Auto-Tagging in Music using Convolutional Neural Network

Nitin Nair (nn1174@nyu.edu)
Trey Bradley (gab431@nyu.edu)

***Abstract*— This paper explores a Deep Learning approach to tackle the tricky problem of Auto-Tagging in music using content based features. We compare the performance of a baseline estimator with different Convolutional Neural Network (CNN) structures. In addition to using different CNN's, we also experiment with using different inputs to gauge the affect of performance of the network on the inputs provided. We measure the performance of the network by calculating the AUC-ROC (Area Under the Receiver Operating Curve). We have used the MagnaTagATune data set [1] for training and testing the performance of the models. This paper is inspired by the work shown in [2], which uses a fully convolutional Neural Network to implement Auto-tagging.**

I. INTRODUCTION

Many of todays digital-media consumer applications use data about their users and content in classification and recognition tasks. The main goal here is to recommend content (in this case music) to their users, based on individual preferences. Naturally, communities in Music Information Retrieval (MIR), Digital Signal Processing (DSP), and Machine Learning (ML) have designed a variety of approaches to this user-engagement task. One user-experience tool that is in wide-spread use for designing recommendation systems are social tags. When tagged with this meta data, a single music asset can reveal high-level information about its genre, instrumentation, tempo, mood, etc. [2]. Drawing from data sets like Last.fm, Million Song data set and MagnaTagATune [1], many researchers have devoted efforts towards developing automatically-generated music tagging systems. This project proposal outlines a content-based automatic music tagging algorithm that uses fully convolution neural networks (FCN), as described in [2]. Following are discussions about the problems involving music tagging, convolutional networks, model evaluation, and results of our experiments.

II. PROBLEM STATEMENT

In this paper we try to tackle the common *cold start* problem in recommendation systems. The cold start problem refers to the failure of a system in recommending new content, since the content does not carry any high-level usage data (tags) upon entering the system.

The scope of music tagging is synonymous with multi-label classification which looks to represent audio content with high-level information. Unlike traditional multi-class classification systems, where classes are mutually exclusive, multi-label classification is a much more complex problem to solve. The reason behind this increase in complexity is because in multi-class classification the output space is limited to a single true value (state or class). This is not the case for multi-label classification systems, where the output space grows exponentially with the number of tags (classes). Since the output can reveal classifications that can be correct, incorrect, or even partially correct, certain aspects of the classification model should be considered, including approaches to system inputs, ML architecture design and evaluation metrics.

Traditional methods have proven to be ineffective in solving this problem, as they are not able to associate the tags with visible features of the song. We plan to tackle this problem by applying Deep Learning methodologies, as Deep Learning networks are designed to extract latent features from the input, which we think can more effectively represent the tags.

III. METHOD

To solve this problem of Auto-tagging, we propose a fully automated Deep Convolutional model. The model will take an audio sample as an input, and perform some pre-processing to generate its mel-spectrogram, a content based feature representation.

We have decided to use a fully convolutional network as our classification model. Two key ML strategies, Batch Normalization and Drop-Out, were added to our optimized CNN architecture to improve the system performance. We discuss in more depth the pre-processing steps and the network itself in the following sections.

Like many music classification tasks, the scope of this work is divided into two core modules. The feature extraction module is tasked with representing raw audio data in a way that allows machines to efficiently discriminate against it among other incoming data. The classification module is tasked with implementing the correct model to properly recognize and learn from incoming data. In order to achieve the desired representation of a signal's spectral envelope with low dimension and high individual expression, our feature extraction task uses mel-spectrograms.

A. Data-set

Launched in 2008 as the TagATune music tagging game, the MagnaTagATune data set[1] resulted, and has been leveraged in past and current works to explore automatic music tagging. The data set consists of over 25,000 songs with 188 user-generated tag annotations that describe the songs' genres, moods, instrument compositions, etc. and meta data

that describes the songs' release information (like artists and albums).

1) Data-Preprocessing: The MagnaTagATune data set comes with 29 seconds of raw audio for each song in *.mp3* file format, sampled at 16,000 Hz. Mel-spectruograms are first calculated and used as spectral-based features [3] that align with human anatomy and perception of sound on a logarithmic scale. The choice for using this spectral feature representation is based on similarities that mel-spectruograms share with images, in computer vision tasks.

As part of the feature extraction module of this system, the mel-spectrograms are normalized before being input into the CNN network. Different normalization criterion have been experimented with here, to test the robustness of the system and also to highlight the role of feature representations at the input stage to our classification system.

The data set[1] comes in the form of raw audio files and *.csv* files which contain the tags annotations and the meta-data relating to the audio files. To make the computation and training faster, we combined the tag-annotations, meta-data and the generated mel-spectrograms into one *python-dictionary data structure*. The dictionary also contains the ground truth labels for each song as a separate key, which is helpful at the time of training the network. The structure of the dictionary is shown below:

main_dictionary:

- *key*: **clip_id1**
  - **tags**-*value(0/1)*...**mel_spectrum**,**outputs**
- *key*: **clip_id2**
  - **tags**-*value(0/1)*...**mel_spectrum**,**outputs**

⋮

For this project we have created three data structures, each following the above shown dictionary form and containing all the same data, except for the field containing the mel-spectrograms. The mel-spectrograms are different across each structure in the way in which they have been normalized. The different types of normalizations that we performed are listed below:

- normalize each value (bin, time-block) by generating statistics from the whole training data (Figure 1)
- normalize each value (bin, time-block), by generating statistics from all values for a specific coefficient index in the same spectrogram (Figure 2)
- normalize each value (bin, time-block) by generating statistics from all values in a single spectrogram (Figure 3)

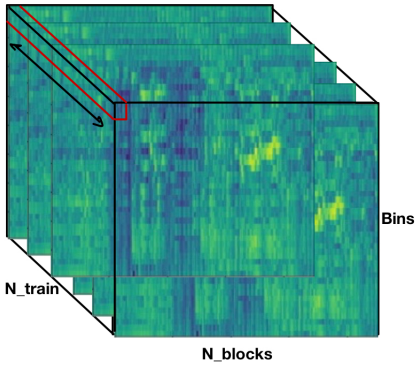

Fig. 1. CNN hidden layer feature representation

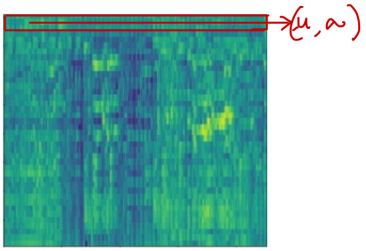

Fig. 2. CNN hidden layer feature representation

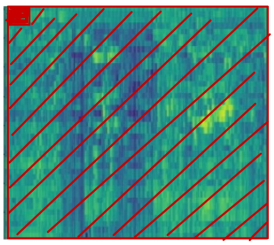

Fig. 3. CNN hidden layer feature representation

B. Network Model

1) CNN: Since artificial neuron and perceptron models were introduced in the late 1950s, neural networks have gained in use in computer vision and MIR communities. The fundamental task of this study is to understand and model human brain processes in perceiving the world, which are assumed to receive raw content and decompose

its features in a sequential manner until high-level features and information can be extracted. Understanding that humans complete this processes naturally and seamlessly in a learning process, CNNs have been proposed to handle this complex classification process.

CNNs are designed to learn hierarchical features over multiple sequential layers, which communicate with each other in a learning process . Mel-spectrograms are introduced in the first CNN layer and are the method of choice in many experiments to depict time and frequency behaviors of the incoming signal. These are considered low-level features which are decomposed into intermediate-level features, perhaps like chords, beats and rhythms, and tonality in subsequent layers of the CNN. Finally, the hierarchy ends with feature vectors that contain high-level information about the signal. In addition to exploiting this hierarchical nature of sound decomposition, the CNN model is also used because of its robustness to variations in translation, distortion and local invariance in the incoming signal [2]. This property is useful for this scope of audio classification since characteristic activity can occur anywhere in time and frequency domains of incoming signals. Figure 1 below illustrates this learning property of the CNN model with a computer vision example. The model receives a raw signal (here, an image of a yellow car) and decomposes the signal into features that it can learn. These models are characterized as black-box, since we do not actually know which features the model learns–simply that it is learning and can ultimately use the features to classify incoming signals.

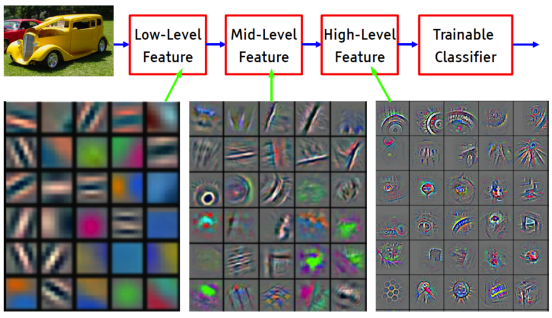

Fig. 4. CNN hidden layer feature representation

CNNs can be configured to test different architectures on the test and training data. These architectures can be customized to handle variations on input representation, data, and classification scope. For example, any number of convolutional and pooling layers can be added or activation functions changed in order to process different audio signal representations, reduce over-parameterization, and adjust sensitivity of the convolutions. The following section discusses the architectures used in [2] to classify audio signals with music tags.

2) Network Components: For our project we have implemented a standard CNN, where the network layers are connected in the following flow: **Convolutional Layer** –> **Activation (ReLU)** –> **MaxPool** In addition, we tried experimenting with additional layers, including Batch normalization and Drop-out, which have been suggested in certain literature to increase model performance. A short description about the functionality of each layer is provided below:

- Convolutional Layers: The convolutional layers are responsible for extracting the features from the preceding layers. The functionality can be described by thinking of the layers as performing convolution between the previous layers and the kernels. The kernels contain the weights which are responsible for generation of the feature maps. These feature maps, as shown in Fig.1, activate, when they find the corresponding features from its input. Kernel size can be used to adjust the sensitivity of the layers to certain features and certain kinds of features.
- ReLU: The Rectified Linear Units (ReLU) is an activation function which adds non-linearity to the output of the convolutional layers. Non-linearity has been noted as one of the ground-breaking facets of neural network models in classifying complex features with complex relationships.
- Batch Normalization [7] is introduced to provide numerical stability to the outputs of each layer. As the name suggests, Batch Normalization normalizes the outputs of each layer by calculating the statistics of each output layer.
- Dropout: Big networks have a tendency to over fit the data. Dropout is added to prevent the network from over fitting.

3) Evaluation: Another big challenge that we had to overcome was to find a method to evaluate the performance of the system. An approach to evaluating the performance of a traditional multi-class classification system is relatively simple to rationalize. This is because the output of the network can either be correct or incorrect. There is no concept of partial correctness. In contrast, with multi-label classification systems, the output can be correct, incorrect or partially-correct. It becomes harder to evaluate the performance, as it becomes a hard choice to weigh the importance of each correct label in making desired predictions.

As referenced in [2], we have opted for AUC-ROC (Area Under the Receiver Operating Curve) as the metric to evaluate the performance of the system, as part of the SciKitLearn Python package. This metric shows the performance of the model to classify the training and testing data at various thresholds. The axes on an AUC-ROC plot include the model's false-positive rate (on the x-axis, related to the model's specificity) and the model's true-positive rate (on the y-axis, related to the model's recall or sensitivity).

IV. EXPERIMENT

In this project we are using the MagnaTagATune data set [1] for training and testing purposes. The data set comes with 188 tags with 0 or 1 labelling for every tag for each song. Using the entire set of 188 tags exponentially increases the output vector space and it becomes hard to train the network as it never converges. Therefore, to be able to train the network , we are using the top 50 tags in the data set, ranked based on the frequency of its occurrence.

A. CNN Classifier

For preliminary training, we used Binary Cross Entropy loss on the output of the network. But, the network failed to converge, even after multiple trials. On inspection, it can be found that, on average, even after using the top 50 tags, every song has, at max, 4-5 tags associated with it. This sparsity of tags makes it hard to train the network.

To overcome this issue, we had to use weighted Binary Cross Entropy loss, where the weights were chosen to give more importance to rare tags. This produced better results, and we were able to train the network with desired decreasing loss.

The mel-spectrogram acts as a 2-D input to the network. The network outputs a 50 dimensional vector. This output vector represents the probabilities of the tags being associated with the song. The probabilities are then converted to a binary 'yes' or 'no' depending on a defined threshold.

We ran the experiment with 2 networks, both with similar architectures, but one incorporating Batch Normalization and Dropout and the other without these techniques. The network architectures are listed and depicted in Table I and Figure 5 below:

CNN - 4 Convolutional Layers Architecture
Mel-Spectrogram (input: 40 x 909 x 1)
Convolution: 3 x 3 x 128
MaxPool (2,4) output:
Convolution: 3 x 3 x 384
MaxPool (2,4) output:
Convolution: 3 x 3 x 768
MaxPool (2,6) output: 1 x 1 x 2048
Convolution: 3 x 3 x 2048
MaxPool (1,7) output: 1 x 1 x 2048
Output: 50 x 1

TABLE I
PROPOSED NETWORK ARCHITECTURE (BATCH NORMALIZATION AND DROPOUT WAS ADDED BETWEEN EACH CONVOLUTIONAL LAYERS IN A SECOND ARCHITECTURE)

The results of the experiment have been documented in table II.

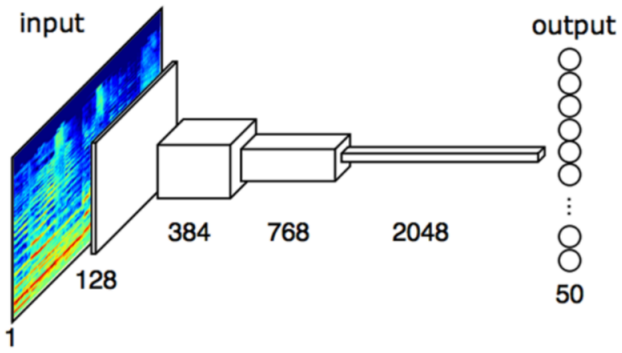

Fig. 5. Network Architecture visualized

B. Baseline Classification

In order to further validate the performance of our CNN model against alternative approaches, we ran an AUC-ROC evaluation on a baseline classifier. As suggested by multi-label classification literature, we used SciKit Learn's Random Forest model, which essentially fits a number of decision tree classifiers to the training data and employs a voting system to determine the probability of a class (label). Results from this baseline classification can be seen in table III. We used the data structure that contained mel-spectrograms that were normalized across the entire training set. Additionally we used SciKit Learn's KNN model with K = 125 and mostly default parameters as a second baseline. These results are also provided in Table III.

V. RESULTS AND DISCUSSION

The above sections describe the methodology behind our implementation of the 4 convolutional layers CNN model. A summary of our results can be seen in Table III. We achieved an AUC-ROC score of .8714 under the architecture that trained for 10 epochs, with mel-spectrograms that were normalized across all examples in training dataset. A score of .5 is assumed to be the classification of randomly guessing model.

As seen from the results, the network becomes a better classifier the more it trains, but the improvement becomes relatively smaller after every epoch. Also, it can be noted that input features play a huge role in the network performance. Just by normalizing the inputs after taking the log of the mel-spectrogram gives us an increase in performance of about 10%. Normalization is a processing step that can affect the input's representation of certain characteristics of the audio, individually and as a part of a larger body of data. For example, normalizing the mel-spectrograms over the entire data set will alter the model's interpretation of silence as features in the samples. Further, should there be a particularly silent audio example, this kind of normalization will reduce the model's learning of silence as a characteristic feature.

Data Set Type	AUC Validation	AUC Test
norm. across all data (10 epochs)	.8585	.8629
norm. across each mel coefficient (10 epochs)	.8593	.8622
norm. within spectrogram (10 epochs)	.8578	.8595
norm. across all data (2 epochs)	.7441	.7524
normalized per mel-spectrogram pre-Log (2 epochs)	.4924	.4964
Batch Norm. & Drop Out (Val / Test)	.8655	**.8714**

TABLE II

MY CAPTION

Baseline Classifier	AUC Validation	AUC Test
Random Forest (nodes = 100)	.502	.506
KNN (N = 125)	.5104	.5104

TABLE III

BASELINE CLASSIFIER RESULTS

VI. CONCLUSIONS

This project showed promising results of CNN models to allocate tags to unseen songs based on low-level features extracted from the training set. The CNN classifiers out perform our baseline by a large margin. The results also show that auto-tagging can be performed on music based on latent features, which is why the CNN model performs well on this task. The results reveal the importance of the feature extraction component to the classification task.

Although the networks perform well, there is room for improvement to increase the model's ability to classify songs. The performance of the network is partly limited by the size of the data set. Future work may include modifying the network to train on a larger data set, like the Million-Song data set, and further experimentation with different spectral representations of audio signals.

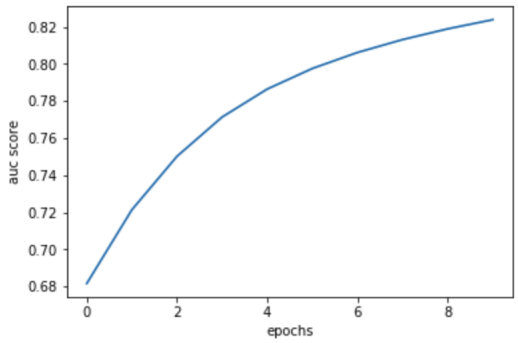

Fig. 6. AUC through training

REFERENCES

[1] Edith Law, Kris West, Michael Mandel, Mert Bay and J. Stephen Downie (2009). Evaluation of algorithms using games: the case of music annotation. In Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR)

[2] Keunwoo Choi, George Fazekas, Mark Sandler (2016). Automatic Tagging using Deep Convolutional Neural Networks. 17th International Society for Music Information Retrieval Conference

[3] Mel Frequency Cepstral Coefficients for Music Modeling - Beth Logan (2000)

[4] Multi-Level and Multi-Scale Feature Aggregation Using Pretrained Convolutional Neural Networks for Music Auto-Tagging - Jongpil Lee and Juhan Nam

[5] Deep content-based music recommendation - Aaron van den Oord, Sander Dieleman, Benjamin Schrauwen

[6] Multi-class AUC metrics and weighted alternatives - Ben Van Calster, Vanya Van Belle, George Condous, Tom Bourne, Dirk Timmerman, Sabine Van Huffel

[7] Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift - Sergey Ioffe,Christian Szegedy

[8] http://tagatune.org/