Spring 2025 Induction Assignment

Society for Artificial Intelligence and Deep Learning

www.saidl.in

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

The deadline is Saturday, March 15, 2025 at 11:59 PM IST.

A Note from Us

Through this assignment, our aim is to introduce you to the key and upcoming 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 will 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 tools and concepts required along the way.

Please, go through the instructions before attempting the questions. The questions may 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 the time (depending on your previous familiarity) on each question will be spent **learning** new concepts. 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 regardless 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 after the second week to stay updated with your progress.

All the very best!

Submission Guidelines

The submission deadline is **Saturday 15/03/2025 at 11:59 PM IST**.
Submission is to be made via this form.

The submission should be in the form of a Github repository titled **SAiDL-Spring-Assignment-2025**. Use this repository as a starter. Your repository must be **kept private up until the submission deadline**, but don't forget to make the repository public when submitting the assignment.

You are required to document your approach and results for each question you attempt and analyses wherever required, in detail. The documentation should be included in the GitHub repository along with the code. The documentation must be made in LaTeX. Make sure your final report for each task is as detailed and clear as possible. You are free to use other code bases, but ensure that the **tasks** in the given question have been **fulfilled**.

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

You'll be allotted a mentor for the assignment **after the first two weeks** and you are required to inform them whenever you complete a question (submit the Google form only after you have completed the assignment or at the deadline). Make sure you join our Slack Workspace to receive all the assignment related updates 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 acknowledge/cite 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 preferred mode of communication for all the questions that you have as it will also benefit others. However, you can reach out to us through other means in case we fail to respond on Slack.

- Karan Bania (SSMs / Multi-modality) - 7069863756
- Yash Bhisikar (SSMs / RL) - 9096794571
- Tejas Agrawal (Multi-modality / Core ML) - 8770544585
- Ankita Vaishnobi Bisoi (Core ML / Multi-modality) - 7788920434
- Shreyas V (Multi-modality / Core ML) - 8660728506
- Harshvardhan Mestha (SSMs / Diffusion) - 9892572823
- Aaron Rock Menezes (SSMs / Core ML) - 8451907244
- Sasmit Datta (Diffusion / Multi-modality) - 8334825566
- Sarang S (Core ML / RL) - 8301827846
- Ashmit Rana (Core ML / Multi-modality) - 7827146531

Resources

The following is a list of resources for 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 or Corey Schafer. These books [1], [2] and the official documentation are good references.

- ***Coursera Specialization on Machine Learning***
  This specialization is the launchpad for most AI enthusiasts and gives you a first hands-on approach to Machine Learning along with the math behind it.

- ***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. You can go to its official tutorials (we recommend version 1.x as 2.x is not as widely adopted yet), this is a library that will become easy to operate once you start using it. Good luck with shape errors!

- ***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. These tutorials by Patrick Loeber also help.

- ***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.

- ***ML and AI***
  For Core Machine Learning, we recommend Bishop's "Pattern Recognition and Machine Learn-

ing" 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 Stanford's cs231n (Deep Learning for 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 (we recommend doing assignments from the latest offering of the course as well) 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.

We also recommend a parallel from Harvard, [CS 197]. It has detailed notes on the implementation and coverage of more recent concepts.

- ***Overleaf***
  For creating your LaTeX report, we highly recommend using Overleaf, a user-friendly editor for working with LaTeX files. Learning to create LaTeX documents is a valuable skill to have. It has a steep learning curve, but this one-time effort has huge benefits. You can explore the official Overleaf tutorials or check out this helpful YouTube playlist designed for beginners.
  PS: This document itself was made using Overleaf.

Assignment

The assignment is made up of the following sections:

1. Core ML Task (**Compulsory**)
2. Domain-Specific Tasks (**any one out of 4**)
   (a) State-Space Models
   (b) Multi-Modality
   (c) Reinforcement Learning
   (d) Diffusion
3. Paper Implementation

You have to attempt this assignment in **one** of the following ways:

1. Core ML ***and*** one of the four domain-specific tasks
2. Paper Implementation task **only**.

Every question has a **BONUS** section. This is only to be done after the preceding tasks for that question have been completed.

Please note that you will be judged primarily on the effort you have put in and your understanding of the concepts involved.

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 **LaTeX report** for each task as detailed and clear as possible.

Core ML

Robustness in machine learning is crucial when working with real-world datasets prone to noisy labels, which can lead to overfitting and poor model performance. It is essential to develop methods that handle such challenges effectively. However, improving robustness often comes at the **cost** of performance on **clean** datasets, as noise mitigation techniques can limit a model's ability to fully leverage accurate data. Thus, balancing robustness and performance is a key challenge in designing reliable systems.

Normalizing losses enhances robustness to noisy labels by bounding loss values within [0, 1]. However, while robust, these losses can suffer from underfitting, limiting performance on clean data. The **Active-Passive Loss (APL) framework** balances robustness and performance. Active losses (e.g., Cross-Entropy, Focal Loss, etc.) maximize the probability of the true class, while passive losses (e.g., Mean Absolute Error, Reverse Cross-Entropy) also minimize the probabilities of incorrect classes. Pairing the active and passive losses to address underfitting and enhance noise tolerance, making APL effective in noisy-label scenarios.

1. **Data Preparation**: Modify the CIFAR-10 dataset by introducing symmetric noise rate $\eta \in [0.2, 0.8]$ by randomly flipping the labels within each class to incorrect labels from other classes.
2. **Normalized Losses**: Train models on the noisy datasets using normalized loss functions, like Normalized Cross-Entropy (NCE) and Normalized Focal Loss (NFL). Record the performance metrics for these loss functions and compare their results with their vanilla counterparts (standard Cross-Entropy and Focal Loss). Generate plots to illustrate the effect of different noise rates on model performance and demonstrate the robustness of normalized losses over their vanilla alternatives.
3. **APL Framework**: Train models using the APL losses (you may use loss functions such as NCE, NFL, MAE, and RCE) and compare their performance with the previous results. Highlight how APL improves robustness to label noise while maintaining or enhancing model performance. Use plots to compare the performance of all methods under different noise rates.
4. **BONUS**: Repeat the experiments but with asymmetric noise in the CIFAR-10 dataset with $\eta \in [0.1, 0.4]$.

Resources:

- Normalized Loss Functions for Deep Learning with Noisy Labels [paper]
- PyTorch Tutorials [link]

Recommended Platform: Google Colab, Local Machine

State-Space Models

State-space models are very recent **architectures** (not models, which are known for over 60 years!) based on signal processing concepts. They are a first theoretical solution to the $O(L^2)$ complexity that transformers are infamous for. Thus, these models have widespread potential applications, as a general-purpose sequence-modeler.

1. **Read** "Efficiently Modeling Long Sequences with Structured State Spaces" [link].
2. **Implement** a simple S4 in PyTorch, and **evaluate** it on the sCIFAR-10 dataset [repo], this is the general *Long Range Arena* repository, you will find the dataset in the *lra_benchmarks/image/* folder. It is **not** required to reproduce their results, design a simpler **proof-of-concept**.
3. Additionally, try **experimenting** with a different discretization scheme from the paper. This paper might help and their repository has implementations for this.
4. **BONUS**: Re-do the question using **only** numpy (you can further reduce the subset size if epoch-time is a problem).

Resources:

- Blog Posts by the authors [1], [2], [3]
- Gilbert Strang's MIT course, 18.06, might help as a refresher on Linear Algebra [Website], especially his lecture on the [Woodbury's Identity].
- Signal Processing primer: Understanding convolutions, Control Theory, and the State-Space Systems.

The paper might (will) seem daunting at first, you **do not** have to read all the proofs in the appendix (though we do expect some familiarity with why and how) try to read the paper multiple times before implementing anything.

Recommended Platform: Local Machine, Google Colab.

Multi-Modality

This task involves ML networks that work between **modalities**. A modality, in general, refers to the way something is experienced (for example, vision is a modality, language is another modality). Multi-modality is on the rise right now. For our case, we will be working with Graphs, Language and Vision.

1. Perform EDA on the dmg777k dataset from kgbench [paper] [repository] [website].
   (a) Plot the types and number of entities (nodes)
   (b) Plot the types and number of edges
   (c) See what type of information is present per node
   (d) The ***triples*** attribute defines your graphs, learn how that works
   (e) The ***e2i / i2e*** fields might also help
   (f) Think about how you might use this to **train** a Graph Neural Network (GNN).
2. Develop a GNN that performs **node-classification** on the dataset.
   (a) The nodes/edges can be diverse! Learn how to operate with these, CLIP might be a nice starter paper
   (b) You might want to start with simple vision feature extraction and a Bag-of-words for text + **something**
   (c) Finally, you could use pre-trained vision and language models from HuggingFace to embed the nodal information.
   (d) Think **when** to perform this embedding!
3. **BONUS**: Jointly fine-tune the pre-trained models.

Resources:

- Stanford's Machine Learning with Graphs course (cs224w) [Website]
- Stanford's Deep Learning for Computer Vision course (cs231n) [Website]
- Stanford's NLP with Deep Learning course (cs224n) [Website]
- Antonio Longa's tutorials on PyG [Playlist]
- HuggingFace has several courses. You can find more tutorials by googling the exact topic you want them on.

These resources are neither exhaustive nor necessary to complete, use them for specific parts when needed!

Create a subset of the dataset if the training time per epoch for your network exceeds **2 minutes**. Clearly document your **subset** creation strategy (yes, you can also get creative here).

A pipeline of this extent is tough to make, especially for beginners, spend some time reading around papers, get a grip on Vision models, Language models etc. before implementing. Perplexity is your friend.

The task is very similar to what you might have to do for a research project, it has a lot of moving components.

Notes for the **bonus** task:

- This is **compute intensive**, so a few epochs as a proof-of-concept will suffice.
- On a high-level, this only requires the addition of loss functions, but it is not trivial to implement.

Recommended Platform: Google Colab (can be done locally, might be slower).

Reinforcement Learning

The exploration-exploitation dilemma presents a fundamental challenge in reinforcement learning (RL): agents must balance exploring new states and actions with exploiting known profitable strategies. Traditional approaches, such as ϵ-greedy, often fall short in deep RL settings, particularly in continuous action spaces. Modern deep RL addresses this challenge through various techniques, including novelty-based exploration, count-based methods, and entropy regularization.

In this task, we will investigate one such technique: adding **parameter space noise** and **learned uncertainty estimates**.

1. **Implement** TD3 (Twin Delayed DDPG) algorithm on Mujoco's Hopper-v4 environment
   (a) Use relevant techniques like Polyak averaging, Dual critic networks, Prioritized experience replay and n-step returns
   (b) Exclude the target policy smoothing component of TD3
2. **Implement** the Independent Gaussian Noise variation from Noisy Nets for Exploration paper for your actor-network from TD3.
   (a) Evaluate the performance gained and experiment with different sample distributions (beta, gamma, etc.) Report your findings and inferences.
   (b) How would you test the adversarial robustness of your setup? Would the performance be affected by different encoder blocks (CNNs, Attention Heads) ***Hint***: Think about how each of the building blocks operate over data

In many real-world RL problems, agents must learn from **raw** sensory inputs like images or sensor data, making it challenging to identify relevant features for decision-making. Good representations can significantly improve sample efficiency by **capturing essential patterns** while ignoring irrelevant variations, enable better transfer learning across related tasks, and make exploration more effective by creating meaningful similarity metrics between states. This is particularly important in deep RL, where the quality of learned representations can dramatically impact the agent's ability to generalize from limited experience and tackle increasingly complex tasks.

In the bonus task, we will evaluate the plug-and-play nature of different representation learning approaches and their impact on sample efficiency and generalization performance

[BONUS] Use any self supervised learning technique (BYOL, SALE, etc) for learning state-action representations and integrate it with your implementation above.

(a) Pick a suitable approach to learn the representations (concatenating them, using shared parameters, state-dependent action representation, etc.) and **justify** your choice.

(b) Demonstrate the effectiveness of your learned representations in relevant ways. ***Hint***: We are not looking for accuracy/loss plots, go through the papers for ideas. You can be creative here.

Deep RL is very finicky with the parameter choices, so it's possible that even with functional code, you might not get good results. That's why we are emphasizing to document every step and report the results, even if you are not able to complete the tasks.

Resources:

- Gymnasium: go through the documentation and tutorials to get started.

- Spinning Up in Deep RL by OpenAI - neat primer on all things deep RL. We highly recommend to go through the "Policy Gradient Algorithms" section to gain an intuitive understanding of TD3
- Reinforcement Learning: An Introduction by Sutton and Barto - the Bible of RL. Might be a difficult read, but we encourage you to stick around and work your way through the equations on your own
- UCL Lectures by David Silver <3 or the newer Hado Van Hasselt's lectures.
- Stanford CS224R [Course Link]: A more accessible introduction to Deep RL. A good place to learn about Actor-Critic methods
- Berkeley CS285 [Youtube Link] [Course Link]: Another Deep RL course, more comprehensive and theoretical

Recommended Platform: Local Machine, Kaggle Notebooks, Google Colab.

glhf!

Diffusion

A diffusion model is most commonly used for generative tasks - images, audio, video, 3D models, etc. The model takes pure noise and iteratively removes said noise to produce the required output. Traditionally, models such as Stable Diffusion use a CNN based backbone called a U-Net. These models are *very capable* but are not *very scalable*.

To remedy this the Diffusion-Transformer (DiT) was introduced by replacing the CNN based UNet with a Transformer block. This shall be the primary focus of your tasks.

1. Clone this repository, and run the ***run_DiT.ipynb*** with any one of the class labels, and answer the following:
   (a) Set CFG to zero, and set CFG to the maximum value → what difference do you see, can you explain **why**?
   (b) Now try changing the sampling steps – try 50 then 250 then 500 → what difference do you see, and **explain**.
2. Since DiT is a scalable transformer – it will useful to try a more efficient attention implementation (or variant) → look at the xformers library, and implement the following:
   (a) Replace DiT's Attention block with the xformers attention block and measure the time taken to sample 50 images – Do you notice a speedup v/s the baseline model?
   (b) Now we'll look into Sliding Window Attention (SWA). Train 2 DiT models on the landscape dataset **without class conditioning** from scratch (one with repo's full attention and with SWA) and compare their samples and FID.
3. Now let's look into another evaluation method: CLIP Mean Maximum Discrepancy (CMMD). This uses CLIP embeddings and Mean Maximum Discrepancy (MMD) instead of Inception embeddings and Fréchet Distance (FD) (google these methods!) to evaluate a generative model.
   (a) Implement a CMMD evaluation code and calculate it for the two DiTs you trained above. Compare it with FID. Does it correlate with better image generation? **Why**? What assumptions are made for FID calculation? How does CMMD improve on those?
   (b) Now replace CLIP with SigLIP and ALIGN in your implementation. Compare the scores you get from these above models with CLIP and FID. Explain the differences in results to the best of your ability.
4. **BONUS:** The DiT block in the given repo uses Adaptive Layer Norm (AdaLN) to add class conditioning to our diffusion models. Train a DiT on the sun397 dataset **using class conditioning** in the form of In-context Conditioning. Compare this method with other conditioning methods like Cross Attention. Are these better than AdaLN? Why? Compare using FID and CMMD.

Notes:

- **sun397 dataset**: Don't train on the entire dataset. Train on a subset of it - take 100 images each from any 40 of 287 total classes the dataset has. You can use the HF dataset stream feature to create this subset.
- Resize your images (both for landscape and sun397 datasets) to 256x256 to get optimal quality from VAE.

- You'll be training latent diffusion models. Don't train in pixel space. Use the pre-trained VAE which the DiT repo uses.
- Use normal attention for the bonus task, and not your xformers implementation in task 2.
- You are free to choose any size or configuration of your DiT model, but use that as your baseline for all experiments (eg: DiT-B/8).
- Answer all the questions mentioned in your report.
- Do attach images (generated from **your implementation**) along with your explanations.

Resources:

- This video covers the core concepts of a diffusion model (highly recommended if this is your first time hearing about diffusion) – watch till 41:40.
- Relevant Papers – DDPM (maths behind diffusion), [VAE], Diffusion models beat gans on image synthesis, Latent Diffusion Models (Important read!).
- Mathematical explanations of DDPMs: [1] [2] [3]
- Additional Videos – KL divergence, explainer for DiT, VAE, VQ-VAE
- Stable Diffusion Explainers – [1] [2]

Recommended Platform - Google Colab, Kaggle 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 recently(NeurIPS, ICML, ICLR, CVPR, AAAI, ACL, etc. [Rankings Portal]). 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 it a thorough reading 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 (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.