Society for Artificial Intelligence and Deep Learning

www.saidl.in

Spring 2024 Induction Assignment

Introduction

This is the official assignment for SAiDL inductions. Please join our Slack Workspace here and fill this form in order to register.

Deadline is **Sunday 10/03/2024 11:59 PM IST**.

A Note from Us

Through this assignment we aim to introduce you to key ideas in Machine Learning, Deep Learning and Artificial Intelligence. We value your time and have designed each question such that the process of solving it should add to your skill set while also giving us an idea of your abilities. We have also provided a collection of resources to help you learn the required tools and concepts along the way.

Please go through the instructions before attempting the questions. The questions might seem intimidating at first glance. Don't panic! They are meant to be challenging and will take some time and effort to complete. Try looking around to understand the concepts involved in the questions and try digging into these topics. It is expected that around 10% - 40% of time (depending on your prior familiarity) on each question will be spent on learning new concepts. The submissions will be judged primarily on the basis of the approach and the thought that went into attempting the answer, so be sure to document everything. Also make sure to submit your attempt irrespective of whether or not you have completed the assignment as a whole.

If you have any doubts, try to search for solutions on the internet first before asking doubts on the slack. We will also be assigning everyone mentors so keep them updated with your progress.

All the best!

Logistics

Submission Guidelines

The submission deadline is **Sunday 10/03/2024 11:59 PM IST**
Submission is to be made via: *TBA*

The submission should be primarily in the form of a Git repo titled *SAiDL-Spring-Assignment-2024*. The repo should be **kept private up until the submission deadline**. Make sure you make the repo public while submitting the assignment.

You are required to document your approach and results for each question you attempt and analysis wherever required, in detail. The documentation should be included in the github repository along with the code and can be made in Markdown, LaTeX etc… Make sure to make your final report for each task as detailed and clear as possible.

Any form of plagiarisation or collaboration with others is strictly prohibited and will be penalized.

You'll be allotted a mentor for the assignment and you are required to inform them whenever you complete a question (submit the google form only after you have completed the assignment or after the deadline). All of this will be done through our Slack Workspace so make sure you join it for receiving all the assignments and for asking queries that you might have. You can also contact those mentioned below for queries.

SAiDL holds the rights to the questions asked in the assignment, if you choose to use this outside the assignment, you need to cite/acknowledge SAiDL.

Contact Details

In case of any doubts, clarifications, or guidance, you can contact one of us. We request that you stick to slack as the medium of communication for all the questions that you have. However, you can reach out to us through other means in case we fail to respond on slack.

- Aditya Agarwal (CV / RL) - 8879459970
- Karan Bania (GNN / Pruning) - 7069863756
- Rishav Mukherji (CV / Pruning) - 9920741703
- Yash Bhisikar (RL/ Pruning) - 9096794571
- Ankita Vaishnobi Bisoi (CV) - 7788920434
- Tejas Agrawal (NLP) - 8770544585
- Shreyas V (NLP / GNN) - 8660728506
- Harshvardhan Mestha (GNN) - 9892572823

Resources

The following is a list of resources to tools and topics you will need to be familiar with in order to solve the assignment questions. This section is more a selection of resources for things you will need along the way. It is not mandatory to complete, you should treat it more as a roadmap of things to learn and as a reference to get back to.

Python

Python is the Lingua Franca of AI (at the moment). Check out this guide by Google to start or the tutorial series by Sentdex. These books (1, 2) and the official documentation are good references.

Coursera Course on Machine Learning

This course is the launchpad for most AI enthusiasts and gives you your first hands-on approach to Machine Learning along with the math behind it. The course requires implementation of ML algorithms in Octave/MATLAB. However, it is recommended that you attempt the Python versions of these exercises which can be found here.

Linux Terminal

This is one of the fundamental requirements for any computer scientist. The following will help you get started (1, 2).

Numpy

Crudely speaking this is MATLAB for Python. We suggest you do this after the Andrew NG course. You can go to its official tutorial or learn it with hands-on deep learning experience via deeplearning.ai's first course.

Pandas

This is one of the most crucial and powerful libraries in data science. To begin with, you end up learning how to read and write CSV and JSON files as well as how to manipulate Data Frame rows, columns, and contents. Again Sentdex to the rescue.

Matplotlib

Learn how to plot basic graphs. The pyplot submodule should be enough for the beginning. Sentdex is your saviour again. The docs are useful too!

PyTorch

PyTorch is a Machine Learning framework built to provide a toolkit for writing Deep Learning research and application code. We would suggest that you get hands-on experience with PyTorch by following the official tutorials.

Git

An integral part of most software projects, Git is a tool for version control and managing changes to code. This is a cool guide for a quick overview, for a more detailed intro check out this course. Make sure to sign up for the Student Developer Pack on GitHub.

Conda

Managing different packages when you are working on multiple projects can be a pain (especially in Python). Conda (and various other similar tools) is a highly featured package manager which makes life much easier. Check it out at the docs and learn how to use it here or here.

Resources for ML and AI

For Core Machine Learning, we recommend Bishop's *Pattern Recognition and Machine Learning* Textbook [Link] and for Core AI, Russel and Norvig's *Artificial Intelligence: A Modern Approach* [Link]. Both of these are very comprehensive textbooks that cover the fundamentals of both areas with easy to understand explanations. You do not need to complete them cover to cover in one go, but can use them more as reference books, learning about specific topics at a time.

For those who are new to Deep Learning, we recommend you start with the Stanford CS231 (Computer Vision): [YouTube Link] [Course Link] course which covers everything from loss functions and basics of Neural Networks to Computer Vision. Go through the video lectures and attempt the accompanying assignments to get a good idea of various fundamental concepts. The Deep Learning Book [Link] is also a great resource for detailed introductions to a wide range of concepts.

Assignment

The assignment is made up of the following sections

1. Pruning and Sparsity
2. Graph Neural Networks
3. Natural Language Processing
4. Reinforcement Learning
5. Computer Vision
6. Paper Implementation

The suggestion would be to attempt **any three out of the first 5 questions**. However, you can attempt the **Pruning and Sparsity problem and a Paper Implementation Task**. Please note that the difficulty of the questions varies, and you will be judged primarily on the effort you have put in.

Along with how well you perform the tasks, you will also be evaluated on how clearly and thoroughly you document your approach and results. So make sure to make your final README or report for each task as detailed and clear as possible.

Pruning and Sparsity

As models become bigger and bigger, (e.g. Google's paper GShard has a model with > 500B parameters!) it becomes necessary to find ways to scale down models. In this question, you will be implementing pruning techniques and then creating an analysis of the techniques. The question will test your understanding of deep learning models as you will have the task of choosing what to prune and to what extent. The task is the following -

1. **Base model implementation** - Implement a very basic (but large enough size > 35 MB at least) vision model. You can also implement very famous models like VGG, which already have implementations online. Test this model on the CIFAR-10 dataset (you should get > 90% accuracy easily!).

2. **Fine-grained pruning by magnitude** - This task has several components, ideally,
    a. you would need to find *appropriate sparsity values* for each convolution layer so proceed by doing a *sensitivity scan* of layers on different sparsity values, (google sensitivity scan, the course listed has really good slides.)
    b. Pick sparsity values for each layer and prune them, document what factors (**Hint: It's not only the effect of more sparsity**) you considered when picking a value for each layer. The resulting size (count only non-zero elements) should be < **25%** of the original model size. You would also observe a substantial drop in accuracy.
    c. *Fine-tune* this new and pruned model and report the accuracy you can achieve, you should again be able to reach ~ 90% accuracy.

    Note: You will be judged more on your analysis, so make sure to document all results, positive and negative, and the corresponding reason.

3. **Bonus tasks** - Implement code to plot the weight distribution for layers pre- and post-pruning. After that check the balance of the activations when input is passed.

Resources -

1. This folder has some basic definitions and starter code from us.
2. MIT's 6.5940 [YouTube][Course Website (2023)].

Recommended Platform - Google Colab Notebook

Graph Neural Networks

With the advent of Graph Neural Networks (GNNs), Machine Learning in molecular research has sped up immensely. For most cases, because the area is new, people will crawl their own custom datasets over the web. The task for this section is broken into 3 (2 + 1 bonus) parts:

1. **Literature review** – prepare a small (~300 words) literature review of the GAT (Graph Attention Network) paper. Pretty neat advice here, although you can do it without referring to that website as well.

2. **Implementation** – implement a very basic GAT and test it out on any molecular dataset from here (make sure there is some baseline so we know that the model is working, although it's not necessary to get baseline results at all!) OR refer to the paper's experiments for datasets (PPI is in PyG's datasets as well). You can implement it from scratch in PyTorch or in PyG, but you cannot directly import and use it. This is a good tutorial on implementing custom Message Passing layers in PyG.

3. **Bonus** – In case you are really interested, first try to import PyG's basic GCN layer and compare it to the results you obtained using a GAT, and if you're done with all other sections, you can also try to implement the GCN layer itself! **It will be much easier compared to implementing a GAT Layer**.

Resources –

a) Stanford's cs224w [YouTube] [Course Website (2020)],

b) Antonio Longa's tutorials [1] [2],

c) PyG documentation can be overwhelming but it is a good place regardless.

If you have (somehow) been keeping up with GNNs, you would know that Google just released their TF-GNN 1.0, it is much more complex **right now** than PyG so try to stick to PyG (**Bonus** for implementing it there). One final warning: setting up PyG can lead to weird bugs, so do create a new environment for this task!

Recommended Platform - Local machine.

Natural Language Processing

Efficiently scaling neural networks in natural language processing (NLP) poses a considerable challenge due to increasing computational demands. Traditional models often struggle to balance capacity and computational efficiency, limiting their ability to handle large-scale language datasets effectively.

To address this, the concept of Mixture-of-Experts (MoE) introduces a novel approach to conditional computation, aiming to boost model capacity without a significant surge in computational costs. This approach involves a collection of specialized sub-networks (experts) and a trainable gating network. Unlike conventional architectures, the MoE layer selectively activates experts based on input examples, effectively engaging a fraction of the model's capacity. This enables a notable increase in overall model capacity without a proportionate rise in computational load.

1. **Implementation**: Implement a small architecture stacked with two LSTM layers, with an MoE layer in between. Check the effectiveness of this model on the designated datasets by varying the sizes of the layers and the number of experts.

2. **Dataset**: The assignment requires you to implement models based on the specified architecture on the two datasets for two different tasks.
    a. Part **(i)** requires you to train a model for Named Entity Recognition on the Dataset CoNLL-2003 (available here).
    b. For part **(ii)**, use the Stanford Question Answering Dataset (SQuAD 1.1) (available here) to evaluate another model on the task of Question-Answering.

3. **Analysis and Comparison**: Evaluate the relative performance of your MoE-augmented language model, comparing it against the baseline model without the MoE layer. Assess the impact of varying the number of experts and explore the trade-offs between model capacity and computational efficiency. What impact does modifying the gating network have on the system's performance?

Resources -

1. Stanford cs224n [YouTube Link] [Course Link]
2. Mixture of Experts Paper - [Link]

Recommended Platform - Google Colab Notebook.

Reinforcement Learning

The credit assignment problem is one of the fundamental challenges in reinforcement learning. This challenge becomes even more difficult when the environment offers sparse and/or delayed rewards. Various strategies have been explored to address this issue, including attention mechanisms, eligibility traces and domain-specific heuristics. In this task, you'll explore one strategy: providing an additional reward signal for each of the agent's actions.

Implement the Intrinsic Curiosity Module (ICM) from this paper on Mujoco's Hopper environment**.** More specifically, your tasks are broken down into the following subtasks:

1. Train an encoder for the environment states. Choose any architecture you want - CNN, Autoencoder, Transformer blocks, etc. Make sure you justify your choice in the report
2. For the inverse model, train an MLP with a regression head or classification head at the end, depending on whether the action space is continuous
3. Implement the forward model with the following modifications: instead of using just the current frame encoding, use a stack of previous frames' encodings and the current action to compute the loss. Is it better than using just the current state? Report it in your results.
4. Integrate the ICM with the DQN algorithm. Report your results

Again, we emphasize that you document every step and report the results, even if you are not able to complete these tasks.

Resources:

1. Reinforcement Learning: An Introduction by Sutton and Barto - A classic text, will be a bit intimidating at the start, but stick with it since it's a pretty amazing resource
2. Lectures at UCL by DeepMind: David Silver is a fantastic teacher; if you are interested in a more thorough coverage, check out Hudo Van Hasselt's lectures
3. Gymnasium - go through the documentation. You'll also find a few tutorials for getting your hands dirty
4. Berkeley cs285: [YouTube Link] [Course Link]

Recommended Platform: Google Colab/Kaggle Notebooks. If you are having trouble using Gymnasium, try local training as well.

Computer Vision

Autoencoders are a staple in computer vision, they encode images in a latent space in a self-supervised manner. The Variational Autoencoder provides a modification to the original autoencoders allowing for variability in the space and also image generation. You are tasked with the following:

1. Implement a simple variational autoencoder trained on the MNIST dataset.
2. Qualitatively test the generation of samples when the latent space is sampled from a Gaussian (1,2) distribution and compare it with the samples from a Normal distribution.
3. Try to improve the quality of generations from the beta distribution without changing the data or the architecture. (This only includes the data preprocessing and model architecture, the forward functions, losses, and weights can be changed).
4. Report the various steps taken even if they do not lead to improvement and your reasoning for the same.

Bonus - Carry out the same task using a Gamma (3,2) distribution.

Resources -

1. Stanford cs231n [YouTube Link] [Course Link]
2. Variational Autoencoder - [Link], [Link]
3. KL Divergence - [Link]

Recommended Platform - Google Colab Notebook.

Paper Implementation

Your task is to examine the central claim of a paper by implementing it and performing additional empirical experiments.

1. Select any one of the papers published in an A or A* conference (NeurIPS, ICML, ICLR, ICCV, AAAI, ACL, etc). First, read the abstract and give a rough reading of the paper you have chosen and identify its central claims. At this point inform us that you are attempting to implement the paper and tell us about the paper chosen.
2. Once you get approval from us, you can go ahead and give a thorough reading of the paper and implement the paper using a framework of your choice (we highly recommend PyTorch).
3. Formulate and perform additional experiments and ablations (such as trying on a different dataset, with different experimental parameters etc...) to test the central claims of the paper. You can discuss these with your mentor.
4. Document your code with a README and a short review of the paper (preferably in LaTeX). The documentation should contain the results of your evaluation and explain how your additional experiments evaluate the paper's claims.

You will be evaluated primarily on your understanding of the paper and the additional evaluations you have performed on its central claims. We are not expecting very clean code, however, it should be able to replicate the results described in the paper and be well-documented. While this task has not been designed for beginners, anyone is welcome to give it a go.