Cheng Zeng (Jane)

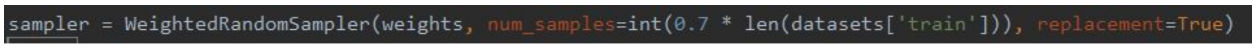
Professor Jafari

Machine Learning II

22 April2020

Individual Report

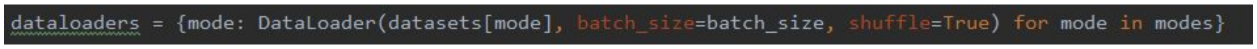
In the beginning, I tried to useWeightedRandomSampler to ensure that each batch sees a proportional number of all classes.



```
sampler = WeightedRandomSampler(weights, num_samples=int(0.7 * len(datasets['train'])), replacement=True)
```

The WeightedRandomSampler obtains the list of target classes and shuffle. Then, get the class counts and calculate the weights and class by taking its reciprocal. Next, it assigned the weight of each class to all the samples. Finally, obtain the corresponding weight for each target sample. Therefore, we received our data proportion is 1:2:1.

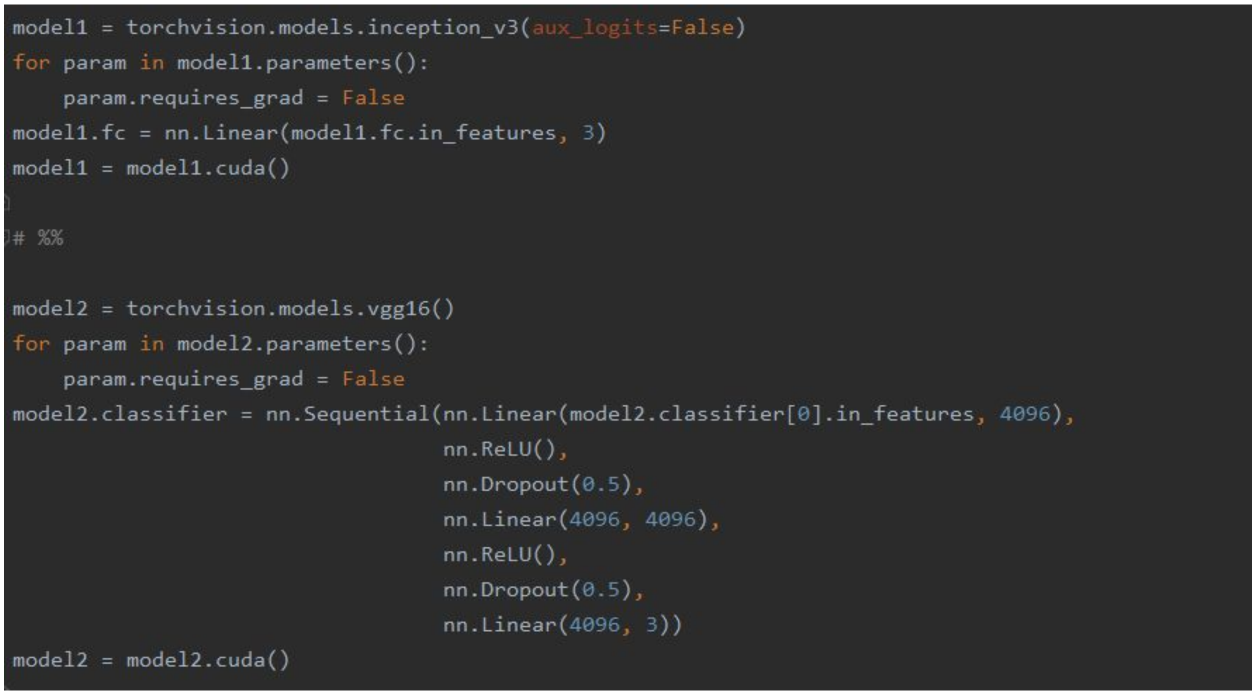
However, after use sampler, my training accuracy became around 90% but the test accuracy was only around 50%. That means my training is overfitting. Therefore, I used the general DataLoader which the result didn't overfit.



```
dataloaders = {mode: DataLoader(datasets[mode], batch_size=batch_size, shuffle=True) for mode in modes}
```

After our group received the good result with ResNet18, I suggest that we try whether adding inception V3 and VGG 16 to ResNet will show better results with only the original dataset instead of after data balancing.

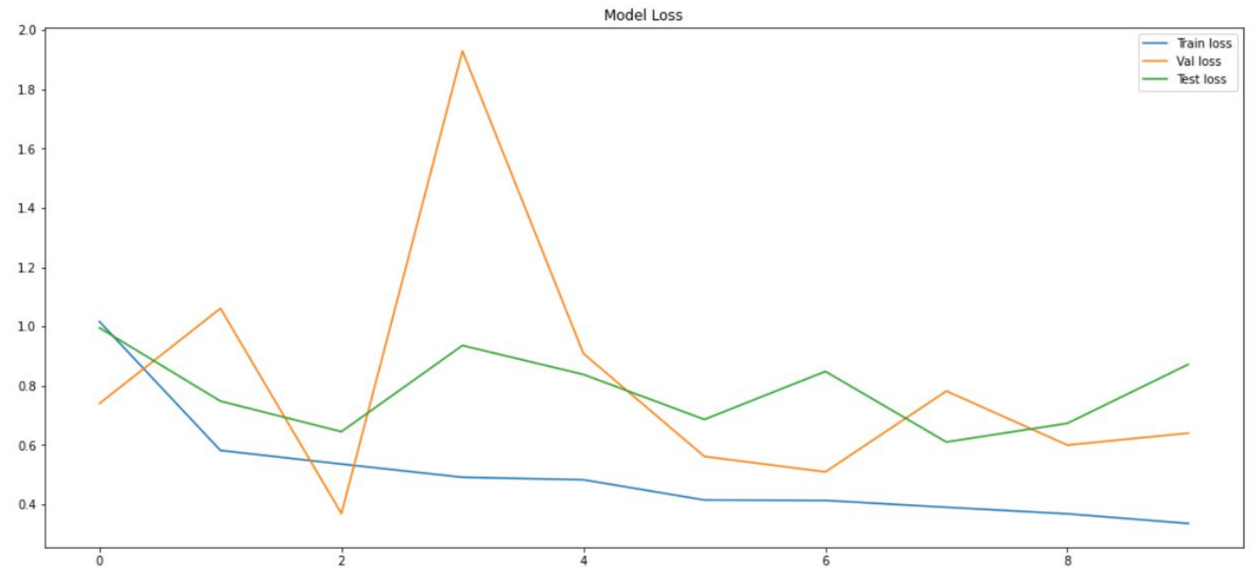
The reason I chose Inception V3 and VGG 16 was because I read several articles about image classification models. People discussed that these two models are actually better to use. Especially the GCP guides shows that the Inception V3 has attained greater than 78.1% accuracy in about 170 epochs on each of these.

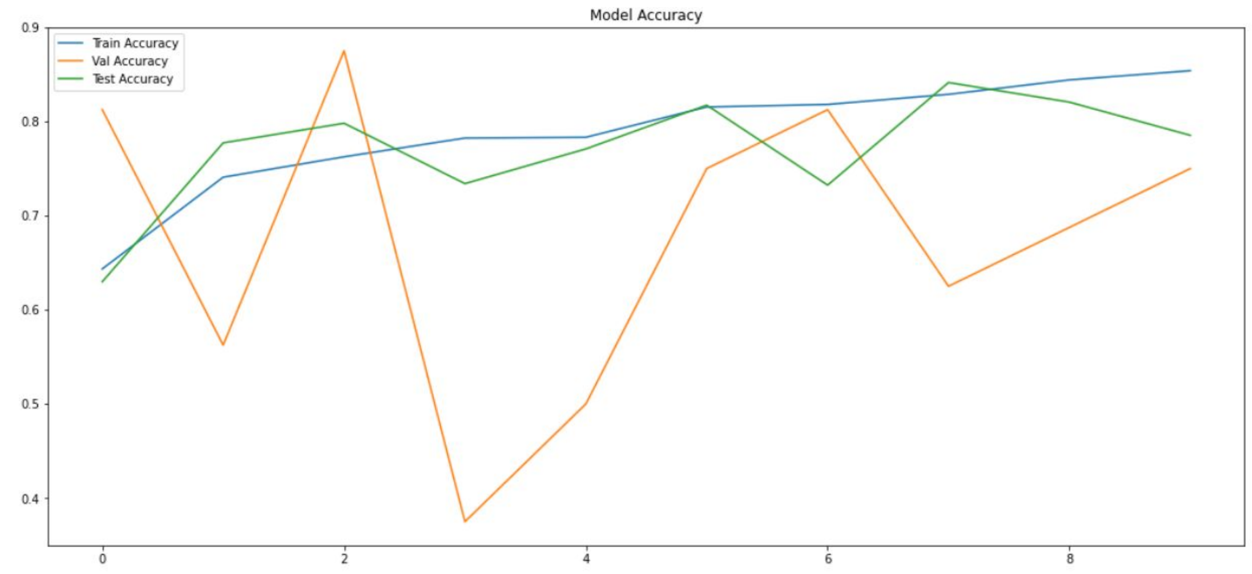


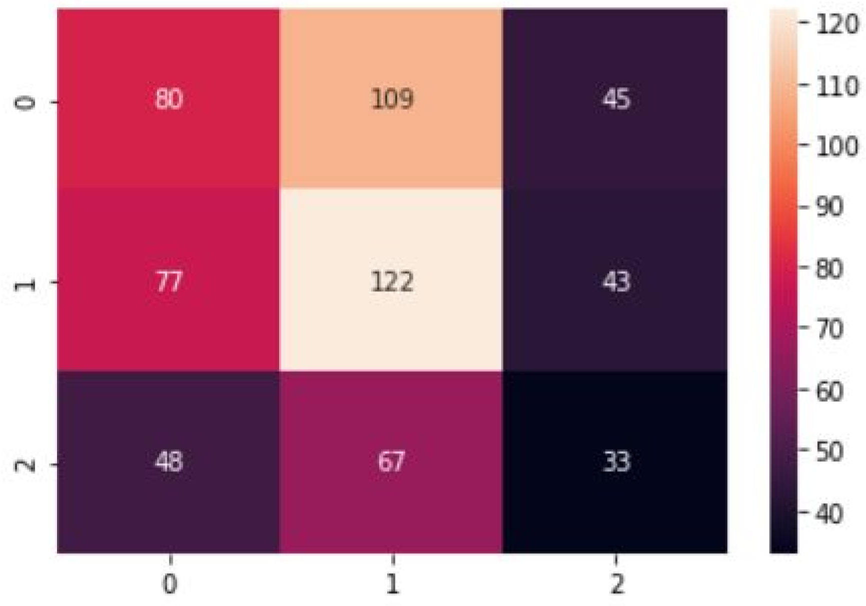
```
model1 = torchvision.models.inception_v3(aux_logits=False)
for param in model1.parameters():
    param.requires_grad = False
model1.fc = nn.Linear(model1.fc.in_features, 3)
model1 = model1.cuda()

# %%

model2 = torchvision.models.vgg16()
for param in model2.parameters():
    param.requires_grad = False
model2.classifier = nn.Sequential(nn.Linear(model2.classifier[0].in_features, 4096),
                                  nn.ReLU(),
                                  nn.Dropout(0.5),
                                  nn.Linear(4096, 4096),
                                  nn.ReLU(),
                                  nn.Dropout(0.5),
                                  nn.Linear(4096, 3))
model2 = model2.cuda()
```

First of all, I used a similar augmentation with CNN model but added ‘transfer.ColorJitter’ to make the picture more recognizable. For the model, I only use the parameters with [brightness = 0.5, contrast = 1.5]. Then, I set up the parameters are 128 batch size, 0.005 LR, 0.001 decay and 25 epoch. Then I received the train accuracy was around 79% and the test accuracy was around 60%. After that we also tried to change the epoch to 50 to see whether the accuracy will increase with that. However, the result was more stable around 20 epochs. Following step, I tried to use SGD and Adam for the optimizer. After multiple trials, Adam optimizer turned out better results with around 80% of training and validation accuracy and 72% of test accuracy. Nonetheless, validation accuracy was very unstable because the original validation folder only contains 16 pictures with no viral pneumonia. Kartik mentioned that vgg16_bn() might lead me to a better result. Suppostly, vgg16_bn() is better than vgg16 based on people's experiments. However, I didn't see the result significantly change with our dataset.

Therefore, the following plots are the best result I have ever got：


Model Loss
Train loss
Val loss
Test loss
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
2
4
6
8


Model Accuracy
Train Accuracy
Val Accuracy
Test Accuracy
0.9
0.8
0.7
0.6
0.5
0.4
0
2
4
6
8



Reference

https://cloud.google.com/tpu/docs/inception-v3-advanced#optimizer

https://forums.fast.ai/t/arch-vgg16-vs-arch-vgg16-bn/7746/2