Machine Learning Final Report

Hongjia Huang hh3043

1 Short Abstract

For this report I would introduce the data prepossessing process and the model I used. Also, my Kaggle name is Hongjia Huang as well.

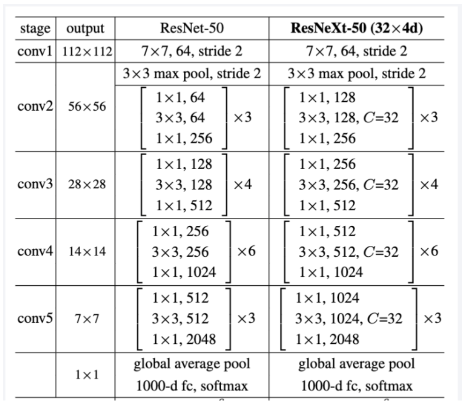
2 Data Prepossessing

We use the package Spleeter to split the music into vocal and instrumental parts. Then, we use the package Librosa to convert the vocal part of music into a tensor of shape [128, 130] which convey the information of frequency with respect to time in the music. Later, I simply perform a slicing to transform the tensor into size [128, 128] and reshape it to become a tensor of size [1, 128, 128] such that the data can be passed into image classification models as a image of 1 channel (gray scale image).

3 Models

3.1 Architectures

The models I tried on this competition: Resnet18, ResNext50, ResNext101, densenet201, LSTM, Shakeshake resnet26, Shakeshake resNext and a variation of resnet that is integrated with attention modules. Finally, I used an ensumable between ResNext50 and Densenet201 for my final submission that achieved 83.04% accuracy on the testing dataset.



stage	output	ResNet-50		**ResNeXt-50 (32×4d)**	
conv1	112×112	7×7, 64, stride 2		7×7, 64, stride 2	
conv2	56×56	3×3 max pool, stride 2		3×3 max pool, stride 2	
		[1×1, 64; 3×3, 64; 1×1, 256]	×3	[1×1, 128; 3×3, 128, C=32; 1×1, 256]	×3
conv3	28×28	[1×1, 128; 3×3, 128; 1×1, 512]	×4	[1×1, 256; 3×3, 256, C=32; 1×1, 512]	×4
conv4	14×14	[1×1, 256; 3×3, 256; 1×1, 1024]	×6	[1×1, 512; 3×3, 512, C=32; 1×1, 1024]	×6
conv5	7×7	[1×1, 512; 3×3, 512; 1×1, 2048]	×3	[1×1, 1024; 3×3, 1024, C=32; 1×1, 2048]	×3
	1×1	global average pool 1000-d fc, softmax		global average pool 1000-d fc, softmax	

Figure 1: Resnet50 v.s. ResNext50 architecture

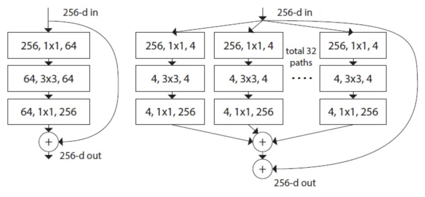


Figure 2: difference between the residual connections of Resnet and ResNext

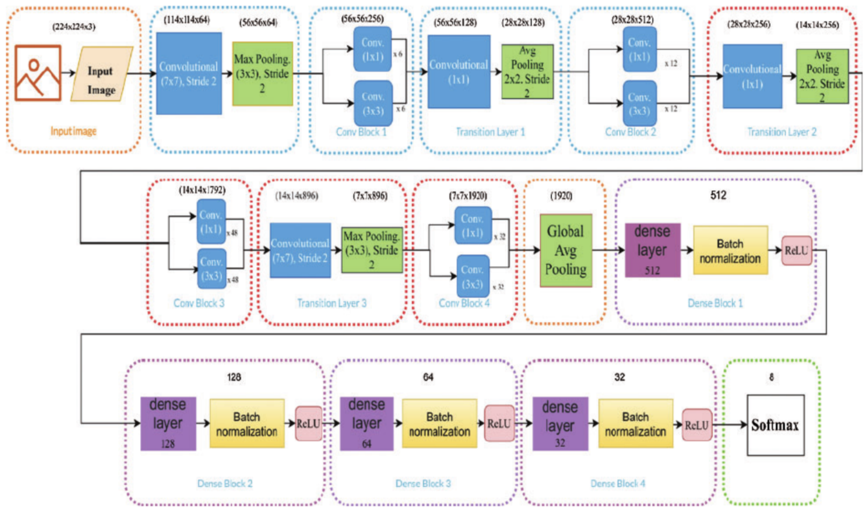


Figure 3: Densenet201 architecture

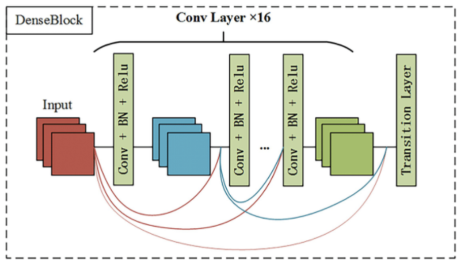


Figure 4: typical denseblock connections

3.2 Computational Consumption

While all models I tried can be trained on RTX8000, it is suggested to use more powerful GPUs like A100, H100, and A800 when trying models with deep layers such as densenet201 and ResNext50. Many thanks to NYU Greene HPC and NYUSH HPC.

4 Training

Apart from using model architectures that are already provided by PyTorch, I also make some modifications to the origin architecture and apply some useful tricks in the training procedure.

4.1 Modification to architectures

Most models I tried on are variants of resnet. In the original resnet architecture in PyTorch, the first layer is usually a convolution layer with filter size 7, stride 2 and padding 3. Then, it is followed by a max pooling layer with kernel size 3, stride2 and padding 1 . However, considering that models such as resnet18 are trained on images with size [224, 224] and the data we applied on this model are gray scale images of size [128, 128], we do not need to down sample the input so much in the early layers. So, for the first layer, I changed the filter to size 3 with stride and padding set to 1. Then, for the second layer, I simply skip the max-pooling part by settling the kernel size to 1 with stride 1 and padding 0.

4.2 Tricks in Training

Firstly, during training, I store the model with the largest validation accuracy and run prediction for testing dataset based on this model checkpoint.

Secondly, I use Cosineannealingwarmsup as the learning rate scheduler, I set the period for a warms up cycle to be roughly the number of iterations needed for the model to converge. Then, with each warmsup, we could (possibly) find a different local optimal. We only need to pick the one with largest validation accuracy. However, this process is very time consuming and higher validation accuracy does not necessary means higher testing accuracy. For these reasons, I did not include multiple warms up steps in my training for the final models I acquire but I still use Cosineannealing as the schedular.

Thirdly, I applied Spec Augment to perform time and frequency masking inside each batch, making the augmentation process more random.

Finally, I use Mixup for data augmentation. It is believed that applying mixup for too many epochs hurt the model's generalization towards new data. For that reason, I tried using mixup for the first half of the training process, and use the simple crossentropy loss for the second half. This helps to achieve higher performance on deep models such as Densenet201.

4.3 Ensemble

I trained different models independently and then perform ensemble by adding the output probabilities together and return the label corresponds to the largest probabilistic value, as my hypothesis is that different models are more likely to capture different features of input data than similar models.

I tried different combinations between the models and the ensemble that works the best for me is using only 1 Desnenet18 model together with 1 ResNext50 model.

I also tried to do a weighted sum of different models' output by training a meta model with a single convolution layer followed by batch normalization. However, the weighted model overfit quickly such that the testing accuracy decrease slightly.

5 Experiments

I tried various models with various method. The conclusions I draw are: 1. converting original data to gray scale image yields higher performance comparing to RGB image. 2. Mixup and Spec Augmentation (time and frequency masking) helps to increase the accuracy around 2%. 3. ensemble without reweigh could increase the performance whereas ensemble with reweigh is easy to overfit.

6 Discussion

I tried to implement GAN and Autoencoder for this competition. However, I did not finished their implementations. I believed that Autoencoder could help in reconstructing images of the same labels and acquire more input data for the training process. This could potentially lead to better performance.

Also, because the dataset are constructed by first dividing different songs into training and testing sets, then songs are sliced into snips of 3 seconds, the manually separated training and validation datasets would have similar musics whereas validation and testing datasets have different musics. Hence, it makes more sense to do a K-fold Cross-Validation, this could potentially lead to better performance as well.