Homework2: Music Auto Tagging

Multi-Label Classification and Metric Learning Approach

20213013 Jiho Kang

October 25th, 2021

**[Algorithm Description]**

***Network Input*** [1] reported that Mel-spectrogram can represent audio input representation better than STFTs and MFCCs. In this assignment, Mel-spectrogram is used as input for all network architectures.

***Building Block*** About abbreviations used in table 2,3, and 4, please refer to table 1.

|  |  |
| --- | --- |
| **Abbreviation** | **Building Block** |
| Mel | Mel-spectrogram + Amplitude to dB + Batch Normalization |
| Conv1d\_B | 1D Convolution + Batch Normalization + Relu + 1D Max Pooling |
| Conv1d\_F | 1D Convolution + Batch Normalization |
| Conv1d\_M | 1D Convolution + Batch Normalization + Relu + 1D Max Pooling |
| ConvY | Vertical Convolution + Batch Normalization + Relu + 2D Max Pooling |
| ConvH | Horizontal Convolution + Batch Normalization + Relu + 2D Max Pooling |
| Conv2d | 2D Convolution + Batch Normalization + Relu + 2D Max Pooling |
| MP1d | 1D Max Pooling |
| MP2d | 2D Max Pooling |
| AP1d | 1D Average Pooling |
| AAP1d | 1D Adaptive Average Pooling |
| FC | Fully Connected |

**Table 1.** Abbreviation for layer’s building block

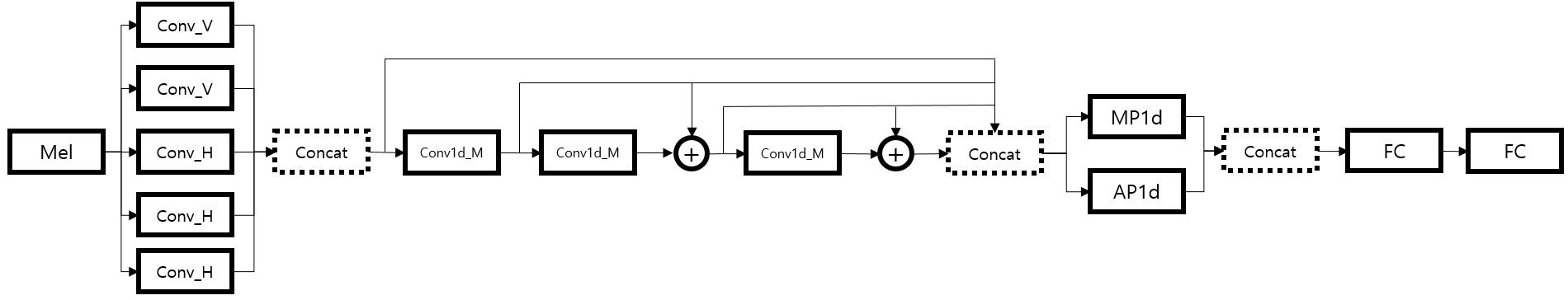
***Network Architectures for Question 2*** Four network architectures were compared; *1D-CNN*, *2D-CNN*, *FCN* (fully convolutional network), and *Musicnn* (pretrained). Refer to table 2 for detail structure of *1D-CNN*, *2D-CNN*, and *FCN*. *Musicnn* was taken from [4]. For more detail, refer to table 3 and figure 1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Order** | **Layer & Output size** | | | | | |
| **1D-CNN** | | **2D-CNN** | | **FCN** | |
| 1 | Mel | (B, 1, 96, 188) | Mel | (B, 1, 96, 188) | Mel | (B, 1, 96, 188) |
| 2 | Conv1d\_B | (B, 32, 64) | Conv2d | (B, 64, 24, 47) | Conv2d | (B, 16, 24, 47) |
| 3 | MP1d | MP2d | MP2d |
| 4 | Conv1d\_B | (B, 32, 22) | Conv2d | (B, 128, 8, 15) | Conv2d | (B, 32, 8, 15) |
| 5 | MP1d | MP2d | MP2d |
| 6 | Conv1d\_B | (B, 32, 8) | Conv2d | (B, 128, 2, 5) | Conv2d | (B, 64, 2, 5) |
| 7 | MP1d | MP2d | MP2d |
| 8 | AAP1d | (B, 32, 1) | Conv2d | (B, 64, 1, 1) | Conv2d | (B, 128, 1, 1) |
| 9 | FC | (B, 50) | MP2d | MP2d |
| 10 |  |  | FC | (B, 50) | Conv1d\_F | (B, 50) |

**Table 2.** Layer & output size of network architecture for question 2

|  |  |  |
| --- | --- | --- |
| **Order** | **Network Architectures & Output Size** | |
| **Musicnn** | |
| 1 | Mel | (B, 1, 96, 188) |
| 2 | [ConvY, ConvY, ConvH, ConvH, ConvH] | (B, 204\*2+51\*3, 188) |
| 3 | Conv1d\_M | (16, 64, 188) |
| 4 | Conv1d\_M | (16, 64, 188) |
| 5 | Conv1d\_M | (16, 64, 188) |
| 6 | [MP1d, AP1d] | (16, 753+753, 1) |
| 7 | FC+BN+Relu | (16, 200) |
| 8 | FC | (B, 50) |

**Table 3.** Layer & output size of Musicnn architecture for question 2



**Figure 1.** Musicnn architecture overview

***Embedding Architectures for Question 4*** Three embedding architectures were compared; *Linear*, *CNN*, and *Musicnn* (pretrained). Refer to table 4 for detail structure.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Order** | **Layer & Output size** | | | | | |
| **Linear** | | **CNN** | | **Musicnn** | |
| 1 | Mel | (B, 1, 96, 188) | Mel | (B, 1, 96, 188) | Mel | (B, 1, 96, 188) |
| 2 | FC | (B, 4096) | Conv2d | (B, 64, 24, 47) | [ConvY, ConvY, ConvH, ConvH, ConvH] | (B, 204\*2+51\*3, 188) |
| 3 |  |  | MP2d | Conv1d\_M | (16, 64, 188) |
| 4 |  |  | Conv2d | (B, 128, 8, 15) | Conv1d\_M | (16, 64, 188) |
| 5 |  |  | MP2d | Conv1d\_M | (16, 64, 188) |
| 6 |  |  | Conv2d | (B, 128, 2, 5) | [MP1d, AP1d] | (16, 753+753, 1) |
| 7 |  |  | MP2d | Conv\_1d\_F | (16, 256) |
| 8 |  |  | Conv2d | (B, 256, 1, 1) |  |  |
| 9 |  |  | MP2d |  |  |
| 10 |  |  | Conv1d\_F | (B, 256, 256) |  |  |

**Table 4.** Layer & output size of embedding architecture for question 4

***Loss Design for Question 4***Triplet is defined as , where is the anchor sample, is positive anchor, and is negative sample. and are sampled from same label but is sample from negative for label . Accordingly, basic triplet loss is defined as shown in equation (1) where equation (2) is a distance metric.In addition to basic triplet loss, disentangled triplet-based model proposed in [2] and track regularization technique proposed in [3] are adopted for question 4.

To utilize disentanglement, the similarity notion with respect to class label is formulated as reported in Table 5. For instance, if is *harpsichord*, then the similarity notion of is *instrument*. The masking function is applied to the embedding feature to achieve disentanglement. Importantly, each mask evenly occupies dimensions based on number of similarity notions and dimension of embedding feature. For better understanding, assume that embedding the feature dimension is 256 (=644). Then, values of 192 (=643) dimensions is killed by masks and only values of 64 dimensions is remained.Thus, the disentangled triplet loss is formed like equation (3).

|  |  |
| --- | --- |
| **Similarity Notion** | **Class label** |
| Genre (12) | classic, techno, rock, indian, opera, country, metal, new age, dance, pop, electronic, classical |
| Mood (5) | soft, quiet, loud, ambient, weird |
| Instrument (31) | harpsichord, sitar, flute, piano, guitar, strings, drums, violin, synth, harp, cello, beat, beats, choral, vocal, vocals, no vocals, male vocal, female vocal, no vocal, solo, female, male, singing, woman, man, choir, voice, male voice, no voice, female voice |
| Tempo (2) | slow, fast |

**Table 5.** Similarity notion and class label for disentanglement

For track regularization, track based triplet is defined as . and are sampled from same song. Different from disentanglement-based loss, it should be noted that mask function is not applied for track regularization term as shown in equation (4). That is regularization is designed considering for all dimensions of embedding space. Total loss function is equation (5) where represents trade-off between semantic similarity and overall track-based similarity [3].

**[Experiments and Results]**

***Question 2***Notation *Musicnn-x* in table 6 indicates that the pre-trained parameters are used up to the *x*th layer, and the layers after that are trained again. The Adam optimizer is used for training. The learning rate to 0.01 is initialized and is reduced by a factor of 2 when the validation loss does not decrease for 30 epochs, up to 4 times, after which early stopping is applied. The results are reported in Table 6.

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Model Params** | **Training Time (s)** | **ROCAUC** |
| 1D-CNN | 17,300 | 196.2 | 85.3 |
| 2D-CNN | 299,892 | 264.6 | 87.7 |
| FCN | 104,184 | 220.0 | 87.8 |
| Musicnn-0 (Fully re-trained) | 774,250 | 633.3 | 88.1 |
| Musicnn-7 (Partially re-trained) | 774,250 | 295.1 | 89.6 |
| Musicnn-8 (Fully pre-trained) | 774,250 | No training | 89.7 |

**Table 6.** Question 2 results

***Question 4***To check effectiveness of track regularization and disentanglement*,* various combinations of loss type were experimented. *Musicnn* embedding model used pre-trained parameters except for last layer. The Adam optimizer is used for training. The learning rate to 0.001 is initialized and is reduced by a factor of 2 when the validation loss does not decrease for 5 epochs, up to 2 times, after which early stopping is applied. The margin for the triplet-based models is set to 0.4. was set to 0.5 for track regularization. The results are reported in Table 7.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Embedding**  **Models** | **Normalization** | **Track**  **Regularization** | **Disentanglement** | **Training**  **Time (s)** | **Multi Label Recall** | | | |
| **R@1** | **R@2** | **R@4** | **R@8** |
| Linear | X | X | X | 179.5 | 42.0 | 56.0 | 69.8 | 81.6 |
| Linear | X | O | X | 188.5 | 45.6 | 59.7 | 72.6 | 82.6 |
| Linear | X | X | O | 184.6 | 40.5 | 55.4 | 67.9 | 79.8 |
| Linear | X | O | O | 195.9 | 45.0 | 59.5 | 72.3 | 82.2 |
| FCN | X | X | X | 179.1 | 44.0 | 57.8 | 70.5 | 81.4 |
| FCN | O | O | X | 191.0 | 45.5 | 59.6 | 72.4 | 82.5 |
| FCN | O | X | O | 183.5 | 43.2 | 57.6 | 71.3 | 82.5 |
| FCN | O | O | O | 198.6 | 45.0 | 59.5 | 71.6 | 82.9 |
| Musicnn | X | X | X | 222.4 | 48.5 | 62.7 | 76.0 | 85.7 |
| Musicnn | O | O | X | 229.0 | 50.0 | 64.2 | 76.9 | 87.1 |
| Musicnn | O | X | O | 238.0 | 50.5 | 64.9 | 77.4 | 86.3 |
| Musicnn | O | O | O | 231.8 | 50.1 | 64.6 | 77.4 | 86.7 |

**Table 7.** Question 4 results for training time, multi label recall

**[Discussion]**

***Experiment Result Analysis for Question 2*** The originally given settings (epochs: 10, optimizer: Nesterov momentum) of *1D-CNN* and *2D-CNN* were changed to (epochs: 30, optimizer: Adam) and the test was performed. As shown in table 6, the results showed that the performance of *1D-CNN* and *2D-CNN* increased from 64, 77 to 85, 87, respectively. It can be seen that the Adam optimizer worked effectively when the epoch was increased.

When comparing *2D-CNN*, *FCN*, and *Musicnn-0*, it was confirmed that the number of parameters and performance were not proportional. Although the number of parameters of FCN was noticeably smaller than that of *2D-CNN* and *Musicnn-0*, there was no difference in performance. (Refer to table 6)

*Musicnn* model was pretrained using all of the MagnaTagATune dataset. The interesting thing found here was that the performance deteriorated as the learning of the pretrained *Musicnn* layer with a subset of MagnaTagATune was further carried out. The reason for this is speculated that the model learned a better representation from a larger amount of data. (Refer to table 6)

***Experiment Result Analysis for Question 4*** To validate effectiveness of implementation for disentanglement and track regularization, an ablation study was conducted. Looking at the case of *Linear* embedding first, the performance was the best when only track regularization was used, and disentanglement did not improve the performance. It was judged that if the masking function was applied in a state where non-linearity could not be expressed, it adversely affected the representation of data. However, in the case of *FCN* and *Musicnn*, performance is improved by using disentanglement. (Refer to table 7)

Similar to the experimental results of Question 2, the pre-trained *Musicnn* had the highest performance. Since it has been trained with a larger amount of data, it seems that Model can have a better representation of data when compared with other models which only were trained using subset of data.

**Reference**

[1] Keunwoo Choi, George Fazekas, Mark Sandler. Automatic Tagging using Deep Convolutional Neural Networks. *arXiv.* <https://arxiv.org/abs/1606.00298>, 2016.

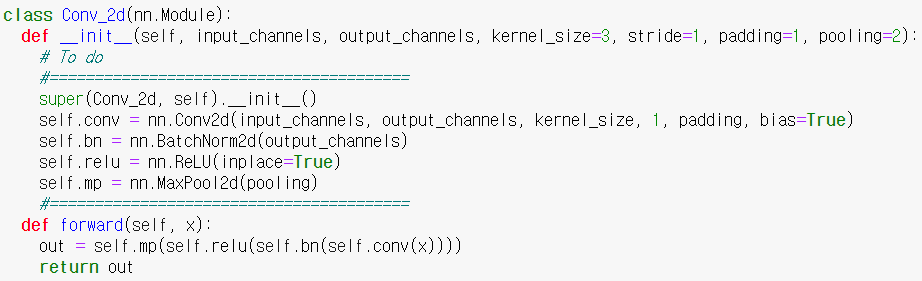
[2] Jongpil Lee, Nicholas J. Bryan, Justin Salamon, Zeyu Jin, Juhan Nam. Metric Learning vs Classification for Disentangled Music Representation Learning. *arXiv*. <https://arxiv.org/abs/2008.03729>, 2020.

[3] Jongpil Lee, Nicholas J. Bryan, Justin Salamon, Zeyu Jin, Juhan Nam. Disentangled Multidimensional Metric Learning for Music Similarity. *IEEE ICASSP,* 2020.

[4] State-of-the-art Music Tagging Models. <https://github.com/minzwon/sota-music-tagging-models>. *Github*.

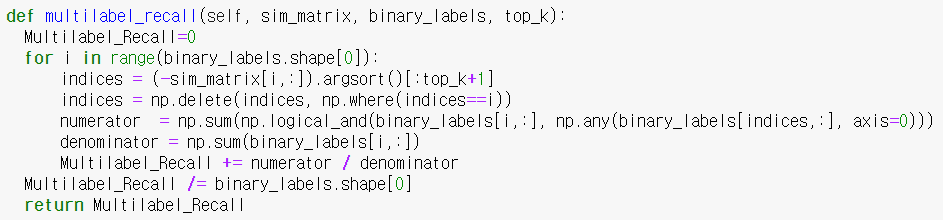
**Appendix**

1. Question 1: Implement a CNN based on a given model specification

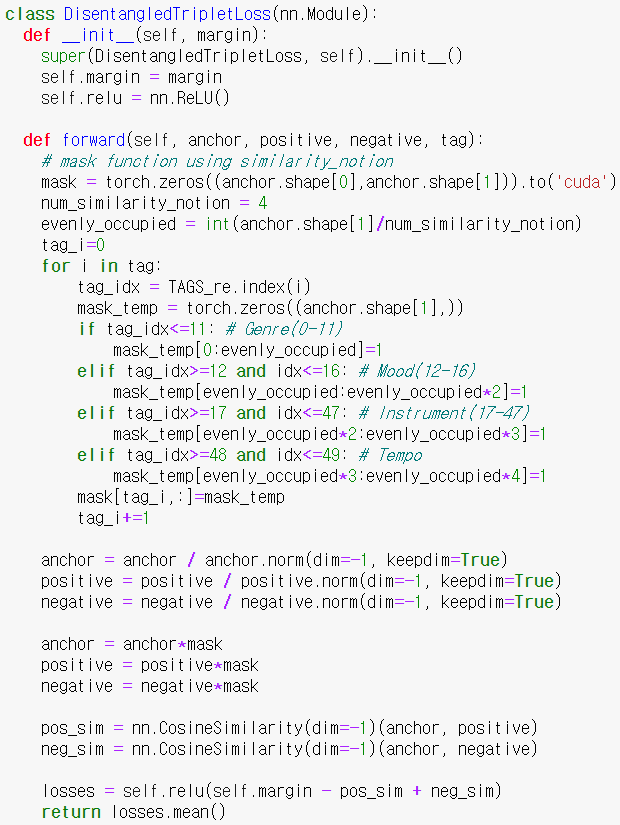




1. Question 3: Implement the evaluation metric



1. Disentangled triplet loss



1. Track Regularization

