COMS E6998 section 012
PRACT DEEP LEARNING SYS PERF
Instructor: Parijat Dube

Homework 4

Due: Nov 13, 2021

Problem 1 - *Transfer learning: Shallow learning vs Finetuning, Pytorch* 30 points

In this problem we will train a convolutional neural network for image classification using transfer learning. Transfer learning involves training a base network from scratch on a very large dataset (e.g., Imagenet1K with 1.2 M images and 1K categories) and then using this base network either as a feature extractor or as an initialization network for target task. Thus two major transfer learning scenarios are as follows:

- *Finetuning the base model*: Instead of random initialization, we initialize the network with a pretrained network, like the one that is trained on Imagenet dataset. Rest of the training looks as usual however the learning rate schedule for transfer learning may be different.
- *Base model as fixed feature extractor:* Here, we will freeze the weights for all of the network except that of the final fully connected layer. This last fully connected layer is replaced with a new one with random weights and only this layer is trained.

1. For fine-tuning you will select a target dataset from the Visual-Decathlon challenge. Their web site (link below) has several datasets which you can download. Select any one of the visual decathlon dataset and make it your target dataset for transfer learning. **Important : Do not select Imagenet1K as the target dataset.**
   (a) *Finetuning:* You will first load a pretrained model (Resnet50) and change the final fully connected layer output to the number of classes in the target dataset. Describe your target dataset features, number of classes and distribution of images per class (i.e., number of images per class). Show any 4 sample images (belonging to 2 different classes) from your target dataset. (2+2)
   (b) First finetune by setting the same value of hyperparameters (learning rate=0.001, momentum=0.9) for all the layers. Keep batch size of 64 and train for 200-300 epochs or until model converges well. You will use a multi-step learning rate schedule and decay by a factor of 0.1 ($\gamma = 0.1$ in the link below). You can choose steps at which you want to decay the learning rate but do 3 drops during the training. So the first drop will bring down the learning rate to 0.0001, second to 0.00001, third to 0.000001. For example, if training for 200 epochs, first drop can happen at epoch 60, second at epoch 120 and third at epoch 180. (8)
   (c) Next keeping all the hyperparameters same as before, change the learning rate to 0.01 and 0.1 uniformly for all the layers. This means keep all the layers at same learning rate. So you will be doing two experiments, one keeping learning rate of all layers at 0.01 and one with 0.1. Again finetune the model and report the final accuracy. How does the accuracy with the three learning rates compare ? Which learning rate gives you the best accuracy on the target dataset ? (6)
2. When using a pretrained model as feature extractor, all the layers of the network are frozen except the final layer. Thus except the last layer, none of the inner layers' gradients are updated during backward pass with the target dataset. Since gradients do not need to be computed for most of the network, this is faster than finetuning.
   (a) Now train only the last layer for 1, 0.1, 0.01, and 0.001 while keeping all the other hyperparameters and settings same as earlier for finetuning. Which learning rate gives you the best accuracy on the target dataset ? (8)
   (b) For your target dataset find the best final accuracy (across all the learning rates) from the two transfer learning approaches. Which approach and learning rate is the winner? Provide a plausible explanation to support your observation. (4)

Homework 4

For this problem the following resources will be helpful.

References

- Pytorch blog. Transfer Learning for Computer Vision Tutorial by S. Chilamkurthy
  Available at https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html
- Notes on Transfer Learning. CS231n Convolutional Neural Networks for Visual Recognition.
  Available at https://cs231n.github.io/transfer-learning/
- Visual Domain Decathlon

Problem 2 - *Weakly and Semi-Supervised Learning for Image Classification* 20 points

This problem is based on two papers, by Mahajan et al. on weakly supervised pretraining and by Yalinz et al. on semi-supervised learning for image classification. Both of these papers are from Facebook and used 1B images wiith hashtags. Read the two papers thoroughly and then answer the following questions. You can discuss these papers with your classmates if this helps in clarifying your doubts and improving your understanding. However no sharing of answers is permitted and all the questions should be answered individually in your own words.

1. Both the papers use the same 1B image dataset. However one does weakly supervised pretraining while the other does semi-supervised . What is the difference between weakly supervised and semi-supervised pretraining ? How do they use the same dataset to do two different types of pretraining ? Explain. (2)
2. These questions are based on the paper by Mahajan et al.
   (a) Are the model trained using hashtags robust against noise in the labels ? What experiments were done in the paper to study this and what was the finding ? Provide numbers from the paper to support your answer. (2)
   (b) Why is resampling of hashtag distribution important during pretraining for transfer learning ? (2)
3. These questions are based on the paper by Yalzin et al.
   (a) Why are there two models, a teacher and a student, and how does the student model leverages the teacher model ? Explain why teacher-student modeling is a type of *distillation* technique. (2+2)
   (b) What are the parameters K and P in stage 2 of the approach where unlabeled images are assigned classes using teacher network ? What was the idea behind taking $P > 1$? Explain in your own words. (2+2)
   (c) Explain how a new labeled dataset is created using unlabeled images ? Can an image in this new dataset belong to more than one class ? Explain. (2+2)
   (d) Refer to Figure 5 in the paper. Why does the accuracy of the student model first improves as we increase the value of K and then decreases ? (2)

References

- Yalniz et al. Billion-scale semi-supervised learning for image classification.
  Available at https://arxiv.org/pdf/1905.00546.pdf
- Mahajan et al. Exploring the Limits of Weakly Supervised Pretraining.
  Available at https://arxiv.org/pdf/1805.00932.pdf

Homework 4

Problem 3 - *PALEO, FLOPs, Platform Percent of Peak (PPP)* 20 points

This question is based on modeling the execution time of deep learning networks by calculating the floating point operations required at each layer. We looked at two papers in the class, one by Lu et al. and the other by Qi et al.

1. Why achieving peak FLOPs from hardware devices like GPUs is a difficult propostion in real systems ? How does PPP help in capturing this inefficiency captured in Paleo model. (4)

2. Lu et al. showed that FLOPs consumed by convolution layers in VG16 account for about 99% of the total FLOPS in the forward pass. We will do a similar analysis for VGG19. Calculate FLOPs for different layers in VGG19 and then calculate fraction of the total FLOPs attributed by convolution layers. (6)

3. Study the tables showing timing benchmarks from Alexnet (Table 2), VGG16 (Table 3), Googlenet (Table 5), and Resnet50 (Table 6). Why the measured time and sum of layerwise timings for forward pass did not match on GPUs ? What approach was adopted in Sec. 5 of the paper to mitigate the measurement overhead in GPUs. (2+2)

4. In Lu et al. FLOPs for different layers of a DNN are calculated. Use FLOPs numbers for VGG16 (Table 3), Googlenet (Table 5), and Resnet50 (Table 6), and calculate the inference time (time to have a forward pass with one image) using published Tflops number for K80 (Refer to NVIDIA TESLA GPU Accelerators). Use this to calculate the peak (theoretical) throughput achieved with K80 for these 3 models. (6)

References

- Qi et al. PALEO: A Performance model for Deep Neural Networks. ICLR 2017. Available at https://openreview.net/pdf?id=SyVVJ85lg
- Lu et al. Modeling the Resource Requirements of Convolutional Neural Networks on Mobile Devices. 2017 Available at https://arxiv.org/pdf/1709.09503.pdf

Problem 4 - *Optimus, Learning and Resource models, Performance-cost tradeoffs* 30 points

Peng et al. proposed Optimus scheduler for deep learning clusters which makes use of a predictive model to estimate the remaining time of a training job. Optimus assumes a parameter-server architecture for distributed training where synchronization between parameter server(s) and workers happen after every training step. The time taken to complete one training step on a worker includes the time for doing forward propagation (i.e., loss computation) and backward propagation (i.e., gradients computation) at the worker, the worker pushing gradients to parameter servers, parameter servers updating parameters, and the worker pulling updated parameters from parameter servers, plus extra communication overhead.

The predictive model proposed in Optimus is based on two sub-models, one to model the training loss as a function of number of steps and the other to model the training speed (training steps per unit time) as a function of resources (number of workers and parameter servers). The training loss model is given by Equation (1) in the paper. It has three parameters β_0, β_1, and β_2 that needs to be estimated from the data.

1. The first step is to generate data for predictive model calibration. You will train Resnet models with different number of layers (18, 20, 32, 44, 56) each with 3 different GPU types (K80, P100, V100). For

Homework 4

these runs you will use CIFAR10, a batch size of 128, and run each job for 350 epochs. You need to collect training logs containing data on training loss and step number for different configuration. **The data collection can be done in a group of up to 5 students. If working as a group each student should pick one of the 5 Resnet models and train it on all three GPU types. So each student in the group will be contributing training data from 3 experiments. If you decide to collaborate in the data collection please clearly mention the name of students involved in your submission.** For each of these 15 experiments, use all the training data and calibrate a training loss model. You will report 15 models one of each of the experimental configuration and their corresponding parameters $(\beta_0, \beta_1, \beta_2)$. (15)

2. We next study how the learned parameters, β_0, β_1, and β_2, change with the type of GPUs and the size of network. Use a regression model on the data from 15 models to predict the value of these parameters as a function of number of layers in Resnet and GPU type. From these regresssion model predict the training loss curve for Resnet-50. Note that we are effectively doing prediction for a predictive model. To verify how good is this prediction, you will train Resnet-50 on a K80, P100, and V100 for target accuracy of 92% and compare the predicted loss curve with the real measurements. Show this comparison in a graph and calculate the percentage error. From the predicted loss curve get the number of epochs needed to achive 92% accuracy. Observe that there are three curves for three different GPU types, but the number of epochs required to reach a particular accuracy (convergence rate) should be independent of hardware. (8)

3. Using the predicted number of epochs for Resnet-50 along with the resource-speed model (use Equation (4) in Peng et al. along with its coefficients from the paper) obtain the time to accuracy of Resnet-50 (to reach 92% accuracy) in two different setting (with 2 and 4 parameter servers respectively) as a function of the number of workers. So you will be plotting two curves, one for 2 and one for 4 parameter server case. Each curve will show how the time to achieve 92% accuracy (on the y-axis) scales with number of workers (on the x-axis). (7)

References

- Peng et al. Optimus: An Efficient Dynamic Resource Scheduler for Deep Learning Clusters Available at https://i.cs.hku.hk/ cwu/papers/yhpeng-eurosys18.pdf

Note

- In 5.2 other than the ResNet layers being different, every other hyperparameter should be the same during the data collection process across different students in a group (i.e.: Learning Rate, optimizer, preprocessing/normalization method. etc.). You should also use SGD optimizer since it's one of the key assumptions by Peng et al.
- When determining the βs, you can use scipy's **curve_fit** function for regression based on k (effective step number) and l (training loss).