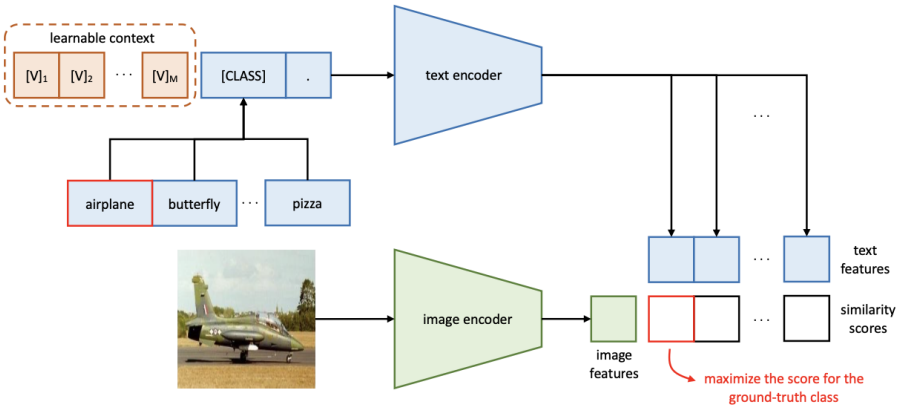
Checkpoint

Smruti Chourasia, Renyi Qu

Project: Replication of CoOp and CoCoOp for Image Classification

Paper Summary:

- Context Optimization (CoOp) is a simple approach specifically for adapting CLIP-like vision-language models.
- **Idea:** Avoid manual prompt tuning by modeling context words with continuous vectors that are end-to-end learned from data while the massive pre-trained parameters are frozen.
- **Method:** models a prompt's context words with learnable vectors, which could be initialized with either random values or pre-trained word embeddings
    - Two implementations are provided to handle tasks of different natures:
        - Unified context - which shares the same context with all classes and works well on most categories;
            - t = [V]1 [V]2 . . . [V]M [CLASS],
        - class-specific context - which learns a specific set of context tokens for each class and is found to be more suitable for some fine-grained categories.
            - [V]i [V]i . . . [V]i [V]j[V]j ...[V]j for i≠ j and i,j ∈ {1,...,K}.
- Training - Learnable context
    - minimize prediction errors using the cross-entropy loss with respect to the learnable context vectors while keeping the entire pre-trained parameters fixed.
    - gradients can be back-propagated all the way through the text encoder, distilling the rich knowledge encoded in the parameters for learning task-relevant context.



Current progress:

We have successfully replicated the model architectures for both CoOp and CoCoOp using PyTorch and the official CLIP package. Specifically, we used the basic RN50 model of CLIP to start with and trained the CLIP+CoOp/CoCoOp wrapper on few-shot examples from the Flower 102 dataset. We evaluated the model performance of Linear Probe, Zero-shot CLIP, and CoOp on it.

We made the following modifications compared to the authors' source code:

- **Save computational cost:**

  We precomputed the image embeddings for all the 1020 images in this small dataset in place of the Vision Encoder part in order to save computational resources.

- **Code modification:**

  We did not build a separate TextEncoder class as they did because we believe this part is rather redundant to the existing "encode_text" function in the CLIP source code. We modified that function directly to fit in the scheme of the project.

  We deleted and recreated the redundant parts of the original modules as they neither contribute to the final performance nor led to better interpretability. Essentially, we focused on constructing the two most important modules: Prompt Learner and the full CLIP+CoOp/CoCoOp wrapper.

- **Dataset for training**
  - The authors did not fix the seed for randomization, which may lead to slightly different evaluation results.
  - The authors did not specify which examples per class they used as their training data, so we fixed our seed and randomly selected data for shots.

- **Optimal settings:**

  In CoOp, the authors attempted 3 different context positions (front, mid, back), 2 different types of soft contexts (class-specific, global), and 2 different types of context vector initializations (fixed via input text, randomized).

  However, in CoCoOp, they only used back position and global context. We

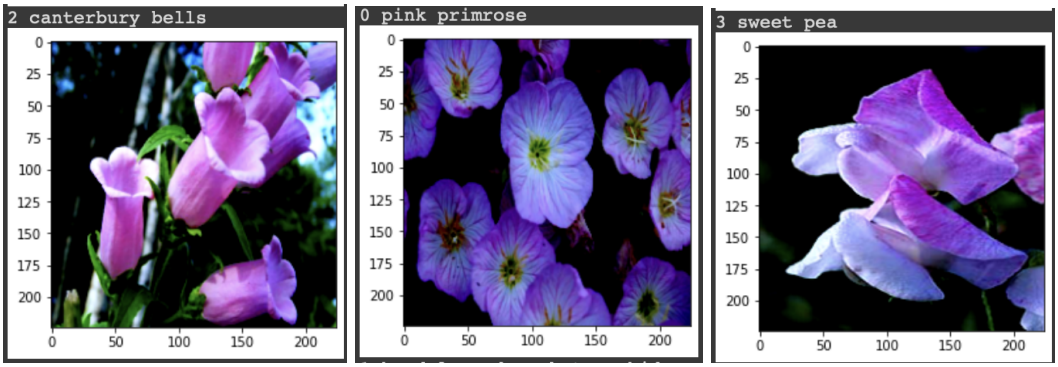
evaluated both models using only the CoCoOp configuration to reduce unnecessary complexities.

- Training/Hardware optimisation (not used)
    - The authors used their own trainer modules named “dassl” for the entire training process, while we wrote the training and evaluation process on PyTorch and PyTorch Lightning. We referred to their paper for hyperparameters, especially for the SGD optimizer and the Cosine learning rate scheduler, to ensure that our replication uses identical configurations as theirs without using their custom “dassl” package.
    - They included hardware optimization parts such as mixed precision and DDP (distributed data parallel), while we chose not to write this part to save time.

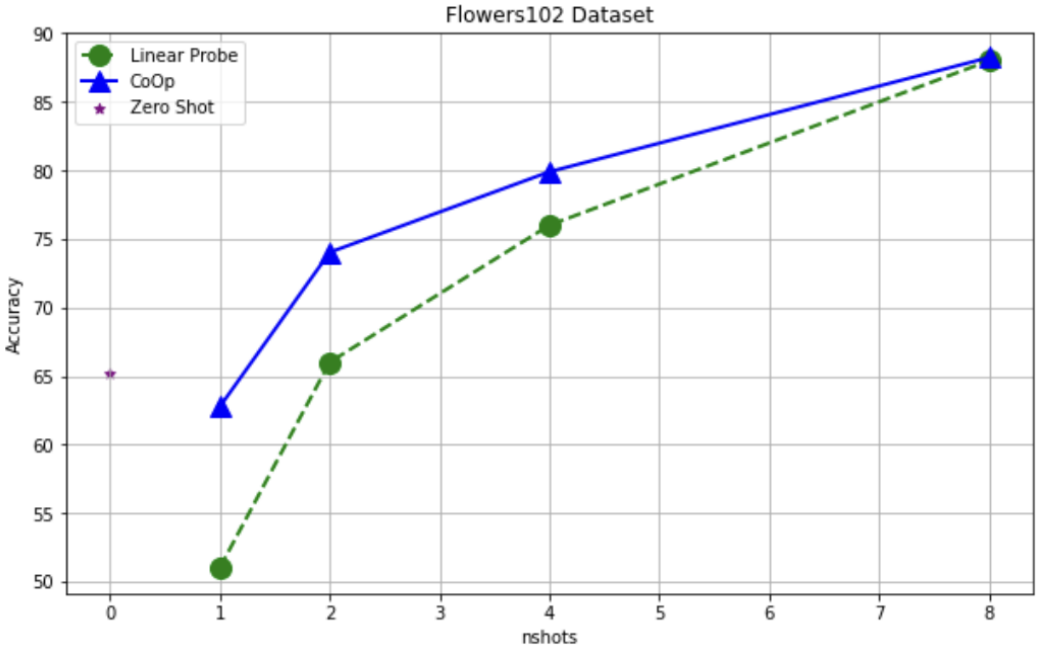
Experimentation:

We started with the Oxford’s 102 Flower Category Dataset. The flowers chosen to be flower commonly occuring in the United Kingdom. Each class consists of between 40 and 258 images. The images have large scale, pose and light variations. In addition, there are categories that have large variations within the category and several very similar categories.

Examples from Dataset



The evaluation results on the Flower102 dataset are shown below:



Using 80:20 split of the data for train(n-shot):test

	Linear Probe	CoOp	Zero Shot
1-shot	51%	62.75%	65.136%
2-shot	66%	74.02%	
4-shot	76%	79.9%	
8-shot	88%	88.24%	

'CoOp helps in the low shot training region'

We can see the benefit gained for 1,2 shots is significant. Hence context based learning can be considered good for tasks with less data annotations. Currently we used a context vector of size 16 (initialized randomly), classname attached to the end - hence the prompt will look like [SOS][V1]...[Vn][CLS][EOS].

Batch size: 12, LR: 0.002 (SGD Optimizer with Cosine Schedular)

Next Steps:

After training and testing on CoCoOp as well, there are several directions that we might go with. We would also want to try other versions of the CLIP model (eg ViT-B/32).

- First, we can train and evaluate our models on other datasets and validate the results from the two papers in comparison against the baselines (linear probe and zero-shot CLIP).
- Second, we can compare few-shot prompt tuning with prompt engineering and see whether it is really necessary to do soft prompting in the case of image classification.
- Third, we can discuss the potential extension of soft prompting on other multimodal tasks (e.g., Image Captioning, VQA).