**Music Auto-Tagging with Capsule Network**

**Abstract:-**

**Music genre recognition (MGR) plays a fundamental role in the context of music indexing and retrieval. Unlike images, music genres consist of immediate characteristics that are highly diversified with abstractions in different levels. However, most representation learning methods for MGR focus on global features and make decisions from features in the same level. To remedy such defects, we integrate a convolutional neural network (CNN) with (NetVLAD) and self-attention to capture the local information across levels and learn their long-term dependencies. A meta classifier is used to make the final MGR classification by learning from aggregated high-level features from different local feature coding networks. Experimental results show that the proposed approach yields higher accuracies than other state-of-the-art models on MagnaTagATune,GTZAN, ISMIR2004, and Extended Ballroom dataset.**

**Introduction:-**

**With the increase of online music databases and user interactive applications, developing effective automatic tools for music classification and retrieval has become an essential issue. Music information retrieval (MIR) aims to retrieve useful information from music and classify it into different categories. For MIR problems, music genre is significantly important because it facilitates both the search for music and the organization of music collections. Furthermore, music genre also reveals the interplay of cultures. Extracting influential features contributes to the automatic classification of music genres**

**Model:-**

**Current Deep Architecture:-**

**In order to facilitate the discussion around the current audio architectures, we divide deep learning models into two parts: front-end and back-end. The frontend is the part of the model that interacts with the input signal in order to map it into a latent-space, and the backend predicts the output given the representation obtained by the front-end. In the following, we present the main front- and back-ends. Deep learning pipeline. Front-ends. These are generally comprised of convolutional neural networks (CNNs),since these can learn efficient representations by sharing weights 1 along the signal. Front-ends can be divided into two groups depending on the used input signal: waveforms or spectrograms. Further, the design of the filters can be either based on domain knowledge or not. For example, one leverages domain knowledge when a front-end for waveforms is designed so that the length of the filter is set to be as the window length of a STFT . Or for a spectrogram front-end, it is used vertical filters to learn timbral representations or horizontal filters to learn longer temporal cues . Generally, a single filter shape is used in the first CNN layer but some recent works report performance gains when using several filter shapes in the first layer. Using many filters promotes a richer feature extraction in the first layer, and facilitates leveraging domain knowledge for designing the filters shape. For example: Which determine the learned feature representations. Waveform front-end using many long filters (of different lengths) can be motivated from the perspective of a multi resolution time-frequency transform; or since it is known that some patterns in spectrograms are occurring at different time-frequency scales, one can intuitively incorporate many (different) vertical and/or horizontal filters in a spectrogram front-end. To summarize, using domain knowledge when designing models allows us to naturally connect the deep learning literature with previous signal processing work. On the other hand, when domain knowledge is not used, it is common to employ a deep stack of small filters, e.g.: 3×1 as in the sample-level frontend used for waveforms [14], or 3×3 filters used for spectrograms [5]. These models based on small filters make minimal assumptions over the local stationary of the signal, so that any structure can be learned via hierarchically combining small-context representations. These architectures with small filters are flexible models able to potentially learn any structure given enough depth and data. Back-ends. Among the different back-ends used in the audio literature, we identified two main groups: (i) fixedlength input back-end, and (ii) variable-length input backend. The generally convolutional nature of the front-end allows it to process different input lengths. Therefore, the back-end unit can adapt a variable-length feature map to a fix-sized output. The former group of models (i) assume that the input will be kept constant – examples of those are front-ends based on feed-forward neural-networks or fully - convolutional stacks. The second group (ii) can deal with different input-lengths since the model is flexible in at least one of its input dimensions – examples of those are back-ends using temporal-aggregation strategies such as max-pooling, average-pooling, attention models or recurrent neural networks. Given that songs are generally of different lengths, these types of back-ends are ideal candidates for music processing. However, despite the different-length nature of music, many works employ fixed-length input back-ends (group i) since these architectures tend to be simpler and perform well.**

**Architecture under study:-**

**After an initial exploration of the different architectures introduced in section 2, we select two models based on opposite design paradigms: one for processing waveforms, with a design that does minimal assumptions over the task at hand; and another for spectrograms, with a design that heavily relies on musical domain knowledge. Our goal is to compare these two models for providing insights in whether domain knowledge is required (or not) for designing deep learning models. This section provides discussion around our architectural choices and introduces the basic configuration setup – which is also accessible online. 4 The waveform model was selected after observing that the sample-level front-end (using a deep stack of 3×1 filters) was remarkably superior to the other waveform-based front ends – as shown in the original paper. This result is particularly compelling because this front-end does not rely on domain-knowledge for its design. Note that raw waveforms are fed to the model without any preprocessing, and the small filters considered for its design make no strong assumptions over the most informative local stationary in waveforms. Therefore, the samplelevel can be seen as a problem agnostic front-end that has the potential to learn any audio task provided that enough depth and data are available. Given that a large amount data is available for this study, the sample-level front-end is of particular interest due to its strong learning potential: its solution space is not constrained by severe architectural choices relying on domain knowledge. On the other hand, when experimenting with spectrogram front-ends, we found domain knowledge intuitions to be valid guides for designing deep architectures. For example, front-ends based on (i) many vertical and horizontal filters in the first layer were consistently superior to front-ends based on (ii) a single vertical filter – as shown in recent publications. Note that the former frontends (i) can learn spectral and (long) temporal representations already in the first layer – which are known to be important musical cues; while the latter (ii) can only learn spectral representations. Moreover, we observed that frontends based on a deep stack of 3×3 filters were achieving equivalent performances to the former front-end (i) when input segments were shorter than 10s – as noted in the literature. But when considering longer inputs (which yielded better performance), the computational price of this deeper model increases: longer inputs implies having larger feature maps in every layer and therefore, more GPU memory consumption. For that reason, we refrained from using a deep stack of 3×3 filters as a front-end – because our 12GBs of VRAM were not enough to input 15s of audio when using a back-end. Hence, making use of domain knowledge also provides guidance for minimizing the computational cost of the model – since by using a single layer with many vertical and horizontal filters, one can efficiently capture the same receptive field without paying the cost of going deep. Finally, note that front-ends using many vertical and horizontal filters in the first layer are an example of deep architectures relying on (musical) domain knowledge for their design. After considering the previous discussion, we select the sample-level front-end as main part of our assumption-free model for waveforms; and we use a spectrogram front-end with many vertical and horizontal (first-layer) filters for the model designed considering domain knowledge. Experiments below share the same back-end, which enables a fair comparison among the previously selected front-ends. Unless otherwise stated, the following specifications are the ones used for the experiments – throughout the document, we refer to these specifications as the basic configuration: Shared back-end. It consists of three CNN layers (with 512 filters each and two residual connections), two pooling layers and a dense layer . We introduced residual connections in our model to explore very deep architectures, such that we can take advantage.**

**Experiment:-**

**Setup:-**

**In this section, we introduce the datasets used in our experiments and describe experimental settings. MTAT dataset we will use in this. Sample-level CNN configuration. For example, in the layer column, the first 3 of “conv 3-128” is the filter length, 128 is the number of filters, and 3 of “maxpool 3” is the pooling length.**

**Dataset:-**

**We evaluate the proposed model on two datasets, MagnaTagATune dataset (MTAT)] and Million Song Dataset (MSD) annotated with the Last.FM tags . We primarily examined the proposed model on MTAT and then verified the effectiveness of our model on MSD which is much larger than MTAT . We filtered out the tags and used most frequently labeled 50 tags in both datasets, following the previous work . Also, all songs in the two datasets were trimmed to 29.1 second long and resampled to 22050 Hz as needed. We used AUC (Area Under Receiver Operating Characteristic) as a primary evaluation metric for music auto-tagging.**

**Optimization:-**

**We used sigmoid activation for the output layer and binary cross entropy loss as the objective function to optimize. For every convolution layer, we used batch normalization and (ReLU) activation. We should note that, in our experiments, batch normalization plays a vital role in training the deep models that takes raw waveforms. We applied dropout of 0.5 to the output of the last convolution layer and minimized the objective function using stochastic gradient descent with 0.9 (Nesterov) momentum. The learning rate was initially set to 0.01 and decreased by a factor of 5 when the validation loss did not decrease more than 3 epochs. A total decrease of 4 times, the learning rate of the last training was 0.000016. Also, we used batch size of 23 for MTAT and 50 for MSD, respectively. In the spectrogram model, we conducted the input normalization simply by dividing the standard deviation after subtracting mean value of entire input data. On the other hand, we did not perform the input normalization for raw waveforms.**

**Result:-**

**mn-DCNN models**

**The evaluation results for the mn-DCNN models on MTAT for different input sizes, number of layers, filter length and stride of the first convolution layer. As the proposed sample-level architecture by performing experiments. The models used in the experiments follow the configuration strategy described . In the spectrogram experiments, 128 melbands are used to match up to the number of filters in the first convolution layer of the raw waveform model. FFT size was set to 729 in all comparisons and the magnitude compression is applied with a nonlinear curve, log(1 + C|A|) where A is the magnitude and C is set to 10.**

**MSD result and the number of filters**

**We investigate the capacity of our sample-level architecture even further by evaluating the performance on MSD that is ten times larger than MTAT. While training the network on MSD, the number of filters in the convolution layers has been shown to affect the performance. According to our preliminary test results, increasing the number of filters from 16 to 512 along the layers was sufficient for MTAT.**

**Comparison to state-of-the-arts**

**we show the performance of the proposed architecture to previous state-of-the-arts on MTAT and MSD. They show that our proposed sample-level architecture is highly effective compared to them.**

**Visualization of learned filters**

**The technique of visualizing the filters learned at each layer allows better understanding of representation learning in the hierarchical network. However, many previous works in music domain are limited to visualizing learned filters only on the first convolution layer.**

**Conclusion:-**

**In this paper, we proposed sample-level DCNN models that take raw waveforms as input. Through our experiments, we showed that deeper models (more than 10 layers) with a very small sample-level filter length and sub-sampling length are more effective in the music auto-tagging task and the results are comparable to previous state-of-the-art performances on the two datasets. Finally, we visualized hierarchically learned filters. As future work, we will analyze why the deep sample-level architecture works well without input normalization and nonlinear function that compresses the amplitude and also investigate the hierarchically learned filters more thoroughly.**

**References:-**

[**https://en.wikipedia.org/wiki/Music\_information\_retrieval**](https://en.wikipedia.org/wiki/Music_information_retrieval)

[**https://archives.ismir.net/ismir2018/paper/000191.pdf**](https://archives.ismir.net/ismir2018/paper/000191.pdf)

[**https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9170628**](https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9170628)

[**https://arxiv.org/pdf/1703.01789.pdf**](https://arxiv.org/pdf/1703.01789.pdf)

**GitHub link:-**

**https://github.com/VishweshSalodkar/Music-Auto-Tagging-and-Capsule-Network**