Homework #2: Musical Genre Classification

1. Algorithm Description

1.1 Data Preprocessing

Feature Extraction

As same as HW#1, before extracting the features, I randomly chose one sound sample for each of 8 genres of music, and listen to the sounds and checked all spectrograms. (*feature_check.ipynb*) With the naked eye, the default parameter setting of STFT(Window size=1024, Hop size=512, FFT size=1024) was enough to distinguish most of music genres, so I kept using that parameter as starting point of feature extraction. Mainly, Mel-Spectrogram feature was used. For Question 2, embeddings from msd-musicnn model was used as an input feature. For extracting all spectral feature, librosa.features library was used.

Split the Dataset

Since the GTZAN dataset had only training and dataset, it was hard to know when the training was overfitted or not. Therefore, validation set was split from the training set. For splitting, StratifiedShuffleSplit from sklearn.model_selection library was used to guarantee splitting equal number of samples of each music genres into validation set; 5 samples of each music genres are used. The ratio of train : validation set is 93 : 7.

Augmentation

Because the GTZAN dataset is a small dataset for music classification(730 audio samples with length of 30 seconds), data augmentation and segmentation were conducted. For augmentation, white noise, time shift, time stretch was used, and the intensity of augmentation was randomly chosen for each training and validation samples. Augmentation leads to double the dataset.(original:augmented=1:1) I tried to augment the data reasonably, which doesn't harm the original sound; to listen the augmented sound sample, please refer to *data_augmentation.ipynb*.

Segmentation

Listening to the whole 30 secs of each audio clip is not necessary to classify the genre of the music. Therefore, instead of using the whole Mel-Spectrogram of an audio clip as an input to the network, it was segmented every 4 seconds and fed to network, which leads

generating 7 segments for each audio clip in average and increasing the total amount of data.

1.4 Models

Baseline and Question 1

Baseline and Question 1 model has 3 and 4 layers consisting of 1D Convolution, BatchNormalization, ReLU, and Maxpooling with reduced kernel size as layer goes deeper. While the Baseline model kept using the equal channel size 32, Question 1 model doubles the channel size for each layer.

Question 2

Question 2 model is 2-layer MLP which consists of Linear layer - BatchNorm - ReLU – Linear Layer. With vanilla 2-layer MLP, the accuracy was fluctuated, test accuracy, and adding batch normalization solved the problem and included in the model.

For this model, hidden unit size was set to 32; the number of hidden units of the layer didn't affect the validation accuracy. Since the model utilizes the extracted feature embeddings of the pre-trained model(mtt-musicnn and msd-musicnn), the model has more powerful performance with simpler structure than Baseline and Question 1.

Question 3 : Customized 2D CNN, Popular Models of ImageNet Challenge

To compare the performance of 1D vs. 2D CNN model, Customized 2D CNN was implemented, which has same kernel size, channel, stride, padding, maxpool kernel size and number of layers(4) with Question 1.

Since the Mel-Spectrogram was regarded as an image when fed to the CNN based models, models which made great improvement in image classification task were considered, to get improved performance with reduced training time. Among of many popular models in ImageNet Challenge, VGG and ResNet, which won the Challenge in 2014-2015, were chosen, because these two models provides pre-trained models with different layer numbers, which leads to further investigation of effect of changing number of layers. Model variants of VGG(VGG16, VGG19) and ResNet(ResNet18, ResNet34, ResNet50, ResNet 101) with pretrained weight were from the torchvision.models library. For finetuning, training was proceeded after final FC layer was modified into 8 classes.

Question 3 : ResNet34+MSD Embed Model

When training, each of ResNet with Mel-Spectrogram and 2-Layer MLP with MSD musicnn's extracted feature had good performance. Therefore, final model(SpecAndEmbed) uses both of the features, by concatenating each network's output (**Figure 1**). Empirically, using BatchNorm on the bottom of the Embed_mlp layer leads to the performance of the model worse, so that it was removed.

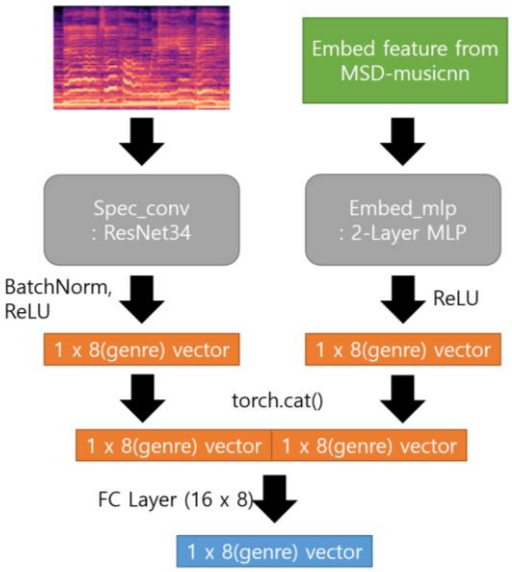

Figure 1 Proposed Algorithm

1.4 Training

Training objective is minimizing the cross-entropy loss between predicted music genre and ground truth. Empirically, performance was better and convergence was faster when using Adam than SGD optimizer, Adam optimizer was used for all experiment. For hyper parameter, learning rate 0.0001, momentum 0.9 was set. For faster train and validation, batch size 32 and 16 were used respectively, and test batch size was 7; test batch size was determined to be same as the number of the segments in 1 audio clip.

While training, validation was conducted after each epoch, and model weight with highest accuracy on validation set was chosen and saved as best model weight. After training, accuracy on test set was calculated based on best model weight. For testing segmented case, the accuracy was calculated by averaging(max pooling) the prediction of each of 7 segment. All train, validation, test set accuracy was calculated by averaging the result of independent 3 train runs with different initialization.

2. Experiments and Results

2.1 Effect of Augmentation and Segmentation

To overcome the shortage of dataset, augmentation and segmentation were used. To investigate the effect of augmentation and segmentation, the validation accuracy was compared with the vanilla dataset. (**Table 1**)

Since the segmentation changes the input size (temporal dimension), it was impossible to compare fairly with the non-segmented input, without changing the model architecture. Therefore, for 1D ConvNet(Baseline, Q1), MLP(Q2) model and Q1+MSD Embed model, accuracy wasn't calculated in segmentation and augmentation plus segmentation condition. In these cases, accuracy was increased after augmentation.

Also, for the SpecAndEmbed models(Q1+MSD, Base2DCNN+MSD, ResNet34+MSD) with segmented condition, non-segmented embedding was fed to the model, due to issue of augmentation and segmentation order. Augmentation was processed in .wav level and segmentation was done in Mel-Spectrogram level, therefore, it was impossible to segment the extracted embeddings of MSD-musicnn from augmented audio. The opposite direction, doing segmentation first and then augmentation was not possible due to the time stretch augmentation.

Nevertheless, for most of models, highest accuracy was achieved when using both of augmentation and segmentation. Therefore, following experiments were conducted, assuming that augmentation plus segmentation improves the accuracy.

2.2 Effect of Deep CNNs

Since torchvision.models library provides VGG, ResNet models with different number of layers, accuracy with different number of layers were compared for these models (**Table 2**). Also, there is a tendency of models to increase and then decrease the performance when the layer is deeper. Above of all, ResNet34 had the highest validation accuracy and test accuracy. Therefore, ResNet34 model was used for the final model, ResNet34+MSD, which results in 86% on average.

	original dataset			augmentation			segmentation			augmentation + segmentation		
Model	train acc(%)	val acc(%)	test acc(%)	train acc(%)	val acc(%)	test acc(%)	train acc(%)	val acc(%)	test acc(%)	train acc(%)	val acc(%)	test acc(%)
Baseline	68.33	58.33	43.78	90.74	77.92	60.67	.	.	.	.	.	.
Base1DCNN(=Q1)	71.6	54.17	54.44	85.22	78.33	61.33	.	.	.	.	.	.
MTT Embed(=Q2)	74.07	66.67	62.0	82.78	90.83	73.78	.	.	.	.	.	.
MSD Embed(=Q2)	83.77	76.67	80.44	86.42	92.08	**86.67**	.	.	.	.	.	.
Base2DCNN	85.62	70.83	69.11	86.73	87.5	73.78	94.53	76.55	70.89	92.4	77.62	73.33
Q1+MSD Embed	84.69	62.5	55.78	92.28	80.0	74.89	.	.	.	.	.	.
Base2DCNN+MSD	87.72	70.83	69.11	91.94	90.0	79.33	95.55	83.1	81.0	94.7	85.54	84.0
resnet34+MSD	94.26	80.0	82.0	96.67	98.33	83.56	97.05	89.17	84.89	99.59	**91.97**	**86.0**

Table 1 Train, Validation, Test accuracy comparison of Different Models

Model	train acc(%)	val acc(%)	test acc(%)
VGG13	97.95	82.96	76.22
VGG16	97.66	85.08	**78.22**
VGG19	93.53	81.01	75.56
ResNet18	98.19	73.33	77.78
ResNet34	99.06	85.08	**80.44**
ResNet50	99.35	83.3	78.44
ResNet101	99.06	82.1	78

Table 2 Train, Validation, Test accuracy comparison of different layers of VGG, ResNet

	PREDICTED								
	Genre	Cla	Cou	Dis	Hip	Jaz	Met	Pop	Reg
TRUE	Classical	20	0	0	0	0	0	0	0
	Country	0	13.3	1	0	2	0.3	**6.3**	0
	Disco	0	0	20.3	0.6	0	0	1	0
	Hiphop	0	0	0	18	0	0	0	0
	Jazz	1	0.6	0	0	15.3	0	0	0
	Metal	0	0	0	0	0	19.6	0.3	0
	Pop	0	1.3	0	1.6	0	0	9.6	0.3
	Reggae	0	1.6	0.3	1.6	0.3	0	0.3	12.6

Table 3 Confusion matrix for each class on test set(avg) of ResNet34+MSD, accuracy=86%

3. Discussion

3.1 1D vs 2D convolution

When comparing the train, validation, test accuracy of 1D(Base1DCNN) vs 2D Conv Net(Base2DCNN), 2D ConvNet's accuracy is much higher than 1D ConvNet, even though they are designed to have same kernel size and similar model structure. It seems that it is due to the Spetrogram's characteristic; Model can learn the not only temporal but also harmonic relationship with 2D convolution, while 1D convolution mainly focused on temporal relationship.

3.2 Why MSD musicnn has the Best Performance?

Although there were many efforts to improve the performance, it was lower than the Q2, extracted musicnn embeddings with MLP. Therefore, it would be useful to know the structure of musicnn and why it can make better performance. According to the documentation of musicnn, musicnn mainly consists of musically motivated CNN, which has several different filter shape(horizontal, vertical), rather than using normal rectangular filter. Also, it has dense layers which enable the fast gradient flow. Above all, since ResNet utilizes the skip connection, using only rectangular filter is one of the main difference with the musicnn and proposed algorithm. Also, since musicnn was trained with 200k training songs, the amount of data also largely impacted on the musicnn's performance.

3.3 Future Work

I wanted to try these within the time limit, but I didn't. Therefore, I write down them as just a suggestion for future works.

Adding Other Spectral Features as Input

As shown in confusion matrix **(Table 3),** even though the model made 86.0% test set accuracy on average, the model sometimes classifies country music as pop music. While observing spectrogram or listening each sample as a human, they are not easy to distinguish. To classify these two classes, it would be possible to use domain knowledge(music). Generally, a drum beat is known for important factor for distinguish the genre of a music. Therefore, adding other temporal related features rather than just using Mel-Spectrogram will be possible.

Using Other Augmentation Method

I used .wav level augmentation using librosa and numpy for this HW#2. However, since Mel-Spectrogram is an input as an image, it could be possible to stretch the data in image domain such as SpecAugment. Also, for this HW#2, I used offline data augmentation for saving time while training. It was possible because the train data was small. However, another suggestion is doing online data augmentation. When doing online data augmentation, the augmented data are different every epoch, which enables more variety of training set than offline augmentation.

4. Conclusion

In this HW#2, 1D and 2D convolution, popular models in ImageNet Challenge(VGG, ResNet) were used and I finally get the highest test accuracy of 86% with ResNet34+MSD. While doing HW#2, I could know more about the importance of the dataset size in Deep Learning, and some audio data augmentation method, and could know the difference between 1D and 2D convolution. Also, from the musicnn's case, I could realize that music domain knowledge is important as well as deep learning knowledge.

5. References

[1] Juhan Nam, 2020, Musical Applications of Machine Learning, Lecture Note 6-8

[2] Data Augmentation for Audio Data, 2019, https://melon1024.github.io/data_aug/

[3] Scikit Learn, 0.23.2, Stratified Shuffle Split,

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html

[4] Librosa version 0.8.0., Features, https://librosa.org/doc/latest/feature.html

[5] Musicnn, https://github.com/jordipons/musicnn

[6] SpecAugment, 2019,

https://ai.googleblog.com/2019/04/specaugment-new-data-augmentation.html