SaiDL Spring Induction Assignment Report

Subroto Majumder

April 6, 2025

1 Introduction

This report details the work undertaken and insights gained during the SaiDL Spring Induction Assignment. The assignment covered two key areas of modern deep learning: robustness in supervised learning (Core ML) and generative modeling with Diffusion Transformers (Diffusion). The following sections summarize the tasks performed, observations made, and knowledge acquired in each area.

---

2 Part 1: Core ML - Robustness to Noisy Labels

2.1 Background: The Challenge of Noisy Labels

Real-world datasets are often imperfect and contain noisy labels (incorrectly assigned classes). Training models on such data can lead to significant challenges:

- **Overfitting:** Models may memorize the incorrect labels, leading to poor generalization on clean data.
- **Performance Degradation:** The presence of noise can hinder the model's ability to learn the true underlying data distribution.

Developing robust machine learning methods capable of handling noisy labels is needed. But there's a trade-off: techniques designed to mitigate noise might restrict the model's capacity, potentially leading to underfitting and suboptimal performance on clean data. The goal is to balance robustness against noise with high performance on accurately labeled data.

2.2 Understanding Robust Loss Functions

My initial research and the provided materials highlighted robust loss functions as a promising approach. Loss functions like Symmetric Cross-Entropy were early examples. The assignment focused on Normalized Losses and the Active-Passive Loss (APL) framework.

- **Normalized Losses:** These losses (e.g., Normalized Cross-Entropy - NCE, Normalized Focal Loss - NFL) constrain the loss value per sample, typically within $[0, 1]$. This bounding prevents individual noisy samples with potentially huge loss values (especially in standard Cross-Entropy) from dominating the gradient updates, thus enhancing robustness. However, this same bounding mechanism can sometimes limit the model's ability to strongly fit the clean samples, potentially causing underfitting.
- **Active-Passive Loss (APL):** APL aims to overcome the limitations of normalized losses by combining two loss components:

- **Active Loss:** Encourages the model to maximize the probability of the *correct* class (e.g., NCE, NFL). It actively pushes the model towards the labeled target.
- **Passive Loss:** Encourages the model to minimize the probabilities of *incorrect* classes (e.g., Reverse Cross-Entropy - RCE, Mean Absolute Error - MAE). It passively discourages the model from being confident about wrong predictions.

This combination seeks to retain the noise robustness of normalized losses while mitigating underfitting by leveraging the complementary nature of the active and passive components.

2.3 Task 1: Data Preparation - Introducing Symmetric Noise

To simulate noisy label scenarios, the CIFAR-10 dataset was modified. Symmetric noise was introduced at rates (η) ranging from 0.2 to 0.8. This involved, for each image, randomly changing its true label to one of the other incorrect labels with probability η, ensuring the new label was different from the original.

2.4 Task 2: Evaluating Normalized Losses

- **Implementation:** Normalized Cross-Entropy (NCE) and Normalized Focal Loss (NFL) were implemented based on the provided paper's definitions.
- **Training & Observations:** Models were trained on the noisy CIFAR-10 datasets using NCE, NFL, and their standard counterparts (Cross-Entropy - CE, Focal Loss - FL).
  - **Performance:** NCE and NFL generally yielded lower peak accuracy compared to standard CE and FL on datasets with lower noise levels.
  - **Robustness:** However, as the noise rate (η) increased, the performance degradation of NCE and NFL was less severe compared to CE and FL. Training curves for normalized losses were observed to be more stable, exhibiting fewer fluctuations compared to standard losses which tended to overfit the noise more aggressively. Plots comparing accuracy vs. noise rate illustrated this trend, visually confirming the enhanced robustness but also highlighting the underfitting issue on cleaner data.
  - **Initial Model:** An initial small CNN (2 Conv layers, 1 MaxPool, 2 Linear layers) was used.

2.5 Task 3: Evaluating the APL Framework

- **Implementation:** An APL loss was implemented by combining NCE (as the active component) and RCE (as the passive component), denoted as APL(NCE+RCE).
- **Training & Observations:** Models were trained using APL(NCE+RCE) and compared against the previous results (CE, FL, NCE, NFL).
  - **Performance & Robustness:** APL(NCE+RCE) demonstrated significantly improved performance compared to both standard losses and normalized losses alone, especially at moderate noise levels ($\eta = 0.2$ to 0.6). It successfully balanced robustness and performance, achieving higher accuracy than NCE/NFL while being more robust than CE/FL. Comparative plots clearly showed APL maintaining better accuracy across increasing noise rates.
  - **High Noise Collapse:** A notable observation was that at a very high noise rate ($\eta = 0.8$), the performance of APL with the initial small model collapsed to near random chance (approx. 10% accuracy on CIFAR-10).

– **Impact of Model Size:** To investigate the collapse at $\eta = 0.8$, a larger, more standard CNN architecture (similar to those commonly used for CIFAR-10 benchmarks, e.g., a ResNet variant or a deeper custom CNN) was employed. Training with this larger model using APL yielded significantly better results at $\eta = 0.8$ (achieving 39% accuracy). While this didn't fully replicate the paper's reported 52% (potentially due to differences in architecture, hyperparameters, or training duration), it clearly indicated that model capacity plays a crucial role, especially under extreme noise conditions, and that APL benefits from sufficient model complexity.

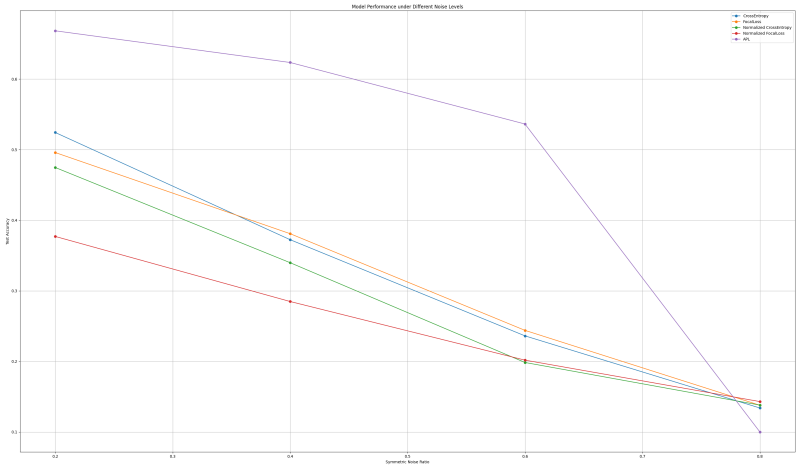
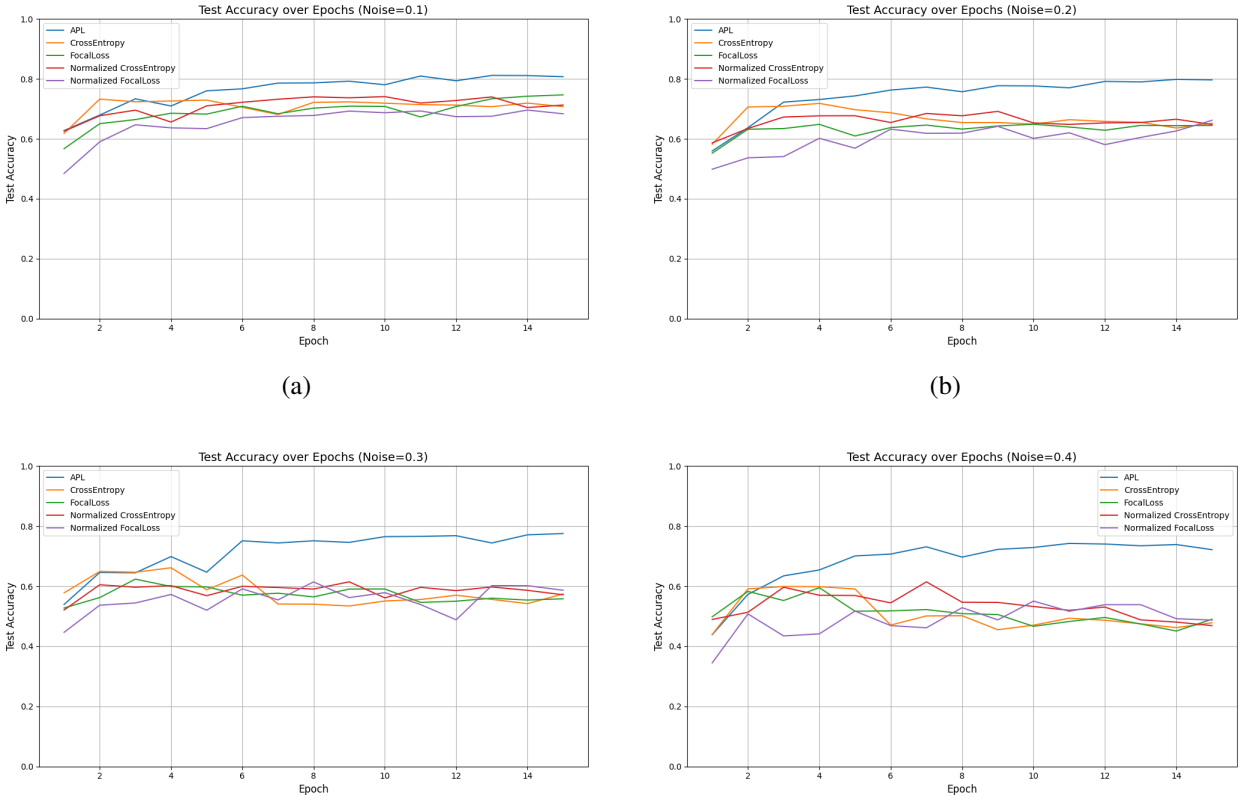


Figure 1: Symmetric noise(x-axis) vs test accuracy(y-axis) graph

2.6 Task 4 (Bonus): Asymmetric Noise

- **Implementation:** Asymmetric noise was introduced to CIFAR-10, where labels of class `c` were flipped specifically to class `(c+1) % num_classes` with probability η (ranging from 0.1 to 0.4). This simulates scenarios where confusion primarily occurs between specific pairs of classes.
- **Training & Observations:**
  - At low asymmetric noise rates ($\eta = 0.1$), all loss functions performed reasonably well.
  - As η increased (0.2 to 0.4), the performance of standard losses (CE, FL) and normalized losses (NCE, NFL) became more erratic and degraded significantly. Training curves showed fluctuations.
  - APL(NCE+RCE) stood out by maintaining consistent performance and stable training curves even as asymmetric noise increased up to 0.4. This suggests the passive component (RCE) is particularly effective at preventing the model from becoming overly confident in the systematically incorrect labels introduced by asymmetric noise.



(a) (b)

(c) (d)

Figure 2: Accuracy of each Losss function at varying levels of Asymmetric noise

2.7 Core ML Summary Key Learnings

- Robust loss functions are vital for handling noisy labels in real-world scenarios.
- Normalized losses (NCE, NFL) improve robustness by bounding loss values but can suffer from underfitting.
- The APL framework effectively balances robustness and performance by combining active (target-seeking) and passive (error-avoiding) loss components.
- APL(NCE+RCE) demonstrated superior performance across various symmetric and asymmetric noise levels compared to baseline and normalized losses alone.
- Model capacity interacts with loss function robustness; larger models can better leverage robust losses, especially under high noise.

---

3 Part 2: Diffusion Models - Diffusion Transformer (DiT)

3.1 Background: Diffusion Models and the Rise of Transformers

Diffusion models represent a powerful class of generative models. They operate by:

1. **Forward Process:** Gradually adding Gaussian noise to data (e.g., an image) over a series of timesteps until it becomes pure noise.
2. **Reverse Process:** Training a model (typically a U-Net) to reverse this process, iteratively removing noise step-by-step, starting from pure noise to generate a sample.

While traditional diffusion models using CNN-based U-Nets are highly capable, their scalability can be limited. The Diffusion Transformer (DiT) architecture replaces the U-Net backbone with Transformer blocks, leveraging the known scalability and performance benefits of Transformers, particularly for large models and datasets.

My initial step involved familiarizing myself with the fundamental principles of diffusion models and the standard U-Net architecture before diving into the DiT specifics.

3.2 Task 1: Exploring DiT Parameters (Using Pre-trained Model)

Experiments were conducted using the provided `run_DiT.ipynb` notebook and a pre-trained DiT model conditioned on ImageNet classes.

3.2.1 (a) Classifier-Free Guidance (CFG) Scale

- CFG is a technique used during sampling to enhance the adherence of the generated output to the class label. It works by interpolating between a purely conditional prediction and an unconditional prediction. The CFG scale (`w`) controls the strength of this guidance: output $=$ unconditional_pred $+$ $w \times$ (conditional_pred $-$ unconditional_pred).
- **Observations:**

- **CFG = 0:** This corresponds to purely *unconditional* generation. The generated images were random samples from the model's learned data distribution, often bearing little resemblance to the specified class label.
- **CFG = Max (e.g., 10.0):** This applies strong guidance. The generated images strongly adhered to the target class label, often appearing more "prototypical" or exaggerated for that class. Diversity among samples for the same class might decrease.
- **Intermediate CFG (e.g., 1.0 to 4.0):** This provided a balance. `CFG=1.0` corresponds to purely conditional generation without amplification. Values slightly higher often produce good quality samples that clearly belong to the target class but retain some diversity.

Increasing CFG scale amplifies the direction pointed towards by the class condition, pushing the sample further away from the unconditional generation path. Too low CFG ignores the condition, while too high CFG can lead to less realistic or overly saturated/sharp images by over-emphasizing class features.

(a) Samples from CFG 0

(b) Samples from CFG 1

(c) Samples from CFG 10

Figure 3: Samples of diffusion model at various values of CFG

3.2.2 (b) Sampling Steps

- The reverse diffusion process is iterative. The number of sampling steps determines how many discrete denoising steps are taken to transform pure noise into a final image.
- **Observations:**
  - **Low Steps (e.g., 50):** Generation was fast but often resulted in lower-quality images. Details might be missing, artifacts could be present, or the overall structure might seem incomplete, especially noticeable at lower CFG values where some samples barely formed the subject.
  - **Medium Steps (e.g., 250):** A good balance between quality and speed. Images were significantly clearer and more detailed than with 50 steps.
  - **High Steps (e.g., 500):** Generation was slowest but typically produced the highest fidelity images with the finest details and fewest artifacts. The improvement from 250 to 500 steps was often less dramatic than from 50 to 250.

  Each sampling step refines the image by predicting and removing a small amount of noise based on the current timestep and conditioning. More steps allow for a more gradual and potentially more accurate reconstruction of the data distribution from noise, leading to higher quality samples at the cost of increased computation time.

(a) 50 Steps, CFG 1

(b) 50 Steps, CFG 10

(c) 250 Steps, CFG 1

(d) 250 Steps, CFG 10

(e) 500 Steps, CFG 1

(f) 500 Steps, CFG 10

Figure 4: Effect of number of sampling steps on images, at two different cfg values

3.3 Task 2: Efficient Attention Mechanisms

3.3.1 (a) xformers Attention

- **Goal:** Replace the standard attention mechanism in the DiT blocks with a memory-efficient and potentially faster implementation from the `xformers` library.
- **Implementation:** The `Attention` block within the DiT model was modified to utilize `xformers.ops.memory_`
- **Observations:** When timing the generation of 50 samples, a slight *slowdown* was observed compared to the baseline implementation.
- **Caveats:** This result was counter-intuitive, as `xformers` is generally expected to provide speedups, particularly through memory savings that allow larger batches or prevent OOM errors. The observed slowdown could be due to several factors:
  - **Overhead:** For the specific model size and image resolution used, the overhead of calling the `xformers` function might outweigh its benefits.
  - **Hardware:** Performance characteristics can be highly dependent on the specific GPU architecture.
  - **Measurement:** The timing methodology might not have been precise enough or isolated the attention computation effectively.
  - **Implementation Details:** Potential subtle differences in implementation or configuration.

  Due to time and resource constraints, a more thorough benchmark across different settings was not feasible. However, the potential memory efficiency benefits remain a key advantage of `xformers`.

3.3.2 (b) Sliding Window Attention (SWA)

- **Concept:** SWA is an approximation of full attention where each token only attends to a local window of neighboring tokens, reducing the computational complexity from $O(N^2)$ to $O(N \times W)$ where N is sequence length and W is window size. This is particularly useful for very long sequences (high-resolution images).
- **Status:** This sub-task, involving training two DiT models (one with full attention, one with SWA) from scratch on a landscape dataset and comparing samples/FID, was **not able to attempt** due to time constraints.

3.4 Task 3: Advanced Evaluation Metrics (CMMD)

- Evaluating generative models is challenging.
  - **FID (Fréchet Inception Distance):** Compares the statistics (mean and covariance) of features extracted by an InceptionV3 model from real and generated images. Lower FID generally indicates better perceptual quality and diversity. It assumes Inception features capture perceptual similarity well and that the features follow a Gaussian distribution.
  - **CMMD (CLIP Mean Maximum Discrepancy):** Uses features extracted from the CLIP model (trained on image-text pairs) and calculates the MMD distance between the distributions of real and generated image features. MMD is a non-parametric distance metric between distributions. CMMD potentially offers advantages as CLIP features might better capture semantic and perceptual similarity relevant to human judgment, and MMD makes fewer assumptions about the underlying feature distributions compared to FID.

- **Status:** This task, involving implementing CMMD, calculating it for the trained models (from Task 2b), comparing with FID, and exploring variants using SigLIP and ALIGN embeddings, was **not attempted** due to time constraints.

3.5 Diffusion Summary Key Learnings

- Diffusion models generate data by learning to reverse a noise-adding process.
- DiT leverages the scalability of Transformers for diffusion, replacing the traditional U-Net.
- CFG scale controls the trade-off between sample diversity and adherence to conditioning.
- The number of sampling steps impacts generation quality and speed.
- Efficient attention mechanisms like `xformers` offer potential speed/memory benefits, though practical gains depend on the specific setup. SWA offers an alternative for handling long sequences.
- Evaluating generative models relies on metrics like FID and emerging alternatives like CMMD, which may offer different perspectives on sample quality and distributional similarity.
- Various methods exist for conditioning diffusion models (AdaLN, cross-attention, in-context), each with potential trade-offs.

---

4 Conclusion

This assignment provided practical experience in Core ML, showing how methods like APL improve robustness to noisy labels, and in Diffusion models, exploring the DiT architecture and key parameters like CFG and sampling steps. While time constraints limited completing all Diffusion tasks, the learning and initial experiments were highly valuable.