UNIVERSITÀ DI TRENTO

Learning prompts for transfer learning with test-time adaptation

Trends and Applications of Computer Vision

Giovanni Scialla, Mattia Franzin, Adnan Irshad, Hira Afzal

Where we left…

Fine-tune the Vision-language model

Improve quality of textual prompts

Prompt Engineering

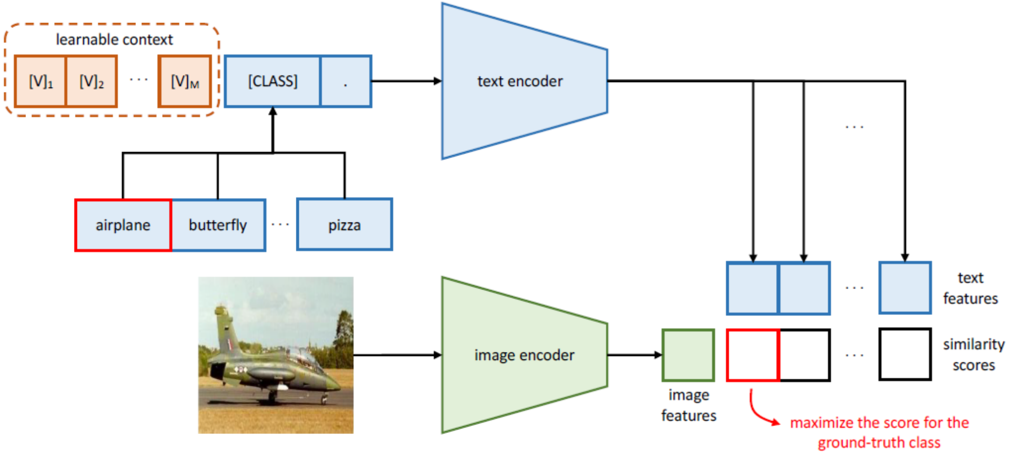
Prompt Learning

Where we left…

CoOp

Learn a Context Vector during training

- **Class-specific context**
  one vector for each class
- **Unified context**
  one vector for all classes



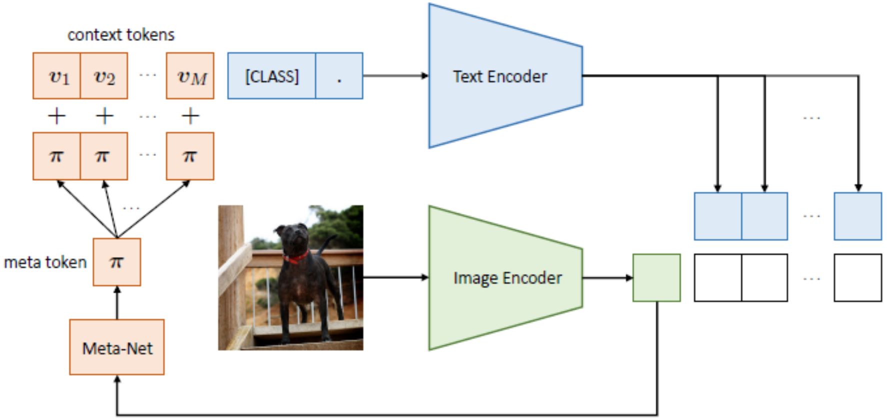

```
    if csc: # Class-Specific-Context
        ctx_vectors = torch.empty(n_cls, n_ctx, ctx_dim, dtype=dtype)
    else: # Generic context
        ctx_vectors = torch.empty(n_ctx, ctx_dim, dtype=dtype)

self.ctx = nn.Parameter(ctx_vectors)  # to be optimized
```

Where we left…

CoCoOp

Condition the Context vector using a learnable token from a **MetaNet**

- use image embedding information to update the conditional tokens



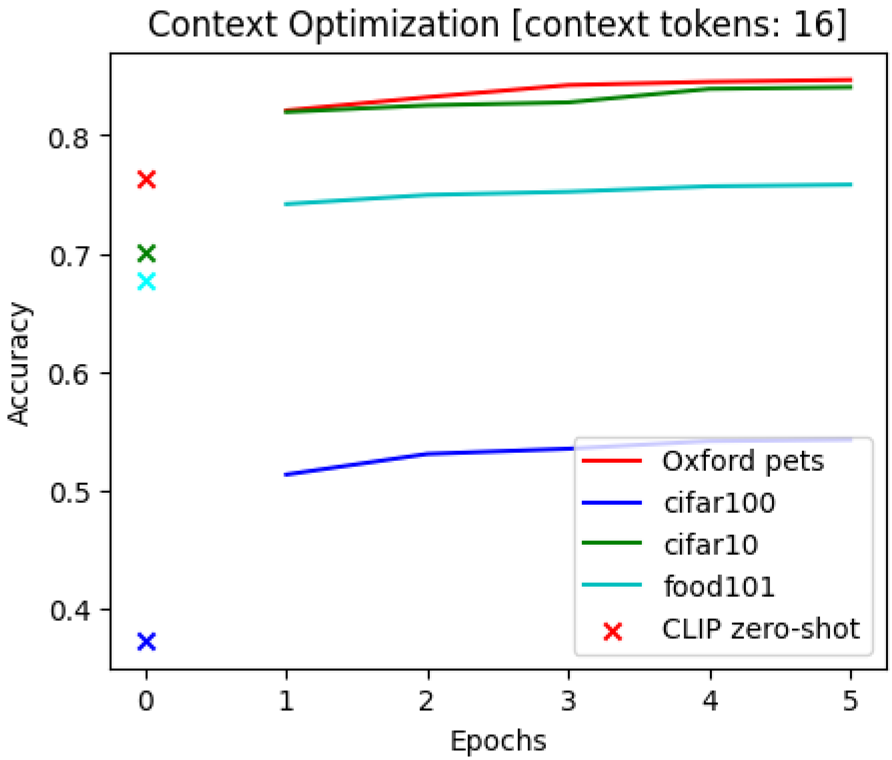

```
    # random initialization
    ctx_vectors = torch.empty(n_ctx, ctx_dim, dtype=dtype)

self.ctx = nn.Parameter(ctx_vectors)

self.meta_net = nn.Sequential(OrderedDict([
    ('linear1', nn.Linear(vis_dim, vis_dim // 16)),
    ('relu', nn.ReLU(inplace=True)),
    ('linear2', nn.Linear(vis_dim // 16, ctx_dim))
]))
```

Datasets

- **CIFAR10** (Alex Krizhevsky, 2009)
- **CIFAR100** (Alex Krizhevsky, 2009)
- **Oxford_pets** (Parkhi et al., 2012)
- **Food101** (Bossard et al., 2014)
- **MNIST** (Deng, L. 2012)

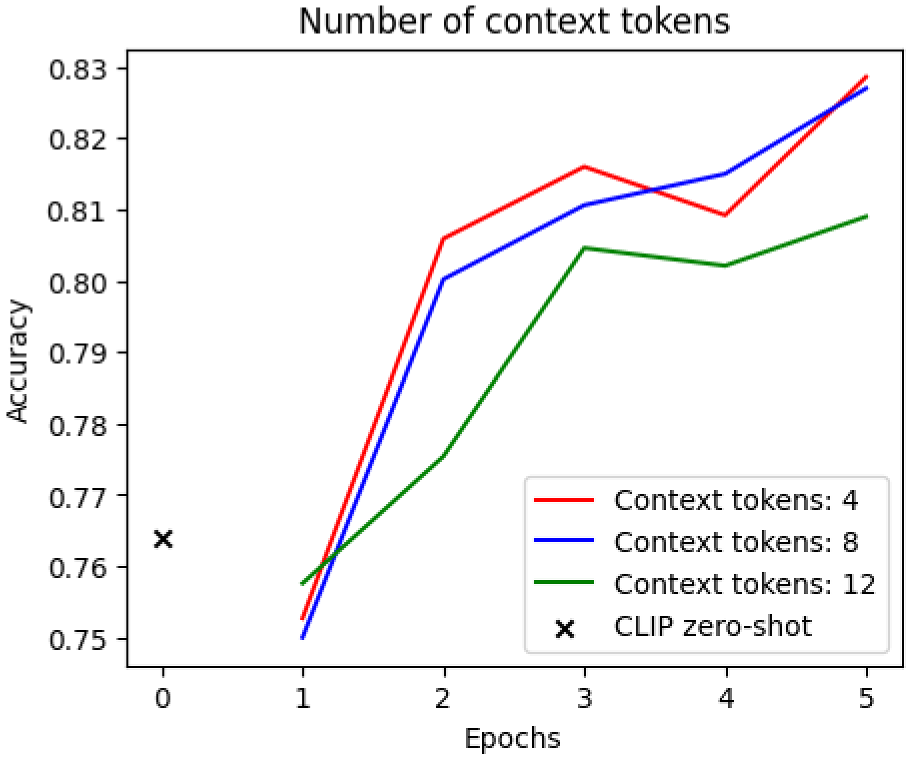
Experiments and Results

Context Optimization performance against Zero-shot CLIP



Dataset	Zero-Shot CLIP Accuracy	CoOp Accuracy
Oxford Pets	0.764	0.8468
Food 101	0.677	0.7583
CIFAR10	0.701	0.8406
CIFAR100	0.374	0.5432

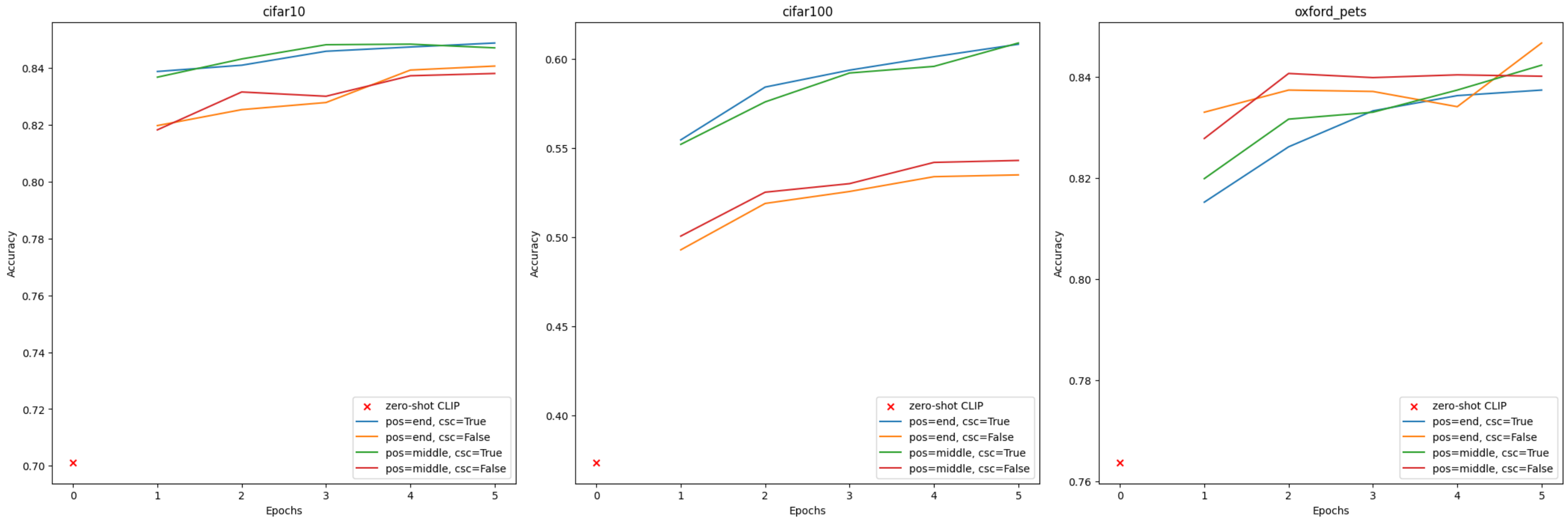
Hyperparameter - Number of Context Tokens



Comparison between different CoOp modalities

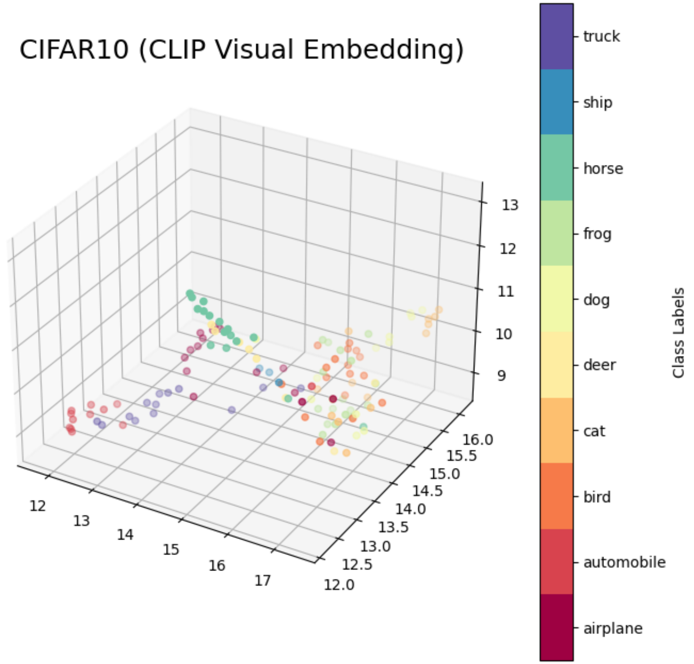
initial context: “X X X X X X X X”

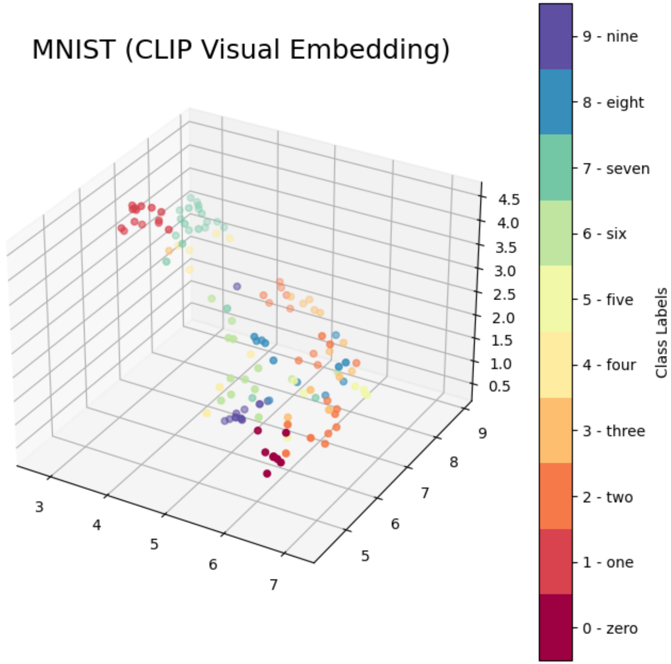
number of context words (tokens): 8



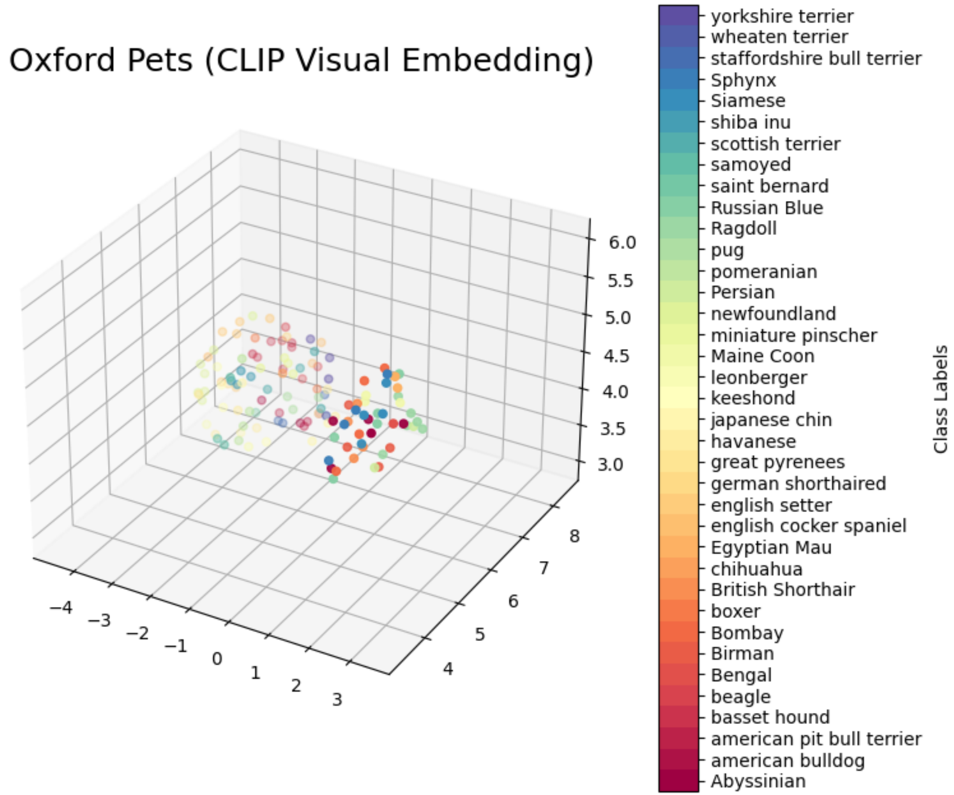
Dimensionality reduction for visualization - UMAP

UMAP: use graph layout algorithms to arrange data to be as structurally similar as possible in a low-dimensional space

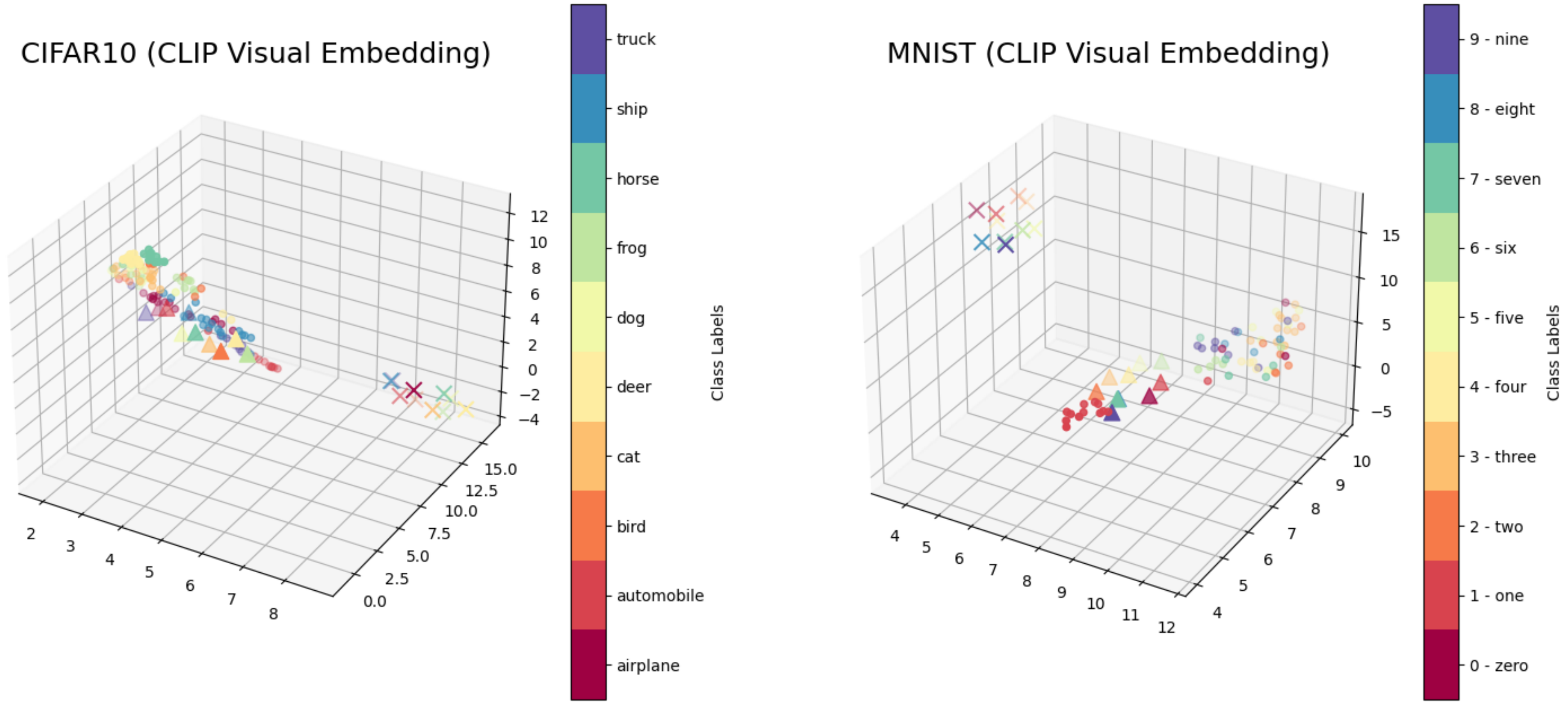




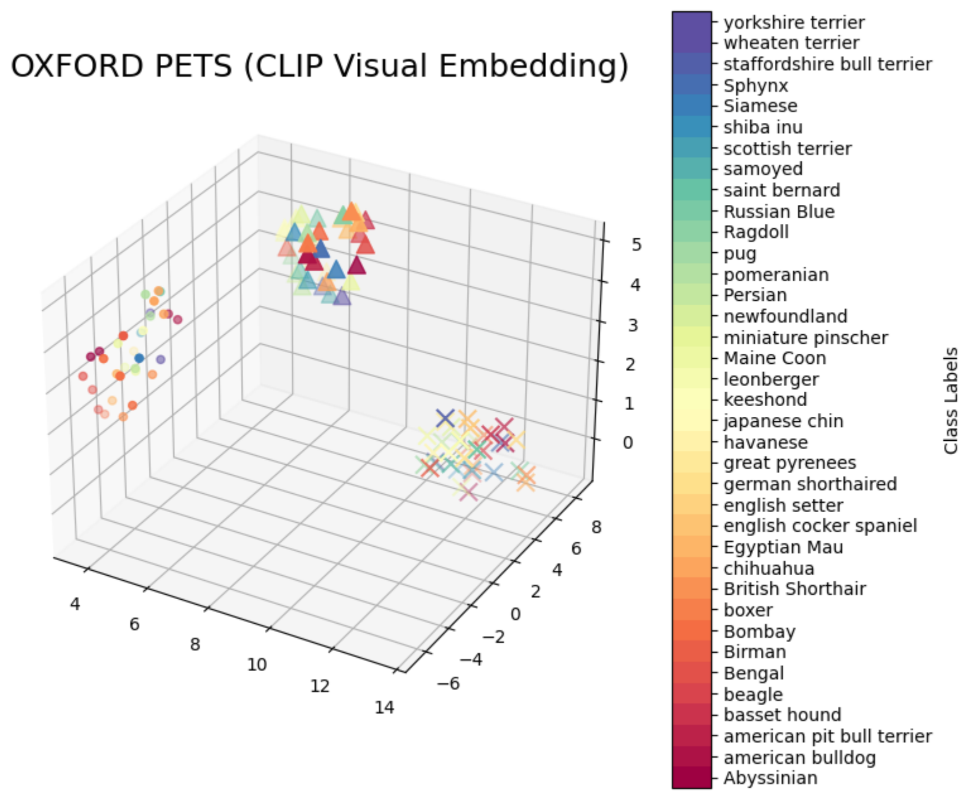
Dimensionality reduction for visualization - UMAP



Learning Prompts for better Embedding alignment



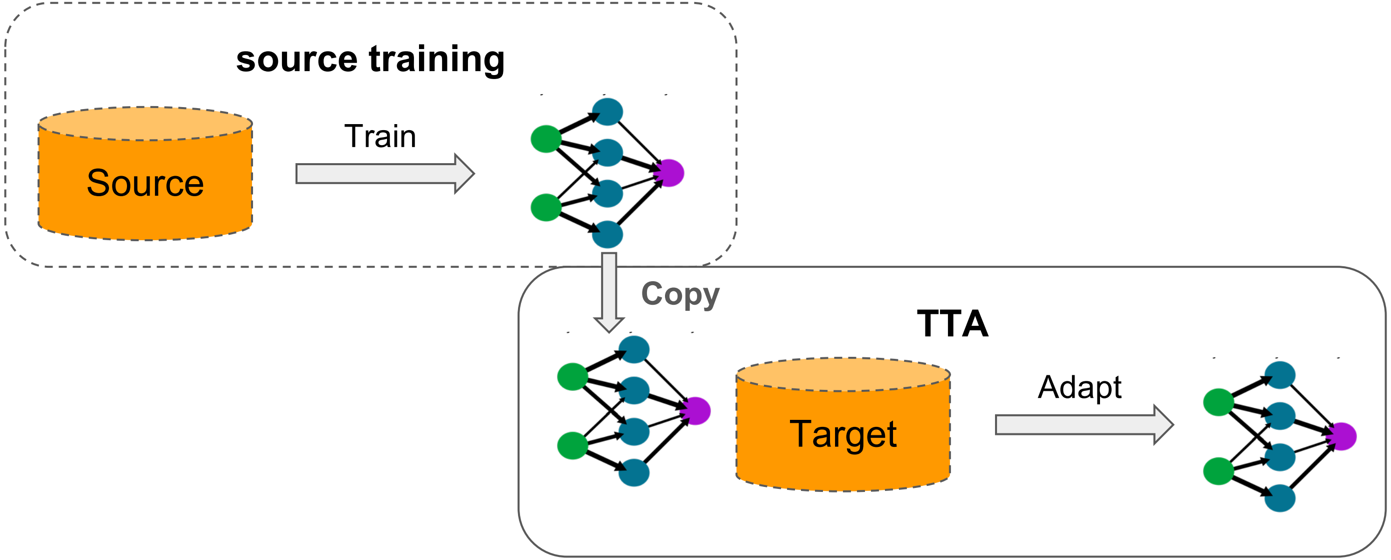
Learning Prompts for better Embedding alignment



Second Part: Prompt Tuning with Test-Time Adaption

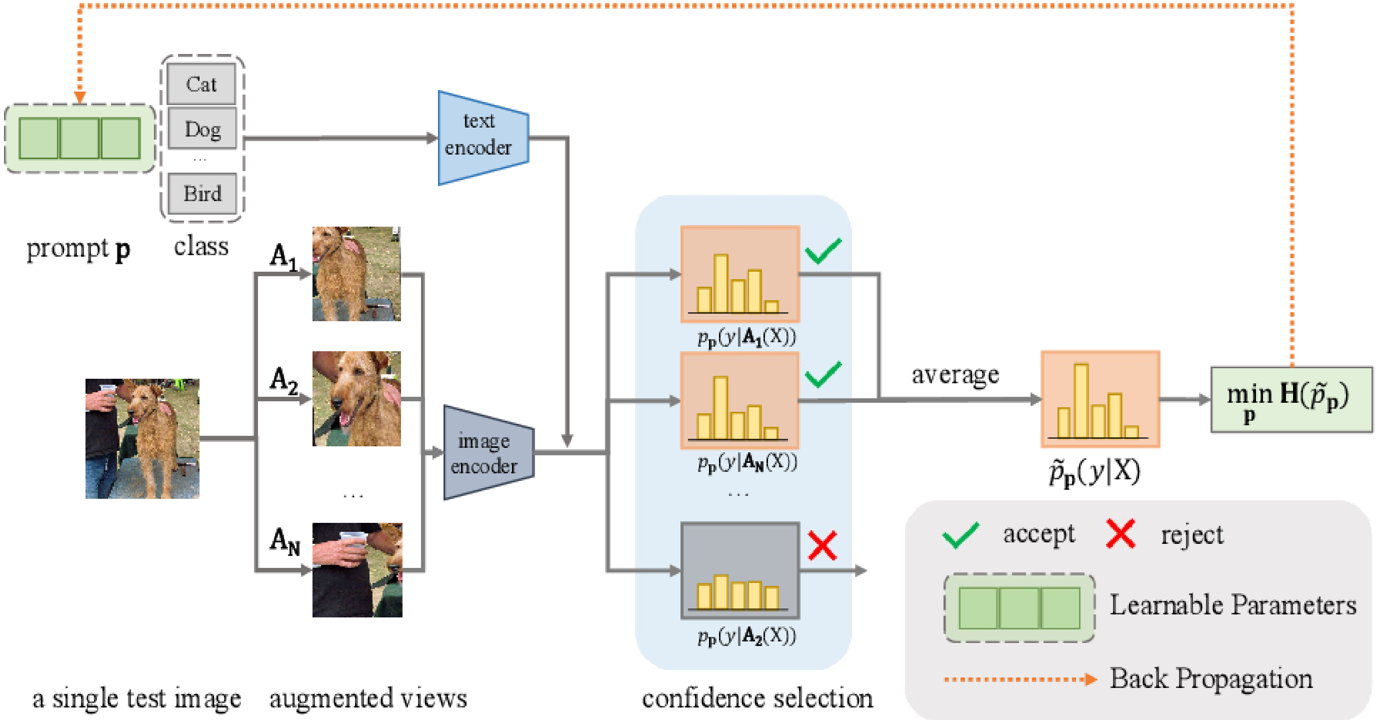
Second Part: Prompt Tuning with Test-Time Adaption

- Test-Time Prompt Tuning for efficient adaptation
  - An effective lightweight adaptation mechanism at test time for foundation models



Test-time prompt tuning for zero-shot generalization in Vision Language Models

TPT: learn adaptive prompts **p** on the fly with a single test sample $\mathbf{X}_{\text{test}}$

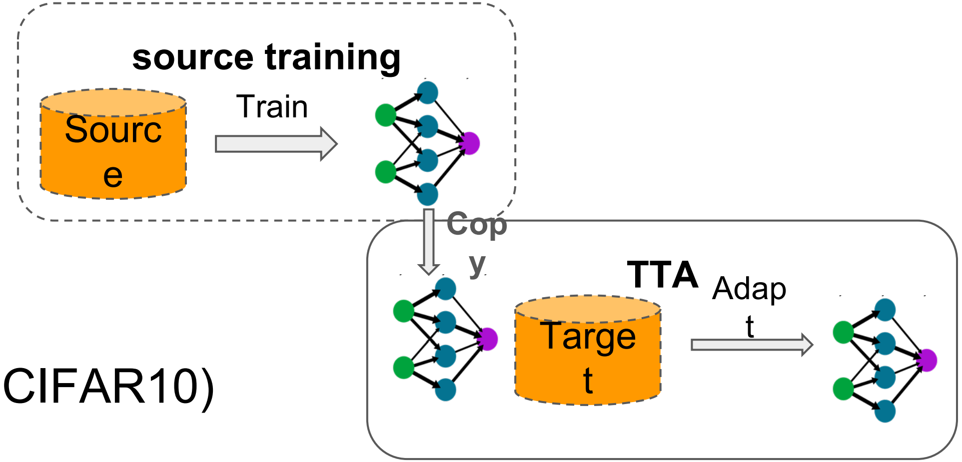


Datasets

- **CIFAR10** (Alex Krizhevsky, 2009)
- **Oxford_pets** (Parkhi et al., 2012)
- **Food101** (Bossard et al., 2014)
- ImageNet-1K (Jia Deng, 2015)
- BongardHOI (Jiang et al, 2022)

TPT Baselines

1. **TPT + CoOP**
    a. Image Classification
    b. Context-Dependent Visual Reasoning
        i. BongardHOI
2. **DiffTPT + CoOp**
    a. Image Classification
        i. Imagenet-R-1K
3. **Experiments**
    a. Cross-dataset evaluation
        i. (Oxford_Pets, Food101, CIFAR10)



TPT Baselines: TPT + CoOP

TPT for image classification task

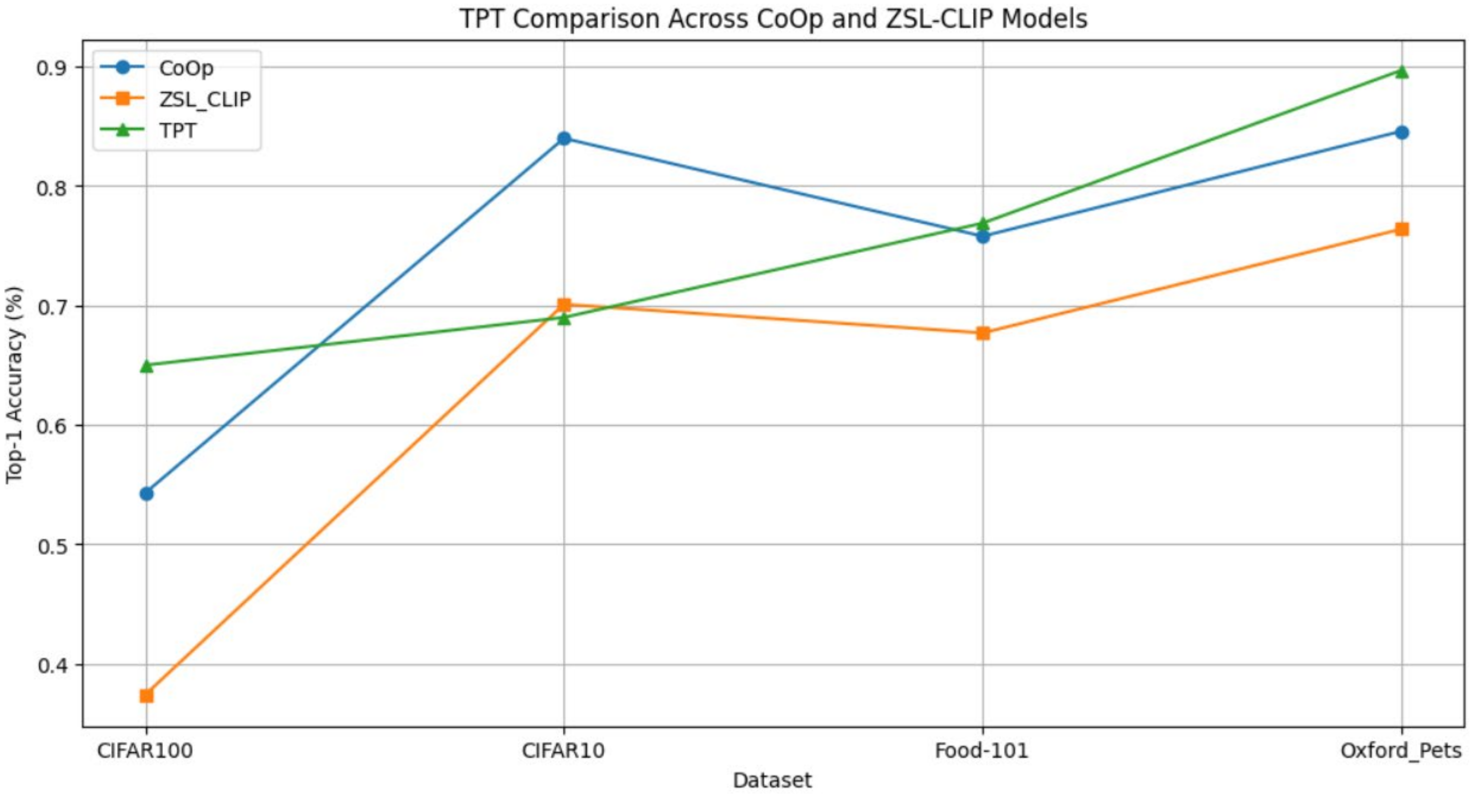
- **Pre-trained CLIP**
  - TextEncoder
  - ImageEncode
- **CoOp**
  - ClipTestTimeTuning
  - PromptLearner
- **AugMix**
  - AugMixAugmenter
- **tpt_classification**

```
class ClipTestTimeTuning(nn.Module):
    def __init__(self, device, classnames, batch_si
                 n_ctx=16, ctx_init=None, ctx_posit
        super(ClipTestTimeTuning, self).__init__()
        clip, _, _ = load(arch, device=device, down
        self.image_encoder = clip.visual
        self.text_encoder = TextEncoder(clip)
        self.logit_scale = clip.logit_scale.data

        # prompt tuning
        self.prompt_learner = PromptLearner(
            clip, class_names, batch_size, n_ctx, 

        self.criterion = criterion
```

```
def tpt_classification(args):
    model = get_coop(args.arch, args.test_sets, args.gpu, args.n_c

    if args.pretrained_model is not None:
        logger.info("Use pre-trained soft prompt (CoOp) as initial
        pretrained_ctx = torch.load(args.pretrained_model)['state_
        assert pretrained_ctx.size()[0] == args.n_ctx
        with torch.no_grad():
            model.prompt_learner.ctx.copy_(pretrained_ctx)
            model.prompt_learner.ctx_init_state = pretrained_ctx

    model_state = None
```

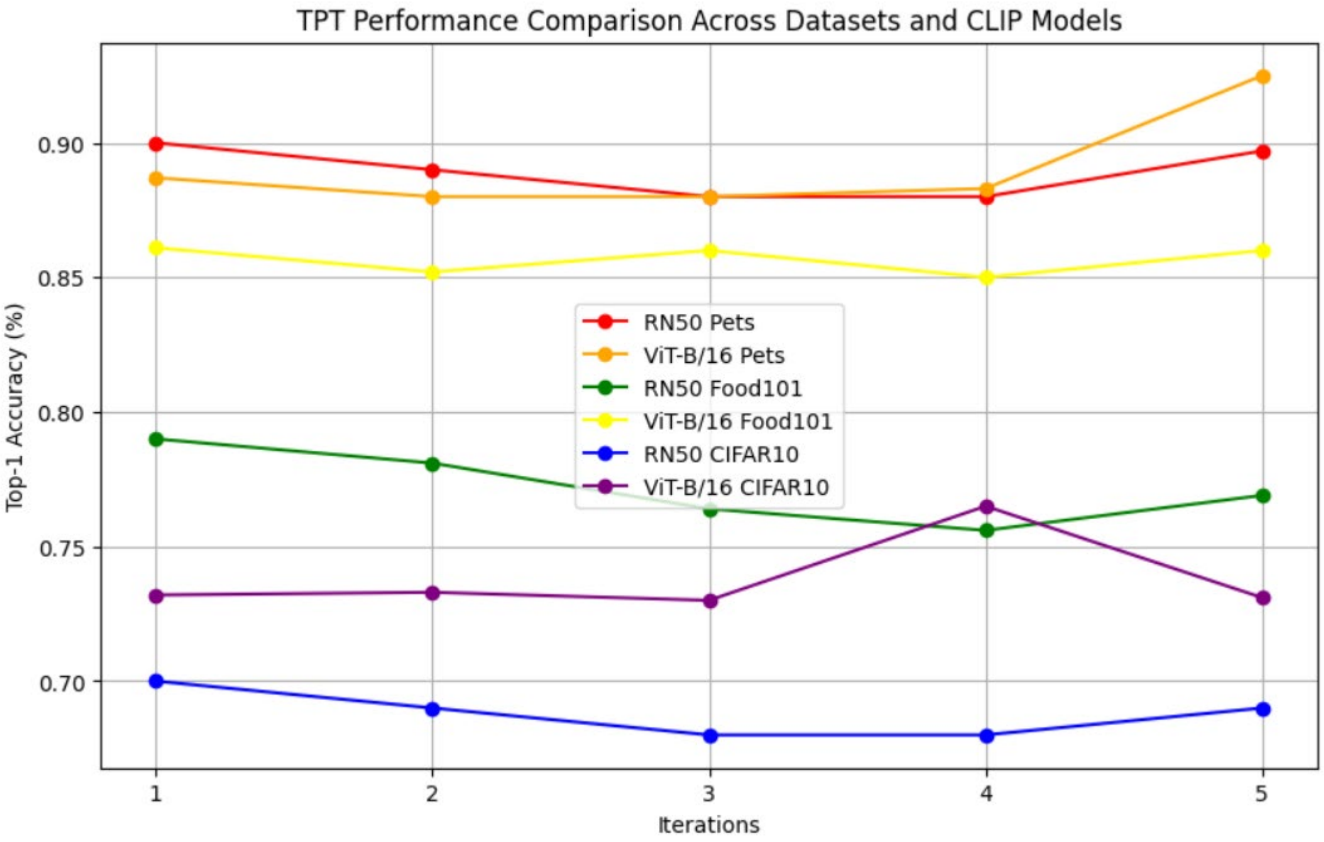
TPT + CoOP: Performance Analysis

TPT improves by around 5%

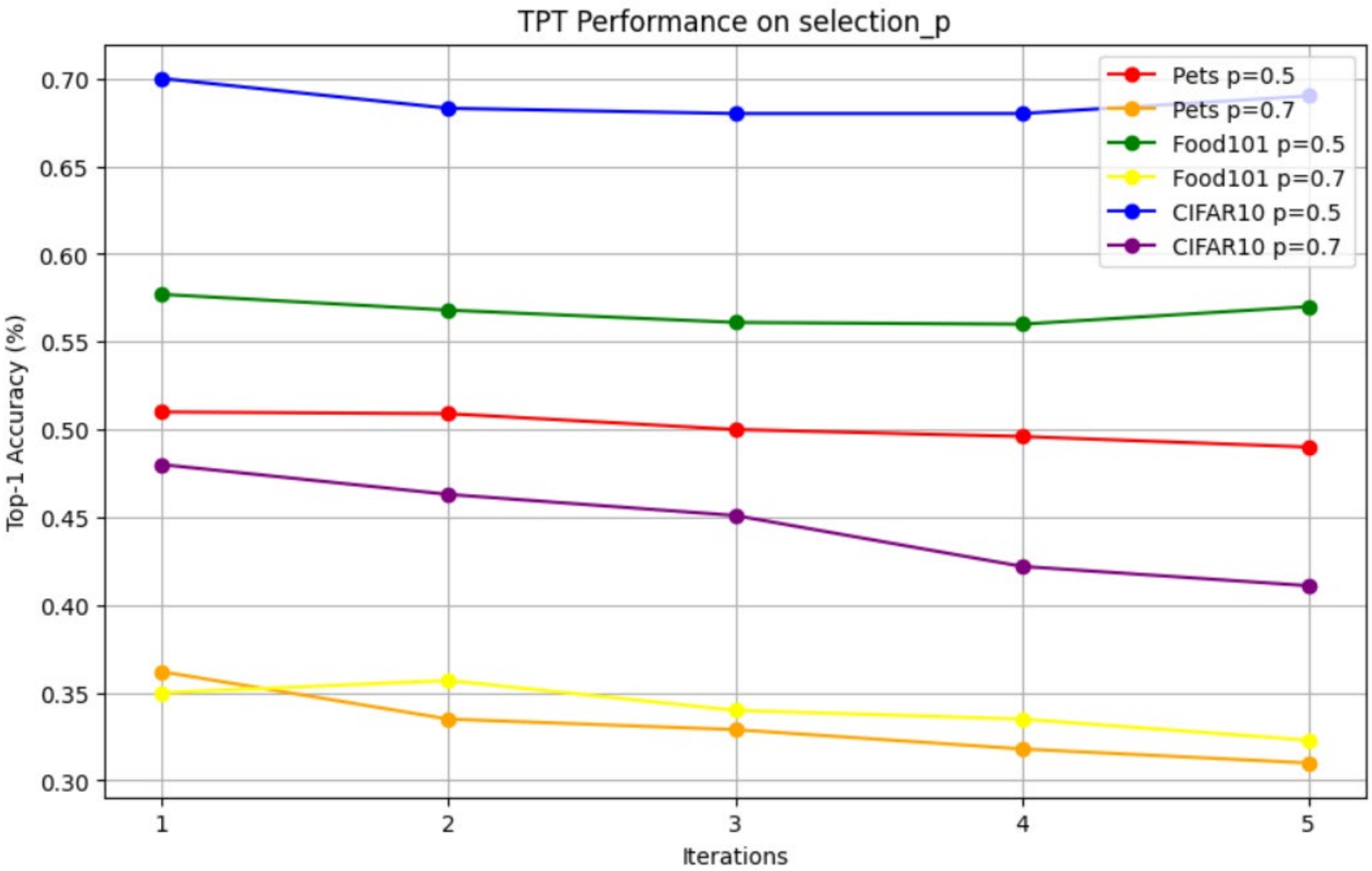


TPT + CoOP: Performance Comparison with ZSL-CLIP

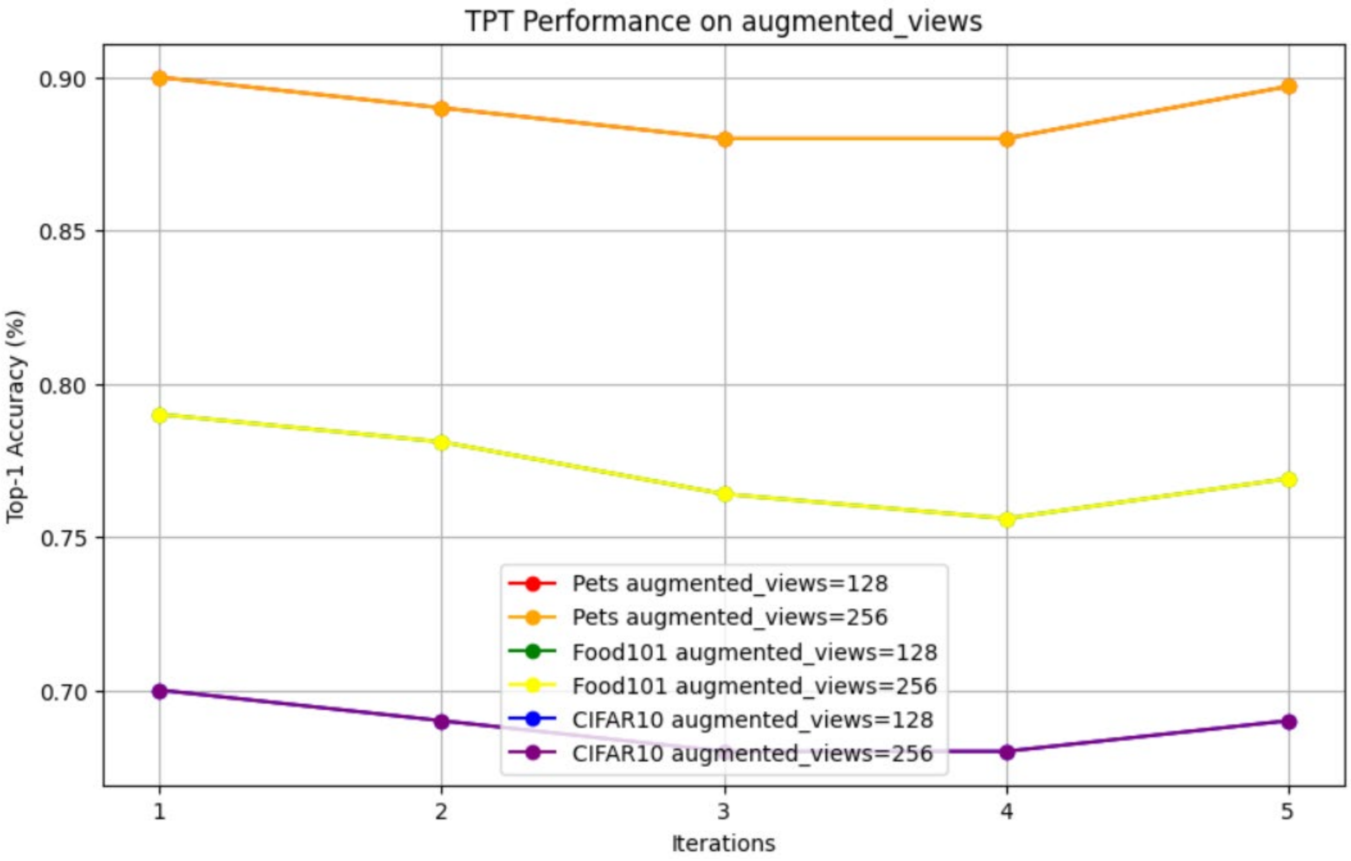
ViT-B/16 Performs Well



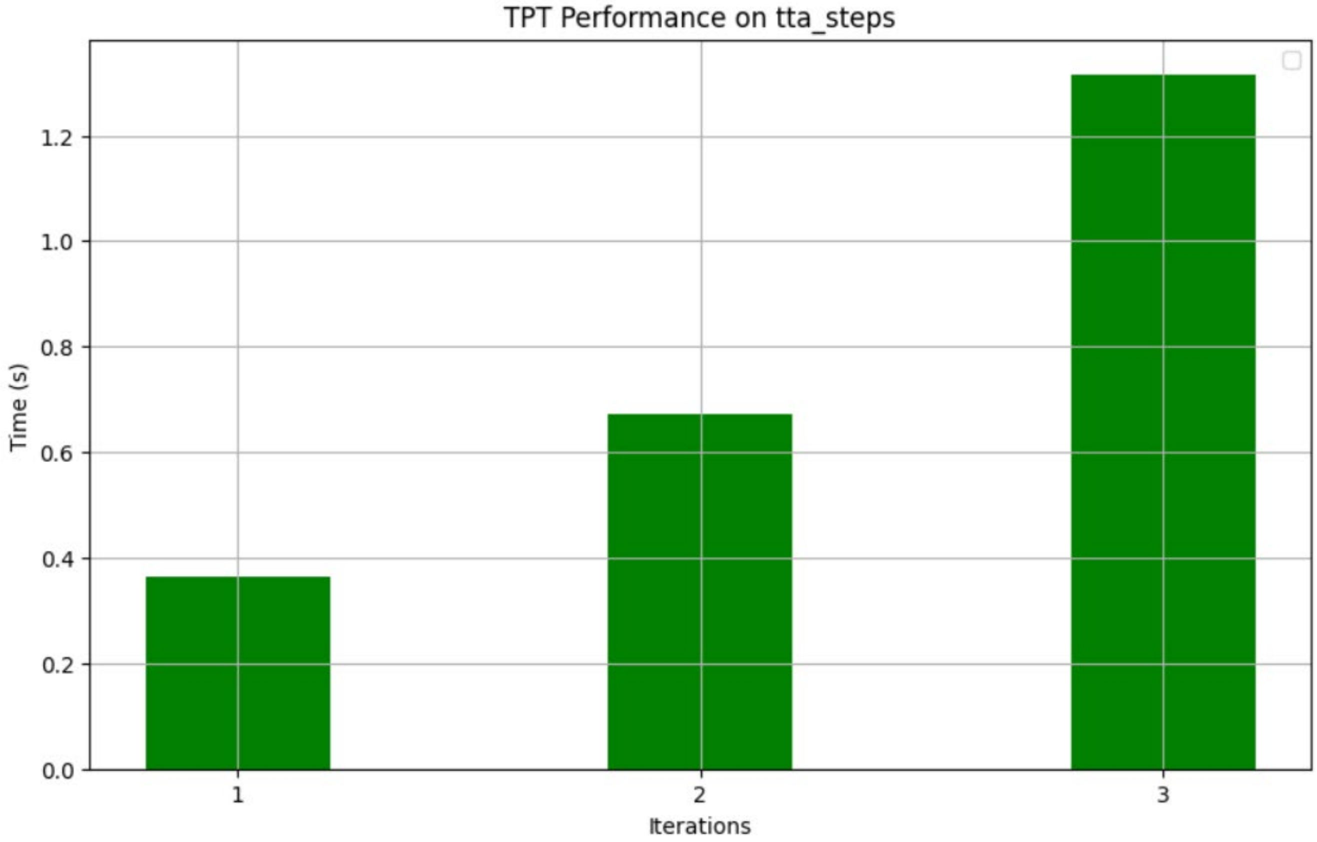
Hyperparameters Exp.: Playing with Augmentation



Hyperparameters Exp.: Playing with Augmentation



Hyperparameters Exp.: Playing with 'tta_steps/Iterations'

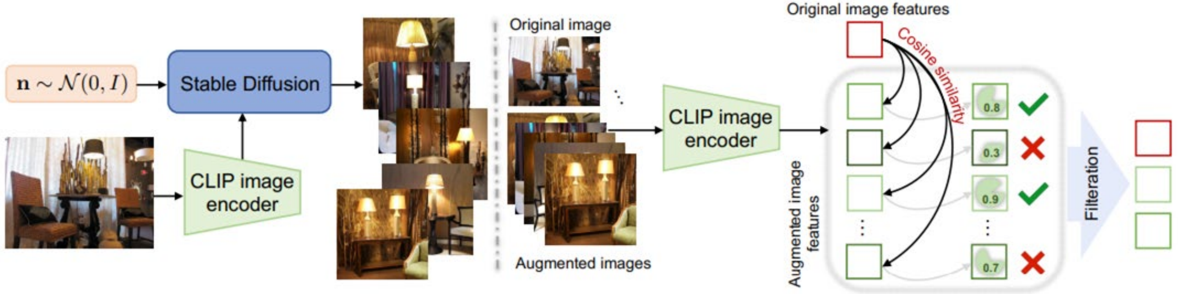


Thoughts on this TPT Analysis

- **Selection Probability ('selection_p'):** Optimal at 0.1, balances data confidence and performance.
- **Augmented Views ('augmented_views'):** Best at 64, ensures adaptability without overloading.
- **Test-Time Adaptation Steps ('tta_steps'):** Ideal at 1, more it, more time it will take and may cause overfitting.
- **Random Seed ('seed'):** Peak performance at 1, crucial for model initialization.
- **Batch Size ('batch_size'):** Optimal at 64, aligns adaptation capacity with training efficiency.
- Also, we see a major drop in performance during playing with Augmentation especially type/number of augmented views and using selection_p.

TPT Baselines: DiffTPT + CoOP

- **Pre-trained CLIP**
  - TextEncoder
  - ImageEncode
- **CoOp**
  - ClipTestTimeTuning
  - PromptLearner
- **AugmentationGenerator**
  - StableDiffusionImageVariationPipeline
- **Diff_tpt_classification**

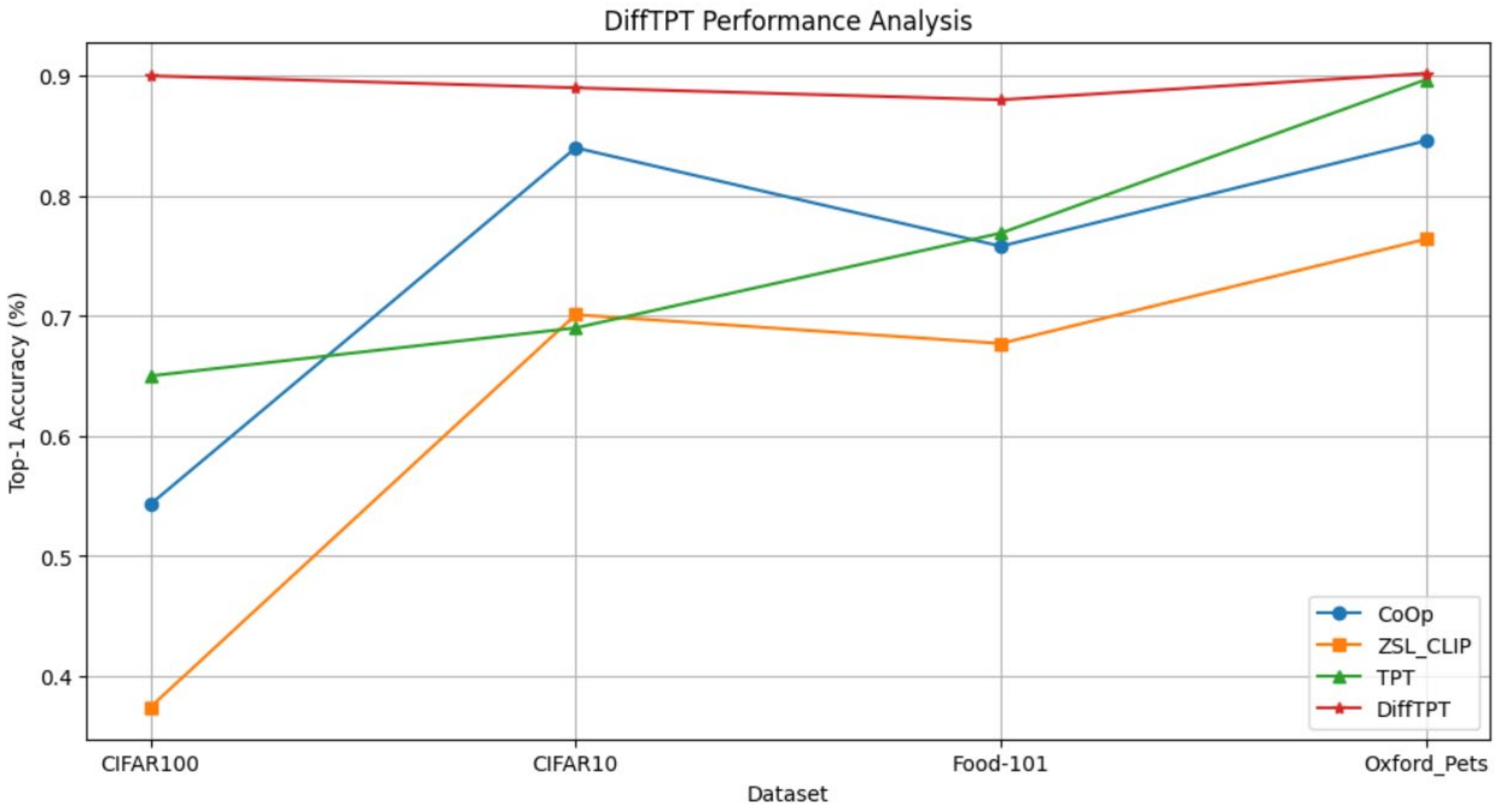


(a) Diverse augmentation with diffusion

(b) Cosine similarity filteration

```
model_name_path = "models/sd-image-variations-diffusers"
pipe = StableDiffusionImageVariationPipeline.from_pretrained(model
tform = transforms.Compose([transforms.ToTensor(), transforms.Resi
        (224, 224), interpolation=transforms.InterpolationMode.BIC
        antialias=False), transforms.Normalize([0.48145466, 0.4578
        [0.26862954, 0.26130258, 0.27577711])])
dataset_ = DatasetImageNetR(args.data_dir, tform)
dataloader = torch.utils.data.DataLoader(dataset_, batch_size=args
generate_images(pipe, dataloader, args)
```

DiffTPT Performance over other Models



Conclusions

We deepened our understanding on prompt learning with test-time adaption by:

- **Studying** several prompt learning methods and test-time adaption techniques drawn from the literature
- **Implementing** from scratch some of those methods
- **Assessing the effectiveness** of the implemented methods by replicating the paper's experiments
- **Visually exploring** in a 3D space the context vector representations of the hand-crafted prompts against the well learned ones