MATH 270 PROJECTS

TONY FENG

1. Guidelines

The purpose of the Project is to do a deep-dive on a particular topic of the course. To do this, you will need to learn the necessary background ahead of time; in addition to the course notes and resources, supplementary references are suggested here and in Schedule.pdf. Then you will closely read the 3 or so seminal papers on the topic that I suggest below. Based on this knowledge, you will then prepare:

- A 30-minute in-class slide presentation (allowing an additional 15 minutes for questions and discussion) on the papers.
- A Colab notebook walking through a toy implementation of the architecture/algorithm in question.

1.1. Presentations.

Each in-class presentation will accompany a lecture that I will give on the topic. My treatment will be theoretical, and the in-class presentation will be used as a concrete illustration of the theory. You will be able to see from the lecture notes roughly I intend to say, except that I will omit the "case study" subsections of the notes. **The target audience for your presentation is your classmates**.

Your presentation should explain enlightening implementation details and the key new insights of the papers, with historical context. Use your judgment in deciding what details to present. Your presentation will be graded on its pedagogical value, not on the amount of material you cover; *presenting too much detail will therefore affect your grade negatively*, but you should be able to answer reasonable questions that may go beyond your presentation material. Note that you will need to understand the papers well enough to lead a 15 minutes of Q&A on them.

Projects will be done in teams of 3-5 students. Team members are expected to play a roughly equal role. **Project sign-ups are** here.

1.2. Tips for presentations.

In slide presentations, speakers typically go too fast and put too much information on slides. Once again: the grading will not be based on how much material you present; in fact, trying to present too much material will certainly tank your grade.

It is almost always helpful to include pictures and visual aids. It is almost always *unhelpful* to include non-trivially complicated mathematical equations. You can also use the chalkboard as necessary to supplement your slides; leave mathematical derivations and manipulations to the chalkboard.

The articles at distill.pub and colah's blog are good models for exposition, although they are neither slides nor Colab notebooks. I recommend spending some time on these websites.

Do practice presentations to people who have not seen your slides before. Auditing students may volunteer themselves to give feedback on practice presentations; check the sign-up sheet here.

1.3. **Colab notebooks.** Your Colab notebook will walk the reader through implementing the relevant architecture or algorithm in PyTorch. These are meant to be read by class participants as a practical grounding for the theoretical lecture material.

The goal is to see a toy model implemented in practice, not to achieve a state of the art performance. For this purpose, it is enough to use a small dataset and a small model; *you will not be graded on how good your trained model actually is, but rather on the pedagogical value of your Colab.*

A model example of an instructional Colab notebook is here. You can aim to emulate its format, except that for this class your Colab notebooks should be written in **PyTorch** instead of TensorFlow. In particular, **your colab should include explanatory text cells** (not just code).

You may use AI tools to help you write your Colab, but you may not simply copy-paste AI generated code. You should think critically about what you write, and consider that *it will be posted publicly*, so do not submit something that you would be embarrassed to subject to public scrutiny.

Datasets can be found on Kaggle and TensorFlow Datsets, for example.

2. List of case studies

2.1. **Week 5: Convolutional neural networks (Oct 6).** This Case Study will cover Convolutional Neural Networks, an architecture designed for image recognition.

2.1.1. *Background preparation.* First, learn about the ImageNet Database and ImageNet Challenge. Then read the section of the course notes on CNNs for background. You may also find [Bishop, §10] to be helpful for background.

2.1.2. *In-class presentation.* Read the following 3 papers and present their key insights in class.

(1) Krizhevsky et. al, *Imagenet classification with deep convolutional neural networks*, Communications of the ACM 60 (2012), 84 – 90.
(2) Simonyan and Zisserman, *Very deep convolutional networks for large-scale image recognition*, CoRR abs/1409.1556 (2014).
(3) He et. al, *Deep residual learning for image recognition*, 2016 IEEE Conference on Computer Vision and Pattern Recognition (2015), 770–778.

Illustrate the architectures used and make sure to include enlightening visuals in your presentation.

Finally, understand and explain how filters can sometimes be interpreted intuitively. For this, you can get started with §10.3 of [Bishop] (e.g., Figure 10.12 there).

2.1.3. *Colab.* Download the MNIST dataset (e.g. from here) and train a miniature version of a VGG style CNN on it to do classification. Try adding some residual skip connections. Plot and discuss the results.

2.2. **Week 6: Recurrent Neural Networks (Oct 13).** This case study will cover Recurrent Neural Networks, an architecture designed for Natural Language Processing.

2.2.1. *Background preparation.* Read the section of the course notes on RNNs, then read this page and Andrej Karpathy's blog post.

2.2.2. *Presentation.* Read the following 3 papers and present their key insights in class.

(1) Cho, et. al, *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation.* (2014).
(2) Sutskever et. al, *Sequence to Sequence Learning with Neural Networks.* (2014).
(3) Bahdanau et. al, *Neural Machine Translation by Jointly Learning to Align and Translate.* ICLR (2015)

Illustrate the architectures used and make sure to include enlightening visuals in your presentation.

Finally, understand and explain some examples which are interpretable. You can get started with the section "Visualizing the predictions and the "neuron" firings in the RNN" in Andrej Karpathy's blog post.

2.2.3. *Colab.* Download a weather dataset, e.g., historical San Francisco Weather Data. Train a (shallow) RNN to predict tomorrow's temperature high given the previous 14 days' temperature highs. Plot and discuss the results.

2.3. **Week 7: Transformers (Oct 20).** This case study will cover Transformers, an architecture originally designed for Natural Language Processing, but flexible enough for multimodal capabilities.

2.3.1. *Background preparation.* Read the section of the course notes on Attention and Transformers. Read through the Harvard NLP group's reproduction of the original Transformer paper. You may find this useful, both for understanding Transformers and for creating your Colab. Your Colab does not need to be as comprehensive as this, but it should also not just copy their work.

2.3.2. *Presentation.* Read the following 3 papers and present their key insights in class.

(1) Devlin et. al, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics.
(2) Dosovitskiy et. al, *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.* International Conference on Learning Representations.
(3) Dai et. al, *Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context.* ACL 2019.

Illustrate the architectures used and make sure to include enlightening visuals in your presentation.

2.3.3. *Colab.* Download an NLP dataset (e.g., tinyshakespeare), and use it to train a (decoder-only) Transformer to generate text. Display and analyze its performance.

2.4. **Week 8: Large Language Models (Oct 27).** This case study will cover the training of Large Language Models, notably OpenAI's early GPT models. Unlike other projects, **this one only has a presentation component**.

2.4.1. *Presentations.* Read the following papers by OpenAI on GPT-1, GPT-2, and GPT-3.

(1) Radford et. al, *Improving Language Understanding by Generative Pre-Training.* OpenAI technical report (2018).
(2) Radford et. al, *Language Models are Unsupervised Multitask Learners.* OpenAI technical report (2019).
(3) Brown et. al, *Language Models are Few-Shot Learners.* Advances in Neural Information Processing Systems (2020).

Illustrate the architectures used, giving a sense of details (e.g., size of hyperparameters) when enlightening. Also explain the training methodology and insights. Familiarize yourself with some of the relevant benchmarks, then discuss the experimental setup and results.

Next, talk about scaling laws. Read the two papers:

(1) Kaplan et. al, *Scaling Laws for Neural Language Models* (2020).
(2) Hoffmann et. al, *Training Compute-Optimal Large Language Models* (2022).

I will explain some of the qualitative conclusions of these papers in my lectures. You should go back over this, detailing exactly what experiments were conducted and how the conclusions were drawn. Note that there is an inconsistency between the quantitative conclusions of these two papers about scaling laws. The current consensus is that Hoffman et. al is correct. If you read that paper carefully then you will see their hypothesis for the discrepancy; this issue is revisited in the papers

- Pearce and Song, *Reconciling Kaplan and Chinchilla Scaling Laws.*
- Porian et. al, *Resolving Discrepancies in Compute-Optimal Scaling of Language Models.*

Officially, I am not asking you to read these last two papers, but it might be helpful to skim them to understand their conclusions.

Finally, give a quick overview of the trend of developments in more modern LLMs by going through these papers:

(1) Touvron et. al, *LLaMA: Open and Efficient Foundation Language Models.*
(2) Touvron et. al, *Llama 2: Open Foundation and Fine-Tuned Chat Models.*
(3) Llama team, *The Llama 3 Herd of Models.*
(4) OpenAI, gpt-oss-120b.

You should include lots of visuals throughout your presentation.

2.4.2. *Colab.* This project does not have a Colab component.

2.5. **Week 9: Generative Adversarial Networks and Variational Autoencoders (Nov 3).** This case study will cover two early generative models, GANs and Variational Autoencoders, which are used especially for image generation.

2.5.1. *Background preparation.* Read the relevant sections (§9 and §10) of the course notes for background. You may also find [Bishop, §17 and §19] helpful.

2.5.2. *Presentations.* Read and present the following papers.

(1) Radford et. al, *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.* ICLR 2015.
(2) Zhu et. al, *Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks.* 2017 IEEE International Conference on Computer Vision (ICCV).
(3) Higgins et. al, *β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework.* ICLR 2016.

Illustrate the architectures used and make sure to include enlightening visuals in your presentation.

2.5.3. *Colab.* Download the Generative Dog Images dataset. Train both a GAN and a VAE to generate more images from this distribution. Realistically, you probably will not have the compute needed to make a decent model, so focus on getting the ideas right without worrying too much about performance.

2.6. **Week 10: Diffusion Models (Nov 10).** This case study will cover Diffusion Models, the current dominant model in image generation (for example, it is the model behind Stable Diffusion and Dall-E 2). It can be viewed as a successor to VAEs.

2.6.1. *Background preparation.* Read the section of the course notes on VAEs and Diffusion Models (§10 and §11). You may also find [Bishop, §19 and §20] to be helpful.

2.6.2. *Presentations.* Read and present the following papers.

(1) Sohl-Dickstein et. al, *Deep Unsupervised Learning using Nonequilibrium Thermodynamics.* ICML (2015).
(2) Ho et. al, *Denoising Diffusion Probabilistic Models.* Advances in Neural Information Processing Systems (2020).
(3) Rombach et. al, *High-Resolution Image Synthesis with Latent Diffusion Models.* Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2022).

Illustrate the architectures used and make sure to include enlightening visuals in your presentation.

2.6.3. *Colab.* Download the Generative Dog Images dataset. Train a diffusion model to generate images from this distribution. Realistically, you probably will not have the compute needed to make a decent model, so focus on getting the ideas right without worrying too much about performance.

2.7. **Week 12: Reinforcement Learning I: value function methods (Nov 24).** This case study will cover Reinforcement Learning as applied to game-playing, including (perhaps most famously) AlphaGo.

2.7.1. *Background preparation.* Read §12 of the course notes and §3–5 of Sutton–Barto.

2.7.2. *Presentations.* Read and present the following papers.

(1) Mnih et. al, *Playing Atari with Deep Reinforcement Learning* (2013). See also accompanying paper in Nature, *Human-level control through deep reinforcement learning.*
(2) Silver et. al, *Mastering the Game of Go with Deep Neural Networks and Tree Search.* Nature 529 (2016), 484-489.
(3) Silver et. al, *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm.* See also accompanying paper in Science, *A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play.*

Illustrate the architectures used, and explain the training methodology and insights. Make sure to include enlightening visuals in your presentation.

2.7.3. *Colab.* Implement a Monte Carlo value optimization algorithm for Blackjack. You do not need to use a neural network for this, although you can if you want.

2.8. **Week 13: Reinforcement learning II: policy optimization (Dec 1).** This case study will cover Reinforcement Learning as applied to develop reasoning capabilities for math and coding in Large Language Models.

2.8.1. *Background.* Read §13 of the course notes and Schulman et. al, *Proximal Policy Optimization Algorithms* (2017).

2.8.2. *Presentations.* Watch this talk by Denny Zhou on reasoning in Large Language Models, and read the slides.

Read and present this, along with the following papers.

(1) Li et. al, *Competition-Level Code Generation with AlphaCode* (ArXiv 2203.07814), and the accompanying paper in Science (2022), *Competition-level code generation with AlphaCode.*
(2) Trinh et. al, *Solving olympiad geometry without human demonstrations* Nature (2024). There is a followup paper on ArXiv by Chervonyi et. al, *Gold-medalist Performance in Solving Olympiad Geometry with AlphaGeometry2*.
(3) DeepSeek-AI, *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning.*

2.8.3. *Colab.* Implement PPO algorithm for Blackjack.