: It indicates where the center of mass of the frequency spectrum is located of the audio.
4. Spectral bandwidth: It indicates the frequency bandwidth for each frame.
5. Spectral rolloff: It is the roll-off frequency for each frame. The roll-off frequency is defined for each frame as the center frequency for a spectrogram bin such that at least roll_percent (0.85 by default) of the energy of the spectrum in this frame is contained in the bin and the bins below.
6. MFCC (Mel Frequency Cepstral Coefficient): In sound processing, the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC. We used the 20 MFCC values. The formula to convert f hertz into m mels is given in Equation 1.

$$m = 2595 log_{10}(1 + \frac{f}{700}) \quad \ldots (1)$$

We took the mean and variance of every feature values. The final dataframe had 480 rows and 50 columns. The last column is the label column, indicating the genre of that particular audio. A snap of the dataframe is given in Table 1.

Name	ZCR	Chroma STFT Mean	MFCC_20 Mean	MFCC_20 Var	Label
electronic.wav	37800	0.324840	0.444263	33.152580	0

experimental.wav	23424	0.174870	12.773882	32.273647	1
folk.wav	29326	0.295099	-1.172379	44.371731	2
hiphop.wav	108342	0.396907	0.094077	40.441700	3
instrumental.wav	41181	0.370636	-1.277175	38.967281	4
international.wav	43718	0.239448	-3.514284	80.465935	5
pop.wav	98404	0.397668	3.882601	29.444992	6
rock.wav	83250	0.520272	-0.011346	18.726589	7

Table 1 - Feature dataframe consisting of 48 feature columns and 1 label column for 480 training examples. The labels range from 0-7 for the 8 genres present.

A. Our Proposed Models

Our basic workflow was to define a baseline model first on the dataset and then enhance the accuracy further from that point. The list of models we used are as follows: -

1. Baseline Model

We trained certain conventional classifiers provided by the sklearn library to build our baseline model. We tried classifiers like Support Vector Classifier, Logistic Regression and fed the feature matrix as input data to them. We got the best performance using a Support Vector Classifier. It had an overall accuracy of 46% on the test dataset.

2. Artificial Neural Network (ANN)

Next we tried an Artificial Neural Network (ANN), which used the feature matrix as its input values. It was a 3 layer neural network which employed basic forward and backward propagation. The activation function used was ReLU and optimizer was Adam.

3. Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a deep learning algorithm which excels particularly in image classification. They generally take image data as input and assign learnable weights to various parts of the image. Then they train over the image dataset and calculate the weights which allow them to differentiate one class of image from the other. Hence CNN's are a practical choice of model owning to their ascendancy in image classification problems. For image input, we converted each audio file into a spectrogram, which is a visual representation of the spectrum of frequencies over time domain. A regular spectrogram is squared magnitude of the Short Term Fourier Transform (STFT) of the audio signal. This regular spectrogram is squashed using mel scale to convert the audio frequencies into something more human understandable. Then it is known as a mel-spectrogram. We used the built in function in the librosa library to convert audio files in to their respective mel-spectrograms (McFee et al., 2015). An example mel-spectrogram is given in Figure 1. The mel-spectrograms for different genres are different images with distinct features which can be fed into a CNN for classification.

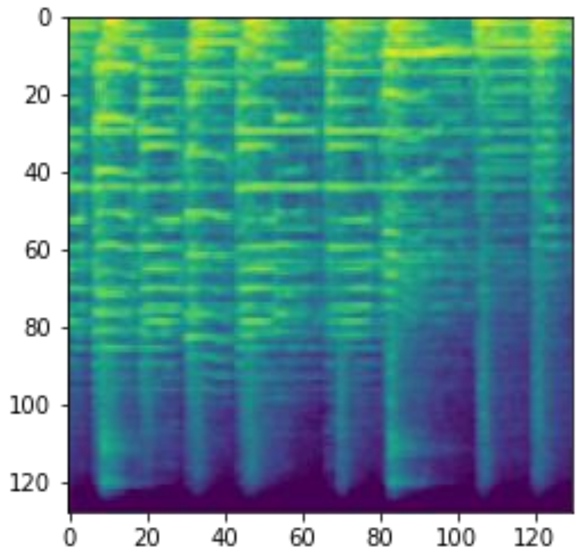

Figure 1 - Example mel-spectrogram of an audio file. The mel-spectrogram is a spectrogram of the audio where the frequencies are converted to the mel scale.

4. Convolutional-Recurrent Neural Network (CRNN)

For our next model, we considered a combination of CNN and RNN in series. A Recurrent Neural Network (RNN) is a modified neural network which has its own memory, i.e. it can remember things from its past set of inputs which can affect output of later stages of the network. RNNs were popularized due to their ability to remember each and every piece of information through time. The basic architecture of the CRNN model is given in Figure 2. We trained the CRNN model using Adam optimizer with a learning rate of 0.001 for 500 epochs.

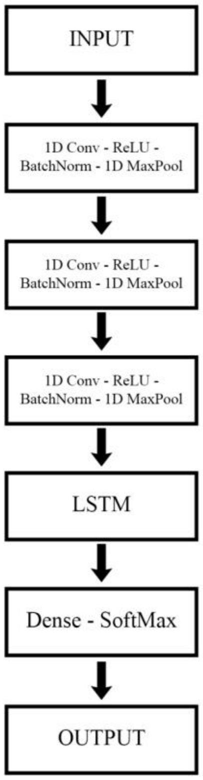
INPUT
1D Conv - ReLU - BatchNorm - 1D MaxPool
1D Conv - ReLU - BatchNorm - 1D MaxPool
1D Conv - ReLU - BatchNorm - 1D MaxPool
LSTM
Dense - SoftMax
OUTPUT

Figure 2 – Basic architecture of the CRNN model, a combination of CNN and RNN in series.

We already discussed the usage of CNN in image classification; we couple that with RNN's excellence in understanding long term sequential data in this model. It utilizes 1D convolution layers that perform convolution operation across the time dimension. Each convolutional layer extracts some features from the input images. ReLU activation function is used after the convolution operation. Batch normalization is done for standardization of the values. Max Pooling is applied to prevent overfitting. The output from the CNN is fed into an LSTM unit. LSTM (Long Short Term Memory) is an advanced RNN which overcomes the problem of vanishing gradients present in vanilla RNN, where it cannot remember long term dependencies. GRU (Gated Recurrent Unit) is also a viable option which can be used in place of LSTM.

5. Parallel Convolutional-Recurrent Neural Network (PCRNN)

Next we tried a parallel CNN-RNN model. The idea behind this model is that there might be a drawback in the series CRNN model. The RNN can only learn long term dependencies on the output of the CNN. It cannot do that on the original untouched raw data, because the data is always passing through a CNN before reaching the RNN. Hence we might consider a parallel system of CNN and RNN and then concatenate their learnings for the final output. The CNN can work on extracting edge-corner based features in the images, while in parallel the RNN can work on temporal time based features. The basic architecture of the PCRNN model is given in Figure 3. We trained the PCRNN model using Adam optimizer with a learning rate of 0.001 for 500 epochs.

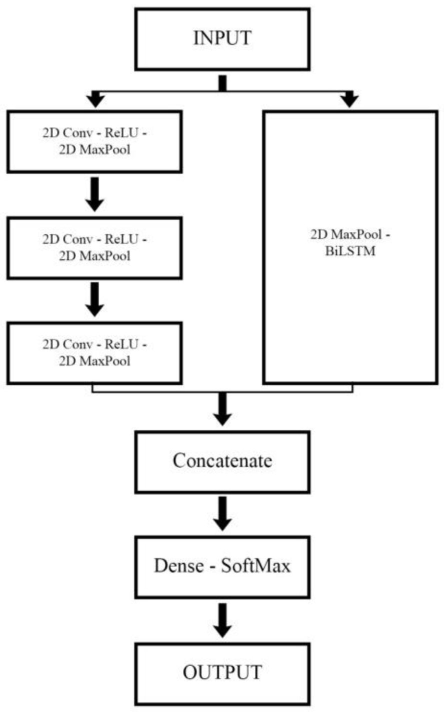
INPUT
2D Conv - ReLU - 2D MaxPool
2D Conv - ReLU - 2D MaxPool
2D Conv - ReLU - 2D MaxPool
2D MaxPool - BiLSTM
Concatenate
Dense - SoftMax
OUTPUT

Figure 3 - Basic architecture of the PCRNN model, a combination of CNN and RNN in parallel.

6. Ensemble Learning

Next we experimented on the dataset with ensemble learning. The meaning of the word ensemble is a group of separate entities viewed as a whole, rather than individually. Similarly, ensembling in machine learning constitutes all those methods that combine multiple learning algorithms to obtain a better performance than any one of the constituent algorithm working alone. As we are indulged in a classification type problem, our ensembling scenario will be multiple base classifiers incisively combined in order to produce a greater working accuracy than any one base classifier. The idea behind ensembling is that we can average the working of the models having low accuracy (weak learners) and models having high accuracy (strong learners) on the dataset.

Some commonly used ensemble techniques are bagging and boosting. Bagging operates on the divide and conquer rule. It breaks down the entire dataset into small sample datasets, fits a base model on each sample dataset and finally averages the result from all the models. Boosting is an iterative technique. It starts off with a base model training on the entire dataset and producing some output. In the next iteration, the boosting algorithm will seek to improve over the performance of the model in the previous iteration. It is a sequential process where each subsequent model attempts to correct the errors of the previous model.

6.1 AdaBoost

We used **AdaBoost**, one of the earliest boosting algorithms that were used. Its full form is **Adaptive Boosting**. **AdaBoost** helps you combine multiple weak learners into a single strong learner. The basic idea of **AdaBoost** is to properly assign weights to both the weak classifiers and each training example present in the subset the classifier trained on. Each weak classifier is trained using a random subset of the overall dataset. After the training is completed, training examples are assigned weights. Examples which were hard to classify are given a greater weight so that their probability of appearing in the next random subset increases. Each weak classifier is also assigned a weight based on its training accuracy. Generally, a classifier with 50% accuracy is given a weight of zero, and a classifier with less than 50% accuracy is given negative weight.

The mathematical formula for the final prediction hypothesis of **AdaBoost** is given in Equation 2.

$$H(x) = sign(\sum_{t=1}^{T} a_t h_t(x)) \quad \ldots (2)$$

Breaking down Equation 2, H(x) is the final predictive output. $h_t(x)$ stands for the output of weak classifier t for input x. It is also known as the weak hypothesis. a_t is the weight assigned to the classifier t for input x. The formula to calculate a_t is given in Equation 3. E is the error rate.

$$a_t = 0.5 * ln(\frac{1-E}{E}) \quad \dots (3)$$

As we can see in Equation 3, when the error rate E is greater than 0.5, the weight assigned to the classifier is negative. In contrast, when the error rate E is lesser than 0.5, the weight assigned to the classifier is positive. A graph of weight assigned to the classifier (a_t) versus error rate (E) is given in Figure 4. Initially all input training example has equal weightage. The weights change as the boosting progresses.

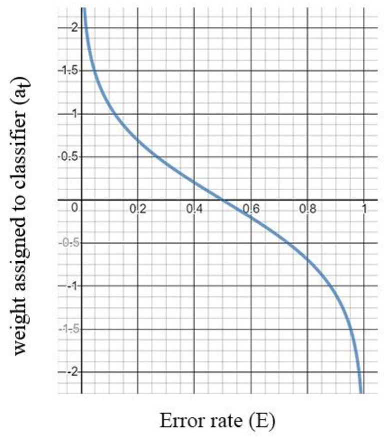

Figure 4 - a_t (weight assigned to the classifier) versus E (error rate) graph. a_t becomes 0 for E=0.5 and tends to reach infinity at E={0,1}.

3. Results and Discussions

The following results were obtained, listed in Table 2.

Models	Accuracy	Precision	Recall	F1 Score
SVC	46%	0.48	0.40	0.44
Logistic Regression	40%	0.61	0.64	0.62
ANN	67%	0.60	0.57	0.58
CNN	77%	0.90	0.48	0.63
CRNN	90%	0.91	0.88	0.89
PCRNN	88%	0.95	0.82	0.88
Without Ensemble	73%	0.68	0.64	0.66
With Ensemble (AdaBoost)	85%	0.78	0.95	0.86

Table 2 – Results obtained using our proposed models on the test dataset.

We applied the non-ensembling algorithm of Random Forest on our entire dataset using 70-30 train test split, which resulted in 73% accuracy on the test data. As we can see from the results, applying ensembling technique of AdaBoost on our dataset enhanced the accuracy a bit compared to it. But ensembling cannot surpass the CRNN model, which gives the best accuracy on our dataset. The accuracy is high because we used a relatively small dataset of 480 audio clips of 30s each. The use of more data may affect the accuracy. We noticed the class-wise performance of the models were starkly different.

Following are the accuracy and loss graphs of the respective models given in Figure 5-12. The combined accuracy and loss graphs are given in Figure 13 to show the graphical comparison between the various algorithms.

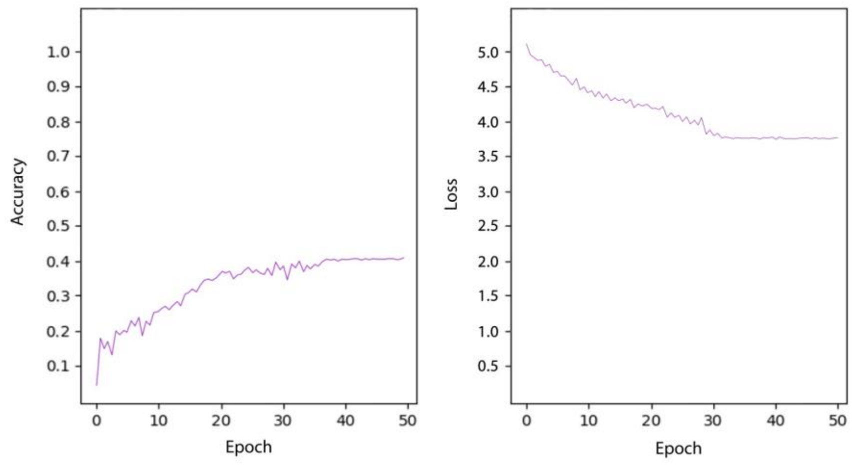

Figure 5 – Accuracy vs Epoch graph on the left & Loss vs Epoch graph on the right for the Logistic Regression model.

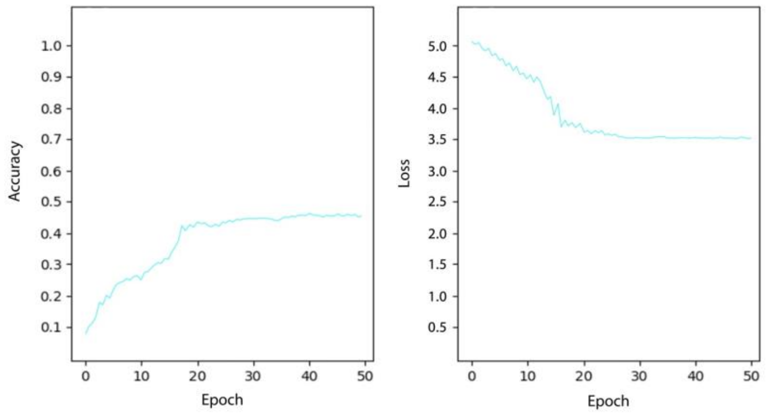

Figure 6 – Accuracy vs Epoch graph on the left & Loss vs Epoch graph on the right for the SVC model.

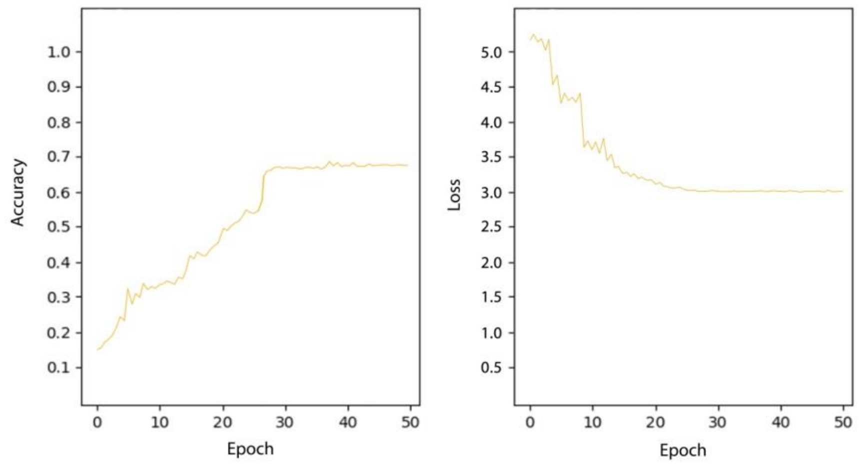

Figure 7 – Accuracy vs Epoch graph on the left & Loss vs Epoch graph on the right for the ANN model.

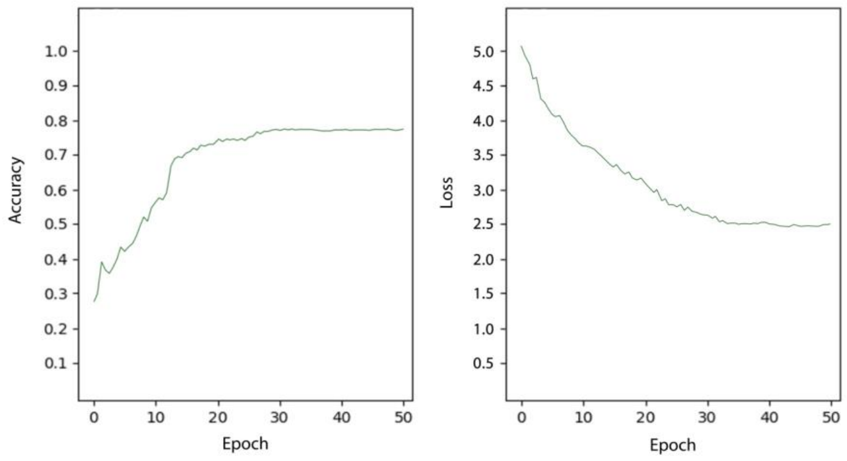

Figure 8 – Accuracy vs Epoch graph on the left & Loss vs Epoch graph on the right for the CNN model.

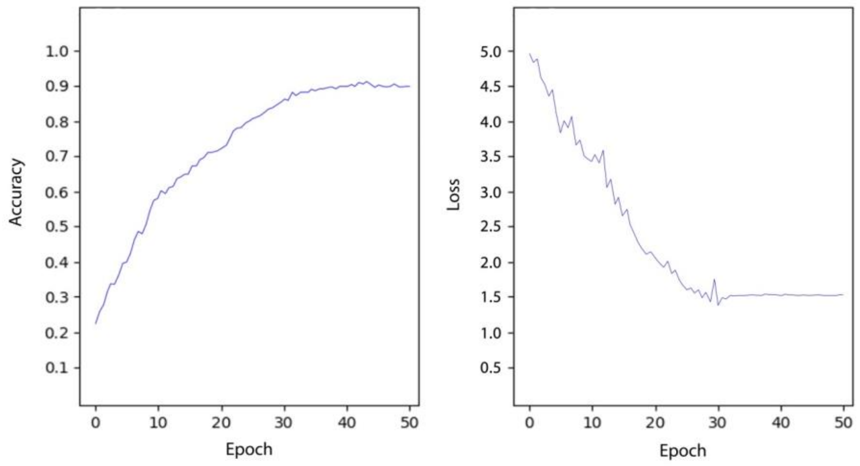

Figure 9 – Accuracy vs Epoch graph on the left & Loss vs Epoch graph on the right for the CRNN model.

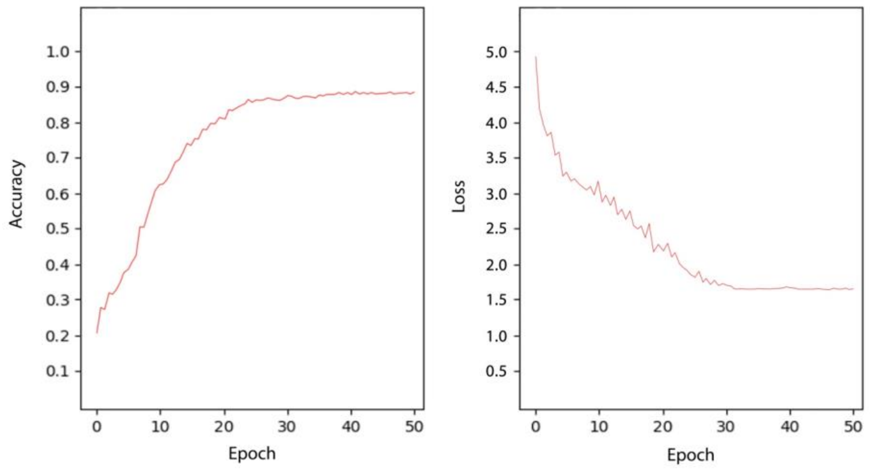

Figure 10 – Accuracy vs Epoch graph on the left & Loss vs Epoch graph on the right for the PCRNN model.

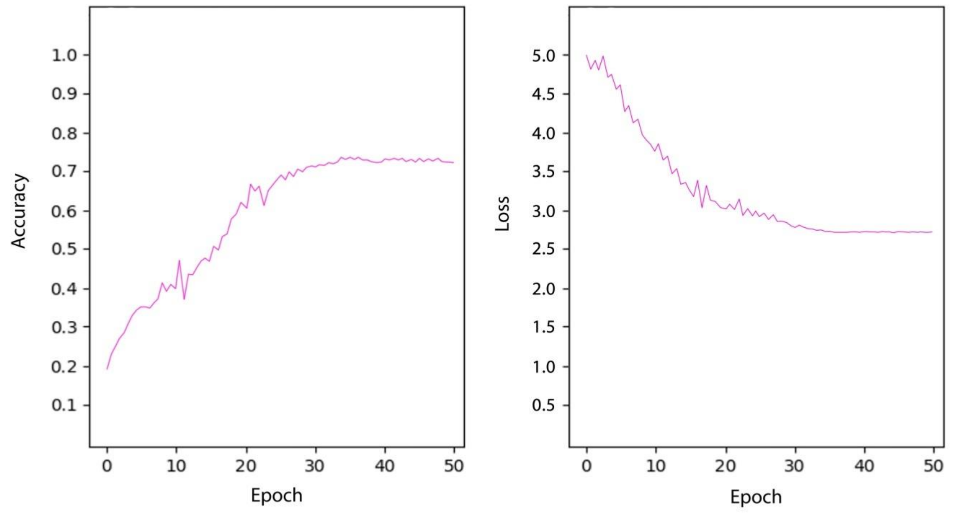

Figure 11 – Accuracy vs Epoch graph on the left & Loss vs Epoch graph on the right for the without ensemble model.

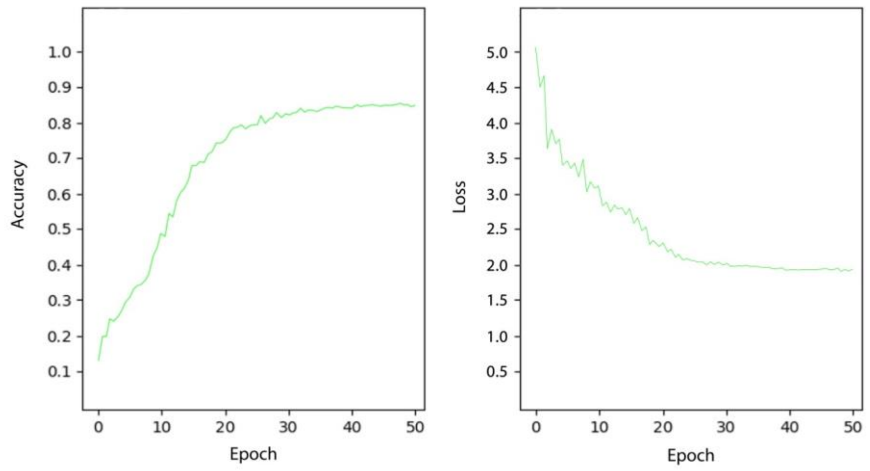

Figure 12 – Accuracy vs Epoch graph on the left & Loss vs Epoch graph on the right for the AdaBoost ensemble model.

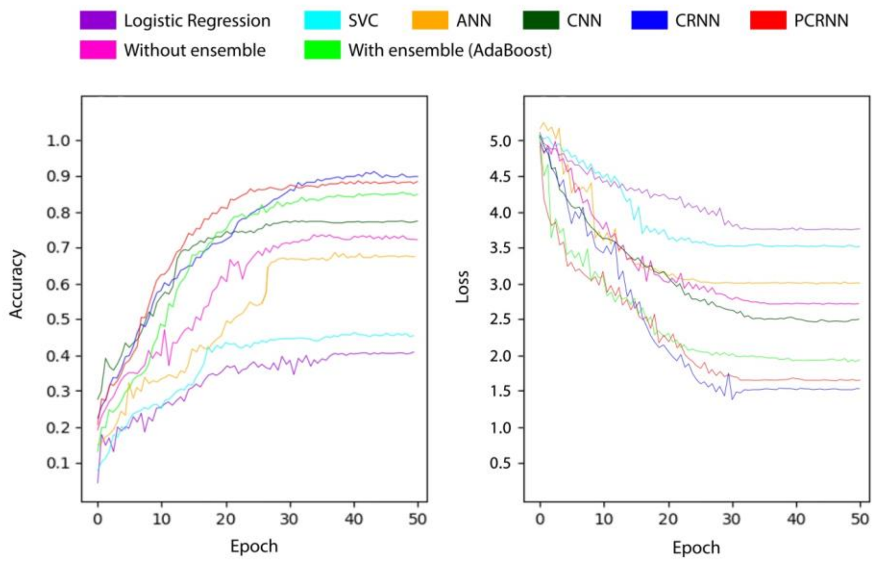

Figure 13 - Accuracy vs Epoch graph on the left & Loss vs Epoch graph on the right for all the models.
.

4. Conclusion

Changes in land use in the Pangkajene watershed caused an increase in the runoff coefficient (C) in 2009 by 0.360 and in 2018 by 0.372. The increase in the value of the flow coefficient will affect the increase in flood discharge. The results of the design flood discharge analysis show that in 2009 and 2018 an increase in flood discharge was obtained by 3.45% or ± 18.83 m3/s in each return period.

It is hoped that there will be similar and ongoing research on this matter, especially in watersheds (DAS) which have a high potential for changing land use into residential areas as a result of the supporting infrastructure development itself.

5. References

Annesi, P., Basili, R., Gitto, R., Moschitti, A., & Petitti, R. (2007). Audio Feature Engineering for Automatic Music Genre Classification. *RIAO*, 702–711.

Bertin-Mahieux, T., Ellis, D. P., Whitman, B., & Lamere, P. (2011). The million song dataset in Proceedings of the 12th International Society for Music Information Retrieval Conference. *Miami, October, 24*, 591–596.

Chang, K. K., Jang, J.-S. R., & Iliopoulos, C. S. (2010). Music Genre Classification via Compressive Sampling. *ISMIR*, 387–392.

Chathuranga, D., & Jayaratne, L. (2013). Automatic music genre classification of audio signals with machine learning approaches. *GSTF Journal on Computing (JoC), 3*, 1–12.

Choi, K., Fazekas, G., Sandler, M., & Cho, K. (2017). Convolutional recurrent neural networks for music classification. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2392–2396.

Feng, L., Liu, S., & Yao, J. (2017). Music genre classification with paralleling recurrent convolutional neural network. *ArXiv Preprint ArXiv:1712.08370*.

Jeong, I.-Y., & Lee, K. (2016). Learning Temporal Features Using a Deep Neural Network and its Application to Music Genre Classification. *Ismir*, 434–440.

Jothilakshmi, S., & Kathiresan, N. (2012). Automatic music genre classification for indian music. *Proc. Int. Conf. Software Computer App.*

Lidy, T., & Schindler, A. (2016). Parallel convolutional neural networks for music genre and mood classification. *MIREX2016, 3*.

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & Nieto, O. (2015). librosa: Audio and music signal analysis in python. *Proceedings of the 14th Python in Science Conference*, *8*, 18–25.

Pelchat, N., & Gelowitz, C. M. (2020). Neural network music genre classification. *Canadian Journal of Electrical and Computer Engineering*, *43*(3), 170–173.