Centre for AI-Fundamentals

RAEng Google DeepMind Summer Internship Programme 2025

Project proposal

Project Title	Fixed Fisher for Training DNN
Lead supervisor	Satya Prakash Dash
Project Description	Stochastic Gradient Descent (SGD) and adaptive optimizers like Adam[1] are widely used in training DNN. This is notably true in the deep learning setting, where gradients can be computed efficiently via backpropagation. Although these methods can provably converge to local minima, the block hessian heterogeneity is imminent. But current widely used optimizer like Adam does not take this factor into consideration while computing the weight update. Alleviating such correlation results in faster convergence and stable minima (wrt output distribution of DNN). Natural gradients, which are invariant under reparameterization and compute the steepest descent direction in parameter space, are a well-studied class of second-order optimizers which uses Fisher Information Matrix (FIM)[2] to compute exact step length for each gradient direction in parameter space. Under large batch sizes (when the batch size is representation of the distribution of dataset), natural gradients could help alleviate Hessian block heterogeneity as the exact step lengths are based on the input minibatch. Kronecker decomposition of natural gradient update[3] has simplified the application and computation of inverse Fisher for DNN. But during training, the Kronecker products can be of low-rank and to efficiently invert these matrices, we regularize the update using Tikhonov regularization. And we express the inverse Fisher as the Kronecker product of inverse of regularized version of activation and error matrices. This update is taken to oversimplify the objective (as we want to go away with calculating the full Kronecker product) but theoretically speaking, this update has two more terms which are not present in Fisher inverse. To this end, this project aims to remove unwanted updates from the Fisher inverse and provide a new solution to approximately obtain the actual Fisher inverse. Project Objectives:

The general progression of the project will start with the intern getting accustomed to nanoGPT[4] repository. Add more transformer architectures like linear trasformer[5] to that repository and then creating scripts to apply KFAC and Fixed Fisher (our method) to the code base. Then we will start with ablation experiments on smaller models and then scale up.

Here is a tentative 7-Week Plan:

1. Week 1 & 2: Literature survey and updating nanoGPT repository with a simple Linear Transformer. architecture.
2. Week 3 & 4: Adding KFAC, EVA & Fixed Fisher algorithm to the repository.
3. Week 5 & 6: Ablation experiments and scaling up to large models.
4. Week 7: Preparing a coherent report which includes the main objectives achieved and results.

Minimum Requirements:

1. Experience in training DNN.
2. Proficiency in Python and ML framework like PyTorch.
3. Experience in CUDA (is a plus).

Learning outcomes / Skill Competences:
With successful completion of the project, the interns would have gained:

1. Keen understanding of how optimization plays a role in pre-training LLMs.
2. Proficiency in using torch.nn package in a distributed environment and extract meaningful features from LLMs during training.
3. Experience in using CUDA kernels to optimize the training process.

References:
[1] Kingma 2015, Adam: A Method for Stochastic Optimization [link].
[2] Amari 1998, Natural Gradient Works Efficiently in Learning [link].
[3] Martens 2015, Optimizing Neural Networks with Kronecker-factored Approximate Curvature [link].
[4] Karpathy 2022, nanoGPT: The simplest, fastest repository for training/finetuning medium-sized GPTs [link].

	[5] Schlag 2021, Linear Transformers Are Secretly Fast Weight Programmers [link].