POUF: Prompt-Oriented Unsupervised Fine-tuning for Large Pre-trained Models

Korawat Tanwisuth [*1] Shujian Zhang [*1] Huangjie Zheng [1] Pengcheng He [2] Mingyuan Zhou [1]

Abstract

Through prompting, large-scale pre-trained models have become more expressive and powerful, gaining significant attention in recent years. Though these big models have zero-shot capabilities, in general, labeled data are still required to adapt them to downstream tasks. To overcome this critical limitation, we propose an unsupervised fine-tuning framework to directly fine-tune the model or prompt on the unlabeled target data. We demonstrate how to apply our method to both language-augmented vision and masked-language models by aligning the discrete distributions extracted from the prompts and target data. To verify our approach's applicability, we conduct extensive experiments on image classification, sentiment analysis, and natural language inference tasks. Across 13 image-related tasks and 15 language-related ones, the proposed approach achieves consistent improvements over the baselines. PyTorch code is available at `https://github.com/korawat-tanwisuth/POUF`.

1. Introduction

The pre-train and fine-tune paradigm has become a standard approach to solve many machine learning applications (Qiu et al., 2020; Du et al., 2022). In this paradigm, a model is first pre-trained on an extensive collection of datasets. While the model may see many examples during pre-training, the target dataset may contain unseen examples with new variations (Wenzel et al., 2022). To address the problem of distribution shift between the source and target domains, practitioners fine-tune the pre-trained model on different downstream tasks with task-specific parameters and objective functions. However, this fine-tuning stage generally needs labeled examples, which are expensive to acquire, to adapt the pre-trained model to a specific task.

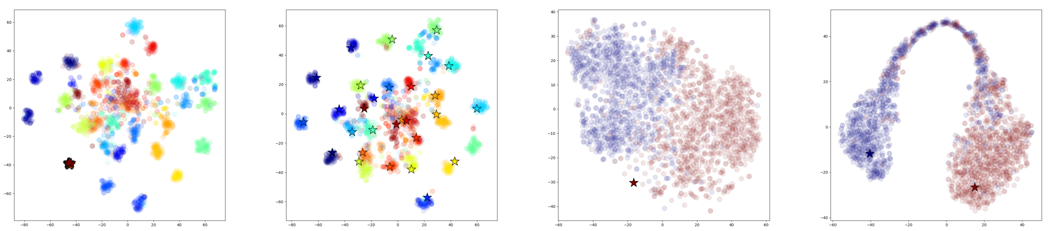
Without introducing task-specific parameters and objective functions for fine-tuning, recent foundation models, such as CLIP (Radford et al., 2021), ALIGN (Jia et al., 2021), and GPT-3 (Brown et al., 2020), leverage the power of language supervision during pre-training to perform zero-shot predictions through language prompting. For example, to make predictions on a new dataset, users only need to convert class names into textual prompts such as "a photo of a {CLASS}.". A prediction is then made by obtaining the prompt yielding the highest similarity score with a given image. Despite the capability to perform zero-shot predictions, the distribution shift problem still persists.

Recent methods focus on prompt engineering to adapt these models to downstream tasks. Prompt engineering refers to the process of finding the most appropriate prompt to allow a language model to solve the task at hand (Liu et al., 2021a). Many works (Wallace et al., 2019; Shin et al., 2020) focus on discrete prompt search (finding the prompt in the space of the vocabularies of the language model). Since the interpretability of the prompt is often not as important as the performance of the model, various works (Li & Liang, 2021; Lester et al., 2021; Zhou et al., 2022b;a) propose using continuous prompts that perform prompting in the embedding space of the model. Notably, Lester et al. (2021) and Zhou et al. (2022b) introduce continuous parameters in the word-embedding space and optimize them by maximizing the likelihood of the model on the labeled target data. While these approaches successfully adapt the pre-trained models, they still require a few annotated samples.

To overcome this limitation, we propose prompt-oriented unsupervised fine-tuning (POUF), a simple yet effective framework for fine-tuning pre-trained prompt-based large models with zero-shot capabilities, directly on the unlabeled target data. We formulate unsupervised fine-tuning as a process of minimizing the statistical distance between the empirical distribution of the textual prototypes and that of the target features. Our framework relies on language prompts to construct class prototypes or target features depending on the model type. For language-augmented vision models, we align the representations of class-specific language prompts,

[*]Equal contribution [1]The University of Texas at Austin [2]Microsoft Azure AI. Correspondence to: Mingyuan Zhou <mingyuan.zhou@mccombs.utexas.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).



(a) (b) (c) (d)

Figure 1: t-SNE visualizations of the embeddings of both the CLIP and RoBERTa models before and after applying POUF, where ($\star$) indicates textual prototypes while ($\cdot$) represents target features. Different colors correspond to different classes. **(a)** shows the features and prototypes of the pre-trained zero-shot CLIP model on the Office-31 (Amazon) dataset, while **(b)** shows those obtained after fine-tuning CLIP with POUF. **(c)** shows the features and prototypes of the pre-trained zero-shot RoBERTa model on the Subjectivity dataset, while **(d)** shows those obtained after fine-tuning RoBERTa with POUF. The prototypes of the zero-shot models are not well aligned with the target features as they are clustered around a single point far away from the target features for both CLIP and RoBERTa models. After adapting with POUF, the prototypes and features of their respective classes are well aligned.

which are class prototypes, with the target image features in the latent space. For masked-language models, we extract the masked-token representations from language prompts and align them with the textual prototypes from the decoder head of the language model. By aligning these distributions, the pre-trained model can better capture the variations in the target data. To this end, we utilize transport-based alignment and mutual-information maximization objective functions to align the latent representations. Our proposed method is compatible with both full model tuning and prompt tuning.

Our contributions include the following: **1)** We propose a prompt-oriented framework for fine-tuning pre-trained models with zero-shot capabilities, directly on the unlabeled target data. **2)** We illustrate how to formulate POUF under both language-augmented vision models and masked-language models and demonstrate its effectiveness in practical tasks, such as image classification, sentiment analysis, and natural language inference. **3)** We perform extensive ablation studies to justify the design decisions of our approach.

2. Prompt-Oriented Unsupervised Fine-tuning

Below we provide a simple recipe for fine-tuning zero-shot models on unlabeled target data for classification tasks. Our aim is to reduce the distribution shift between the data used to pre-train zero-shot models and the target data. Motivated by the observation that the class prototypes represent source domain information (data for pre-training), we propose to align them with the target representations in the latent space. We demonstrate how to formulate POUF for both language-augmented vision and masked-language models. Figure 1 provides motivating examples of the latent features learned using our method on different models. It is evident that before applying our method, the textual prototypes are not well aligned with target features, whereas after applying POUF, the prototypes and target features become well aligned. A schematic diagram of POUF is shown in Figure 2.

2.1. POUF for Language-Augmented Vision Models

Given an input image $\boldsymbol{x}$ from a target dataset $\mathcal{X}$, the CLIP model encodes it through an image encoder to obtain a latent representation as $\boldsymbol{f} = F(\boldsymbol{x})$. Each class name is included in a prompt template as an input, x^k_{prompt}, to the text encoder, $G(\cdot)$, to obtain textual representation $G(x^k_{prompt}) = \boldsymbol{w}_k$ for class k. Radford et al. (2021) suggest using the template as "a photo of a {CLASS}." in CLIP as it leads to better performance than using the class names alone. The prediction probabilities over classes are then given as

$$P(y = k \mid \boldsymbol{x}) = \frac{\exp\left(\cos\left(\boldsymbol{f}, \boldsymbol{w}_k\right)/T\right)}{\sum_{k'=1}^{K}\exp\left(\cos\left(\boldsymbol{f}, \boldsymbol{w}_{k'}\right)/T\right)}, \quad (1)$$

where T is a temperature parameter.

2.1.1. Textual-prototype construction

While the model has seen a large collection of data during pre-training, it may not capture the variation in the target data, leading to a sub-optimal performance at the test time without any adaptation. For example, the model may see an image of a polar bear on a white background during pre-training but fail to correctly identify another photo of a polar bear on an unseen background. If we regard the pre-training data as the source domain data, we can apply domain adaptation methods to bridge the domain gap.

Existing methods in domain adaptation align the distribution of source with that of the target data (Tzeng et al., 2014; Ganin & Lempitsky, 2015; Long et al., 2015; 2017; Tzeng et al., 2017; Long et al., 2018; Zhang et al., 2021b).

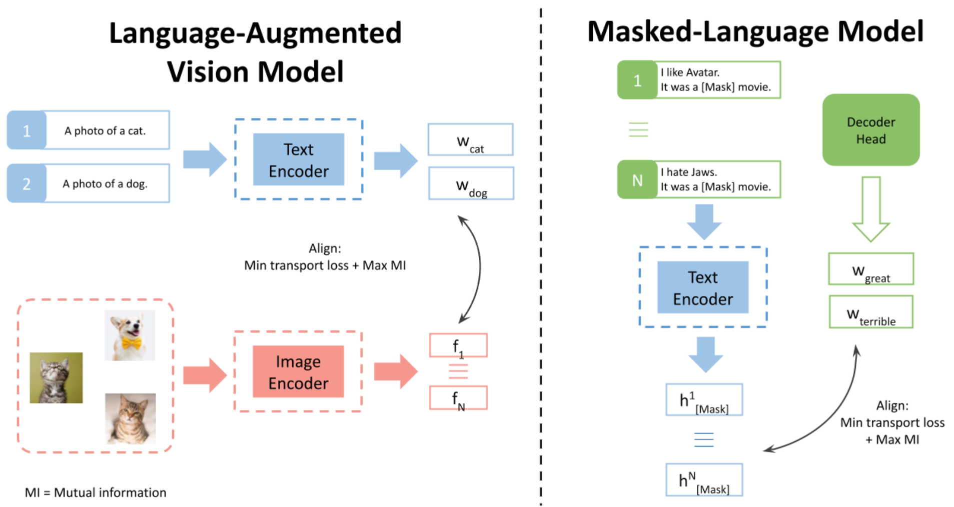


Figure 2: A schematic diagram of POUF for language-augmented vision models and masked-language models. Our method addresses the distribution shift problem by aligning the class prototypes with the target features. To this end, POUF minimizes the transport cost between the prototypes and target features while maximizing the mutual information between them. For language-augmented vision models, as shown on the left, the textual prototypes are the representations of the prompts. For masked language models, as shown on the right, the textual prototypes are extracted from the decoder head of the underlying language model.

However, this approach presents three challenges. **1)** Computational inefficiency: The amount of pre-training data is enormous. As an illustration, the CLIP model is trained with 400 million image and text pairs (Radford et al., 2021). **2)** Privacy issue: Access to the pre-training data can be restricted. **3)** Class mismatch: Pre-training data is likely to contain classes that are not present in the target data (Cao et al., 2018).

To overcome these challenges, we propose constructing a prototype for each class in the latent space to represent the source data (pre-training data). By using these prototypes to represent source images, the computation is now feasible as there are as many textual prototypes as classes, whose number is considerably smaller than the number of pre-training data. This strategy also solves the access issue of the pre-training data because we only need to come up with the class names to obtain these textual prototypes. Finally, the target users have control over which classes they want to classify, meaning that there will not be class mismatching.

Prior works use average latent feature (Snell et al., 2017; Pan et al., 2019; Yue et al., 2021) or learnable weight vectors (Saito et al., 2019; Tanwisuth et al., 2021; 2023) to construct prototypes. The former is computationally expensive as we need to perform multiple forward passings to construct reliable prototypes, while the latter is not applicable since we do not have labeled data. Unlike these approaches, we utilize the textual representations $G(x^k_{prompt}) = \boldsymbol{w}_k$, which represent each class in the latent space, as our prototypes. The textual representations will be close to the images of their respective categories. For example, the textual representation of the prompt "A photo of a cat." will be close to cat images. Thus, they represent the prototypes of the classes. To align the distribution of textual prototypes and target data, we first discuss how to construct these distributions and then present different options for distribution alignment.

2.1.2. Aligning Prototypes with Target Data

Denoting δ as the Dirac delta function, we express the distributions over textual representations and target data as

$$P = \textstyle\sum_{k=1}^K u_k \delta_{\boldsymbol{w}_k}, \quad Q = \textstyle\sum_{i=1}^N v_i \delta_{\boldsymbol{f}_i}, \tag{2}$$

where both $\boldsymbol{u} \in \mathbb{R}^K_+$ and $\boldsymbol{v} \in \mathbb{R}^N_+$ have non-negative elements that sum to one. Unless specified otherwise, we assume $u_k = 1/K$ for $k = 1, \dots, K$ and $v_i = 1/N$ for $i = 1, \dots, N$ in what follows. To align distributions P and Q, we present two alternative methods and discuss the benefits and drawbacks of each.

Comparing Two Distributions. Our objective is to align the distribution of textual prototypes and that of the target data. To achieve this, we need to quantify the difference between two discrete distributions.

One principled approach is to consider Optimal Transport (OT) (Villani, 2008). Given two discrete distributions P and Q shown in (2), the OT between them is defined as

$$\mathrm{OT}(P, Q) := \min_{\mathbf{T} \in \Pi(\boldsymbol{u}, \boldsymbol{v})} \operatorname{Tr}\left(\mathbf{T}^\top \boldsymbol{C}\right), \tag{3}$$

where $\mathbf{T} \in \mathbb{R}^{K \times N}_+$ is a doubly stochastic transport matrix such that $\Pi(\boldsymbol{u}, \boldsymbol{v}) = \left\{\mathbf{T} \mid \mathbf{T}\mathbf{1}_N = \boldsymbol{u}, \mathbf{T}^\top \mathbf{1}_K = \boldsymbol{v}\right\}$, T_{ij} is the transport probability between x_i and y_j, $\boldsymbol{C} \in \mathbb{R}_+^{K\times N}$ is the transport cost matrix with $C_{ki} = c\left(\boldsymbol{w}_k, \boldsymbol{f}_i\right)$, and $\mathrm{Tr}(\cdot)$ denotes the matrix trace. However, OT can be sensitive to outliers due to the two marginal constraints (Chizat et al., 2018). Moreover, solving (3) without any relaxation is a linear programming problem, which has a complexity of $\mathcal{O}(M^3 \log M)$, where M denotes the size of the mini-batch of the target examples (Mérigot & Oudet, 2016). This complexity makes it unfitting for deep learning applications. Cuturi (2013) introduces Sinkhorn divergence, an entropy-regularized OT, to speed up the optimization. While it significantly lowers the computation, it still relies on a two-stage optimization strategy to apply to deep learning models: solving for the transport plan and then updating the network.

To overcome this computational issue while being deep-learning friendly, Zheng & Zhou (2021) have introduced the Conditional Transport (CT) framework. Instead of solving for the OT plan, CT defines a bi-directional transport plan, whose two directions correspond to softmax probabilities normalized across the classes and across the data, respectively. This choice makes it easy to integrate it into the deep-learning framework while also effectively aligning the two distributions (Tanwisuth et al., 2021; Wang et al., 2022). The CT between two discrete distributions can be written as

$$\mathrm{CT}(P, Q) := \mathcal{L}_{t\to\boldsymbol{w}} + \mathcal{L}_{\boldsymbol{w}\to t}, \tag{4}$$

where $\mathcal{L}_{t\to\boldsymbol{w}}$ is the transport cost from target features to textual prototypes and $\mathcal{L}_{\boldsymbol{w}\to t}$ is the transport cost in the opposite direction. The transport cost from target features to prototypes can be expressed as

$$\begin{aligned}\mathcal{L}_{t\to\boldsymbol{w}} &= \mathbb{E}_{\boldsymbol{x}_i\sim\mathcal{X}}\mathbb{E}_{\boldsymbol{w}_k\sim\pi_{\boldsymbol{\theta}}(\boldsymbol{w}_k\,|\,\boldsymbol{f}_i)}\left[c(\boldsymbol{w}_k, \boldsymbol{f}_i)\right] \\ &= \mathbb{E}_{\boldsymbol{x}_i\sim\mathcal{X}}\left[\sum_{k=1}^{K} c(\boldsymbol{w}_k, \boldsymbol{f}_i)\pi_{\boldsymbol{\theta}}(\boldsymbol{w}_k\,|\,\boldsymbol{f}_i)\right],\end{aligned} \tag{5}$$

where $c(\boldsymbol{w}_k, \boldsymbol{f}_i) = 1 - \cos(\boldsymbol{w}_k, \boldsymbol{f}_i)$ is the point-wise transport cost, $\pi_{\boldsymbol{\theta}}(\boldsymbol{w}_k\,|\,\boldsymbol{f}_i) = \frac{p(\boldsymbol{w}_k)\exp(\boldsymbol{w}_k^T\boldsymbol{f}_i)}{\sum_{k'=1}^{K} p(\boldsymbol{w}_{k'})\exp(\boldsymbol{w}_{k'}^T\boldsymbol{f}_i)}$, and $p(\boldsymbol{w}_k)$ is the prior probability of prototype k.

If we regard the prototypes as the modes of the target distribution, this transport direction has a mode seeking effect, meaning that target features will be close to some prototypes. This direction alone, however, could lead to a degenerate solution where target features are close to only a few prototypes (mode collapse). To counteract this potential issue, CT introduces the transport cost in the opposite direction.

$$\begin{aligned}\mathcal{L}_{\boldsymbol{w}\to t} &= \mathbb{E}_{\{\boldsymbol{x}_i\}_{i=1}^N\sim\mathcal{X}}\mathbb{E}_{\boldsymbol{w}_k\sim p(\boldsymbol{w}_k)}\mathbb{E}_{\boldsymbol{f}_i\sim\pi_{\boldsymbol{\theta}}(\boldsymbol{f}_i\,|\,\boldsymbol{w}_k)}\left[c(\boldsymbol{w}_k, \boldsymbol{f}_i)\right] \\ &= \mathbb{E}_{\{\boldsymbol{x}_i\}_{i=1}^N\sim\mathcal{X}}\left[\sum_{k=1}^{K} p(\boldsymbol{w}_k)\sum_{i=1}^{N} c(\boldsymbol{w}_k, \boldsymbol{f}_i)\pi_{\boldsymbol{\theta}}(\boldsymbol{f}_i\,|\,\boldsymbol{w}_k)\right],\end{aligned} \tag{6}$$

where $\pi_{\boldsymbol{\theta}}(\boldsymbol{f}_i\,|\,\boldsymbol{w}_k) = \frac{\exp(\boldsymbol{w}_k^T\boldsymbol{f}_i)}{\sum_{i'=1}^{N}\exp(\boldsymbol{w}_k^T\boldsymbol{f}_{i'})}$. In contrast to $\mathcal{L}_{t\to\boldsymbol{w}}$ which has a mode-seeking effect, this transport direction has a mode-covering effect, meaning that each prototype will get target features assigned close to it. Thus, we can avoid the possible mode collapse. We note the definition of modes differs from that in Zheng & Zhou (2021), in which one can find a more detailed discussion regarding the mode-covering and mode-seeking behaviors of CT.

The prior term, $p(\boldsymbol{w}_k)$, can be set to $\frac{1}{K}$ (a uniform distribution over the classes). However, this approach is not ideal if the class proportions are not balanced. To address the potential class-imbalanced problem, we can learn the prior term from the target data. We discuss an estimation strategy in Appendix G.

Compared to OT, CT has a lower complexity of $\mathcal{O}(d_f MK)$, where d_f is the dimension of the target feature, M denotes the mini-batch size, and K refers to the number of categories. The optimization of CT is also end-to-end as both the cost and transport plan are parameterized by deep neural networks.

While POUF is compatible with both OT and CT, we empirically find that CT leads to better performance. We show this result in the ablation study in Section 4.3. The CT-based transport cost is thus expressed as:

$$\mathcal{L}_{transport}\left(F, G; \mathcal{X}\right) = CT(P, Q). \tag{7}$$

2.1.3. Decision-boundary refinement

Because of the distribution shift problem, some samples may lie close to decision boundaries, meaning that they are far away from the prototypes. Motivated by the cluster assumption (Grandvalet et al., 2005), which states that decision boundaries should not cross high-density data regions, we propose incorporating another module to refine the predictions of the model on the target data. If the violation of the cluster assumption is minimized, the input examples will be close to the textual prototypes in the latent space.

Mutual-Information Maximization. Many existing works in domain adaptation minimize the conditional entropy of the target predictions (Vu et al., 2019; Saito et al., 2019; 2020; Wang et al., 2020). However, this has a major shortcoming since this objective alone could lead to a degenerate solution (all samples are clustered around one prototype) (Morerio et al., 2017; Wu et al., 2020). To overcome this issue, we utilize the information maximization objective (Krause et al., 2010; Shi & Sha, 2012; Liang et al., 2020). This objective adds a regularization term to conditional entropy minimization. This regularization encourages the model to maximize the marginal entropy of the label distribution (Zhang et al., 2021c), making the predictions globally diverse while individually certain. The mutual information objective has the following form:

$$\begin{aligned}\mathcal{L}_{mi}(F,G;\mathcal{X}) &= -[H(\mathcal{Y}) - H(\mathcal{Y} \mid \mathcal{X})] \\ &= -[h(\mathbb{E}_{\boldsymbol{x}\in\mathcal{X}} p(y|\boldsymbol{x})) - \mathbb{E}_{\boldsymbol{x}\in\mathcal{X}} h(p(y|\boldsymbol{x}))], \end{aligned} \quad (8)$$

where $H(\mathcal{Y})$ and $H(\mathcal{Y} \mid \mathcal{X})$ denote the marginal entropy and conditional entropy of the text categories $\mathcal{Y}$, respectively, and $h(p) = -\sum_i p_i \log p_i$.

Putting it all together, we write the final loss function as

$$\mathcal{L}_{LVM}(F,G;\mathcal{X}) = \mathcal{L}_{transport}(F,G;\mathcal{X}) + \lambda\mathcal{L}_{mi}(F,G;\mathcal{X}), \quad (9)$$

where λ is a hyper-parameter controlling the weight of the mutual-information objective. We then update both the image encoder, F, and the text encoder, G, through gradient backpropagation of this loss function. Alternatively, we can introduce soft-prompt parameters in the word embedding space and update only these parameters while keeping the model parameters fixed. We discuss this alternative design consideration in more detail in Section 2.3.

2.2. POUF for Masked-Language Models

In a conventional setting, to fine-tune masked-language models such as BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) on a task, one first converts input x_1 to a sequence of token $\tilde{x}$. The language model, M, then maps $\tilde{x}$ to a sequence of hidden vectors $\{h_k \in \mathbb{R}^d\}$. One then introduces a task-specific head $\mathbf{W}_0 \in \mathbb{R}^{|\mathcal{Y}|\times d}$, which is randomly initialized, to classify the hidden representations through softmax($\mathbf{W}_0\mathbf{h}_{[\text{CLS}]}$) into a class in $\mathcal{Y}$. Since $\mathbf{W}_0$ does not capture any semantic meaning, it is trained together with the parameter of the pre-trained model by minimizing the negative log probability over the labels $y \in \mathcal{Y}$. The drawbacks of this approach are that one needs to not only introduce additional task-specific parameters but also utilize labeled examples to fine-tune the model.

An alternative way to fine-tune the language model on this task is to formulate it as a masked-language modeling problem, mimicking the pre-training process (Schick & Schütze, 2020; Gao et al., 2020). During this process, one first maps the labels to the words in the vocabulary. As an example, for a sentiment classification task, a positive label $(+)$ can be mapped to the word "great" whereas a negative label $(-)$ can be mapped to the word "terrible". The language model is then tasked to fill the mask token with the label words ("great" or "terrible"). Given an input x_1, one can then construct a prompt from the text input as:

$$x_{prompt} = \text{[CLS] } x_1 \text{ It was [MASK]. [SEP].}$$

The prompt can be manually or automatically generated, so long as the input contains a mask token. Having a mask token in the input allows one to treat the problem as a masked language modeling task, *i.e.*, directly modeling the probability of each label as that of the corresponding word in the language model's vocabulary as

$$\begin{aligned} P(y \mid x_{\text{in}}) &= p([\text{ MASK }] = \mathcal{M}(y) \mid x_{\text{prompt}}) & (10) \\ &= \frac{\exp\left(\mathbf{w}_{\mathcal{M}(y)} \cdot \mathbf{h}_{[\text{MASK}]}\right)}{\sum_{y'\in\mathcal{Y}} \exp\left(\mathbf{w}_{\mathcal{M}(y')} \cdot \mathbf{h}_{[\text{MASK}]}\right)}, & (11) \end{aligned}$$

where $\mathbf{w}_v$ corresponds to the weight vector of the vocabulary $v \in \mathcal{V}$ and $\mathbf{h}_{[\text{MASK}]}$ denotes the hidden representation of the mask token, and $\mathcal{M}: \mathcal{Y} \to \mathcal{V}$ represents the mapping from the set of labels to the words in the vocabulary.

2.2.1. Textual-prototype construction

To apply POUF to masked-language models, we first construct the textual prototypes, which contain information about the classes. Unlike language-augmented vision models' prompts which encode class information, Masked-language models' prompts contain information about the input examples. Thus, we cannot use them as our class prototypes. To construct textual prototypes, we propose using the weight of the last layer of the decoder head, which is tasked to predict the mask tokens during pre-training. These weights capture the meanings of the individual words in the vocabulary of the language model. For example, $\boldsymbol{w}_{\text{great}}$ will be close to the masked-token representations of the sentences which contain the word "great". We then use the alignment module to align

$$P = \sum_{y\in\mathcal{Y}} \frac{1}{|\mathcal{Y}|}\delta_{\boldsymbol{w}_{\mathcal{M}(y)}} \text{ and } Q = \sum_{i=1}^{N} \frac{1}{N}\mathbf{h}^i_{[\text{MASK}]}. \quad (12)$$

The information maximization objective is directly computed via the predictive distribution of the language model.

Putting it all together, we write the final loss function as

$$\mathcal{L}_{MLM}(M;\mathcal{X}) = \mathcal{L}_{transport}(M;\mathcal{X}) + \lambda\mathcal{L}_{mi}(M;\mathcal{X}), \quad (13)$$

where λ is a hyper-parameter controlling the weight of the mutual-information objective.

2.3. Design Considerations

As briefly discussed in the previous section, we could perform full model tuning or prompt tuning. **1)** *Full-model tuning* refers to updating the parameters of both the text and image encoders for language-augmented vision models and the text encoder for masked-language models. This design generally leads to better performance but requires more resources for tuning. **2)** *Prompt tuning (Lester et al., 2021)* refers to introducing additional tunable parameters in the input embedding space. Although using fewer resources for fine-tuning, it may not necessarily lead to optimal performance.

We now give a more formal description of prompt tuning. Given a sequence of n tokens, $\{x_1, x_2, \ldots, x_n\}$, a

Table 1: Data and model requirements for different transfer learning settings. POUF requires neither source data nor target labels.

Setup	Source data	Target data	Parameter update
Fine-tuning	N/A	Labeled	Full model
Domain adaptation	Labeled	Unlabeled	Full model
Prompt tuning	N/A	Labeled	Prompt
POUF	N/A	Unlabeled	Prompt/full model

transformer-based language model maps this sequence to a matrix $\mathbf{X}_h \in \mathbb{R}^{n\times h}$ where h is the hidden dimension. Instead of tuning the model parameter, prior works (Lester et al., 2021; Zhou et al., 2022b) introduce soft-prompt parameter $\mathbf{P}_h \in \mathbb{R}^{p\times h}$ where p is the prompt length. The embedded prompt and input are then concatenated to form a matrix $[P_h; X_h] \in \mathbb{R}^{(p+n)\times h}$. This matrix then goes through the encoder, but only the parameters of the soft-prompts $\mathbf{P}_h$ are updated.

3. Related Work

We summarize how POUF differs from related work in Table 1 and provide more details below.

Learning under Distribution Shift. Traditional domain adaptation methods jointly optimize on labeled source data and unlabeled target data (Tzeng et al., 2014; Ganin & Lempitsky, 2015; Long et al., 2015; 2017; Tzeng et al., 2017; Long et al., 2018; Zhang et al., 2021b;d). The requirement for labeled source data limits their use in many applications. POUF, however, can directly adapt to the target domain data, greatly increasing its general applicability. This also means that the performance of domain adaptation methods heavily depends on the source dataset, whereas POUF is source domain agnostic. Another related line of work is “source-free” domain adaptation (Liang et al., 2020; Li et al., 2020b; Kundu et al., 2020b;a; Kurmi et al., 2021). In this paradigm, the pre-trained model is adapted in two steps. First, it is fine-tuned on labeled source data. Then, it is adapted to the target data without accessing the source data. In this sense, they are not completely source-free. Different from this line of work, POUF leverages textual prototypes in models with zero-shot capabilities to adapt to the target data directly, bypassing the fine-tuning step on the labeled source data.

Prompt-based Learning. For a full review of this topic, we refer the readers to the survey by Liu et al. (2021a). Several works that focus on prompt tuning can be broadly classified into two categories: hard and soft prompt tuning. Soft prompt tuning (Li & Liang, 2021; Lester et al., 2021; Zhou et al., 2022b;a) introduces continuous parameters in the word embedding space and tunes these parameters instead of the full model, whereas hard prompt tuning (Wallace et al., 2019; Shin et al., 2020; Zhang et al., 2022) searches for discrete tokens in the vocabulary of the language model to optimize the performance of the model. These works focus on prompt design and engineering and rely on labeled data. Unlike these methods, ours is unsupervised, making it more generally applicable. However, as we show in the experiments, our work is complementary to these approaches since we can apply these methods to obtain better prototypes and then further fine-tune using our framework. Recently, Huang et al. (2022) propose a method for unsupervised prompt tuning for language-augmented vision models. Different from this work, our work provides a general framework for both language-augmented vision and masked-language models.

Prototype-based Learning. The idea of using prototypes has been studied in few-shot classification (Snell et al., 2017; Guo et al., 2022), self-supervised learning (Asano et al., 2019; Caron et al., 2020; Li et al., 2020a), and domain adaptation (Pan et al., 2019; Kang et al., 2019; Liang et al., 2020; Tanwisuth et al., 2021). In these works, the prototypes are defined as the average latent features or learnable weight vectors. Different from them, we propose using textual representations to represent our prototypes. This is the key idea of our method as it allows us to adapt directly to the target data.

4. Experiments

We evaluate POUF on both language-augmented vision and masked-language models and compare it to a rich set of baselines in various vision and language modeling tasks.

4.1. POUF for Language-Augmented Vision Models

Datasets. **1)** *Office-31 (Saenko et al., 2010)* contains 4,652 images with 31 classes from three domains: Amazon (A), Webcam (W), and DSLR (D). **2)** *Office-Home (Venkateswara et al., 2017)* has 15,500 images with 65 classes from four domains: Artistic images (Ar), Clip art (Cl), Product images (Pr), and Real-world (Rw), making it more challenging than Office-31. **3)** *DomainNet (Peng et al., 2019)*, a large-scale dataset, consists of 569,010 images with 345 categories from six domains: Clipart, Infograph, Painting, Quickdraw, Real, and Sketch. More details on the datasets can be found in Appendix D.

Baselines. **1)** *Clip (zero-shot) (Radford et al., 2021)* refers to using the CLIP model to perform zero-shot predictions on the target dataset. **2)** *Tent (Wang et al., 2020)* refers to adapting the model with entropy minimization before predicting on the target dataset. We note that we have slightly modified Tent since the vision transformer (ViT) architecture does not have modulation parameters (batch normalization). Thus, we introduce soft-prompt parameters and update them instead. **3)** *Unsupervised prompt learning (UPL) (Huang et al., 2022)* refers to using the top-K confident predictions for each class as pseudo labels. After obtaining the pseudo labels, we train the soft-prompt parameters with the cross-entropy loss. For a fair comparison, we do not use model ensemble as done in Huang et al. (2022). To help understand the gap between unsupervised methods and supervised ones, we also provide the results of a representative few-shot learning method, **4)** *Context Optimization (Zhou et al., 2022b)* (CoOp). We note that this baseline should not be directly compared to the unsupervised ones, but rather serves as a reference to understand how much labeled data can help boost performance.

Implementation Details. We build our method using the open-source CLIP codebase (Radford et al., 2021) and TLlib transfer learning library (Jiang et al., 2022). For all experiments, we adopt the ViTB-16 for the image encoder and the default transformer from the CLIP paper for the text encoder. All the unlabeled target samples are used for fine-tuning. The learning rate schedule is set to $\eta_{\text{iter}} = \eta_0(1+\gamma\text{iter})^{-\alpha}$, where η_0 is the initial learning rate. We adopt the following default hyper-parameters: $\gamma = 2\text{e}{-4}$, and $\alpha = 0.75$. We set $\eta_0 = 5\text{e}{-7}$ for all experiments except for prompt tuning on Office-31 where $\eta_0 = 1\text{e}{-3}$. We use a mini-batch SGD with a momentum of 0.9 and a batch size of 96 for Office-31 and Office-Home and 16 for DomainNet. The weight of the mutual-information objective, λ, is set to 0.3 for all experiments.

Main Results for Language-Augmented Vision Models. We conduct systematic experiments on 13 image tasks with varying data sizes. In each experiment, we fine-tune the CLIP model with unlabeled target images. Table 2 exhibits the results of POUF and the baselines. Tent and UPL both achieve consistent gains over the zero-shot CLIP model on both the Office-31 and Office-Home datasets. However, on the more challenging DomainNet, both baselines suffer from negative transfers. By contrast, POUF improve the CLIP model by 11.8%, 3.7%, and 3.6% on the Office-31, Office-Home, and DomainNet datasets, respectively. The consistent improvements illustrate that POUF is an effective strategy to fine-tune the model with unlabeled target data. Moreover, POUF is compatible with both prompt tuning and model tuning as evident in the performance gains. Prompt tuning yields slightly worse performance than model tuning, as we expected. However, its memory footprint is significantly lower. We provide parameter and run-time analyses (Fan et al., 2020; Zhang et al., 2021a) of the two approaches in Appendix B.

POUF with model tuning outperforms the few-shot learning method (CoOp) in 5 out of 13 tasks. This result illustrates that there is still a gap between the supervised and unsupervised methods. However, the benefit of a large amount of unlabeled data can sometimes outweigh the benefit of a small amount of labeled data.

To understand the generalization ability of the model after adapting with POUF, we further conduct an unseen class experiment in Appendix E. POUF yields improved performance for the seen class but slightly weaker generalization for the unseen class. This outcome highlights a minor trade-off between the model's specialization and generalization. However, if target users have access to unseen class names, training POUF with both seen and unseen class text prototypes results in superior performance compared to the zero-shot model for both seen and unseen classes. See Appendix E for a detailed analysis.

4.2. POUF for Masked-Language Models

Datasets. We perform experiments on 15 English-language tasks, 8 of which are single-sentence tasks and 7 of which are sentence-pair tasks. These datasets include popular classification tasks such as SST-5, MR, CR, MPQA, Subj, and TREC as well as the GLUE benchmark (Wang et al., 2018). We provide a table of different tasks in Appendix D. Given an input sentence, the goal of the single-sentence task is to predict the label. E.g., given a movie review, we need to predict if it is positive or negative. Similarly, the aim of the sentence-pair task is to predict the relationship between a pair of input sentences. As an illustration, given a premise and a hypothesis, we need to predict whether the hypothesis is an entailment, neutral, or contradiction of the premise.

Baselines. We directly take the baselines and results from Gao et al. (2020). **1)** *Majority* denotes the proportion of the majority class in the data. **2)** *RoBERTa-large (zero-shot)* refers to using prompt-based prediction by RoBERTa-large for zero-shot predictions. **3)** *Few-shot fine-tuning* refers to standard fine-tuning with few-shot examples. **4)** *"GPT-3" in-context learning* refers to prompt-based predictions with augmented context (32 randomly sampled demonstrations). The model is still RoBERTa-large (not GPT-3). **5)** *LM-BFF (man)* refers to the complete method by Gao et al. (2020), where the prompt is manually selected using the template.

Implementation Details. We follow the experimental protocols in Gao et al. (2020). Specifically, for each task, the data is split into $\mathcal{D}_{train}$, $\mathcal{D}_{dev}$, and $\mathcal{D}_{test}$. The authors tune the hyper-parameters on $\mathcal{D}_{dev}$ and report the performance of the model on $\mathcal{D}_{test}$. We take the same hyper-parameters as the original paper: batch size $= 8$, learning rate $= 1e-5$, training steps $= 1,000$, and few-shot examples per class $= 16$. The weight of the mutual-information objective, λ, is set as 0.6 and the weight for the transport cost is set as one for all experiments except for the unsupervised setting on the MRPC and QQP tasks. In these two tasks, the weights are set to $1\text{e}{-4}$ for the mutual-information objective and to $1\text{e}{-2}$ for the transport cost. We validate the model's performance every 100 steps on $\mathcal{D}_{dev}$ and take the best validated checkpoint for the final evaluation on $\mathcal{D}_{test}$.

Table 2: Accuracy (%) on three different datasets for methods based on CLIP.

(a)

Category	Methods	Office-31				Office-Home				
		A	D	W	Avg	Ar	Cl	Pr	Rw	Avg
Unsupervised	CLIP (zero-shot) (Radford et al., 2021)	79.0	77.5	74.7	77.1	82.7	68.1	89.1	89.8	82.4
	Tent (Wang et al., 2020)	81.5(0.1)	80.7(0.7)	82.8(0.2)	81.7	83.2(0.1)	67.8(0.2)	91.9(0.1)	90.4(0.1)	83.3
	UPL (Huang et al., 2022)	81.4(0.1)	82.6(0.1)	83.6(0.3)	82.5	83.3(0.1)	67.7(0.1)	91.5(0.1)	90.7(0.1)	83.3
	POUF (prompt tuning)	83.6(0.5)	89.9(2.5)	90.6(2.3)	88.0	83.7(0.1)	71.2(0.1)	91.4(0.1)	90.8(0.1)	84.3
	POUF (model tuning)	**84.4(0.4)**	**91.1(2.1)**	**91.3(2.5)**	**88.9**	**86.2(0.4)**	**73.8(0.3)**	**92.7(0.1)**	**91.7(0.1)**	**86.1**
Few-shot	CoOp (Zhou et al., 2022b)	83.7(2.1)	98.8(0.6)	98.2(0.8)	93.6	85.6(0.4)	73.2(0.2)	92.7(0.1)	91.6(0.1)	85.8

(b)

Category	Methods	DomainNet						
		C	I	P	Q	R	S	Avg
Unsupervised	CLIP (zero-shot) (Radford et al., 2021)	70.9	48.2	65.9	14.0	83.6	63.6	57.7
	Tent (Wang et al., 2020)	71.4(0.2)	47.8(0.5)	66.2(0.1)	14.2(0.1)	83.9(0.1)	64.1(0.2)	57.9
	UPL (Huang et al., 2022)	71.7(0.1)	47.5(0.4)	66.3(0.2)	14.4(0.3)	83.8(0.1)	64.3(0.1)	58.0
	POUF (prompt tuning)	72.8(0.1)	53.1(0.3)	68.6(0.1)	15.9(0.1)	84.4(0.0)	66.2(0.1)	60.2
	POUF (model tuning)	**73.8(0.1)**	**55.7(0.3)**	**68.6(0.1)**	**18.8(0.7)**	**84.6(0.0)**	**66.4(0.1)**	**61.3**
Few-shot	CoOp (Zhou et al., 2022b)	75.3(0.2)	55.7(0.2)	71.9(0.2)	20.7(0.2)	83.4(0.1)	67.4(0.1)	62.9

Table 3: Results on various datasets for methods based on RoBERTa-large.

Category	Methods	SST-2 (acc)	SST-5 (acc)	MR (acc)	CR (acc)	MPQA (acc)	Subj (acc)	TREC (acc)	CoLA (Matt.)
Unsupervised	Majority	50.9	23.1	50.0	50.0	50.0	50.0	18.8	0.0
	RoBERTa-large (zero-shot)	83.6	35.0	80.8	79.5	67.6	51.4	32.0	**2.0**
	POUF	**89.4(0.0)**	**44.9(1.6)**	**86.4(0.9)**	**86.0(5.6)**	**78.1(7.8)**	**80.8(0.1)**	**36.9(15.1)**	1.5(0.1)
Few-shot	Few-shot fine-tuning	81.4(3.8)	43.9(2.0)	76.9(5.9)	75.8(3.2)	72.0(3.8)	90.8(1.8)	88.8(2.1)	**33.9(14.3)**
	"GPT-3" in-context learning	84.8(1.3)	30.6(0.9)	80.5(1.7)	87.4(0.8)	63.8(2.1)	53.6(1.0)	26.2(2.4)	−1.5(2.4)
	LM-BFF (man)	92.6(0.5)	50.6(1.4)	86.6(2.2)	90.2(1.2)	87.0(1.1)	92.3(0.8)	87.5(3.2)	18.7(8.8)
	LM-BFF (man) + POUF	**93.2(0.0)**	**53.6(0.2)**	**87.6(0.2)**	**91.8(0.2)**	**88.0(0.2)**	**92.9(0.1)**	**90.1(0.1)**	20.6(0.6)
		MNLI (acc)	MNLI-mm (acc)	SNLI (acc)	QNLI (acc)	RTE (acc)	MRPC (F1)	QQP (F1)	STS-B (Pear.)
Unsupervised	Majority	32.7	33.0	33.8	49.5	52.7	**81.2**	0.0	–
	RoBERTa-large (zero-shot)	50.8	51.7	49.5	50.8	51.3	61.9	49.7	−3.2
	POUF	**55.0(2.3)**	**56.3(3.4)**	**64.0(0.2)**	**68.2(4.0)**	**64.5(2.5)**	**81.2(0.1)**	**53.8(0.1)**	**20.9(9.3)**
Few-shot	Few-shot fine-tuning	45.8(6.4)	47.8(6.8)	48.4(4.8)	60.2(6.5)	54.4(3.9)	76.6(2.5)	60.7(4.3)	53.5(8.5)
	"GPT-3" in-context learning	52.0(0.7)	53.4(0.6)	47.1(0.6)	53.8(0.4)	60.4(1.4)	45.7(6.0)	36.1(5.2)	14.3(2.8)
	LM-BFF (man)	70.7(1.3)	72.0(1.2)	79.7(1.5)	69.2(1.9)	68.7(2.3)	77.8(2.0)	**69.8(1.8)**	73.5(5.1)
	LM-BFF (man) + POUF	**73.8(1.1)**	**74.9(0.1)**	**80.0(0.2)**	**76.3(0.3)**	**70.5(0.6)**	**81.8(0.7)**	69.0(1.0)	**82.1(0.1)**

Main Results for Masked-Language Models. To validate the effectiveness of POUF on language tasks, we fine-tune the RoBERTa-large model with POUF on the unlabeled text data. We also incorporate POUF into LM-BFF to show the compatibility of our method with few-shot learning frameworks. We report the results on the 15 language tasks in Table 3. Under the unsupervised category, zero-shot RoBERTa model yields consistently higher performance than majority class. This result indicates that the model contains knowledge that can be exploited by prompt-based prediction. After applying POUF, the model performance significantly improves on 14 tasks. Under the few-shot category, few-shot fine-tuning does not always lead to a better performance than prompt-based prediction. Similarly, "GPT-3" in-context learning sometimes hurts the prompt-based prediction performance. We speculate that the size of the language model may have an impact on the performance. Since Gao et al. (2020) use RoBERTa-large instead of the original GPT-3 model, the smaller size of RoBERTa may have a significant impact on the performance of in-context learning. While POUF does not rely on labeled data, it outperforms both few-shot fine-tuning and "GPT-3" in-context learning on 10 and 14 tasks, respectively. For the few-shot setting, we incorporate POUF into LM-BFF by using the few-shot examples without labels to compute POUF's loss and add it to LM-BFF's objective. The results show that POUF also consistently boosts the performance of LM-BFF across 14 out of 15 tasks, illustrating the compatibility of our method with few-shot learning frameworks.

4.3. Analysis of Results

Ablation Studies. To verify the effect of each component of our method, we conduct ablation studies on both the Office-31 (image) and RTE (language) datasets and report the results in Table 4. **1)** *Distribution-matching options.* We provide two alternatives for CT: OT and OT-Sinkhorn. OT solves the exact linear program to obtain the optimal couplings, while OT-Sinkhorn solves a relaxation of the OT

Table 4: Average accuracy (%) of POUF on Office-31 (CLIP) and RTE (RoBERTa) under different variants.

Datasets	OT for $\mathcal{L}_{transport}$	OT-Sinkhorn for $\mathcal{L}_{transport}$	POUF w/o $\mathcal{L}_{transport}$	POUF w/o $\mathcal{L}_{mi}$	$c(\boldsymbol{\mu}_k, \boldsymbol{f}_j^t) = \exp(-\boldsymbol{\mu}_k^T \boldsymbol{f}_j^t)$	POUF (default setting)
Office-31	86.1 ± 0.2	86.5 ± 0.2	83.3 ± 0.4	84.2 ± 0.2	87.6 ± 0.5	88.0 ± 0.9
RTE	58.0 ± 5.5	63.3 ± 6.0	56.9 ± 1.5	58.4 ± 1.3	61.3 ± 6.3	64.5 ± 2.5

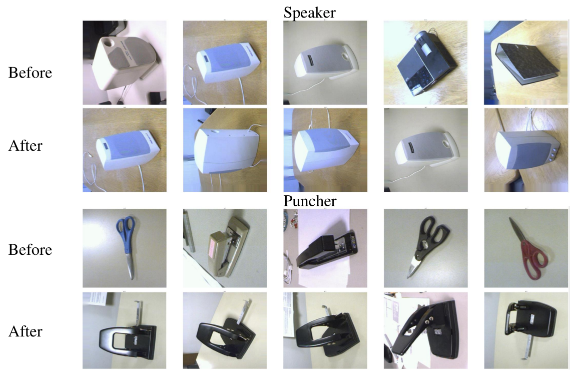


Figure 3: K-nearest neighbors of the prototypes of two different categories in Office-31 - Webcam: "Speaker" and "Puncher". For each category, the top row exhibits the top-5 neighbors of the prototype of the CLIP model before applying POUF, whereas the bottom row shows the top-5 neighbors of the prototype of the CLIP model after applying POUF.

problem with the Sinkhorn algorithm (Cuturi, 2013). In each variant, we replace CT in the $\mathcal{L}_{transport}$ while keeping $\mathcal{L}_{mi}$. In both image and language tasks, POUF with CT outperforms POUF with OT and OT-Sinkhorn. **2)** *Significance of each loss.* We remove each loss from the framework to understand the effect of each part. Without the transport cost, the performance drops by 4% and 7% on the image and text tasks, respectively. Similarly, after removing the mutual-information objective, the accuracy decreases by 4% and 6% on the image and text tasks, respectively. These results illustrate the significance of each loss and the synergistic effect of the two losses. **3)** *Cost function.* In the transport framework, we utilize the cosine distance as the cost function. Alternatively, we could use other cost functions. We experiment with another cost function inspired by the radial basis kernel. The result shows that the cosine distance function leads to better performance in both modalities, justifying our design decisions.

Visualization. Q: Does POUF learn more meaningful prototypes? A: Yes In Figure 3, we visualize the K-nearest neighbors ($K = 5$) of the prototypes to understand what the model learns. Before applying POUF, the prototypes of the zero-shot CLIP model contain images that do not semantically represent the classes. The fourth and fifth images among the nearest neighbors of the "Speaker" prototype include a picture of a projector and a ring binder, respectively, while the five nearest neighbors of the "Puncher" prototype do not have a single image of a puncher. The "Puncher" prototype seems to be close to the images of Scissors and

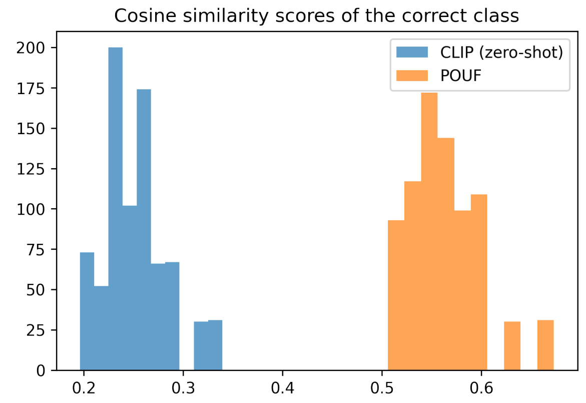


Figure 4: Histograms of cosine similarity scores between the correct classes' prototypes and target features in Office-31 - Webcam.

Staplers. After applying POUF, we observe significant improvement qualitatively. Both the "Speaker" and "Puncher" prototypes are now close to the images of their respective classes, demonstrating that POUF helps learn more meaningful prototypes. **Q: Does POUF reduce the violation of the cluster assumption? A: Yes.** In Figure 1, we visualize the t-SNE plots of the prototypes and target features. After applying POUF, the prototypes are assigned to their respective clusters of target features. To further quantify this point, for each example, we compute the cosine similarity between the target feature and the prototype of its label. We then visualize the histograms of the similarity scores in Figure 4. It is evident that the cosine similarity scores of POUF are higher than those of the zero-shot CLIP model, meaning that the target examples are moved closer to the prototypes of their corresponding classes.

5. Conclusion

In this paper, we present POUF, a simple yet effective framework for directly adapting prompt-based zero-shot models to the unlabeled target data. We propose aligning the prototypes and target data in the latent space through transport-based distribution alignment and mutual information maximization. Across 13 image and 15 language tasks, POUF achieves consistent performance gains over the baselines.

Acknowledgments

We thank Camillia Smith Barnes for providing valuable feedback on the paper. K. Tanwisuth, S. Zhang, H. Zheng, and M. Zhou acknowledge the support of NSF-IIS 2212418, NSF-IFML, and TACC.

References

Asano, Y. M., Rupprecht, C., and Vedaldi, A. Self-labelling via simultaneous clustering and representation learning. *arXiv preprint arXiv:1911.05371*, 2019.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *NeurIPS*, 2020.

Cao, Z., Long, M., Wang, J., and Jordan, M. I. Partial transfer learning with selective adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2724–2732, 2018.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924, 2020.

Chizat, L., Peyré, G., Schmitzer, B., and Vialard, F.-X. Unbalanced optimal transport: Dynamic and kantorovich formulations. *Journal of Functional Analysis*, 274(11): 3090–3123, 2018.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26:2292–2300, 2013.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Du, Y., Liu, Z., Li, J., and Zhao, W. X. A survey of vision-language pre-trained models. *arXiv preprint arXiv:2202.10936*, 2022.

Fan, X., Zhang, S., Chen, B., and Zhou, M. Bayesian attention modules. *Advances in Neural Information Processing Systems*, 33:16362–16376, 2020.

Ganin, Y. and Lempitsky, V. S. Unsupervised domain adaptation by backpropagation. In Bach, F. R. and Blei, D. M. (eds.), *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1180–1189. JMLR.org, 2015.

Gao, T., Fisch, A., and Chen, D. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*, 2020.

Grandvalet, Y., Bengio, Y., et al. Semi-supervised learning by entropy minimization. In *CAP*, pp. 281–296, 2005.

Guo, D., Tian, L., Zhang, M., Zhou, M., and Zha, H. Learning prototype-oriented set representations for meta-learning. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=WH6u2SvlLp4`.

Huang, T., Chu, J., and Wei, F. Unsupervised prompt learning for vision-language models. *arXiv preprint arXiv:2204.03649*, 2022.

Jia, C., Yang, Y., Xia, Y., Chen, Y.-T., Parekh, Z., Pham, H., Le, Q., Sung, Y.-H., Li, Z., and Duerig, T. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, pp. 4904–4916. PMLR, 2021.

Jiang, J., Shu, Y., Wang, J., and Long, M. Transferability in deep learning: A survey, 2022.

Kang, G., Jiang, L., Yang, Y., and Hauptmann, A. G. Contrastive adaptation network for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4893–4902, 2019.

Krause, A., Perona, P., and Gomes, R. Discriminative clustering by regularized information maximization. *Advances in neural information processing systems*, 23, 2010.

Kundu, J. N., Venkat, N., Babu, R. V., et al. Universal source-free domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4544–4553, 2020a.

Kundu, J. N., Venkatesh, R. M., Venkat, N., Revanur, A., and Babu, R. V. Class-incremental domain adaptation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*, pp. 53–69. Springer, 2020b.

Kurmi, V. K., Subramanian, V. K., and Namboodiri, V. P. Domain impression: A source data free domain adaptation method. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 615–625, 2021.

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Li, J., Zhou, P., Xiong, C., and Hoi, S. C. Prototypical contrastive learning of unsupervised representations. *arXiv preprint arXiv:2005.04966*, 2020a.

Li, R., Jiao, Q., Cao, W., Wong, H.-S., and Wu, S. Model adaptation: Unsupervised domain adaptation without source data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9641–9650, 2020b.

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Liang, J., Hu, D., and Feng, J. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*, pp. 6028–6039. PMLR, 2020.

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*, 2021a.

Liu, X., Gong, C., Wu, L., Zhang, S., Su, H., and Liu, Q. Fusedream: Training-free text-to-image generation with improved clip+ gan space optimization. *arXiv preprint arXiv:2112.01573*, 2021b.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Long, M., Cao, Y., Wang, J., and Jordan, M. I. Learning transferable features with deep adaptation networks. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pp. 97–105, 2015.

Long, M., Zhu, H., Wang, J., and Jordan, M. I. Deep transfer learning with joint adaptation networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2208–2217. PMLR, 06–11 Aug 2017.

Long, M., Cao, Z., Wang, J., and Jordan, M. I. Conditional adversarial domain adaptation. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Mérigot, Q. and Oudet, E. Discrete optimal transport: complexity, geometry and applications. *Discrete & Computational Geometry*, 55(2):263–283, 2016.

Morerio, P., Cavazza, J., and Murino, V. Minimal-entropy correlation alignment for unsupervised deep domain adaptation. *arXiv preprint arXiv:1711.10288*, 2017.

Pan, Y., Yao, T., Li, Y., Wang, Y., Ngo, C.-W., and Mei, T. Transferrable prototypical networks for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., and Wang, B. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1406–1415, 2019.

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., and Huang, X. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10): 1872–1897, 2020.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763. PMLR, 2021.

Saenko, K., Kulis, B., Fritz, M., and Darrell, T. Adapting visual category models to new domains. In *European conference on computer vision*, pp. 213–226. Springer, 2010.

Saito, K., Kim, D., Sclaroff, S., Darrell, T., and Saenko, K. Semi-supervised domain adaptation via minimax entropy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8050–8058, 2019.

Saito, K., Kim, D., Sclaroff, S., and Saenko, K. Universal domain adaptation through self supervision. *arXiv preprint arXiv:2002.07953*, 2020.

Schick, T. and Schütze, H. Exploiting cloze questions for few shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*, 2020.

Shi, Y. and Sha, F. Information-theoretical learning of discriminative clusters for unsupervised domain adaptation. *arXiv preprint arXiv:1206.6438*, 2012.

Shin, T., Razeghi, Y., Logan IV, R. L., Wallace, E., and Singh, S. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*, 2020.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

Tanwisuth, K., Fan, X., Zheng, H., Zhang, S., Zhang, H., Chen, B., and Zhou, M. A prototype-oriented framework for unsupervised domain adaptation. *Advances in Neural Information Processing Systems*, 34:17194–17208, 2021.

Tanwisuth, K., Zhang, S., He, P., and Zhou, M. A prototype-oriented clustering for domain shift with source privacy. *arXiv preprint arXiv:2302.03807*, 2023.

Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., and Darrell, T. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7167–7176, 2017.

Venkateswara, H., Eusebio, J., Chakraborty, S., and Panchanathan, S. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5018–5027, 2017.

Villani, C. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Vu, T.-H., Jain, H., Bucher, M., Cord, M., and Pérez, P. Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2517–2526, 2019.

Wallace, E., Feng, S., Kandpal, N., Gardner, M., and Singh, S. Universal adversarial triggers for attacking and analyzing nlp. *arXiv preprint arXiv:1908.07125*, 2019.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Wang, D., Shelhamer, E., Liu, S., Olshausen, B., and Darrell, T. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.

Wang, D., Guo, D., Zhao, H., Zheng, H., Tanwisuth, K., Chen, B., and Zhou, M. Representing mixtures of word embeddings with mixtures of topic embeddings. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=IYMuTbGzjFU`.

Wenzel, F., Dittadi, A., Gehler, P. V., Simon-Gabriel, C.-J., Horn, M., Zietlow, D., Kernert, D., Russell, C., Brox, T., Schiele, B., et al. Assaying out-of-distribution generalization in transfer learning. *arXiv preprint arXiv:2207.09239*, 2022.

Wu, X., Zhou, Q., Yang, Z., Zhao, C., Latecki, L. J., et al. Entropy minimization vs. diversity maximization for domain adaptation. *arXiv preprint arXiv:2002.01690*, 2020.

Yue, X., Zheng, Z., Zhang, S., Gao, Y., Darrell, T., Keutzer, K., and Vincentelli, A. S. Prototypical cross-domain self-supervised learning for few-shot unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13834–13844, 2021.

Zhang, S., Fan, X., Chen, B., and Zhou, M. Bayesian attention belief networks. In *International Conference on Machine Learning*, pp. 12413–12426. PMLR, 2021a.

Zhang, S., Fan, X., Zheng, H., Tanwisuth, K., and Zhou, M. Alignment attention by matching key and query distributions. In *Neural Information Processing Systems*, Dec. 2021b.

Zhang, S., Gong, C., and Choi, E. Capturing label distribution: A case study in nli. *arXiv preprint arXiv:2102.06859*, 2021c.

Zhang, S., Gong, C., and Choi, E. Learning with different amounts of annotation: From zero to many labels. *arXiv preprint arXiv:2109.04408*, 2021d.

Zhang, S., Gong, C., Liu, X., He, P., Chen, W., and Zhou, M. Allsh: Active learning guided by local sensitivity and hardness. *arXiv preprint arXiv:2205.04980*, 2022.

Zheng, H. and Zhou, M. Exploiting chain rule and Bayes' theorem to compare probability distributions. *Advances in Neural Information Processing Systems*, 34:14993–15006, 2021.

Zhou, K., Yang, J., Loy, C. C., and Liu, Z. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16816–16825, 2022a.

Zhou, K., Yang, J., Loy, C. C., and Liu, Z. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, 2022b.

POUF: Prompt-Oriented Unsupervised Fine-tuning for Large Pre-trained Models: Appendix

A. Detailed Implementation Details

A.1. Language-Augmented Vision models

We build our method using the open-source CLIP codebase (Radford et al., 2021; Liu et al., 2021b) and TLlib transfer learning library (Jiang et al., 2022). For all experiments, we adopt the ViTB-16 for the image encoder and the default transformer from the CLIP paper for the text encoder. All the unlabeled target samples are used for fine-tuning. The learning rate shcedule is set to $\eta_{\text{iter}} = \eta_0(1 + \gamma\text{iter})^{-\alpha}$, where η_0 is the initial learning rate. We adopt the following default hyper-parameters: $\gamma = 0.0002$, and $\alpha = 0.75$. We set $\eta_0 = 5\text{e}{-7}$ for all experiments except for prompt tuning on Office-31 where $\eta_0 = 1\text{e}{-3}$. For prompt tuning, the context length, p, is set to 4. We then initialize the soft-prompt parameters with the embeddings of the words "A photo of a". We use a mini-batch SGD with a momentum of 0.9 and a batch size of 96 for Office-31 and Office-Home and 16 for DomainNet. The weight of the mutual-information objective, λ, is set to 0.3 for all experiments. We run all experiments for $5,000$ iterations using the seeds $\{0, 1, 2\}$ and report the average accuracy. All experiments are conducted using a single Nvidia Tesla V100 GPU.

For all the baselines, we use the same hyper-parameters as our method for prompt tuning. For Tent, we set the weight of the entropy objective to 0.3, the same as the weight of the mutual-information objective, λ. For UPL, we follow the original paper to set the number of pseudo labels as 16 examples per class. For CoOp, we set the number of few-shot examples per class as 16 per the original paper.

A.2. Masked-Language Models

We follow the experimental protocols in Gao et al. (2020). Specifically, for each task, the data is split into $\mathcal{D}_{train}$, $\mathcal{D}_{dev}$, and $\mathcal{D}_{test}$. The authors tune the hyper-parameters on $\mathcal{D}_{dev}$ and report the performance of the model on $\mathcal{D}_{test}$. We take the same hyper-parameters as the original paper: batch size $= 8$, learning rate $= 1e - 5$, training steps $= 1,000$, and few-shot examples per class $= 16$. The weight of the mutual-information objective, λ, is set to 0.6 for all experiments except for the unsupervised setting on the MRPC and QQP tasks. In these two tasks, the weights are set to $1\text{e}{-4}$ for the mutual-information objective and to $1\text{e}{-2}$ for the transport cost. We validate the performance of the model every 100 steps on the development set and take the best validated checkpoint for the final evaluation on the test set. For the unsupervised setting, we use the dataset provided by Gao et al. (2020) without the labels. For the few-shot setting, we incorporate POUF into LM-BFF by using the few-shot examples without labels to compute POUF's loss and add it to LM-BFF's objective. We run all experiments using the seeds $\{42, 21, 87, 13, 100\}$ and report the average accuracy. All experiments are conducted on four Nvidia Tesla V100 GPUs.

B. Parameter Size and Runtime Analysis

Table 5: Parameter and run-time analyses of prompt v.s. model tuning the CLIP model.

Settings	Parameter analysis		Run-time analysis
	Tunable parameters	Total parameters	Time per iteration (s)
Prompt tuning	$2,049$	$149,622,785$	0.4
Model tuning	$149,620,737$	$149,620,737$	1.8

In Table 5, we present parameter and run-time analyses of prompt and model tuning for the CLIP model. Prompt tuning introduces h (Embedding dimension) $\times$ p (Context length) $= 512 \times 4 = 2048$ number of parameters to the model, increasing the total number of parameters slightly. However, given the size of the CLIP model, the increase is negligible. We note that the number of tunable parameters is $2,049$ since we also adjust the temperature parameter, a scalar, in the CLIP model. Since the number of tunable parameters is significantly lower in prompt tuning than in model tuning, it is not surprising that the training time per iteration for prompt tuning is one-quarter that of model tuning.

C. Limitation and Societal Impact

POUF leverages the power of language representations to adapt to the unlabeled target data directly. While the language model may have a large vocabulary size, it is possible that it does not capture some exotic words. This means that if the categories that we want to predict are outside the vocabulary of the language model, we cannot use our method to fine-tune the model, as we cannot even make predictions. Future research can focus on how to extend POUF to adapt to unknown classes beyond the vocabulary of the model.

POUF relies on large-scale pre-trained models to adapt to the target data. As with any computationally intensive venture, practitioners should consider using sustainable energy resources. On the positive side, POUF enables practitioners to efficiently fine-tune on the target data, saving the computation.

D. Datasets

Table 6: Dataset information for image experiments.

Dataset	Number of images	Number of categories	Domains
Office-31 (Saenko et al., 2010)	$4,652$	31	Amazon, DSLR, Webcam
Office-Home (Venkateswara et al., 2017)	$15,500$	65	Artistic images, Clip Art, Product images, Real-World
DomainNet (Peng et al., 2019)	$569,010$	345	Clipart, Infograph, Painting, Quickdraw, Real, Sketch

Table 7: Dataset information for language experiments from Gao et al. (2020). $|\mathcal{Y}|$ denotes the number of categories. L refers to the average number of words in a sentence. The license of the GLUE benchmark is cc-by-4.0.

Category	Dataset	$\lvert\mathcal{Y}\rvert$	L	#Train	#Test	Type	Labels (classification tasks)
single-sentence	SST-2	2	19	$6,920$	872	sentiment	positive, negative
	SST-5	5	18	$8,544$	$2,210$	sentiment	v. pos., positive, neutral, negative, v. neg.
	MR	2	20	$8,662$	$2,000$	sentiment	positive, negative
	CR	2	19	$1,775$	$2,000$	sentiment	positive, negative
	MPQA	2	3	$8,606$	$2,000$	opinion polarity	positive, negative
	Subj	2	23	$8,000$	$2,000$	subjectivity	subjective, objective
	TREC	6	10	$5,452$	500	question cls.	abbr., entity, description, human, loc., num.
	CoLA	2	8	$8,551$	$1,042$	acceptability	grammatical, not grammatical
sentence-pair	QNLI	2	$11/30$	$104,743$	$5,463$	NLI	entailment, not entailment
	MNLI	3	$22/11$	$392,702$	$9,815$	NLI	entailment, neutral, contradiction
	RTE	2	$49/10$	$2,490$	277	NLI	entailment, not entailment
	MRPC	2	$22/21$	$3,668$	408	paraphrase	equivalent, not equivalent
	QQP	2	$12/12$	$363,846$	$40,431$	paraphrase	equivalent, not equivalent
	STS-B	$\mathcal{R}$	$11/11$	$5,749$	$1,500$	sent. similarity	-

E. Unseen Class Experiments

Testing POUF under an open class scenario is an interesting setting. POUF is developed to address domain shift (changes in feature distribution) but does not specifically target label space shift. To investigate the impact of POUF on generalization to unseen classes, we designed two experiments: in-domain generalization and out-of-domain generalization. The setup of each experiment is explained below. POUF (text prototypes = seen classes): we adapt POUF with the textual prototypes from only the seen classes. POUF (text prototypes = seen + unseen classes): in many cases, the target users may not have image data of the unseen classes but know the class names in advance. In this version, we adapt POUF with the textual prototypes from both the seen and unseen classes. 1) Experiment 1: In-domain generalization. In this experiment, our goal is to investigate the impact of POUF on both seen and unseen image classes within the same domain. To do so, we randomly divide the classes in one of the Office-31 dataset's domains into two groups: half as seen and the other half as unseen. We employ images from the seen classes for adaptation using POUF. To evaluate the model, we assess its performance on both seen and unseen classes within this domain. The findings are presented below. 2) Experiment 2: Out-of-domain generalization. In this experiment, we aim to examine the impact of POUF on seen and unseen image classes across different domains. To achieve this, we randomly split the classes into two groups: half as seen and the other half as unseen. We apply POUF to adapt the model using the seen classes from one domain and then evaluate the adapted model on the seen and unseen classes of another domain. The findings are presented in Tables 8 and 9.

In both tables, POUF enhances the performance of zero-shot models for seen classes. However, when trained exclusively with seen-class prototypes, POUF yields improved performance for the seen class but slightly weaker generalization for the unseen class when evaluated on the Amazon domain, in both in-domain and out-of-domain generalization settings. This outcome highlights a minor trade-off between the model's specialization and generalization. If target users have access to unseen class names, training POUF with both seen and unseen class text prototypes results in superior performance compared to the zero-shot model for both seen and unseen classes in both scenarios. The outcome implies that the essential concept is to maintain the semantics in the text space for improved generalization. A possible explanation for this observation is that, during pre-training, both text and image features are plentiful. However, during fine-tuning, we continue to have abundant image features but limited text prompts, which come from the target user. If we fine-tune both text and image encoders, we might end up with representations that are particularly tailored to the seen classes.

Table 8: POUF on unseen classes for Webcam (W).

	In-domain Setting		Out-of-domain Setting	
Adaptation	seen class of Webcam (W)	seen class of Webcam (W)	seen class of Webcam (W)	seen class of Webcam (W)
Testing	seen class of W	unseen class of W	seen class of A	unseen class of A
CLIP (zero-shot)	65.2	87.5	76.5	83.0
POUF (text prototypes = seen classes)	**85.1**	92.8	**83.6**	82.7
POUF (text prototypes = seen + unseen classes)	75.5	**95.7**	79.6	**85.8**

Table 9: POUF on unseen classes for Amazon (A).

	In-domain Setting		Out-of-domain Setting	
Adaptation	seen class of Amazon (A)	seen class of Amazon (A)	seen class of Amazon (A)	seen class of Amazon (A)
Testing	seen class of A	unseen class of A	seen class of W	unseen class of W
CLIP (zero-shot)	76.5	83.0	65.2	87.5
POUF (text prototypes = seen classes)	**84.1**	81.7	**81.1**	89.4
POUF (text prototypes = seen + unseen classes)	80.6	**87.2**	72.2	**93.1**

F. Additional Ablation Study on the Hyper-parameter λ

λ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Accuracy	87.5	88.3	90.6	91.7	90.8	89.2	90.6	89.8	88.9	91.6

Table 10: Accuracy of POUF on the Webcam domain of the Office-31 dataset for different λ values.

G. Class Proportion Estimation

The transport framework provides a nice way of handling the class-imbalanced issue. If we know the proportions beforehand, we can adjust the marginal distribution, $p(\boldsymbol{w}_k)$, by setting it to the corresponding proportions. Classes with higher proportions will then receive higher transport costs. However, in practice, the proportions are unknown so we need to estimate them. To estimate the proportion of class, we can marginalize the predictive distribution of the classes given the data as follows

$$p(\boldsymbol{w}_k) = \sum_i \pi_{\boldsymbol{\theta}}(\boldsymbol{w}_k \mid \boldsymbol{f}_i) = \sum_i \frac{p(\boldsymbol{w}_k)\exp(\boldsymbol{w}_k^T \boldsymbol{f}_i)}{\sum_{k'=1}^K p(\boldsymbol{w}_{k'})\exp(\boldsymbol{w}_{k'}^T \boldsymbol{f}_i)}$$

Since the marginalization is done over the data, mini-batch update is often required to overcome the computational expense. Thus, the above estimation can be noisy. To overcome this issue and estimate the global proportions from a local mini-batch of data, we can iteratively learn the global proportions over multiple iterations. We can then learn the proportions as follows

$$\tilde{p}(\boldsymbol{w}_k)^{l+1} = \frac{1}{M}\sum_i \pi_{\boldsymbol{\theta}}(\boldsymbol{w}_k \mid \boldsymbol{f}_i)$$

$$p(\boldsymbol{w}_k)^{l+1} = \alpha^l \tilde{p}(\boldsymbol{w}_k)^{l+1} + (1-\alpha^l) p(\boldsymbol{w}_k)^l,$$

where α^l is the learning rate of iteration l and follows a cosine learning rate schedule.

H. Pseudo Code

Algorithm 1 POUF Pseudocode for language-augmented vision models, PyTorch-like

```
# F: image encoder
# G: text encoder
# prompts: textual inputs (i.e. "A photo of {CLASS}")

for x in loader: # Load a minibatch x with M samples
   f, prototypes = F(x), G(prompts) # Compute embeddings

   f = normalize(f, dim=1)
   prototypes = normalize(prototypes, dim=1)

   sim_mat = f @ prototypes.T # M-by-K

   L_transport = compute_transport_loss(sim_mat, T)
   L_mi = compute_mi_loss(sim_mat/T)
   L = L_transport + lambda_mi * L_mi # Loss

   L.backward() # Back-propagate
   update(F, G) # SGD update

def compute_transport_loss(sim_mat, T):
   # Compute transport cost
   cost = 1 - (sim_mat)

   # Compute transport plans
   forward_plan = softmax(sim_mat/T, dim=0)
   backward_plan = softmax(sim_mat/T, dim=1)

   # Compute final loss
   forward_cost = (cost * forward_plan).sum(0).mean()
   backward_cost = (cost * backward_plan).sum(1).mean()
   return forward_cost + backward_cost

def compute_mi_loss(sim_mat):
   # Compute conditional entropy
   softmax_out = softmax(sim_mat, dim=1)
   entropy_loss = sum(-softmax_out * log(softmax_out + eps), dim=1).mean()

   # Compute marginal entropy
   mean_softmax = softmax_out.mean(0)
   regularization = sum(-mean_softmax * log(mean_softmax + eps))

   # Compute final loss
   return entropy_loss - regularization
```

Algorithm 2 POUF Pseudocode for masked-language models, PyTorch-like

```
# M: text encoder

for x in loader: # Load a minibatch x with M samples. The input is assumed to be in a prompt.
   outputs = M(x) # Compute embeddings

   sequence_output, pooled_output = outputs[:2]

   sequence_mask_output = sequence_output[:, mask_pos] # Extract masked-token representations

   prediction_mask_scores = lm_head(sequence_mask_output)[:, label_word_list] # Compute logits

   f = get_features_before_last_layer(sequence_mask_output, lm_head) # Features before decoder head
   prototypes = lm_head.decoder.weight[label_word_list]

   f = normalize(f, dim=1)
   prototypes = normalize(prototypes, dim=1)

   sim_mat = f @ prototypes.T # M-by-K

   L_transport = compute_transport_loss(sim_mat, T)
   L_mi = compute_mi_loss(prediction_mask_scores)
   L = L_transport + lambda_mi * L_mi # Loss

   L.backward() # Back-propagate
   update(M) # SGD update

def get_features_before_last_layer(features, lm_head):
   x = lm_head.dense(features)
   x = gelu(x)
   x = lm_head.layer_norm(x)
   return x
```