CS 441

HW 5: Deep Learning and Applications

Due Date: Nov 17, 2025 11:59pm

In this assignment, we explore application of deep learning models and adaption to custom tasks, including fine-tuning, linear probe, and zero-shot prediction. We also introduce two new visual datasets: Oxford 102 Flower and Oxford-IIIT Pet.

The aims of this homework are:

1. Learn about an application of AI and reflect on its potential impact, good and bad.
2. Fine-tune a pretrained ImageNet network to perform a custom classification task
3. Learn how to use the vision-language model CLIP to perform new classification tasks, either zero-shot, linear probe, or nearest neighbor.
4. The stretch goals offer opportunities to experiment with word tokenizers, design a custom network, and/or work with custom data.

Read this document and the report template and tips and tricks before beginning to code.

- Report template
- Tips and Tricks
- Starter code

1. Applications of AI [30 pts]

Choose an application area (e.g. AI for agriculture, advertising, recommendation systems, self-driving cars/trucks, manufacturing, construction, home monitoring, virtual assistants), read at least two related media articles or papers, cite your sources, and answer the following questions:

1. How is AI used in that application area? What is the problem that AI is trying to solve, and what are the key AI/ML technologies involved? What are the technical challenges? (100+ words)
2. What is the actual or potential positive impact? Who is impacted? (50+ words)
3. What is the actual or potential negative impact? Who is impacted? (50+ words)

If you use media articles, they should be from highly respected sources, e.g. The Economist, Wall Street Journal, NY Times, Washington Post, Atlantic, MIT Technology Review. This will be graded as complete/incomplete. Any citation format is OK, as long as it clearly indicates the source, article title, and date of article.

2. Fine-tune ImageNet Model for Pets Classification [30 pts]

A convolutional neural network (CNN) is a type of deep learning artificial neural network commonly used for image processing tasks. In this section, you will load a pretrained network, finetune it for a new task, and evaluate it to achieve a good image classification accuracy on the **Oxford-IIIT Pet dataset**. The task is to classify the breed of cat or dog.

To save you time and focus on the interesting parts, we've provided helper functions and much of the code structure. Read it carefully and complete it as you go. Your key steps are:

1. Load a pretrained ResNet34 and replace the last layer of the network so that the output dimension matches the number of Pet classes. Use the printout of network structure and parameters to answer questions in your report.
2. Set the learning rate and learning scheduler, train for at least 10 epochs, and evaluate. You may need to modify the training settings to get a better performance. You can consider the data augmentation, learning rate, learning rate scheduler, and batch size. Attach the accuracy and loss plots for training_set and test_set, and report your best testing accuracy. You should be able to reach an accuracy of at least 90% within 10 epochs, and make sure that you plot at least 10 epochs.

3. Zero-shot and Linear Probe using CLIP [40 pts]

CLIP (Contrastive Language-Image Pre-Training), developed by OpenAI, is a neural network trained on 400 million (image, text) pairs. CLIP encodes text and images into the same vector space, so that it can match images to text descriptions by encoding each and computing the dot product of the encodings. This enables zero-shot classification, i.e. assigning images to text labels from a dataset without training on that dataset. The following are some resources you may find useful for understanding CLIP and finishing your tasks in this homework.

- GitHub Repo (Make sure to look at this for examples of how to use CLIP)
- Learning Transferable Visual Models From Natural Language Supervision (paper)
- CLIP Website
- Labels for Flowers 102 dataset (json file)

The starter code includes sections for loading the Flowers 102 dataset, the CLIP model, and examples of usage. The code splits the data into a "train" and "test" (validation) set.

Your tasks are to use CLIP for a new classification task in three ways:

Zero-shot: Use pre-trained text and image representations to classify images. For zero-shot recognition (no training), text features are computed from the CLIP model for phrases such as "An image of <flower_name>, a type of flower" for varying <flower_name> inserts. Then, image features are computed using the CLIP model for an image, and the cosine similarity between each text and image is computed. The label corresponding to the most similar text is assigned to the image. We've included an example in the starter code. You'll get that working using a data loader, which enables faster batch processing; then, compute the accuracy over the test set. You should see top-1 accuracy (i.e. percent of times the highest confidence predicted label is correct) in the 60-70% range.

Linear probe: Often, it is possible to improve performance for a particular classification task by training a linear classifier on the model's features. This is called a "linear probe". We can do that by extracting the CLIP pre-trained image features and then training a linear logistic regression classifier. We do not use text features for the linear probe method. Train on the train set, and evaluate on the test set and report your performance. You can get top-1 accuracy in the 90-95% range. If you see accuracy in the 80's, try both normalizing and not normalizing the features.

Nearest neighbor: We can also extract pretrained CLIP features and apply a nearest neighbor classifier. You can use your own implementation of nearest neighbor or a library like sklearn or FAISS for this. Try K={1, 3, 5, 7, 11, 21}. Report performance for best K on the test set. You should see top-1 accuracy in the 80-90% range.

4. Stretch Goals [up to 80 pts]

a. Compare word tokenizers [20 pts]

Train at least two 8K token word tokenizers (e.g. BPE, WordPiece, SentencePiece) on the WikiText-2, and compare their encodings. You can use existing libraries, such as those linked below to train and encode/decode. Report the encodings for "I am learning about word tokenizers. They are not very complicated, and they are a good way to convert natural text into tokens." E.g. "I am the fastest planet" may end up being tokenized as `[I, _am, _the, _fast, est, _plan, et]`. Also, report the tokenizations of an additional sentence of your choice that results in different encodings by the two models.

https://github.com/huggingface/tokenizers

b. Implement your own network [up to 30 pts]

For the Oxford Pets dataset, try to write the network by yourself. You can get ideas from existing works, but you cannot directly import them using packages, and the parameter number should be lower than 20M. Train your network from scratch. You would get points if your network can reach a test accuracy of 35% (15 pts), and another 15 pts if it reaches 45%. You would want to pay more attention to data augmentation and other hyper-parameters during this part. This will require use of a GPU.

Submission Instructions

Append two files: (1) completed report template; (2) a pdf version of your Jupyter notebook. Be sure to include your name and acknowledgments. **The report is your primary deliverable**. The notebook pdf is included for verification/clarification by the grader. To create PDF of notebook: Use "jupyter nbconvert" -- see starter code. To combine PDFs: use https://combinepdf.com/ or another free merge tool. *The first pages should correspond to the report, followed by the code.*

Submit the combined file to Gradescope HW5. **In Gradescope, select pages in your template *and* notebook that correspond to each subproblem**. Failure to correctly identify pages will result in a penalty.