Deep Learning Assignment 2: RNNs, Transformers, and Self-Supervised Learning

Due Date: **January 11th, 2026, 23:59**

In this assignment:

- You will implement recurrent neural networks (RNNs) and Long Short-Term Memory (LSTM) networks from scratch.
- You will build an image captioning model using RNNs, LSTMs, and attention mechanisms.
- You will implement the Transformer architecture from the "Attention is All You Need" paper.
- You will explore self-supervised learning with SimCLR contrastive learning.
- You will work with state-of-the-art pretrained models: CLIP and DINO.

Setup: Same as Assignment 1 - use the free tier of Google Colab and upload the assignment folder to `My Drive/dl/assignments/assignment2/`. All notebooks require GPU acceleration for training.

Assignment Parts

This assignment consists of four main parts. Complete all parts in the order presented.

Part 1: Image Captioning with RNNs and LSTMs (25 points)

Notebook: `rnn_lstm_captioning.ipynb`

In this part, you will:

- Implement vanilla Recurrent Neural Networks (RNNs) from scratch
- Implement Long Short-Term Memory (LSTM) networks
- Build an image captioning model that generates natural language descriptions of images
- Implement attention mechanisms to improve caption quality
- Train your models on the COCO Captions dataset

You will implement RNN and LSTM cells with forward and backward passes, then combine them with a pre-trained CNN (RegNet-X 400MF) to create an end-to-end image captioning system. Finally, you'll augment your model with spatial attention to focus on relevant image regions while generating captions.

Part 2: Transformers for Arithmetic Operations (25 points)

Notebook: `Transformers.ipynb`

In this part, you will:

- Implement the Transformer architecture from "Attention is All You Need"
- Build core components: Multi-Head Attention, Layer Normalization, Feed-Forward blocks
- Implement both Encoder and Decoder blocks with residual connections
- Implement positional encoding (simple and sinusoidal)
- Train a Transformer model on a toy arithmetic dataset (addition and subtraction)
- Visualize attention weights to understand what the model learns

This implementation focuses on a simplified sequence-to-sequence task with fixed-length inputs and outputs, making it easier to understand the core Transformer mechanisms before applying them to more complex problems.

Part 3: Self-Supervised Learning with SimCLR (25 points)

Notebook: `Self_Supervised_Learning.ipynb`

In this part, you will:

- Learn about self-supervised learning and contrastive learning
- Implement data augmentation for SimCLR
- Implement the contrastive loss function (both naive and vectorized versions)
- Train a SimCLR model and compare it with a baseline model
- Observe the power of self-supervised pretraining for downstream classification tasks

SimCLR learns visual representations by maximizing agreement between differently augmented views of the same image. You will see how models pretrained with self-supervised learning outperform randomly initialized models on classification tasks.

Part 4: CLIP and DINO (25 points)

Notebook: `CLIP_DINO.ipynb`

In this part, you will:

- Explore CLIP (Contrastive Language-Image Pre-Training) for zero-shot classification
- Implement similarity computation between text and image features
- Build a zero-shot classifier and an image retrieval system using CLIP
- Explore DINO (self-DIstillation with NO labels) features for semantic segmentation
- Train a simple segmentation model on DINO features using the DAVIS dataset

CLIP learns to align image and text representations in a shared embedding space, enabling powerful zero-shot capabilities. DINO learns fine-grained visual features that are useful for dense prediction tasks like segmentation.

Submission Instructions

When you have completed all parts of the assignment, follow these steps to generate and submit your work:

Step 1: Generate Submission Files

1. Ensure all code cells in all notebooks have been executed and their outputs are visible
2. **IMPORTANT:** Manually save all `*.ipynb` and `*.py` files before proceeding
3. In the `CLIP_DINO.ipynb` notebook, navigate to the last cell
4. Run the submission cell, which will automatically generate:
   - `a2_code_submission.zip` – A ZIP archive containing:
     - `transformers.py`
     - `Transformers.ipynb`
     - `rnn_lstm_captioning.py`
     - `rnn_lstm_captioning.ipynb`
     - `Self_Supervised_Learning.ipynb`
     - `CLIP_DINO.ipynb`
     - `rnn_lstm_attention_submission.pt`
   - `a2_inline_submission.pdf` – A PDF containing all notebook outputs

Step 2: Create Student Information File

Manually create a text file named `students.txt` containing both students' information in the following format:

```
Student1_Full_Name Student1_ID
Student2_Full_Name Student2_ID
```

For example:

```
John_Doe 123456789
Jane_Smith 987654321
```

Note: Use underscores (_) instead of spaces in the names, and ensure the student IDs match your official registration records.

Step 3: Submit to Lemida

Download all files from your Google Drive to your local computer, then submit the following to Lemida:

1. `a2_inline_submission.pdf`
2. `a2_code_submission.zip`
3. `students.txt`

Good luck!