EEEM071 coursework report template

Nam Tran

Abstract

Among 21 challenges for VeRi, 16 were about the sample quality and 2 were about the lack of data. As such, it is reasonable to finetune pretrained models on VeRi in hope that the generalization power of pretrained model can overcome those 18 issues. However, in the theme of consistency, this report demonstrates the need to revisit the common finetuning rules.

1. Introduction

Vehicle Re-identification (VeRi) is "Image Retrieval" task [1] applied for VeRi dataset where the dataset was created by vehicles captured on traffic cameras. Among 21 challenges for VeRi, 16 were about the sample quality and 2 were about the lack of data (section 7: Challenges Regarding Vehicle Re-identification in [2]). When there's lack of quality samples, the general common sense is to finetune a pretrained model.

Nearly a decade of experience with Convolution Neural Network (CNN) has developed institutional common senses in how to properly finetune it. Yet there is a lack of recent sanity check on the institutional common sense in finetuning itself. In this report, I will revisit the following rules with counter-rule:

- Rule 1 - Keep the consistency between pretrain and finetune datasets.
- Rule 2 – It is okay to finetune with different optimizer and learning rate-scheduler (lr-scheduler)
- Rule 3 – Using fully-connected layer in small dataset results overfitting and difficulty in convergence.
- Rule 4 – Aim for at least 30 epochs.

Different from standard format in writing research paper the discussion part is shown straight away in introduction part due to space constraint of the report and allow the readers to engage straight away to key experiment findings before reading the experiments.

1.1. Rules and counter-rules

Rule 1 – Keep the consistency between pretrain and finetune datasets. It is a common practice to normalize finetuned dataset with the mean (0.485, 0.456, 0.406) and standard deviation (0.229, 0.224, 0.225) of ImageNet to keep the data-augmentation consistency. The impact of data augmentation is demonstrated by [3] when transferring different domains, which applies to this coursework as VeRi and ImageNet are also different domains.

Counter-rule 1 – It is beneficial to add new data augmentations on top of pretrained augmentation. In fact, the most consistent finetuning recipe in this report produces the least accuracies on both resnet50 and resnet50 with fully-connected layer (resnet50_fc512), see my table 1. Thus, demonstrating the necessity of adding more data augmentation methods despite the risk of inconsistency.

Rule 2 – It is okay to finetune with different optimizers, loss, lr-schedulers. I will use hyperparameters to refer to (optimizers, loss, lr-schedulers). This rule has become a prevalent practice as there is yet any formal study about this topic though there are papers emphasizing the need to use small learning rate in finetuning [4]. On the other hand, breaking free from hyperparameters consistency enable freedom to use better hyperparameters like label smoothing for regularization, Sharpness-Aware Minimization for optimizer [5], contrastive loss…

Counter-Rule 2 – It is not okay to finetune with different optimizers, lr-schedulers. To isolate this counter-rule from the catastrophic forgetting problem by using large lr in finetuning, I deliberately finetune with large lr (0.5) and use PyTorch new training recipe [6] to keep lr consistent, then use large lr for PyTorch old training recipe which has smaller lr (0.1). Table 2 on next page shows the adverse effect of optimizers and lr schedulers inconsistencies. With this new finding, I address the need to consider optimizer and lr-scheduler consistency in any future study that used pretrained models from third-party (i.e., PyTorch, HuggingFace…). The work in this direction is left for future works.

Rule 3 – Using fully-connected layer in small dataset results overfitting and difficulty in convergence. When the number of samples < number of parameters (which is 99% the case in finetuning), the risk of overfitting is very prevalent.

(Partially) Counter-rule 3 – Fully-connected layers always lead to better accuracy when used w.r.t regularizer like label smoothing on small epoch (10). In fact, throughout all this coursework which conducts 10 experiments as shown in table 1, 2, 3, overfitting only occurred twice. On the other hand, given that PyTorch old

pretraining recipe gives out worse weight initialization on weight-space than its new pretraining recipe. It is very noticeable about extra parameter from fully-connected layer slows finetuning convergence in table 1 when comparing between resnet50 and resnet50_fc512.

Rule 4 – Aim for at least 30 epochs. This is a common practice for finetuning resnet, mobilenet… and was the recommended setting by the teaching assistant for this coursework.

Counter-rule 4 – Larger epoch only makes small impact. This has been demonstrated in figure 3 and table 5 in [7]. In this report, the accuracy improvement for finetuning recipe 1 when going from 10 epochs → 15 epochs (before overfit occurs) is only 0.6% for resnet50 and 0.2% for resnet50 with fully-connected layer.

2. Experiment

2.1. Methodology

I follow the theme of incremental improve as done by [7] for table 5. First, I start with counter-rule 1 where I keep the hyperparameters setting between PyTorch new training recipe [6] and my 4 finetuning recipes the same, and only add-on 1 data augmentation for each recipe (see my table 1). Then for counter-rule 2, I keep the data augmentation and hyperparameters settings the same for 4 finetuning recipes but use the old pretrained resnet50-family instead (the reasoning to use old pretrained resnet is already discussed in section 1.1 for counter-rule 2). To experiment counter-rule 3, I employ resnet50_fc512 without dropout in the experiments for counter-rule 1 and 2 to exhaustively squeeze out the overfitting phenomenon as mentioned in the ML communities, overfitting is rare. PyTorch new pretraining recipe produces resnet50-ImageNet1k-v2 checkpoint and old recipe produces resnet50-ImageNet1k-v1. Both were used on resnet50 and resnet50-fc512. The hyperparameters for old and new recipe is shown in table 2 and table 1, respectively.

2.2. Experiments

Data augmentation	Pretraining 1	Finetune 1	Finetune 2	Finetune 3	Finetune 4
label_smoothing	0.1	0.1	0.1	0.1	0.1
color_augmentation		Yes			Yes
color_jitter			Yes		Yes
random_erase	0.1	0.1	0.1	0.1	0.1
train_crop_size	176	224	224	224	224
auto_augment	ta_wide'				
mixup_alpha	0.2				
cutmix_alpha	1				
interpolation	bilinear'	bilinear'	bilinear'	bilinear'	bilinear'
epoch	600	10	10	10	10
resnet50		88.4%	87.1%	85.3%	88.8%
resnet50_fc512		90.5%	89.8%	86.0%	89.3%

Table 1: Experiments for counter-rule 1 and 3 where the consistent data augmentation is random_erase and add-on data augmentation are color_jitter and color_augmentation. Here, the most consistent finetune recipe (3) has the least accuracy on both resnet. Most hyperparameters used for PyTorch new pretraining recipe is shown in this table.

Hyperparameters	Pretraining 2	Finetune 1	Finetune 2	Finetune 4
optimizer	sgd	sgd	sgd	sgd
lr	0.1	0.5	0.5	0.5
lr_scheduler	step	sequential	sequential	sequential
main_lr_scheduler		cosine anneal	cosine anneal	cosine anneal
warmup_lr_scheduler		linear	linear	linear
warmup_epochs	5	5	5	5
warmup_decay	0.01	0.01	0.01	0.01
weight_decay	1E-04	2E-05	2E-05	2E-05
step_size	30			
gamma	0.1			
epoch	600	10	10	10
resnet50		76.4%	74.5%	78.2%
resnet50_fc512		71.3%	75.7%	72.7%

Table 2: Experiments for counter-rule 2 and 3 where the data augmentation settings used in table 1 are kept the same. The only difference is the pretraining resnet50 to be used. Most hyperparameters used for PyTorch old pretraining recipe is shown in this table.

Data augmentation	Finetune 1	
resnet50	89 (epoch=14)	88.9 (epoch=15)
resnet50_fc512	90.7 (epoch=13)	89.1 (epoch=14)
	Most accuracy	Overfit drop

Table 3: Experiment for counter-rule 4 and 3 where I push the training epoch from 10 → 15. Only 1 finetune recipe is conducted because workload of this report putting on AISurrey-condor, Colab and my timetable has been stretched to the limit.

3. Conclusion

From the 10 experiments above, future finetuners can be more relaxed on data augmentations-consistency and using fully-connected layers, should be more concern on hyperparameters inconsistency (i.e., optimizers, lr and lr-scheduler inconsistencies) and less concern on pushing for higher epoch. Due to time constraint and study overlap, I refer readers to [7] where the authors exhaustively experiment fully-connected layers in different width and depth scaling for resnet and different regularization methods

References

[1] Papers with Code, "Image Retrieval," [Online]. Available: https://paperswithcode.com/task/image-retrieval.

[2] Zakria, J. Deng, M. S. Khokhar, M. U. Aftab, J. Cai, R. Kumar and J. Kumar, "Trends in Vehicle Re-identification Past, Present, and Future: A Comprehensive Review," 19 February 2021. [Online]. Available: https://arxiv.org/abs/2102.09744.

[3] J. Liang, D. Hu and J. Feng, "Do We Really Need to Access the Source Data? Source Hypothesis Transfer for Unsupervised Domain Adaptation," 20 February 2020. [Online]. Available: https://arxiv.org/abs/2002.08546.

[4] S.-A. Rebuffi, H. Bilen and A. Vedaldi, "Efficient Parametrization of Multi-domain Deep Neural Networks," 27 May 2018. [Online]. Available: https://arxiv.org/abs/1803.10082.

[5] P. Foret, A. Kleiner, H. Mobahi and B. Neyshabur, "Sharpness-Aware Minimization for Efficiently Improving Generalization," 29 April 2021. [Online]. Available: https://arxiv.org/abs/2010.01412.

[6] PyTorch, "How to Train State-Of-The-Art Models Using TorchVision's Latest Primitives," 18 November 2021. [Online]. Available: https://pytorch.org/blog/how-to-train-state-of-the-art-models-using-torchvision-latest-primitives/.

[7] I. Bello, W. Fedus, X. Du, E. D. Cubuk, A. Srinivas, T.-Y. Lin, J. Shlens and B. Zoph, "Revisiting ResNets: Improved Training and Scaling Strategies," 9 November 2021. [Online]. Available: https://openreview.net/forum?id=dsmxf7FKiaY.