DLSR\_lab2\_0516310

Introduction

The lab is going design customized dataset with data augmentation. Also, try to use Weighted Random Sampler to balance the number of each class.

Experiment Setup

* Python 3.7.4
* Pytorch 1.2.0
* Pytorchvision 0.4.0
* Imgaug 0.4.0
* Pillow 6.1.0
* Matplotlib 3.1.3

Workflow

1. Data Preprocessing
2. Test on Validation Data
3. Loop : Train on Training Data
4. Loop : Test on Validation Data
5. Loop : If the Validation Accuracy is better, then save the weights
6. Test on Testing Data

Data Preprocessing

1. Baseline
   1. Shuffle = True (equal to Random Sampler)
   2. Normalization ([0.4965, 0.3980, 0.3058], [0.3071, 0.2927, 0.2835])
2. Augmentation
   1. Weighted Random Sampler
   2. GaussianBlur with p = 0.5
   3. iaa.Fliplr with p = 0.5
   4. iaa.Affine(rotate = (-45, 45), mode='symmetric') with p = 0.5
   5. iaa.OneOf([iaa.Dropout(p = (0, 0.1)), iaa.CoarseDropout(0.1, size\_percent=0.5)]) with p = 0.5
   6. iaa.AddToHueAndSaturation(value = (-10, 10), per\_channel=True) with p = 0.5
   7. Normalization ([0.4965, 0.3980, 0.3058], [0.3071, 0.2927, 0.2835])
3. Augmentation 2
   1. Weighted Random Sampler
   2. GaussianBlur with p = 0.5
   3. iaa.Fliplr with p = 0.5
   4. iaa.Affine(rotate = (-45, 45), mode='symmetric') with p = 0.5
   5. Normalization ([0.4965, 0.3980, 0.3058], [0.3071, 0.2927, 0.2835])

Model

1. Build-in model: wide\_resnet101\_2
2. Fully-connected layer (1000, 512) + relu
3. Fully-connected layer (512, 128) + relu
4. Fully-connected layer (128, 11)

Result

1. Baseline (Epoch 20, Learning Rate 0.001, Batch Size 8)
   1. Total Accuracy: 90.56%
   2. Average Per-class Accuracy: 90.04%
   3. Class 0: 368 cases / 90.22%
   4. Class 1: 148 cases / 79.73%
   5. Class 2: 231 cases / 83.98%
   6. Class 3: 500 cases / 91.20%
   7. Class 4: 335 cases / 85.67%
   8. Class 5: 287 cases / 88.85%
   9. Class 6: 432 cases / 94.94%
   10. Class 7: 147 cases / 95.24%
   11. Class 8: 96 cases / 94.79%
   12. Class 9: 303 cases / 92.08%
   13. Class 10: 500 cases / 94.20%
2. Data Augmentation



1. Data Sampler



1. Result of Data Augmentation (Epoch 20, Learning Rate 0.001, Batch Size 8)
   1. Total Accuracy: 88.98%
   2. Average Per-class Accuracy: 87.72%
   3. Class 0: 368 cases / 82.61%
   4. Class 1: 148 cases / 70.27%
   5. Class 2: 231 cases / 80.09%
   6. Class 3: 500 cases / 90.40%
   7. Class 4: 335 cases / 85.37%
   8. Class 5: 287 cases / 91.29%
   9. Class 6: 432 cases / 94.44%
   10. Class 7: 147 cases / 93.20%
   11. Class 8: 96 cases / 90.62%
   12. Class 9: 303 cases / 91.42%
   13. Class 10: 500 cases / 95.20%
2. Result of Date Augmentation (Transfer Learning from Baseline, Epoch 20, Learning Rate 0.0001, Batch Size 8)
   1. Total Accuracy: 92.47%
   2. Average Per-class Accuracy: 91.94%
   3. Class 0: 368 cases / 91.85%
   4. Class 1: 148 cases / 79.05%
   5. Class 2: 500 cases / 89.61%
   6. Class 3: 335 cases / 91.20%
   7. Class 4: 287 cases / 91.64%
   8. Class 5: 432 cases / 91.99%
   9. Class 6: 147 cases / 93.29%
   10. Class 7: 96 cases / 95.92%
   11. Class 8: 303 cases / 95.83%
   12. Class 9: 500 cases / 93.73%
   13. Class 10: 231 cases / 97.20%
3. Result of Date Augmentation 2(Transfer Learning from Baseline, Epoch 20, Learning Rate 0.0001, Batch Size 8)
   1. Total Accuracy: 93.28%
   2. Average Per-class Accuracy: 92.80%
   3. Class 0: 368 cases / 93.75%
   4. Class 1: 148 cases / 83.11%
   5. Class 2: 500 cases / 90.48%
   6. Class 3: 335 cases / 91.00%
   7. Class 4: 287 cases / 92.45%
   8. Class 5: 432 cases / 91.99%
   9. Class 6: 147 cases / 94.68%
   10. Class 7: 96 cases / 94.56%
   11. Class 8: 303 cases / 96.88%
   12. Class 9: 500 cases / 93.40%
   13. Class 10: 231 cases / 98.40%

Discussion

1. The average per-class accuracy doesn’t improve after data augmentation if I re-train the model, but it improves if I start training from baseline. Why?
   1. Data augmentation increases variance of data, so it may need more epochs on training.
   2. If we start training from baseline, we can avoid local minimum.
2. If I take out some data augmentation, the average per-class accuracy improves. Why?
   1. If we observe the changes of per-class accuracy, some of them increase and the others decrease. These data augmentation I applied may help with specific categories, but also lowering accuracy of other classes.

Other

1. Why my customized dataset consumes more memory?
2. Quick python script to get mean and std.
   1. torch.stack([t.mean(1).mean(1) for t, c in trainset]).mean(0)
   2. torch.stack([t \*\* 2 for t, c in trainset], dim = 1).std((1, 2))
3. The system order of classes is sorted by character instead of number.