COMSM0138: Applied Deep Learning Coursework

A REPLICATION OF A END-TO-END LEARNING ARCHITECTURE FOR MUSIC AUDIO TAGGING

Hanbin Zhang
School of Computer Science
University of Bristol
Bristol, United Kingdom
ey20699@bristol.ac.uk

Yichen Dong
School of Electrical, Electronic and Mechanical Engineering
University of Bristol
Bristol, United Kingdom
ow20717@bristol.ac.uk

***Abstract*—This paper presents a replication of a study that used an end-to-end learning architecture for music audio tagging, focusing on the application of deep learning techniques. The proposed model is based on the work of [1], utilizing convolutional neural networks (CNNs) on raw audio waveforms. The MagnaTagATune dataset is employed for training and evaluation. The replication includes a detailed analysis of the model architecture, dataset characteristics, implementation details, quantitative results replication, training curves, qualitative results, and proposed improvements. Our works aim to reproduce results from [1] and attempt to extend the performance of the model similar to this architecture.**

***Index Terms*—feature learning, end-to-end learning, convolutional neural networks, music information retrieval, automatic tagging**

A. Team Ackonwledgement

We agree that all members have contributed to this project (both code and report) in an approximately equal manner.

Hanbin Zhang Yichen Dong

B. Introduction

S. Dieleman and B. Schrauwen [1] address the challenge of exploring end-to-end learning for music audio using convolutional neural networks (CNNs) in their article. Traditionally, music information retrieval (MIR) tasks involve a two-stage approach, requiring expertise on this field and significant engineering effort. The authors investigate the feasibility of applying feature learning directly to raw audio signals, eliminating the need for mid-level representations like spectrograms. While their study on automatic tagging tasks did not surpass the performance of spectrogram-based approaches, it revealed that CNNs can autonomously discover frequency decompositions and learn phase- and translation-invariant features from raw audio. The research aims to reduce reliance on engineered features and advance the understanding of end-to-end learning capabilities in music audio processing.

C. Related Work

In 2017 Y. Tokozume and T. Harada [2] tried to achieve better performance on tasks similar to MIR in end-to-end environmental sound classification by developing a deep convolutional neural network architecture that operates directly on raw audio waveforms as been done in this paper. The key aspects of their model are the use of multiple 1D convolutional layers with very small 8-sample filters to hierarchically extract time-frequency features from the raw signals. This takes the place of hand-engineered spectrogram features as well. Additionally, aggressive data augmentation is performed by randomly cropping 1.5-second segments during training and aggregating predictions at test time. These contributions allow the model to finally exceed the performance of log-mel spectrogram features on the ESC-50 dataset.

In 2018 T. Kim, J. Lee, and J. Nam [3] also employed a deep enhanced convolutional neural network architecture that operates based on raw audio waveforms, intending to achieve better performance in end-to-end music auto-tagging tasks. However, the model described in this paper is inspired by ResNets and SENets and introduces the ReSE-n block as the building block of this network. In other words, it modifies the structure of conventional convolutional neural network building blocks (Conv1D, BatchNormal, ReLU, MaxPool) by combining the building blocks of SENet and ResNet. Additionally, it reduces the filter size of Conv1D, ultimately forming the ReSE-n block. The original audio is first subjected to 1 layer of strided convolution, followed by 9 layers of ReSE-n blocks and 2 layers of conventional fully connected layers. This process yields probabilities corresponding to each tag, ultimately completing the classification. The results indicate that, on the *MagnaTagATune* dataset, this model shows significant improvement in accuracy compared to conventional convolutional neural network models.

D. Dataset

The *MagnaTagATune* dataset is utilized for model training. It is a collection of annotated audio clips designed for music information retrieval tasks, in particular automatic tagging. Containing more than 25,000 audio clips 29.1 seconds long each with 12KHz sampling rates, covering a wide range of genres and moods, *MagnaTagATune* is a diverse and comprehensive resource for training and evaluating automatic tagging algorithms. In addition, each audio clip in the dataset is associated with a combination of 188 binary tags describing

various musical features or characteristics. This makes the dataset ideal for raw audio-based automatic labeling.

1) Training and test set split: The model uses 16963 songs from this dataset as a training set and an additional 4392 songs as a test set.

2) About tags: Originally, the number of tags in the dataset was 189, however, only 50 tags are used within the training set and validation set. The representation of the tags corresponding to each song is a tensor of length 50, where each index represents a tag (e.g. baroque, classical, country, etc.), and when an index corresponds to a value of 1 in the tensor, it means that the song has this characteristic, while a value of 0 means that it does not have this characteristic.

E. CNN Architecture (Dieleman et al.)

Our model is based on the model described in *"Sample-Level CNN Architectures for Music Auto-Tagging Using Raw Waveforms" [1]*, which is shown in Figure 1. Also, the output dimensions of each layer are denoted inside (#) indicating the number of units, (^) stands for the number of channels, and (&) is filter sizes and pooling sizes. Apart from that, the data

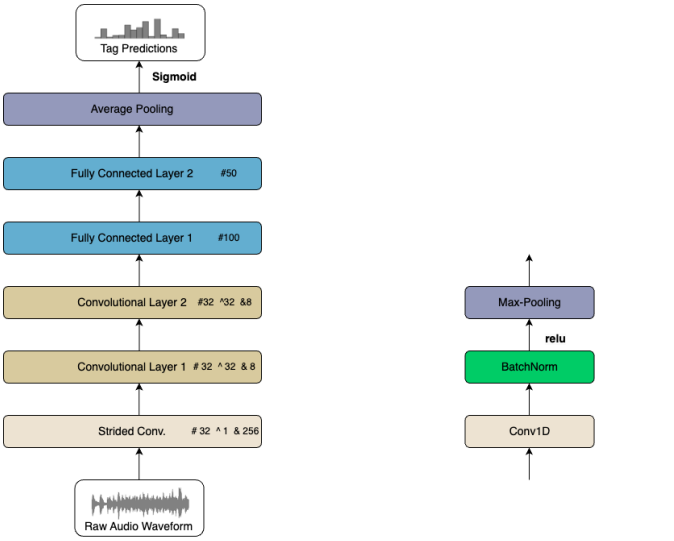


Fig. 1. Proposed architecture for auto-tagging of raw audio. (a) The model consists of a strided convolutional layer, 2 blocks of convolutional layer, and two fully-connected layers. (b) The building block for each convolutional layer.

is entered into strided convolution as 10 tunes in each batch of 10 clips each, with a single channel. Then, before passing through the fully connected layer, the data is stacked, and the 10*10 clips become whole to enter the fully connected layer. Finally, after leaving the average pooling layer, the 10 tunes are each mapped to 50 labels representing different music genres, yielding 10*50 corresponding probabilities.

F. Implementation Details

Among the details of the implementation, the following points are worth noting:

1) Training and Evaluation: In the training phase, the data is processed in batches, each containing 10 songs. The overall training process repeats over 5 epochs. Throughout training, we log per-batch metrics such as loss and step time for analysis. Notably, the legacy logging code includes per-batch accuracy, but our code omits this value, using a placeholder (4396) solely for display purposes in the terminal log, with no actual utilization.

For model evaluation, the test set is employed to calculate the average AUC score across all tags and samples. This score is assessed after each epoch to validate the model's performance.

2) Batch Normalization: When this model is passing forward, a batch normalization step is added by default to all layers except the final fully-connected layer, i.e.: the results of the convolutional layers for the first and second layers, and the first fully-connected layer are batch normalized before being passed into the rectified linear unit (ReLU). Batch normalization has several benefits, such as: reducing sensitivity to weight initialization, acting as a regulariser, etc.

3) Adaptive input size for first fully connected layer: The length of the input tensor of the fully connected layer depends entirely on the size of the output tensor of the previous step, i.e., the second convolutional layer.

4) The utilization of Sigmoid function and Average pooling before the outcome: Before finally mapping each tune to its corresponding label, it is made to pass through the average pooling layer so that the 50 values corresponding to each of the 10 clips can be averaged. This becomes 50 values corresponding to each of the 10 tunes, facilitating the passing of the Sigmoid function in the next step, which maps these values to the interval between 0 and 1 to get the intuitive probability, i.e., the tag corresponding to each song.

5) Adaptive padding for different stride length: As the length of the strided convolution changes, the length of the output result of each convolutional layer changes as it is passed forward. However, when inputting a fully connected layer, it is necessary to re-split the tensor that was put together from 10 small clips into a single tune to use the convolutional layer for computation back into 10 small clips, sometimes resulting in inconsistent lengths. This is where automatic padding is used to make the length of each small segment of audio consistent.

6) Optimizer and hyperparameters: The optimizer updates the model parameters to minimize the loss. The update rule employed by the optimizer encompasses a variety of variations. In our approach, we used Stochastic Gradient Descent (SGD) with momentum as our optimizer. This modification SGD, compared to the original one, introduces a momentum term, facilitating accelerated convergence and the attenuation of oscillations in parameter updates. This not only contributes to a faster training pace but also yields a more robust and well-converged model.

For the momentum parameter beta, a frequently employed value of 0.9 has been selected, proving effective in a majority of scenarios. Additionally, a learning rate of 0.1135 has been chosen. Although relatively larger than some of our alternative

learning rates, such as 0.01 and 0.05, this choice still gives favorable results. This can be attributed to its ability to drive faster progress, leading to a more rapid convergence and adeptly overcoming plateaus during the descent process.

G. Replicating Quantitative Results

Length	Stride	AUC(train)	AUC(test)
1024	1024	0.7810	0.7665
1024	512	0.7703	0.7591
512	512	0.7852	0.7748
512	256	0.7769	0.7705
256	256	0.7859	0.7873

TABLE I
AUC SCORES WITH DIFFERENT STRIDES

In an attempt to reproduce AUC scores, various configurations of stride convolution layers were tested with five epochs on training. The obtained scores are summarized in Table **??**. The replication results generally align with the trends observed in the original paper [1], demonstrating that as the stride step size and kernel size decrease, the AUC scores increase. Notably, the only deviation from this trend is observed in the case of a length of 512 and a stride step size of 256, where the AUC score is lower compared to the configuration with a stride step size of 512. Notably, comparing to the original result from [1] is higher comparing to our replicated result which, however, is still usable comparing to the coursework guidance. This might due to some better optimizer, hyperparameters and data augmentation technics were used by S. Dieleman and B. Schrauwen.

H. Training Curves

In examining the training and loss curves of our model, we employed a convolutional layer with a stride set at 256 and a length of 256, as these parameters consistently exhibited similar trends.

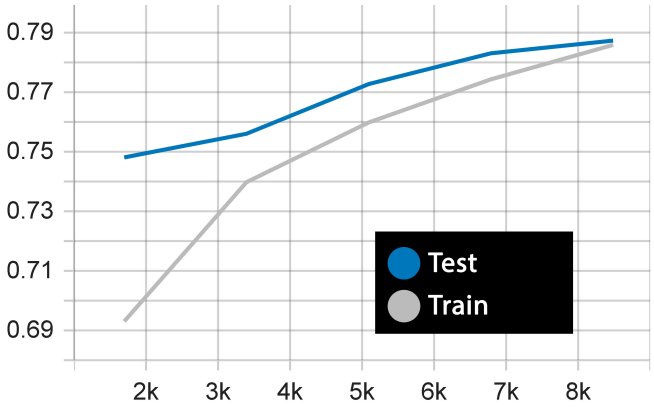


Fig. 2. Accuracy curve when stride=(256, 256)

Upon examining both Figure 2 and Figure 3, it becomes evident that neither overfitting nor underfitting manifests prominently during the training process. As training progresses, scores on both the training and test sets converge towards

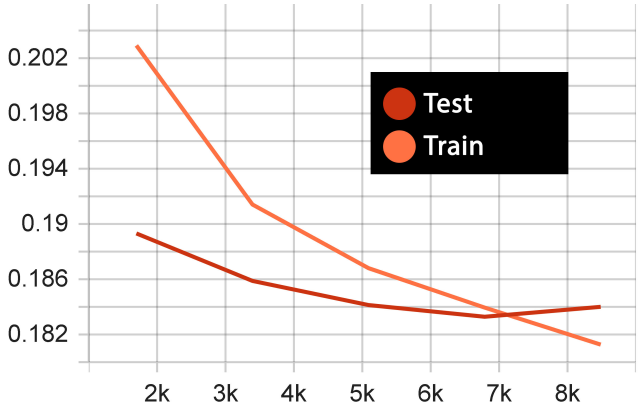


Fig. 3. Loss curve when stride=(256, 256)

higher accuracy and lower loss values. A noteworthy observation is the model's superior accuracy on the test set compared to the training set, challenging conventional expectations. This discrepancy may be attributed to the relative simplicity of the test set, making it inherently more predictable.

Moreover, Figure 3 indicates a slight increase in the loss of the test set towards the end of training, hinting at a potential risk of overfitting. Addressing this concern is essential to ensure the model's generalization to new data. Exploring adjustments to hyperparameters or incorporating regularization techniques may be beneficial in mitigating overfitting and optimizing overall model performance.

I. Qualitative Results

From our best model, we analyzed which examples show good and bad performance based on our model with a (256, 256) stride convolution layer.

1) Example with High Confidence: One exemplary audio yielding outstanding results is titled *kenji_williams-faces_of_epiphany-03-illusion_bedrock_mix-523-552.npy*. The model achieved an impressive AUC score of nearly 100% on this particular audio excerpt. This audio is associated with the following tags: [*'beat', 'dance', 'drum', 'electro', 'fast', 'synth', 'techno'*]. The model's predictions exhibit a notably high probability for these tags, ranging between 0.7 to 0.9, in stark contrast to other tags, which mostly hover below 0.01. This stark difference in probability values distinctly identifies the tags the audio possesses and those it does not. However, the model displayed some imperfections; specifically, it struggled with explicit predictions for the *'drum'* and *'techno'* tags compared to other categories. Nonetheless, the overall performance remains commendable.

2) Example with Low Confidence: Conversely, the model faced challenges in producing accurate predictions for certain audio clips. An illustrative example is *the_headroom_project-jetuton_andawai-12-lost_world-0-29.npy.* The model's prediction yielded a considerably lower AUC score of 0.14 for this audio. This particular audio is associated with only the *'quiet'* tag. The model's predicted probability for this tag is not significantly higher than for other tags that are not

present in the audio. One possible explanation for this subpar performance could be the limited number of tags (only one) in comparison to the total set of 50 tags. This limitation may pose a challenge for the model in accurately predicting whether an audio is quiet or not.

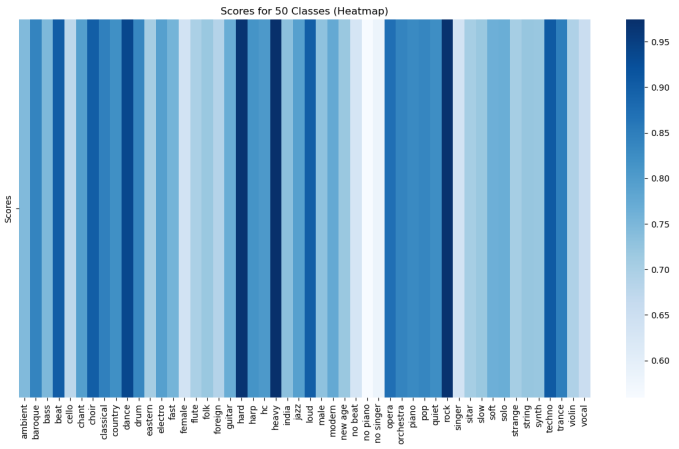


Fig. 4. Heat map showing the per class accuracy

3) Per-class Accuracy: Moreover, the model's per-class accuracy is assessed through AUC scores assigned to each class, with a visual representation provided by the heatmap shown in Figure 4. Examining Figure 4, it's evident that the strips corresponding to *'rock'*, *'hard'*, and *'heavy'* exhibit the darkest colors, indicating the model's strong performance in discerning whether a piece of music possesses these specific tags. However, the model shows subpar performance on certain classes. For instance, the score for the 'no piano' tag is notably lower compared to all other classes. Such variations may stem from the inherent ease or difficulty of identifying certain genres. For instance, *'rock'* exhibits a robust and generalized pattern, making it more conducive for a CNN model to recognize. On the other hand, *'no piano'* presents a challenge due to its less explicit pattern, making it harder to generalize.

J. Improvements

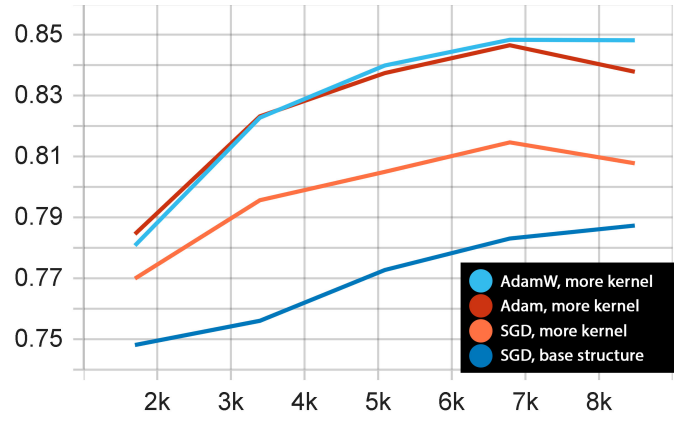


Fig. 5. Different improvement methods with accuracy curve

1) Utilizing AdamW as optimizer: The primary reason for using AdamW was that it would apply an adaptive learning rate to each parameter and adjust it separately based on past gradients. Also, weight decay correction has been introduced in AdamW. In this current model, where the optimal learning rate varies from parameter to parameter, this adaptive ability can make a significant difference. However, for Stochastic Gradient Descent with Momentum(SGD with Momentum), although momentum can help SGD converge faster, relying on manually adjusted momentum may not be able to handle the changes in the loss situation during the training process very precisely. As shown in Figure 5, overfitting may still occur in the later stages of training. As for Adam, it maintains two moving averages for each parameter: the first moment (mean) and the second moment (uncentered variance) to achieve an adaptive learning rate. It updates the parameters using a combination of the gradient and a moving average of past gradients. The update includes a bias correction term to account for initialization bias in the moments. But, In standard Adam, weight decay is applied to both the weights of the parameters and the moving average. On the other hand, AdamW decouples weight decay from the moving average [4]. It applies weight decay only to the actual weights, not to the additional terms used in the moving average. This leads to better convergence results. The model was iteratively tested and tuned several times for the AdamW optimizer to be effective. In the end, the model with a learning rate of 0.003 demonstrated a robust accuracy rate. As Figure 5 shows, during the test period, using AdamW significantly improves the accuracy compared to SGD with momentum and Adam. AdamW proves to be a more suitable optimizer for this model compared to SGD with momentum.

2) Using more kernels in convolutional layer: There may be several accuracy bottlenecks encountered by the model at this stage, but one of them is caused by the smaller model's own smaller parameter set. The smaller set of hyperparameters may not be able to adequately adjust for the complexity of the model, while complex tasks often require models with more parameters and more layers. Therefore, the model expands the 32 kernels fixed in the second layer by a factor of 32, i.e., 1024 kernels, from the original model. From the training results, the new model with more parameters significantly outperforms the old model on both the training and test sets. The only cost is the relatively longer training time.

3) Introducing ResNet and more Convolutional layers to design: In addition, as a derivation of the extension part, An example of hybrid model architecture is presented in Figure 6. This model combines 5 convolutional layers, 2 fully connected layers, and a ResNet dropout layer (shown in Figure 7). It is worth noting that ResNet will implement Dropout between the fourth and fifth convolutional layers to alleviate the overfitting behavior that may be caused by more complex models. With the number of convolutional layers significantly increased while maintaining the number of kernels, and ResNet was added, this hybrid model was expected to give significantly better accuracy. However, the

accuracy of the model did not essentially improve significantly compared to before the changes (i.e., using more kernels in each layer and using adamW as the optimizer), with the best score of only 85.93 (%) . In addition to this, more epochs were introduced overtraining, however, this can lead to overfitting in the later stages of training.

K. Conclusion

In summary, this paper provides a comprehensive replication of the end-to-end learning architecture for music audio tagging proposed by Tokozume and Harada [1]. Our replication successfully reproduces quantitative results, indicating the robustness of the model across various configurations. The training curves demonstrate convergence without prominent overfitting or underfitting, although a potential risk of overfitting towards the end of training is acknowledged. Qualitative analysis reveals prediction accuracy, offering insights into the model's strengths and limitations on particular samples. Furthermore, proposed improvements, such as utilizing AdamW as an optimizer and increasing the number of kernels in the convolutional layer, show promise in enhancing model performance. Future work involves modifications to the architecture, and expanding the model's capacity to handle more complex tasks. This replication contributes to the understanding of deep learning approaches in the context of music audio tagging.

L. Future Work

1) Modification on architecture: For the hybrid model mentioned in *"Sample-Level CNN Architectures for Music Auto-Tagging Using Raw Waveforms" [1]*, the 9 layers of the ReSe network perform well on the same dataset: *MagnaTagATune*. However, when the 2 layers of ResNet are tested in the current architecture, the learning rate drops dramatically. One speculation is that the model currently has fewer parameters and ResNet drops out of updating some neurons during the dropout process. This makes it more difficult for the already smaller model to obtain accurate classifications.

Apart from that, although the 8-layer model introduced in Figure 6 showed an improvement in accuracy, but, there's not much progress. As a conjecture, the number of kernels per layer did not change much, even though the number of layers of the model got much larger. In other words, the parameter scales of the model have not particularly improved, so the accuracy has not stepped up a level.

Therefore, based on the model obtained so far, the next step is to further expand the number of kernels in each layer of the convolutional neural network while further increasing the number of layers of the model to obtain a larger set of parameters and also keep introduce the dropout process of the ResNet to reduce the chance of the occurrence of overfitting phenomena. This further revised hybrid architecture will make it possible to make some progress on the existing models.

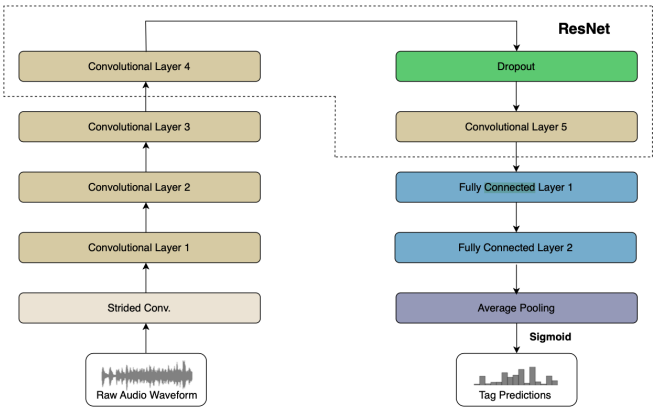
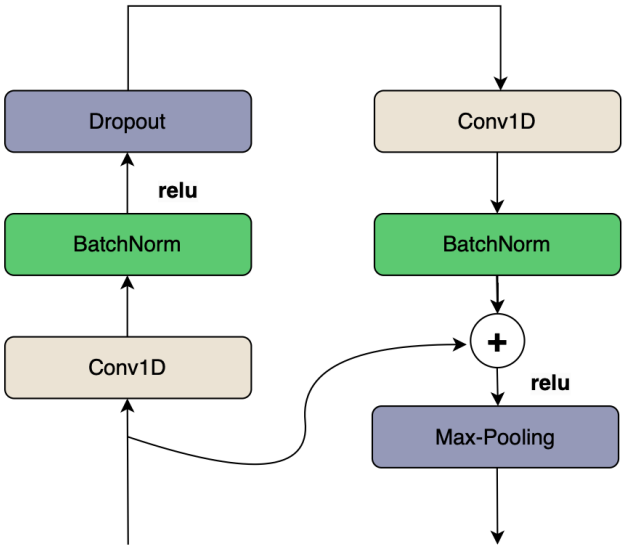


Fig. 6. Proposed new hybrid architecture



b) Building block for ResNet Dropout Block with Previous an subsequent Convolutional layer

Fig. 7. Details about ResNet in proposed architecture

REFERENCES

[1] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, Italy, 2014, pp. 6964-6968, doi: 10.1109/ICASSP.2014.6854950.

[2] Y. Tokozume and T. Harada, "Learning environmental sounds with end-to-end convolutional neural network," 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, USA, 2017, pp. 2721-2725, doi: 10.1109/ICASSP.2017.7952651.

[3] T. Kim, J. Lee and J. Nam, "Sample-Level CNN Architectures for Music Auto-Tagging Using Raw Waveforms," 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 2018, pp. 366-370, doi: 10.1109/ICASSP.2018.8462046.

[4] I.Loshchilov and F.Hutter, "Decoupled Weight Decay Regularization," 2017 International Conference on Learning Representations.