Music Genre Classification Capabilities of Enhanced Neural Network Architectures

Joshua Engelkes
enge0479@morris.umn.edu
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

Abstract

With the increase of digital music audio uploads, applications that deal with music information have been widely requested by streaming platforms. Automatic music genre classification is an important function of music recommendation and music search applications. Since the music genre categorization criteria continually shift, data-driven methods such as neural networks have been proven especially useful to music information retrieval. An enhanced CNN architecture, the Bottom-up Broadcast Neural Network, uses mel-spectrograms to push music data through a network where important low-level information is preserved. An enhanced RNN architecture, the Independent Recurrent Neural Network for Music Genre Classification, takes advantage of the sequential aspects of music while combating typical RNN shortcomings. Both enhanced neural networks performed the best in their studies, but the IndRNN has a higher average classification accuracy.

Keywords: RNN, CNN, music genre classification

1 Introduction

The amount of digital music being uploaded onto the Internet is rapidly increasing. As music distribution technology becomes readily available to music artists, the need for music information applications increases. Applications such as music recommendation and music search have been implemented to manage music data. Music Information Retrieval (MIR) is an area of study that attempts to automatically classify and annotate music data. Demands for MIR have increased as the rate of Internet music uploads continues to swell.

Music classification has become essential to MIR by categorizing music tracks into cohorts. Music recommendation and search applications use music classification by finding new music tracks with similar characteristics to the tracks that the user enjoys. For example, if a user enjoys a piano concerto, a music recommendation application would recommend other piano concertos to the user. Genre classification, a core function of MIR, assigns a specific music genre to a music track. Expert annotation is impractical for large digital music collections, as 120,000 tracks were uploaded in the first quarter of 2023 [12]. Additionally, using experts for MIR tasks is notoriously expensive [6]. Therefore, genre recognition applications are a highly valuable and sought after MIR system.

Earlier versions of music genre classification algorithms mostly rely on handcrafted features for extraction and summarizing of music data [6]. Following feature extraction, the classification of features can be used to directly determine genre. These features require extensive tweaking when adding new genres or adjusting existing genres as they evolve. It is difficult, if not impossible, for handcrafted classification algorithms to automatically design appropriate features for new tasks. Applications that can automatically update as music and genres change are largely in demand. Therefore, it is beneficial to adopt a data-driven learning method rather than relying on rigid handcrafted methods.

Rafi et al. [10] chose two neural network architectures, enhanced for music genre classification, for an in-depth discussion. After discussing background information on related topics and techniques in Section 2, we will discuss two of the enhanced neural network architectures in Section 3 and Section 4. Afterward, we will analyze the accuracy of the enhanced architectures in music genre classification in Section 5. Finally, we will conclude in Section 6.

2 Background

2.1 Neural Networks

Neural Networks are collections of algorithms that are inspired by the communication between neurons in the human brain [7]. The primary idea of a neural network is to gradually improve the accuracy of the results through trial and reflection. The main architecture of a neural network consists of interconnected nodes. These nodes have multiple layers: the input layer, one or more hidden layers, and the output layer. The data is transformed as it travels through the hidden layers of a network, summarizing the data of the previous layer in each node before passing the data forward to nodes in other hidden layers or the final output layer. A generic calculation within a node is displayed below:

$h_{n,l} = \sigma(\sum_{i=1}^{N}(x_{i,l-1} * w_{i,l-1}) + b_{n,l})$

where $h_{n,l}$ is the calculation result in node n of layer l, $x_{i,l-1}$ is an input of the value of the i-th node in the previous layer, $w_{i,l-1}$ is a weight applied to each of the input from the previous layer, $b_{n,l}$ is a bias term added to the summation,

N is the number of neurons in the previous layer, and σ is an activation function (explained later in this section). The transformations of data within the hidden layers of a neural network are reliant on numeric weights and biases, which are initially designated random values. These weights and biases are adjusted through the process of *training*.

The main goal of training is to systematically readjust the weights and biases of the neural network so that the output of each layer directs the data to the appropriate conclusion [18]. In this instance, the weights and biases are readjusted to redistribute the probabilities of a music track being classified toward the correct genre. Training requires data with predetermined outputs. The neural network will readjust each weight and bias so that given the training data as inputs, the outputs of the neural network are accurate in as many cases as possible. To reflect how well a change in bias/weight affects the outcome of the network, an *error function* is used to calculate how close the actual outputs and network outputs are to each other. Using the direction of the derivative of the error function, the value of the weight/bias is adjusted by a preset percentage value: the *learning rate*. This process is repeated until the amount of change in a training round is extremely close to zero.

One of the primary tools of a neural network is an *activation function*, which generates the output of a layer. The networks in Section 3 and Section 4 will employ a *Rectified Linear Unit* (ReLU) as the activation function. A ReLU outputs its input unless the input is negative [4]. Every negative input in a ReLU outputs a 0.

Since neural networks can initially be sensitive to the random values assigned to weights and biases before training, many employ *Batch Normalization* (BN) to accelerate training. Batch normalization is a technique that standardizes the inputs of a layer [13]. The mean and standard deviation of the data in a layer are calculated assuming a normal distribution. The points are refit to a normal distribution with mean 0 and standard deviation 1. Afterward, the BN algorithm re-scales and offsets the input with two parameters adjusted through training. BN increases the speed of training and ensures that the data does not stretch, shrink, or offset in a way that could throw off a neural network.

For generating the conclusion of both neural network architectures in discussion, a softmax function is employed [2]. A softmax function takes a vector of values as input. The output of a softmax function is a vector of probabilities between 0 and 1. For music genre classification, each value in the output vector corresponds to a genre. For instance, if the value for the rock genre is 0.10, then the network has concluded that the digital music track has a 10% chance of being a rock song. The genre with the highest corresponding value is the output of the neural network.

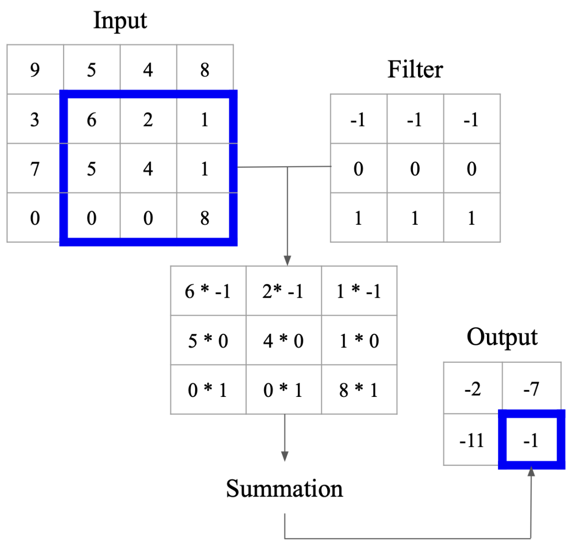

Figure 1. Convolutional Layer [11]

2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are commonly used neural networks that recognize patterns and objects in 2D data, such as images [15]. CNNs can classify images by calculating which of the given categories the image most likely belongs in. The input of a CNN is an image processed into a grid of values, where each value represents a portion of the initial image. For example, an image could be taken as input to a CNN if each pixel is converted into a numeric value (based on color or intensity).

One of the primary tools of a CNN is convolutional matrices. Convolutional matrices are employed in convolutional layers and are used as the weights of a CNN. A convolutional matrix is generally a square matrix with dimensions of odd numbers. For example, common sizes of convolutional matrices are 1x1, 3x3, and 5x5. A convolutional layer example is shown in Figure 1. The convolutional matrix is applied to every area of the input grid that is the same size. This matrix application multiplies each value in the convolutional matrix with each corresponding value in the designated area. The sum of the resulting products is the corresponding output to the initial value. The output of this process is a new grid of summed products. In the example, in Figure 1, the output grid doesn't have the same dimensions as the input grid. There are only four 3x3 sub-grids in a 4x4 grid, so there are only four values in the output. Through the process of applying multiple convolutional layers, the original data can be scaled-down enough to be easily classified.

Throughout a CNN, a non-linear transformation called *pooling* is utilized to condense or summarize the data [3]. Pooling is applied in a CNN in various ways:

- *Max Pooling* takes the maximum value from a specified size of data. The map of maximum values from every

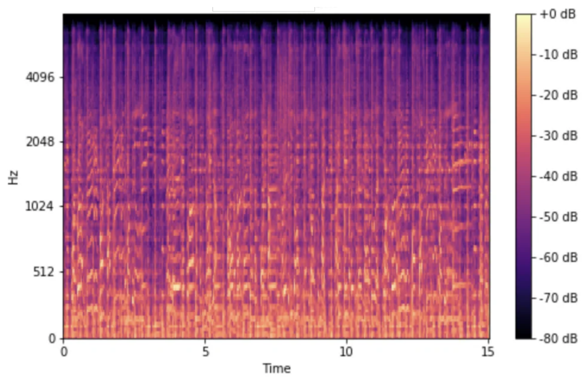

Figure 2. Mel-Spectrogram [11] y-axis: frequency (pitch) in hertz, x-axis: time in seconds, color: decibels (loudness)

instance of the original specified size of data is the output of max pooling.

- *Average Pooling* works similarly to max pooling but takes the average value from the specified size of data.
- *Global Average Pooling* takes a grid as input and outputs the average of its values.

CNNs have been used for MIR tasks such as music recommendation and automatic tagging. A *mel-spectrogram*, displayed in Figure 2, is a 2D visual representation of audio frequencies over a period of time and is a popular input of CNNs for MIR tasks [10]. A mel-spectrogram displays the loudness of frequencies throughout a music track [14]. Each horizontal line of pixels corresponds to a specific frequency. Each vertical line of pixels corresponds to a specific time in the music track.

Although previous CNN architectures have achieved good performance in music genre classification, there are two main issues:

1. Many music genre classification network structures focus on abstracting high-level features in each layer [9], like the vertical and horizontal trends in the mel-spectrogram denoting musical pulse and harmonic progression respectively. This naturally creates a deficit of lower-level features, like the purity of a frequency or the steepness at which a frequency is left. These have been shown to contain critical decision-making information [5].
2. Previous music genre classification applications predict genre from the same scale of time and frequency; however, if the duration of input or distribution of frequencies displayed in the mel-spectrogram is changed after training, the classification system would be affected.

Based on the reasoning above, Liu et al. proposed a novel CNN model that can make full use of low-level/high-level information while malleable to the duration of the input and distribution of frequencies: the Bottom-up Broadcast Neural Network (Section 3) [9].

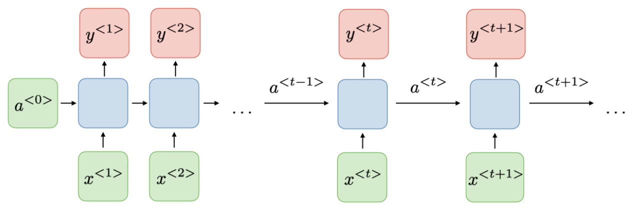

Figure 3. Traditional RNN Architecture [1].
$y^{<t>}$: the output of the recursive node.
$a^{<t>}$: a sequence of hidden states of the network at previous time steps/recursive input to future nodes.
$x^{<t>}$: input to node at time step t

2.3 Recurrent Neural Networks

The *Recurrent Neural Network* (RNN) is a brand of neural network architecture that is designed to model sequential data [8]. The information in an RNN cycles through a hidden layer where every sequential output is influenced by a conglomeration of the preceding output calculations.

A traditional architecture of a *recursive network* in an RNN is shown in Figure 3. RNNs are composed of recursive neurons that generate a series of outputs, or *hidden states*, from an input stream and their own output (recursive inputs) from previous time steps. For each hidden layer (blue squares) at time step t, a traditional RNN will apply the same type of weights, biases, and activation functions to each value in the input sequence $x^{<t>}$ as well as the recursive input sequence $a^{<t>}$. The hidden layer will output the sequence for the output layer $y^{<t>}$, which will be added to the recursive input sequence for the next step in the hidden layer $a^{<t>}$.

Although RNNs are widely used for sequential data, the traditional RNN is vulnerable to undesirable tendencies while training [17]. An RNN applies weights, biases, and activation functions to an input in every step of a hidden layer. Since the calculation of hidden states of neurons applies the same weights and biases to each time step, an RNN architecture must be carefully designed. An increase or decrease in weights could easily nullify the effect of inputs or increase them at a dramatic rate. For this reason, Wu [17] applies the Independent Recurrent Neural Network for music genre classification (Section 4), an enhanced RNN architecture.

3 Bottom-up Broadcast Neural Network

Liu et al. [9] constructed an enhanced CNN architecture, the *Bottom-Up Broadcast Neural Network* (BBNN), for music genre classification. The BBNN contains interconnected building blocks, which ensure the maintenance of low-level information to the decision layer. This information would have been lost in other CNN architectures. The BBNN also

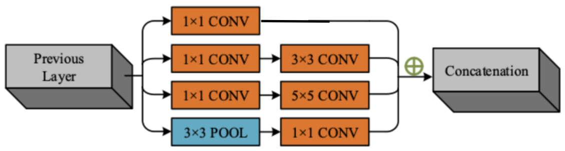

Figure 4. An inception block [9]

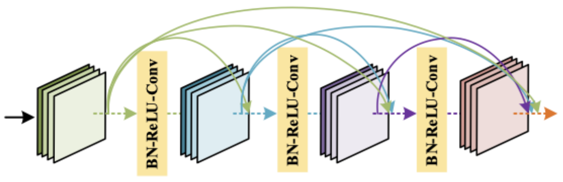

Figure 5. The BBNN broadcast module dense connectivity, the data is transmitted both through the inception blocks (squares) and skipping the inception blocks (arrows overhead) [9]

contains a Broadcast Module (BM) made of a network of blocks that apply different sizes of convolutional matrices to the same input and concatenate the results. This captures patterns of different sizes and overlays them atop one another.

The BBNN is composed of inception blocks, as seen in Figure 4. The inception blocks are stacked atop one another as the basic pattern extraction unit. The BM consists of 3 identical inception blocks. As displayed in Figure 5, the input of the l-th block is the concatenation of the output produced by blocks $1, \ldots, l-1$ and the initial input to the BM. Before every convolutional layer, batch normalization and rectified linear activation are applied. This standardizes the grid of values while ensuring that no values are redistributed below zero. Each inception block, in Figure 4, sends the input layer in 4 directions. Three paths contain 1x1 matrix convolutions before another pass of matrix convolutions of size 1x1, 3x3, and 5x5. The fourth path contains a max-pooling layer with matrix size of 3x3, which is used to reduce the resolution of the grid. The output of the inception block is the concatenation of the four paths results.

Dense connection paths are utilized to connect all inception blocks, which transmits the input/output layers forward to all subsequent blocks in the BM. These dense connection paths propel the low-level information forward.

The entire map of BBNN is displayed in Figure 6. There are 4 main parts of the BBNN:

1. Shallow pattern extraction layer – information is extracted from the mel-spectrogram, and initial patterns are extracted in a 3x3 convolution layer. The result is sent through a BN, ReLU, and a max-pooling operation to standardize and scale down the amount of information in the input.
2. BM – Previously explained
3. Transition layer – BN, ReLU, and average-pooling,
4. Decision layer – global average pooling takes the average of each grid to form a vector that is fed into a softmax function, producing a vector of probabilities with each probability corresponding to a music genre.

4 Independent Recurrent Neural Network

The *Independent Recurrent Neural Network* (IndRNN) takes an input of values as a sequence of sequences, giving a 2D representation of values [17]. The input to the IndRNN is similar to the BBNN. Each sequence of values represents the volume of a frequency at a certain point in the digital music track. When placed next to each other in order of time, the sequences resemble a 2D grid of values, similar to the input of the BBNN. Each sequence for a certain time is the input to a layer of the IndRNN.

Each neuron in an IndRNN processes independently of other neurons in the layer. In a typical RNN, the hidden layer input could contain information from previous time steps of other neurons in the same layer. In the IndRNN, the input stream into each neuron may be previously generated through a neuron of another layer, but no individual neuron can see another neurons hidden states within the same layer. The hidden state of a neuron n at time t in an IndRNN can be expressed as:

$$h_{n,t} = \sigma(w_n x_t + u_n h_{n,t-1} + b_n)$$

- $h_{n,t}$: the t-th hidden state of neuron n
- σ: ReLU
- x_t: the t-th value from the input layer
- w_n, u_n: weights applied to the input from the input layer and previous hidden state respectively
- b_n: a bias

Connections are formed between neurons when two or more layers of an IndRNN are stacked atop one another. Connections between layers are more similar to the inputs of a typical neural network, where a summation of multiple weighted inputs is used to pull information between layers.

The architecture of IndRNN (Figure 7) employs batch normalization to standardize the inputs of each IndRNN layer. Finally, a softmax function is used to create a distribution of probabilities over the genres. The genre with the highest probability is the output of the IndRNN.

5 Comparison

5.1 Testing Data Set

The enhanced CNN and RNN architectures both use the GTZAN data set during training. This data set contains mel-spectrograms of music tracks that span across various music genres. The GTZAN data set is a collection of music created to test an early automatic music genre classification system [16]. The data set includes 20 musical genres, where each

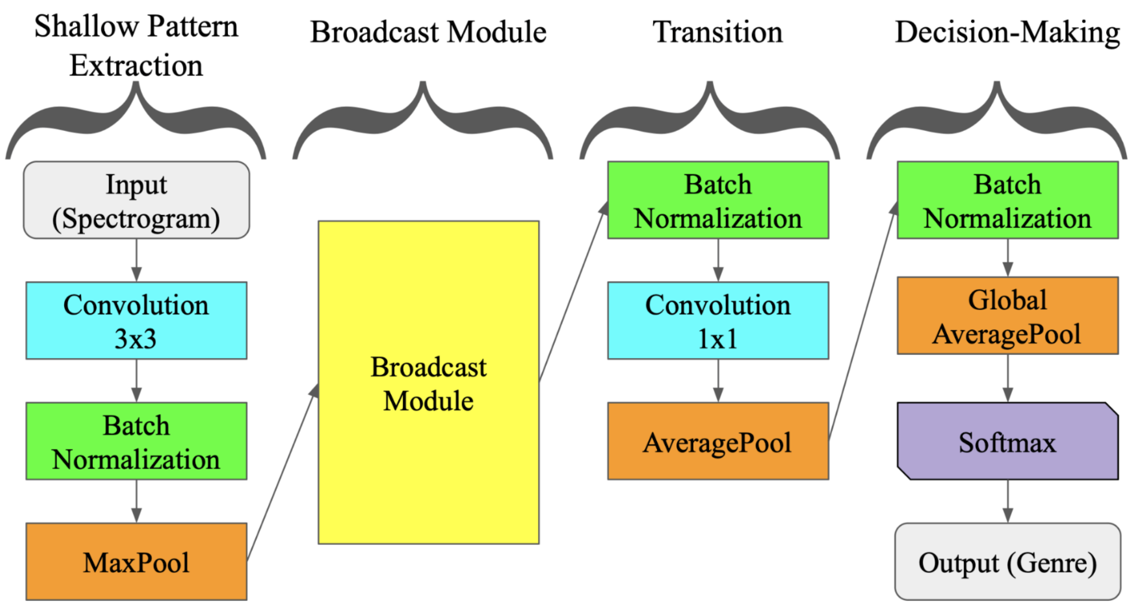

Figure 6. The BBNN full network [9]

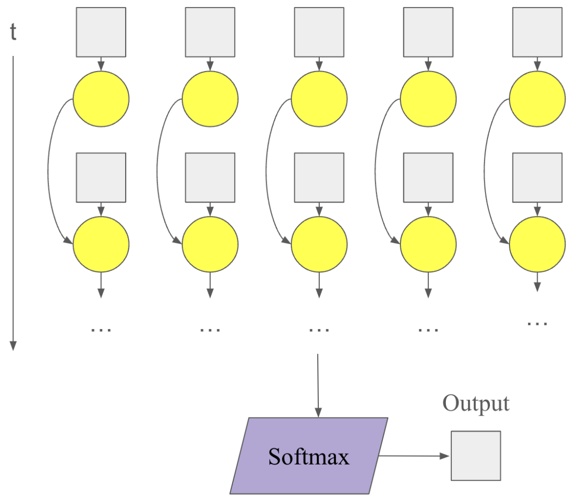

Figure 7. IndRNN architecture [17], moving right across timesteps, moving down across neurons

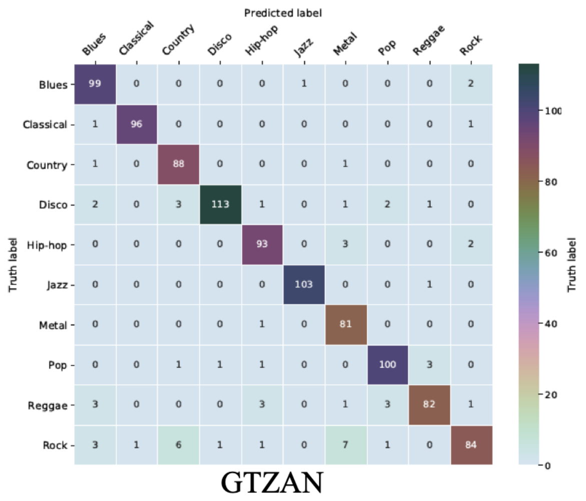

Figure 8. BBNN Results per genre [9]

genre contains 100 30-second track examples for testing. The genres in the GTZAN data set includes classical, country, disco, hip-hop, jazz, rock, blues, reggae, pop, and metal.

5.2 Results

BBNN and IndRNN were trained and tested using GTZAN data [16] set alongside different deep learning models with similar architectures in their respective papers. The input of the BBNN were mel-spectrograms displaying 30 seconds of excerpts from songs [9], while the IndRNN took a collection of stacked sequences where each sequence was related to a corresponding frequency [17]. The data was split into 2 groups to evaluate classification accuracy across experiments. 90% of the songs in the GTZAN data set, or 900 songs, were used for training while the other 10%, or 100 songs, were used for testing. The classification accuracy is the percentage of the testing data set that the neural network sorted into the accurate genre.

In Liu et al. 2021 [9], the BBNN was tested against 5 alternative CNN architectures as well as a traditional, non-data-driven method. During training, the training data was

	BBNN	IndRNN
Average Classification Accuracy	93.9%	96%

Table 1. Classification Accuracies [9] [17]

passed through the neural network 100 times, or 100 *epochs*. The BBNN outperformed all of the architectures in its introductory paper with an average classification accuracy of 93.9% on the GTZAN data set. The genre-specific results of the BBNN on the GTZAN data set are displayed in Figure 8. The BBNN has difficulty classifying the rock genre correctly, commonly misclassifying a rock song as a country or metal song. The BBNN performed better on the alternative data sets in the study, but the GTZAN data set is the only data set in common between the BBNN and IndRNN study. There is no information regarding the training time of the BBNN.

In Wu et al. 2018 [17], the IndRNN was tested against a base RNN architecture as well as a *Long Short-Term Memory* (LSTM) network, another RNN variant. The number of training epochs in this study varied from 5 to 200 epochs in factors of 5. The classification accuracy for the IndRNN study is the average of the accuracies between all epoch amounts. The average classification accuracy of the IndRNN was 96%, outperforming the RNN averaging 89% accurate, but not the LSTM averaging 97% accurate. The IndRNN performed optimally at 100% accuracy with 75 or more epochs. LSTM performed better than the IndRNN at lower epoch amounts, which ultimately led to a higher average classification accuracy. LSTM performed near optimally above 75 epochs as well.

The average epoch calculation time of the IndRNN was 0.23 seconds per epoch, beating both the RNN at averaging 0.27 seconds per epoch and the LSTM at averaging 0.68 seconds per epoch. Overall, the IndRNN performed the best of the RNN variants in terms of classification accuracy and training time on the GTZAN data set.

There is a lack of information about training times for the BBNN, and therefore the only common result between the two enhanced network studies is in classification accuracy. The overall classification accuracies are in Table 1. The IndRNN outperformed the BBNN in average classification accuracy, but the number of training epochs is not held constant during the IndRNN training phase. The classification accuracy of the IndRNN over epoch number is displayed in Figure 9. If the number of epochs is held constant at 100 epochs, as is done in the BBNN study, the IndRNN has a classification accuracy of 100%. Furthermore, the IndRNN has a classification accuracy of 100% for any amount greater than 75 epochs.

6 Conclusion

In this paper, we have compared two enhanced neural networks on their abilities to classify digital music into genres as outlined in Rafi et al. [10]. The BBNN, an enhanced CNN, had a classification accuracy of 93.7%. The IndRNN, an enhanced RNN, has a classification accuracy of 96%. At 100 epochs during training, the IndRNN has a perfect classification accuracy. Therefore, the IndRNN is a more suitable tool for classifying digital music into genres than the BBNN.

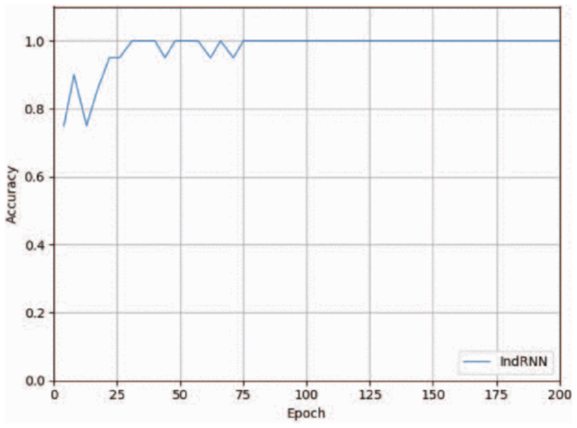

Figure 9. IndRNN accuracy by number of epochs (steps of 5 from 5 to 200) [17]

Acknowledgments

I want to thank my senior seminar advisor, Elena Machkasova, for guidance and feedback during the development of this paper. I would also like to thank Nic McPhee for structuring and leading the senior seminar course. In addition, I would like to thank UMM alum Justin Mullin for his vital and detailed feedback.

References

[1] Afshine Amidi and Shervine Amidi. [n. d.]. Recurrent Neural Networks Cheatsheet. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks Acessed: 11/29/23.

[2] Jason Brownlee. [n. d.]. Softmax Activation Function with Python. https://machinelearningmastery.com/softmax-activation-function-with-python/ Acessed: 11/18/23.

[3] Jason Brownlee. 2019. A Gentle Introduction to Pooling Layers for Convulutional Neural Networks. https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/ Acessed: 10/13/23.

[4] Jason Brownlee. 2020. A Gentle Introduction to the Rectified Linear Unit. https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/ Acessed: 10/13/23.

[5] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. 2017. Transfer learning for music classification and regression tasks. arXiv:1703.09179

[6] Zhouyu Fu, Guojun Lu, Kai Ming Ting, and Dengsheng Zhang. 2011. A Survey of Audio-Based Music Classification and Annotation. *IEEE Transactions on Multimedia* 13, 2 (2011), 303–319. https://doi.org/10.1109/TMM.2010.2098858

[7] IBM. [n. d.]. What are Neural Networks? https://www.ibm.com/topics/neural-networks Acessed: 10/26/23.

[8] Debasish Kalita. 2023. A Brief Overview of Recurrent Neural Networks (RNN). https://www.analyticsvidhya.com/blog/2022/03/a-brief-overview-of-recurrent-neural-networks-rnn/ Acessed: 10/24/23.

[9] Caifeng Liu, Lin Feng, Guochao Liu, Huibing Wang, and Shenglan Liu. 2021. Bottom-up broadcast neural network for music genre classification. *Multimedia Tools and Applications* 80 (2021), 1–19. https://doi.org/10.1007/s11042-020-09643-6

[10] Quazi Ghulam Rafi, Mohammed Noman, Sadia Zahin Prodhan, Sabrina Alam, and Dip Nandi. 2021. Comparative Analysis of Three Improved Deep Learning Architectures for Music Genre Classification. *International Journal of Information Technology and Computer Science* 13 (04 2021), 1–14. https://doi.org/10.5815/ijitcs.2021.02.01

[11] Leland Roberts. 2020. Understanding the Mel Spectrogram. https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53 Acessed: 10/5/2023.

[12] Ronan Ruiz. 2023. New Report Shows 120,000 Tracks Are Being Uploaded To Streaming Services Per Day. https://music.mxdwn.com/2023/05/26/news/new-report-shows-120000-tracks-are-being-uploaded-to-streaming-services-per-day/ Acessed: 10/26/23.

[13] Shipra Saxena. 2023. Introduction to Batch Normalization. https://www.analyticsvidhya.com/blog/2021/03/introduction-to-batch-normalization/ Acessed: 10/13/23.

[14] Barry Truax. 1999. MEL. https://www.sfu.ca/sonic-studio-webdav/handbook/Mel.html Accessed: 10/2/2023.

[15] Barry Truax. 1999. What's the difference between CNN and RNN? https://www.sfu.ca/sonic-studio-webdav/handbook/Mel.html Accessed: 10/2/2023.

[16] George Tzanetakis and Perry Cook. 2002. Musical Genre Classification of Audio Signals. *IEEE Transactions on Speech and Audio Processing* 10 (08 2002), 293 – 302. https://doi.org/10.1109/TSA.2002.800560

[17] Wenli Wu, Fang Han, Guangxiao Song, and Zhijie Wang. 2018. Music Genre Classification Using Independent Recurrent Neural Network. In *2018 Chinese Automation Congress (CAC)*. 192–195. https://doi.org/10.1109/CAC.2018.8623623

[18] Victor Zhou. 2019. Training a Convolutional Neural Network From Stratch. https://towardsdatascience.com/training-a-convolutional-neural-network-from-scratch-2235c2a25754 Acessed: 10/10/23.