Final Project

Cho-Jui Hsieh

Important dates:

- Project proposal due: Feb 11 (Mon), 11:59pm.
- Proposal format: 1 page; describe what you want to do and give some references.
- Project discussion with each group: TBD (probably Feb 13)
- Final presentation: last week of class.
- Final report due: March 20, 11:59pm.

1 Improving Model size / Prediction speed

Model size is important, especially when we want to put models on mobile devices. Also, prediction speed of a ML model is important for real-time systems. One type of recommended project is to pick some machine learning model and try to improve the model size or prediction speed. For example:

- In general you can try to take any ML model and try to compress it using all the different approaches (including quantization, pruning, low-rank approximation, or any other ideas you come up with). You can also combine those approaches to see what's the best result you can get. Potential models include object detection models, video classification/recognization models, text classification models, neural machine translation models, contextual representations (BERT, Elmo), ...
- Speed up maximum inner product search (prediction speed): In many ML algorithms, including neural network with large output space and recommender systems, the bottleneck is on Maximum Inner Product search (MIPS): given a data set with vectors $\{v_1, \ldots, v_n\}$ and a query vector u, how to find the v_i with maximal inner product with u ($\arg\max_i v_i^T u$). Several algorithms including graph-based (small world) approach [1], greedy approach [2], and other clustering/sampling-based approaches [3]. You can compare those algorithms. You can also go beyond to see whether you can predict which algorithm should you use, given a query vector.

- Quantization: Study what's the best way to quantize a general neural network. Current algorithms include [4, 5, 6, 7]. Try to implement these and compare, or come up with some other ideas.

- Pruning: how to get a compressed network by setting many weights to 0 [8]. Naive way is to threshold but you could try different ideas.

2 Robustness of machine learning models

It has been shown recently that small perturbation of data can easily make machine learning models misclassify [9]. There are two types of attacks: test time attack and data poisoning (training time) attack. In test time attack, we assume the model is fixed and given a natural example x, an attacker tries to generate an "adversarial example" $\bar{x} = x + \Delta$ with a small Δ, such that the machine learning model will misclassify $\bar{x}$. In data poisoning attack, the attacker tries to perturb the training data to make the trained model having some desired properties (e.g., low accuracy, or injecting a backdoor). There are several interesting problems in this area:

- Attack: How to generate adversarial examples for a given model? In the white-box setting (the attacker knows the model structure & weights) and for standard image classification this is considered a solved problem [10], but how to do this in a different task (e.g., object detection, video classification) is still interesting. Also, it is interesting to study how to generate adversarial examples in the probability black box setting (when you can observe the probability of model's output), see some recent work [11, 12]. A more challenging problem is the black-box hard label setting where you can only query the model and get the hard label output, see recent work [13, 14]. Black-box attacks are still open problem since they typically require large number of queries. A comparison between current methods or improvement will be a good project.

- Defense: Defense (improving robustness with respect to adversarial attacks) is still an open problem. Two viable approaches are adversarial training [15] and randomization [16]. A comparison between current defense algorithms or improvement will be a good project.

- Verification: The verification problem of neural network can be cast this way: given a neural network and a data point x, determine the largest region r such that $f(x+\Delta)$ always give the same prediction when $\|\Delta\| \leq r$. Several recent work [17, 18].

3 Training Algorithms

Training algorithms for standard ML problems or neural networks have been deeply studied. You could pick some interesting harder models and try to im-

prove the training performance, or discuss how to train a model in some special cases. For instance:

- Important sampling: Current SGD algorithm selects each batch with uniform sampling. Intuitively this is not the best way to sample—there should be some "easy" samples that already well-trained and doesn't need more updates; while some "hard" samples require more frequent updates. The question is how to estimate the "importance" of each sample, and how to do it efficiently? Several references [19, 20]. You can try to implement these methods and investigate their weakness, and try to come up with better approaches.
- Hard negative sampling for multi-label problems: The same "important sampling" idea is even more useful for some large-scale multi-label learning problems, when you have a lot of labels. Examples include language model, translation, extreme classification, and some recommender systems. See some recent work [21] and references therein. You can try to implement these methods and investigate different sampling approaches.
- Large-batch training: Large-batch training is the most important problem when we want to scale up ML training. When you have many GPUs, you need to scale up the batch size, but people found SGD will converge to bad solutions with large batch size [22]. Some recent work on large-batch training [23, 24]. You could try to compare different large batch training approaches or test on some different tasks. (Will probably need several GPUs).
- GAN training: GAN training is an unsolved problem. You could try different algorithms and discuss your findings.

4 Application: Recommdender systems

- Traditional approach (only given partial user-item ratings): Matrix factorization or Bayesian Personalized Ranking (BPR).
- Applying deep learning for recommender systems?
  - Sequential recommendation: LSTM/Attention models
  - Other typse of data?

5 Application: Graph-based algorithms

- Graph embedding learning: given a graph, what's the best embedding to represent each node? There are several different approahces but not clear which one is better. You can do a comparative study (comparing different methods on different graphs), or try to implement some papers and come up with some new idea.

- Graph convolutional network [25, 26, 27] is a new techinque developed recently. However it is not clear what's the best way to make it scalable, and how to train deep GCNs.

6 Application: NLP

7 Application: Computer vision

8 Other applications

References

[1] Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. Approximate nearest neighbor algorithm based on navigable small world graphs. *Information Systems*, 45:61–68, 2014.

[2] Hsiang-Fu Yu, Cho-Jui Hsieh, Qi Lei, and Inderjit S Dhillon. A greedy approach for budgeted maximum inner product search. In *Advances in Neural Information Processing Systems*, pages 5453–5462, 2017.

[3] Yoram Bachrach, Yehuda Finkelstein, Ran Gilad-Bachrach, Liran Katzir, Noam Koenigstein, Nir Nice, and Ulrich Paquet. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 257–264. ACM, 2014.

[4] Soroosh Khoram and Jing Li. Adaptive quantization of neural networks. In *ICLR*, 2018.

[5] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *Journal of Machine Learning Research*, 18:187–1.

[6] Christos Louizos, Matthias Reisser, Tijmen Blankevoort, Efstratios Gavves, and Max Welling. Relaxed quantization for discretized neural networks. In *ICLR*, 2019.

[7] Yu Bai, Yu-Xiang Wang, and Edo Liberty. Proxquant: Quantized neural networks via proximal operators. In *ICLR*, 2019.

[8] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[9] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[10] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

[11] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 15–26. ACM, 2017.

[12] Andrew Ilyas, Logan Engstrom, Anish Athalye, Jessy Lin, Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Black-box adversarial attacks with limited queries and information. In *Proceedings of the 35th International Conference on Machine Learning,{ICML} 2018*, 2018.

[13] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *ICLR*, 2018.

[14] Minhao Cheng, Thong Le, Pin-Yu Chen, Jinfeng Yi, Huan Zhang, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. In *ICLR*, 2019.

[15] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[16] Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. Towards robust neural networks via random self-ensemble. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 369–385, 2018.

[17] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Duane Boning, Inderjit S Dhillon, and Luca Daniel. Towards fast computation of certified robustness for relu networks. In *ICML*, 2018.

[18] Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy Mann, and Pushmeet Kohli. A dual approach to scalable verification of deep networks. *arXiv preprint arXiv:1803.06567*, 2018.

[19] Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *ICML*, 2018.

[20] Tyler B. Johnson and Carlos Guestrin. Training deep models faster with robust, approximate importance sampling. In *Advances in Neural Information Processing Systems 31*, 2018.

[21] Guy Blanc and Steffen Rendle. Adaptive sampled softmax with kernel based sampling. In *ICML*, 2018.

[22] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *ICLR*, 2017.

[23] Yang You, Zhao Zhang, Cho-Jui Hsieh, James Demmel, and Kurt Keutzer. Imagenet training in minutes. In *Proceedings of the 47th International Conference on Parallel Processing*, page 1. ACM, 2018.

[24] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[25] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[26] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[27] Jianfei Chen, Jun Zhu, and Le Song. Stochastic training of graph convolutional networks with variance reduction. In *ICML*, 2018.