Active Test-Time Prompt Learning in VLMs Supplementary Material

A Broader Impact

The goal of deep learning is to learn discriminative invariant feature representations. One roadblock in that goal is that models which are trained on a particular dataset often develop a bias towards it via overfitting which is essentially the emergence of spurious correlations. These spurious correlations pose both technical and ethical concerns that have to be mitigated so that deep learning models can be widely used. In the most common scenario, domain generalization helps us in eliminating these spurious correlations by applying certain techniques before the model's deployment. However, it is not always possible to gather data before the model's deployment, and thus, it becomes imperative to instead adapt the model during inference. We do not see any immediate ethical concerns which are raised by this paper as it does not release any specific dataset nor does it use any human as a subject. However, as with any scientific work, it has the potential for misuse, and thus, we support a continued assessment of methods like ours, which advocate test-time adaptation of large models using expert advice.

B Related Work

B.1 Active Learning

Active Learning promotes label efficiency by imposing a label budget. It can be used in a variety of settings to gain knowledge about some aspect that is not already known and the model is uncertain about via an oracle. The queried samples are then sent to an oracle who returns the true label. Other than choosing on the basis of uncertainty (Lewis & Catlett, 1994; Yang & Loog, 2016; Roth & Small, 2006; Holub et al., 2008), in latest works, the model also tries to maintain diversity (Parvaneh et al., 2022; Sener & Savarese, 2017) in its choice of samples to query. There is typically a dynamic buffer that is maintained by the Active Learning algorithm wherein the annotated samples are put. The buffer is enlarged as more samples which are both diverse and informative are added to it. There have been incorporations of Active Learning into Domain Adaptation as well (Prabhu et al., 2021). In (Wang et al., 2022; Kothandaraman et al., 2022) incorporated Active Learning into Source Free Domain Adaptation which, while not directly dependent on the sourced data, are also not suited for continuous data streams, unlike our TTA setting. In (Saran et al., 2023), they take up the task of actively labelling samples in a streaming setting. However, their work significantly differs from ours as they don't continuously adapt their parameters as the data stream progresses. Instead, they reinitialise their parameters to the original parameters after each new data point is acquired. A contemporary work that is more closely related to ours is (Gui et al., 2024), where they provide some foundational theoretical work on Active Test Time Adaptation. However, their work is also significantly different from ours because they assume a batch setting wherein at each timestep, a minibatch comes for inference while we only assume a single sample. They also do multiple gradient updates in each time step while we do only one. Their labelling is also not done in real-time but effectively postponed by placing the unlabelled samples in a buffer from which they have to be selected for active labelling later. This may not always be possible due to privacy and storage concerns where it may not be feasible to postpone the decision of sending a sample to an oracle and instead retaining it. On the other hand, we make the active labelling decision in real-time based on our dynamically adjusted threshold.

B.2 Test Time Adaptation

Adaptation of pre-trained models is necessary so that they can be used optimally for the given task at hand. Fully Test Time Adaptation (Nado et al., 2020; Schneider et al., 2020; Sun et al., 2023; Liang et al., 2023) is the highly realistic and practical setting wherein a pre-trained model has to optimise and adapt its parameters to a situation that it faces during inference, that is, amidst real-time use. A prominent example includes that of a self-driving car where the car has to adapt to unforeseen conditions while it is being used. A popular way to achieve fully test time adaptation has been to update the statistics of the Batch Normalisation layer during inference. In TENT (Wang et al., 2020), the Batch Normalisation parameters are updated using a self-entropy objective. However, TENT makes the batched input assumption. In MEMO (Zhang et al., 2022), they use the more general case of a single test input by taking multiple augmentations of a single image. TPT (Shu et al., 2022) essentially extends the MEMO philosophy to prompt tuning to make VLMs adapt at test-time by updating prompts. Especially relevant to this paper is the newly introduced paradigm of Active Test Time Adaptation Gui et al. (2024), where the model has the option of querying a few samples during test time adaptation. It was found that at a some latency cost as compared to FTTA methods, the ATTA framework provides much superior results, and thus, its use-case may not completely overlap with that of FTTA.

B.3 Prompt Learning in VLMs

Prompt Learning has been proposed as a method of fine-tuning VLMs in compute-constrained scenarios due to its parameter efficiency. In CoOp (Zhou et al., 2022b) and CoCoOp (Zhou et al., 2022a), the prompts are appended to the textual tokens and help provide context to the input of the VLM instead of finetuning the entire VLM model. Learning good prompts has been shown to dramatically improve the performance of the CLIP (Radford et al., 2021) model. Maple (Khattak et al., 2023)introduced Multimodal Prompt learning where, along with the text encoder, prompts are also learnt for the vision encoder. PromptAlign(Abdul Samadh et al., 2024) extends the multimodal framework of MaPle to a test time setting, and adds Distribution Alignment to this by considering Imagenet to be the proxy source dataset and calculating the alignment loss between different layers of the encoders for each image and precomputed statistics. Prompt Learning has been extended as a transfer learning and adaptation method in various ways and settings. Of special interest to us is the Test Time setting (Shu et al., 2022). The Test Time scenario is highly practical given that foundation models like VLMs are becoming more mainstream and adapting them at test time by optimising a small parameter group like prompts is likely to be the way forward. More recently, in (Bang et al., 2024) they introduce Active Learning to Vision Language models where it is noted that diversity in the form of class-balancing is important for non-trivial gains in VLM performance via Active Learning. However, our method is significantly different from theirs because it is in a test time scenario, whereas theirs was in a supervised learning setting.

C Potential Applications

For the sake of fair comparison with previous arts, we actively label our samples only after evaluating them. However, it is important to note that, when it comes to practical application, we can also reverse the order. That is, we can ask the expert for their advice and take that as the ground truth. This is especially true because we are not restricted by our methodology to postpone the active labelling decision but instead do so in real-time, unlike (Gui et al., 2024) where they collect the unlabelled samples in a buffer and select which ones to query later. Our single test sample in a time-step assumption makes our setting even more practical.

Ideal scenarios for practical application are high-risk ones like autopilot systems and medical diagnosis, where the extra cost and latency in consulting an expert is justified by the potential avoidance of hazards. In autopilot systems, when the system is uncertain, it can hand over control to the pilot, who then demonstrates the correct way of handling that particular scenario. The system then stores the pilot's actions in memory and uses them to learn the correct way so that it does not need human intervention in a similar scenario in future. In medical diagnosis, whenever the system is uncertain, instead of making a wrong diagnosis, it sends the sample over to a medical practitioner who then provides his expert opinion.

Our average inference latency per sample, when the buffer size is at its maximum, is around 0.63s, which is about 50% more than that of PromptAlign (Abdul Samadh et al., 2024), whose latency we found to be 0.41s when averaged over all 14 datasets. However, these figures must be contextualized in comparison to those from (Gui et al., 2024), where they observed up to almost **10x** increase in latency compared to their baselines.

D ALGORITHMS

Algorithm 1 Dynamic Threshold Selection

```
Require: $t, N_{queried}, \hat{\mu}_t, \hat{\sigma}_t$
  if $t < T_{min}$ then
      # Initially select a static threshold till $T_{min}$ number of test samples
      Output:$\tau_0$
  else
      # $\alpha$ is the query selection percentile
      if $\frac{N_{queried}}{t} \geq \alpha$ then
          # Select higher value of z ($z_{high}$) if currently over-querying
          $\tau_t = \hat{\mu} + z_{high}\hat{\sigma}$
      else
          Select standard value of z with respect to $\alpha$ otherwise $\tau_t = \hat{\mu} + z_{selection}\hat{\sigma}$
      end if
      Output:$\tau_t$
  end if
```

Algorithm 2 Class Balanced Eviction from Buffer

```
Require: Unlabeled Dataset $\mathcal{D}_u, Oracle(.)$ $N_{queried} = 0, \hat{\mu}_0 = 0, \hat{\sigma}_0 = 0$
  for $t = 1, 2, 3, ..., |\mathcal{D}_u|$ do
      $\hat{\mu}_t, \hat{\sigma}_t \leftarrow \hat{\mu}_{t-1}, \hat{\sigma}_{t-1}$
      $\tau_t =$ Dynamic Threshold Selection$(t, N_{queried}, \hat{\mu}_t, \hat{\sigma}_t)$
      if $H(\hat{x}_t) > \tau_t$ then
          $y_t = Oracle(x_t)$
          $N_{queried} \leftarrow N_{queried} + 1$
          if the buffer is full then
              # Select class k, the class with the most samples in $\mathcal{D}_l$
              $m = \underset{k \in \{1,2,...,K\}}{\arg\max} \; |c_k|$
              if $|m| = 1$ then
                  # Select sample with lowest CE loss in max class
                  $j = \underset{i \in \{1,2,...J\}}{\arg\min} \; L_{CE}(x_i, y_i) | y_i = c_m$
                  Remove $(x_j, y_j)$ from $\mathcal{D}_l$
              else if $|m| > 1$ then
                  # Select the max class with least avg CE loss
                  $l = \underset{k \in m}{\arg\min} \, \bar{L}_{CE}(x_i, y_i) | y_i = c_k$
                  # Select sample with lowest CE loss in the max class
                  $j = \underset{i \in (1,2,...J)}{\arg\min} \; L_{CE}(x_i, y_i) | y_i = c_l$
                  Remove $(x_j, y_j)$ from $\mathcal{D}_l$
              end if
          end if
          Add $(x_t, y_t)$ to $\mathcal{D}_l$
      end if
  end for
```