Recurrent Neural Networks with Attention for Genre Classification

Jeremy Irvin
Stanford University
jirvin16@stanford.edu

Elliott Chartock
Stanford University
elboy@stanford.edu

Nadav Hollander
Stanford University
nadavh@stanford.edu

1 Introduction

Automated genre classification of songs through audio input alone has depended on manual feature engineering to achieve good performance [1]. Recently, Recurrent Neural Networks (RNN) have shown success in many sequential tasks including Machine Translation and Speech Recognition, out-performing state-of-the-art systems which use hand-crafted features [2,3,4]. In this direction, we present a novel application of recent advances in RNNs to genre classification on the GTZAN Genre Collection dataset[1]. More explicitly, we use Long Short Term Memory Networks (LSTM) with (and without) a soft attention mechanism [4] to sequences of audio signals in order to classify songs by genre. The best model is an LSTM without attention, achieving 79% accuracy on the test set.

2 Related Work

Musical genre classification is an increasingly prevalent problem in the field of music information retrieval. Quick, accurate genre classification has clear applications in the highly lucrative field of intelligent music recommendation systems. One such example is Spotify, which has a full-time team of "data alchemists" who are dedicated to the sole task of classifying their 60 million songs into about 1,000 sub-genres [5]. Implementing these procedures in practice requires efficient algorithms which can be run in real-time.

Classifying a song by genre has traditionally relied on feature engineering which fall under three broad classes: timbral texture features, rhythmic features, and pitch content features [6]. Examples include spectral zero crossings or rhythmic beat detection [1]. Other attempts at extracting auditory features of music, such as tempo and acoustic character have yielded auspicious results in the field of automatic genre classification [1,7]. Additionally, a recently popular approach to the genre classification problem relies on features known as Mel Frequency Cepstral Coefficients (MFCCs) [8]. These features are then input into an off-the-shelf classifier, such as Support Vector Machines (SVM), Nearest Neighbor Classifiers, or Gaussian Mixture Models such as Linear Discriminant Analysis. However, many of these features tend to be intractable to compute in settings with large quantities of data.

Recently, audio spectrograms have successfully been used with RNNs in speech recognition, significantly simplifying much of the previous pipeline [9]. Inspired by this success, we apply recurrent neural networks on spectrograms of audio song data to the music genre classification problem in an attempt to simplify and expedite the classification pipeline.

3 Approach

3.1 Problem Statement

Our model can be formulated as a simple multivariate classification problem on variable-length sequences as follows:

Given a variable-length sequence of feature vectors $x = x_1, ..., x_T$, assign x a class label from $c_1, ..., c_C$ (note that N varies across all input vectors x, but the number of classes C and dimension D of x_t remain fixed). Each x_t is a feature vector capturing aspects of the tth 'timestep' of the sequence x. In our setting, x_t is a vector which represents a multi-second audio clip of the song (see Section **??**).

Hence we define our classification problem as

$$\hat{y} = f(x)$$

[1] Available at http://marsyas.info/downloads/datasets.html

where f is an arbitrary function mapping and $\hat{y}$ is the predicted class for input x. We explore choices of f from multiclass SVM, multinomial logistic regression, and Vanilla MLP as baselines, to Vanilla RNN, Vanilla LSTM, and LSTM with a soft attention mechanism.

3.2 Models

In this section we will briefly describe the more complex models that were implemented for the task. Note that all bias vectors are ignored for notational convenience. For each of the models below, the loss $\mathcal{L}$ is the cross entropy loss between the predicted distribution over classes $\tilde{y}$ and true (one-hot) distribution over classes. Note that these models have been adapted for our particular setting (*many to one* mapping versus a common *many to many* mapping as in Machine Translation, for example).

3.2.1 Vanilla RNN

Given a sequence of vectors $x_1, ..., x_T$, a Vanilla RNN with L layers computes, at a single timestep t and layer l,

$$h_t^1 = Wx_t, \quad h_t^l = \tanh(W_{hh}^l h_{t-1}^l + W_{xh}^l h_t^{l-1}), \quad y_t = W_{hy} h_t^L$$

where h_t^l is the hidden vector at the tth timestep and layer l, and $W, W_{hh}, W_{xh}, W_{hy}$ are parameters to the model. In our classification setting (where the output sequence is a single class), we compute

$$\hat{y} = \arg\max_{c \in C} (\tilde{y})_c$$

where $(\cdot)_c$ denotes the cth index operator, and $\tilde{y} = \text{softmax}(y_T^L)$. All vectors y_t^L are not computed except for y_T^L.

3.2.2 Vanilla LSTM

Given a sequence of vectors $x_1, ..., x_T$, a Vanilla LSTM with L layers computes, at a single timestep t and layer l,

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

$$c_t^l = f \odot c_{t-1}^l + i \odot g, \quad h_t^l = o \odot \tanh(c_t^l), \quad y_t = W_{hy} h_t^L$$

where $\odot$ indicates element-wise multiplication, i, f, o, g, c_t^l are the H dimensional input gate, forget gate, output gate, block gate, and context vector respectively at the tth timestep and lth layer. We compute $\tilde{y}$ and predict as above.

3.2.3 LSTM with Soft Attention

A soft attention mechanism is an additional layer added to the LSTM defined as follows

$$\alpha_t = \frac{\exp(c_T^L \cdot h_t^L)}{\sum_{t'} \exp(c_T^L \cdot h_{t'}^L)}, \quad c = \sum_{t=1}^{T} \alpha_t h_t^L$$

$$\tilde{h} = \tanh(W_c[c; c_T^L]), \quad \tilde{y} = \text{softmax}(W_{hy}\tilde{h})$$

where c_T^L is the memory state output output by the cell at the last timestep T and topmost layer L, h_t^L is the hidden state output by the LSTM at time t and topmost layer L, c is the context vector, $\tilde{h}$ is the attentional hidden state (where $[\cdot;\cdot]$ is concatenation and W_c are learnable parameters), and $\tilde{y}$ is the predicted distribution over classes for the input x. Intuitively, this structure allows the model to learn to attend to particular parts of the input sequence when performing the classification.

A visualization of the general model architecture can be seen in Figure 1. The attention layer is optional in our experiments, and each x_i is a 513-dimensional vector composing a single training example $x_1, ..., x_T$ with T fixed to 22 in all of the training examples. The value of T, however, can be variable if desired.

4 Data

Using the GTZAN Genre Collection [1], we start with a set of 1000 30-second song excerpts labelled into one of 10 genres: Blues, Classical, Country, Disco, Hip-Hop, Jazz, Metal, Pop, Reggae, and Rock. In order to reduce dimensionality, we downsample the songs to 4000 Hz, and further split each excerpt into 5-second clips. For each clip, we compute a spectrogram using Fast Fourier Transforms, giving us 22 timestep vectors of dimensionality 513 for each clip. Spectrograms separate out component audio signals at different frequencies from a raw audio signal, and provide us with a tractable, loosely structured feature set for any given audio clip that is well-suited for deep learning techniques. Overall, this gives us 6000 different song excerpts to work with, each of which has 22 timesteps vectors with 513 values representing spectral amplitude at a range of frequencies. We use 4200 songs for the training set, 600 songs for the validation set, and 1200 songs for the test set.

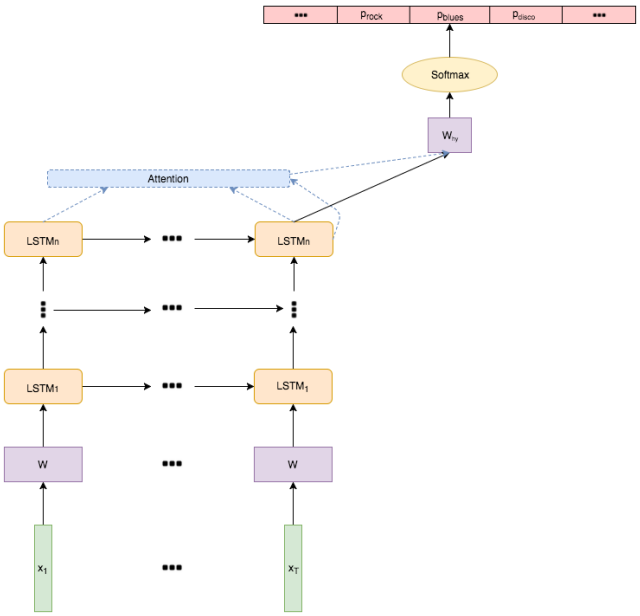

Figure 1: General network architecture. n denotes the number of layers and T the length of the input sequence. The top attention layer is optional. Each LSTM cell is replaced with a regular RNN cell in some of the experiments.

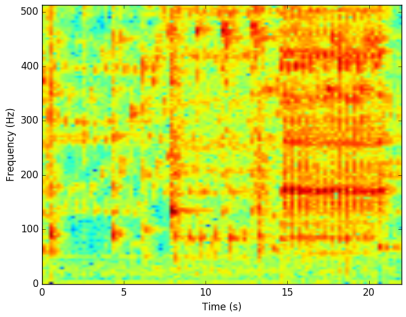

Figure 2: Spectrogram generated from an excerpt of a Jazz song in the training set.

5 Results and Analysis

5.1 Experiments

We have implemented several baselines using the concatenation of the 5 second feature vectors (each vector is $22 \cdot 513 = 11286$-dimensional) as input, including variants of multiclass SVM, multinomial logistic (aka softmax) regression, and neural networks with a softmax layer as the final output layer.

Many different recurrent architectures were tested with differing number of layers, cell sizes, and attention types (different scoring functions; the best scoring function is the one described in Section 3.2.3 and is the only one described in this paper). The best architectures for each broad type (Vanilla RNN, Vanilla LSTM, LSTM with Attention) were all tested with these hyperparameter variations. For every experiment, the standard SGD optimizer was used with a learning rate of 1. Data was passed in batch sizes of 128. The model was chosen using early stopping based on the validation set. Further regularization was implemented using Dropout of 0.2 after each LSTM (or RNN) cell and the projection into softmax layer. Gradients were clipped at a norm of 5 to combat the exploding gradient problem. Each model was run in TensorFlow on a GPU and finished in less than 30 minutes.

5.2 Results

Figure 3 outlines the baseline model evaluations on the test set[2]. The neural network architectures significantly outperform the other baselines, with the best model achieving 70% on the test set.

A few of the common (external, meaning not implemented in this work) baselines on the GTZAN dataset are shown in Figure 4. There is no reported state-of-the-art on the dataset as methods of evaluation have high variance across the literature, making results hard to compare. The listed accuracies in Figure 4 use very similar evaluations (a subset of the dataset is set aside for test evaluation) as this work, and thus are roughly comparable to the numbers in Figure 3 and Figure 5.

Finally, a summary of all experimental results are shown in Figure 5. The best 1 and 2 layer Vanilla RNN, Vanilla LSTM, and LSTM with Attention are shown[3]. The best architecture was a Vanilla LSTM with 2 layers and 250 dimensional cells, achieving a high accuracy of 79% on the test set. This beats the highest baseline by 9%, and compares with many of the external baselines with hand-engineered features.

Model	Test Accuracy
Linear multiclass ovr SVM: L2 Regularization, Squared Hinge Loss	0.343
Linear multiclass ovr SVM: L1 Regularization, Squared Hinge Loss	0.353
RBF kernel multiclass ovr SVM	**0.405**
Polynomial kernel multiclass ovr SVM	0.169
Sigmoid kernel multiclass ovr SVM	0.0958
ovr Logistic Regression:	0.3825
Multinomial Logistic Regression	**0.3892**
2 Layer Neural Network, 125 Hidden Dimension, ReLu Nonlinearity	0.68
2 Layer Neural Network, 125 Hidden Dimension, Sigmoid Nonlinearity	**0.7**

Figure 3: Summary of our baseline results on test set (1200 examples). ovr stands for *one-versus-rest* in contrast with a *one-versus-one* scheme. The best SVM, logistic regression, and neural network classifiers are bolded.

Model	Test Accuracy
Bergstra et. al [11]	**0.825**
Li et. al [12]	0.785
Lidy et. al [13]	0.768
Benetos et. al [14]	0.75
Holzapfel et. al [15]	0.74
Tzanetakis et. al [7]	0.61

Figure 4: Examples of common baselines on the GTZAN dataset [6]. The best external baseline is bolded, and uses AdaBoost on many different extracted features [19].

Model	Test Accuracy
Vanilla RNN: 1 Layer, 125 Dimensional Cell	0.54
Vanilla RNN: 2 Layer, 125 Dimensional Cell	**0.655**
Vanilla LSTM: 1 Layer, 250 Dimensional Cell	0.745
Vanilla LSTM: 2 Layer, 250 Dimensional Cell	**0.79**
LSTM with Attention: 1 Layer, 500 Dimensional Cell	0.7316
LSTM with Attention: 2 Layer, 60 Dimensional Cell	**0.758**

Figure 5: Summary of experimental results. The best vanilla RNN, vanilla LSTM (without attention), and LSTM with attention are bolded. Only the best 1 and 2 layer networks are shown in each case.

5.3 Analysis

The LSTM architectures with and without attention outperform all of our baselines, verifying the intuition that these models can capture a temporal dependency that is more effectively modeled by these sequential architectures. Moreover, the RNNs

[2]Many other MLP's were tested as well, including 250, 500, 1000 dimensional hidden layers with nonlinearities leaky ReLu and ELU [10]. None of these models performed as well as the ones reported in Figure 3

[3]Many different cell sizes were used during experimentation, including 60, 125, 250, 500, and 1000. The models shown in Figure re-fresults outperform all of the other dimensions tested.

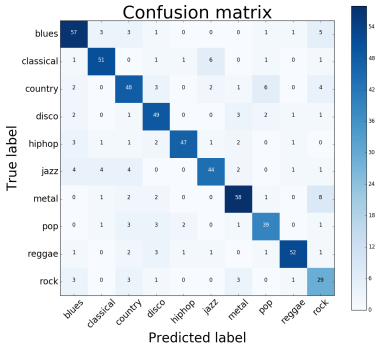

Figure 6: Confusion matrix of the best vanilla LSTM model on the test set. The model makes sensible mistakes, for example confusing rock with metal, jazz with blues, etc.

did not perform well compared to the LSTM models, and further, were outperformed by the neural network baselines. We believe this is due to the vanishing gradient problem that is common when training recurrent neural networks, namely that back-propagating through time over long sequences can cause the gradient to vanish (the norm approaches 0) causing parameters to be unchanged by earlier parts of the sequences [16]. This is also supported by the success of the LSTM networks which are designed to combat this problem through the use of learned gates which allow information (gradients) to pass through the long sequences. It is also important to note that the attention mechanism hurt the vanilla LSTM architecture. We believe this is likely because the additional complexity of the model provided no additional modeling power, only making it harder to train. This isn't completely surprising as attention mechanisms are typically used for alignment in tasks such as machine translation with an encoder and decoder, which may not be appropriate for this setting (attending to specific parts of the input may not be useful in genre classification). Importantly, the best LSTM architecture beats all but one of the external baselines.

6 Conclusions and Future Work

In this paper, we apply deep learning techniques to genre classification. The LSTM model correctly classified 79% of the test set, outperforming all but one of other known classifiers of the GTZAN dataset. We believe that a larger and more robust dataset would yield even higher genre classification accuracy.

In the future, we plan to develop techniques for reverting the learned features of our model back to an interpretable form, akin to what [17] do for visual recognition with Convolutional Neural Networks. Our primary idea is to auralize the learned features through song composition by transforming the weights to audio files using an inverse spectrogram. The aurilization technique will also be applied to interpret what information the model contains about a specific genre. For example, if we make optimizations with respect to Jazz input vectors (leaving learned parameters fixed), and reverse the learned vector back to a wav file using the inverse spectrogram, we can listen to the quintessential Jazz sound as learned by our LSTM.

Most importantly, we have demonstrated that applying Recurrent Neural Networks to genre classification yields promising results. We hope that one today these techniques will be ubiquotous in the field of music information retrieval.

References

[1] George Tzanetakis and Georg Essl and Perry Cook. Automatic musical genre classification Of audio signals. *Speech and Audio Processing, IEEE* pp. 293302, 2002.

[2] Alex Graves and Abdel-rahman Mohamed and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pp. 66456649, 2013.

[3] Ilya Sutskever and Oriol Vinyals and Quoc V. Le. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, pp. 3104-3112, 2014.

[4] Dzmitry Bahdanau and Kyunghyun Cho and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *Proc. International Conference on Learning Representations*, 2015.

[5] https://www.thestar.com/entertainment/2016/01/14/meet-the-man-classifying-every-genre-of-music-on-spotify-all-1387-of-them.html

[6] I. Panagakis and E. Benetos and C. Kotropoulos. Music genre classification: A multilinear approach. *ISMIR*. 2008.

[7] Charles Tripp and Hochak Hung, Manos Pantikakis. Waveform-Based Musical Genre Classification.

[8] Tom LH. Li. and Antoni B. Chan, Genre classification and the invariance of MFCC features to Key and Tempo *International Conference on MultiMedia Modeling, Taipei*. 2011.

[9] A. Hannun and C. Case and J. Casper and B. Catanzaro and G. Diamos and E. Elsen and R. Prenger and S. Satheesh and S. Sengupta and A. Coates and and A.Y. Ng. DeepSpeech: Scaling up end-to-end speech recognition. 2014.

[10] D. Clevert and T. Unterthiner and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). *ICLR*. 2016.

[11] J. Bergstra and N. Casagrande and D. Erhan and D. Eck and B. Kegl. Aggregate features and AdaBoost for music classification. *Machine Learning*, Vol. 65, No. 2-3. 2006.

[12] T. Li and M. Ogihara and Q. Li. A comparative study on content-based music genre classification. *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*. 2003

[13] T. Lidy and A. Rauber and A. Pertusa and J. Inesta. Combining audio and symbolic descriptors for music classification from audio. *Music Information Retrieval Information Exchange (MIREX)*. 2007.

[14] E. Benetos and C. Kotropoulos. A tensor-based approach for automatic music genre classification. *Proceedings of the European Signal Processing Conference, Lausanne*. 2008.

[15] A. Holzapfel and Y. Stylianou. Musical genre classification using nonnegative matrix factorization-based features. *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 16, No. 2. 2008.

[16] R. Pascanu and T. Mikolov and Y. Bengio. On the difficulty of training recurrent neural networks. *ICML*, Vol. 28. 2013.

[17] http://cs231n.github.io/understanding-cnn/

[18] B. Sturm. The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use. 2013.