CSCI-GA 3033-091
INTRO DEEP LEARNING SYS
Instructor: Parijat Dube
Due: May 4, 2022

Homework 5

Problem 1 - *SSD, ONNX model, Visualization, Inferencing* 25 points

In this problem we will be inferencing SSD ONNX model using ONNX Runtime Server. You will follow the github repo and ONNX tutorials (links provided below). You will start with a pretrained Pytorch SSD model and retrain it for your target categories. Then you will convert this Pytorch model to ONNX and deploy it on ONNX runtime server for inferencing.

1. Download pretrained pytorch MobilenetV1 SSD and test it locally using Pascal VOC 2007 dataset. Show the test accuracy for the 20 classes. (3)
2. Select any two related categories from Google Open Images dataset and finetune the pretrained SSD model. Examples include, Aircraft and Aeroplane, Handgun and Shotgun. You can use `open_images_downloader.py` script provided at the github to download the data. For finetuning you can use the same parameters as in the tutorial below. Compute the accuracy of the test data for these categories before and after finetuning. (4+4)
3. Convert the Pytorch model to ONNX format and save it. (2)
4. Visualize the model using net drawer tool. Compile the model using `embed_docstring` flag and show the visualization output. Also show doc string (stack trace for PyTorch) for different types of nodes. (4)
5. Deploy the ONNX model on ONNX runtime (ORT) server. You need to set up the environment following steps listed in the tutorial. Then you need make HTTP request to the ORT server. Test the inferencing set-up using 1 image from each of the two selected categories. (4)
6. Parse the response message from the ORT server and annotate the two images. Show inferencing output (bounding boxes with labels) for the two images. (4)

For part 1, 2, and 3, refer to the steps in the github repo. For part 4 refer to ONNX tutorial on visualizing and for 5 and 6 refer to ONNX tutorial on inferencing.
References

- Github repo. Single Shot MultiBox Detector Implementation in Pytorch.
  Available at https://github.com/qfgaohao/pytorch-ssd
- ONNX tutorial. Visualizing an ONNX Model.
  Available at https://github.com/onnx/tutorials/blob/master/tutorials/VisualizingAModel.md
- ONNX tutorial. Inferencing SSD ONNX model using ONNX Runtime Server.
  Available at https://github.com/onnx/tutorials/blob/master/tutorials/OnnxRuntimeServerSSDModel.ipynb
- Google. Open Images Dataset V5 + Extensions.
  Available at https://storage.googleapis.com/openimages/web/index.html
- The PASCAL Visual Object Classes Challenge 2007.
  Available at http://host.robots.ox.ac.uk/pascal/VOC/voc2007/

Problem 2 - *ML Cloud Platforms* 15 points

In this question you will analyze different ML cloud platforms and compare their service offerings. In particular, you will consider ML cloud offerings from IBM, Google, Microsoft, and Amazon and compare them on the basis of following criteria:

1. Frameworks: DL framework(s) supported and their version. (2)
   Here we are referring to machine learning platforms which have their own inbuilt images for different frameworks.
2. Compute units: type(s) of compute units offered, i.e., GPU types. (2)
3. Model lifecycle management: tools supported to manage ML model lifecycle. (2)
4. Monitoring: availability of application logs and resource (GPU, CPU, memory) usage monitoring data to the user. (2)
5. Visualization during training: performance metrics like accuracy and throughput (2)
6. Elastic Scaling: support for elastic scaling compute resources of an ongoing job. (2)
7. Training job description: training job description file format. Show how the same training job is specified in different ML platforms as YAML file. Identify similar fields in the training job file for any two ML platforms through an example. (3)

Problem 3 - *Kubeflow, MiniKF, Kale* 20 points

In this problem we will follow Kubeflow-Kale codelab (link below). You will follow the steps as outlined in the codelab to install Kubeflow with MiniKF, convert a Jupyter Notebook to Kubeflow Pipelines, and run Kubeflow Pipelines from inside a Notebook. **For each step below you need to show the commands executed, terminal output, and screenshot of visual output (if any). You also need to give a new name to your GCP project and any resource instance you create, e.g., put your initial in the name string.**

1. *Setting up the environment and installing MiniKF*: Follow the steps in the codelab to:
   (a) Set up a GCP project. (2)
   (b) Install MiniKF and deploy your MinKF instance. (2)
   (c) Login to MiniKF, Kubeflow, and Rok. (2)
2. *Run a Pipeline from inside your Notebook*: Follow the steps in the codelab to:
   (a) Create a notebook server. (2)
   (b) Download and run the notebook: We will be using **pytorch-classification** notbeook from the example repo. *Note that the codelab uses a different example from the repo (titanic_dataset_ml.ipynb).* (3)
   (c) Convert your notebook to a Kubeflow Pipeline: Enable Kale and then compile and run the pipeline from Kale Deployment Panel. Show output from each of the 5 steps of the pipeline (5)
   (d) Show snapshots of ”Graph” and ”Run output” of the experiment. (3)
   (e) *Cleanup:* Destroy the MiniKF VM. (1)

References

- From Notebook to Kubeflow Pipelines to KFServing: the Data Science Odyssey
  Available at https://codelabs.arrikto.com/codelabs/minikf-kale-katib-kfserving/0
- https://github.com/kubeflow-kale/examples

Homework 5

Problem 4 - *Deep Reinforcement Learning* 20 points

This question is based on Deep RL concepts discussed in Lecture 8. You need to refer to the papers by Mnih et al., Nair et al., and Horgan et al. to answer this question. All papers are linked below.

1. Explain the difference between episodic and continuous tasks? Given an example of each. (2)
2. What do the terms exploration and exploitation mean in RL ? Why do the actors employ ϵ-greedy policy for selecting actions at each step? Should ϵ remain fixed or follow a schedule during Deep RL training ? How does the value of ϵ help balance exploration and exploitation during training. (1+1+1+1)
3. How is the Deep Q-Learning algorithm different from Q-learning ? You will follow the steps of Deep Q-Learning algorithm in Mnih et al. (2013) page 5, and explain each step in your own words. (2)
4. What is the benefit of having a target Q-network ? (2)
5. How does experience replay help in efficient Q-learning ? (2)
6. What is prioritized experience replay and how is priority of a sample calculated? (3)
7. Compare and contrast GORILA (General Reinforcement Learning Architecture) and Ape-X architecture. Provide three similarities and three differences. (3)
8. Why the performance improves with number of actors in Ape-X? (2)

References

- Mnih et al. Playing Atari with Deep Reinforcement Learning. 2013
  Available at https://arxiv.org/pdf/1312.5602.pdf
- Nair et al. Massively Parallel Methods for Deep Reinforcement Learning. 2015
  Available at https://arxiv.org/pdf/1507.04296.pdf
- Horgan et al. Distributed Prioritized Experience Replay. 2018
  Available at https://arxiv.org/pdf/1803.00933.pdf

Problem 5 - TTA metric, Stability, Generalization 20 points

This question is based on efforts around DL benchmarking and standardization of performance metrics. We will study TTA metric properties using CIFAR10 dataset. Observe that, MLPerf and DAWNBench both use Imagenet1K for benchmarking but we will use CIFAR10, purely for convenience. In particular, we want to study (i) TTA stability, and (ii) Generalization performance of models optimized for TTA. A similar study was done in Coleman et al. using MLPerf and DAWNBench models. You should read this paper (especially Section 4.1 and 4.2) to have a better understanding of what you will be doing in this question. To study stability we will compute the coefficient of variation of TTA on different hardware by running several times the same training job. You will train Resnet-50 model in Pytorch with 2 different hardware types (V100, TPU pod) using CIFAR10. You will use a batch size of 128 and train the model till you get 92% validation accuracy.. From the training logs get the total (wall-clock) time to reach 92% validation accuracy for each of the runs. **We need data from at least 5 different training runs for the same configuration. This data collection can be done in a group of 5 students. Each of the student (in a group of 5) can run the same two training jobs, one with V100 and one with a TPU pod. If you decide to collaborate in the data collection please clearly mention the name of students involved in your submission.**

CSCI-GA 3033-091
INTRO DEEP LEARNING SYS
Instructor: Parijat Dube
Due: May 4, 2022

Homework 5

1. Calculate the coefficient of variation of TTA for both the hardware configurations. Compare the value you obtain with that reported in Table 3(a) in the paper by Coleman et al for Resnet-50, 1xTPU. (4)

2. Collect 5 images from the wild for each of the 10 categories in CIFAR10 and manually label them. **Again in a group of 5 students, each one of you can collect 5 images for 2 categories. In this way each one of you need to collect only 10 images (5 for each of the 2 categories that you choose)**. These images should not be from CIFAR10 dataset. Next test the trained models for these 50 images. What is the accuracy obtained from each of the 10 trained models ? Quantify the mean and standard deviation of accuracy obtained using the 10 models for TPU and the 10 models for V100. (5+5)

3. Measure the GPU utilization using nvidia-smi while training with V100 and report both the time series of GPU core and memory utilization and their average over a period of 3 mins of training. Is the GPU utilization close to 100% ? (3)

4. If the GPU utilization is low, what can you do to increase the GPU utilization? Try your trick(s) and report if you are successful or not in driving GPU utilization close to 100%. (3)

References

- Coleman et al. Analysis of DAWNBench, a Time-to-Accuracy Machine Learning Performance Benchmark. Available at https://cs.stanford.edu/ matei/papers/2019/sigops_osr_dawnbench_analysis.pdf