Vector-Quantized Input-Contextualized Soft Prompts for Natural Language Understanding

Rishabh Bhardwaj[*] [§‡†] **Amrita Saha**[*] [§] **Steven C.H. Hoi** [§] **Soujanya Poria** [‡]

[§]Salesforce Research
[‡]Singapore University of Technology and Design

Abstract

Prompt Tuning has been largely successful as a parameter-efficient method of conditioning large-scale pre-trained language models to perform downstream tasks. Thus far, soft prompt tuning learns a fixed set of task-specific continuous vectors, i.e., soft tokens that remain static across the task samples. A fixed prompt, however, may not generalize well to the diverse kinds of inputs the task comprises. In order to address this, we propose **V**ector-quantized **I**nput-contextualized **P**rompts (VIP)[1] as an extension to the soft prompt tuning framework. VIP particularly focuses on two aspects—**contextual prompts** that learns input-specific contextualization of the soft prompt tokens through a small-scale sentence encoder and **quantized prompts** that maps the contextualized prompts to a set of learnable codebook vectors through a Vector quantization network. On various language understanding tasks like SuperGLUE, QA, Relation classification, NER and NLI, VIP outperforms the soft prompt tuning (PT) baseline by an average margin of 1.19%. Further, our generalization studies show that VIP learns more robust prompt representations, surpassing PT by a margin of 0.6% - 5.3% on Out-of-domain QA and NLI tasks respectively, and by 0.75% on Multi-Task setup over 4 tasks spanning across 12 domains.

1 Introduction

With the increase in network size, it has become more computationally expensive to fine-tune pre-trained language models (PLMs) on downstream tasks. Recent studies have shown that prompt-based learning is a quite effective parameter-efficient method of conditioning the behavior of PLMs for a given predictive task (Lester et al., 2021; Wu et al., 2022). The goal of prompt-based learning, in contrast to traditional supervised learning approaches, is to find the task-specific template that is augmented to the input text in order to guide the PLM toward generating the correct output (Liu et al., 2021). The initial efforts in this direction involved manually designed templates to solve natural language processing (NLP) tasks (Brown et al., 2020; Schick and Schütze, 2021, 2020).

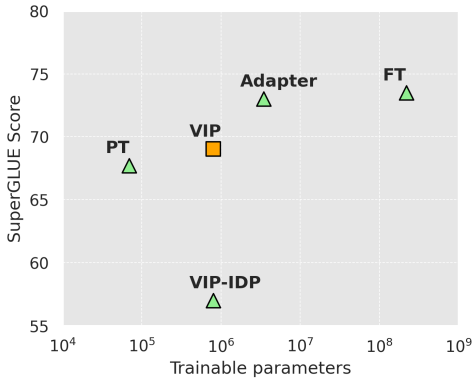

Figure 1: Comparison of model performance vs parameter size on SuperGLUE dataset. PT and FT denotes prompt tuning (2021) and fine-tuning respectively while Adapter (2019) learns adapter modules inserted between layers of the PLM. VIP-IDP denotes instance dependent prompt generation (2022) under text-to-text framework. VIP is our proposed approach of quantized contextual soft prompts.

In order to reduce the human effort required in finding appropriate task and LM-specific prompt templates, more recent works aim to automate the process of prompt designing. One of the popular ways is to search for templates in the discrete space of tokens (Jiang et al., 2020; Haviv et al., 2021; Yuan et al., 2021). These are known as hard prompts. Another popular line of research aims at learning prompt token embeddings in a continuous vector space, i.e., soft prompts (Li and Liang, 2021; Lester et al., 2021). Our work is an extension of the parameter-efficient soft prompt tuning framework,

[*]Equal contribution.
[†] This work was performed while the first author was interning at Salesforce Research. Correspondence to: <amrita.saha@salesforce.com>
[1]Our implementation of VIP is made publicly available https://github.com/declare-lab/VIP

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 6776–6791
December 7-11, 2022

PT, which we describe below.

Given a frozen PLM, the soft prompt tuning method PT (Lester et al., 2021) aims to learn a set of soft tokens (vectors) to solve a downstream task. There is a critical limitation of this approach, that is, the learned prompt tokens are expected to perform well across the samples of the task. Thus they are static against the change in input. Henceforth, we refer to them as static tokens. We hypothesize that learning to attend and adapt the prompt tokens to the input provides more flexibility to the system to find the distribution of prompts that generalizes well to diverse forms of the input. To address the limitation of PT, i.e., input-inadaptability of static tokens, we introduce a novel soft prompt tuning technique, **Vector-quantized Input-contextualized Prompts** (VIP), with two integral properties that are our main contributions:

- **Prompt Contextualization.** We propose a "contextualizer"– a transformer-based sentence encoder that generates input-adapted prompt tokens from the input text and static soft prompts. While there has been some recent interest in input-dependent prompt generation (Levine et al., 2022; Clive et al., 2021; Wu et al., 2022), we observe that a straightway utilization of the contextualized tokens as a replacement for static tokens leads to performance degradation (VIP-C in Table 3). The potential cause is training instability due to noise in the contextual prompt representations that propagates through the frozen PLM and leads to erroneous predictions.

- **Prompt Quantization.** We tackle the aforementioned challenge by introducing a prompt "Quantizer" to reduce the noise in the contextual prompt representations. The quantizer discretizes the continuous space of contextualized prompts, thus allowing us to control its representational capacity. It maps each contextual prompt token to a set of learnable vectors called codebook. These mapped vectors are used as final quantized prompts, with the codebook acting as a parametric discrete latent variable model over the contextual prompt space. Therefore, it is worth noting, quantization limits the space of contextual prompts based on the representation capacity of the codebook vectors.

Similar to the other prompting methods, the original input text is concatenated with the obtained quantized (contextual) prompt tokens and subsequently fed into the frozen PLM. The contextualizer and static tokens are learned using backpropagation to minimize the task-specific prediction loss. Whereas the codebook vectors are learned using a more stable update method of exponential moving average (Roy et al., 2018; Angelidis et al., 2021).

Empirically, we demonstrate the utility of our proposed approach VIP on a wide range of NLP tasks. On SuperGLUE, QA, Relation Classification, NER and NLI, VIP improves over the baseline soft prompt tuning method PT by an average margin of 1.19%. Additionally we find that VIP outperforms PT by a margin of 0.6% - 5.3% on out-of-domain QA and NLI tasks respectively and by 0.75% on Multi-Task setup over 4 different tasks. Overall, our experiments showcase that VIP learns more richer robust prompt representations than static soft prompts PT.

2 Problem Formulation

Following Raffel et al. (2019), we cast all the tasks into a text-to-text format. Therefore, tasks such as text and relation classification are reformulated as text generation and the model is expected to generate the class label y. Formally, given a pretrained Language Model (PLM) parameterized by θ, we formulate a task $\mathcal{T}$ as conditional text generation $\Pr_\theta(\mathrm{Y} \mid \mathrm{X})$ where X and Y are respectively the sequences of token embeddings corresponding to the texts x and y, generated by the tokenizer provided by the frozen PLM.

The prompting approach of Lester et al. (2021) prepends X with the set of learnable soft tokens $P = \{p_1, \ldots, p_n\}$. Soft tokens p_i are vectors that lie in a d-dimensional continuous space $\mathbb{R}^{\mathrm{d}}$. The idea is to condition the model output Y directly on P and X as $\Pr_\theta(\mathrm{Y} \mid P, \mathrm{X})$, where the static vectors in P are conditionally independent of X given task $\mathcal{T}$ i.e. $\Pr(P \mid \mathrm{X}, \mathrm{T}) = \Pr(P \mid \mathrm{T})$. T denotes the set of tokens obtained from task-specific attributes such as task description and class labels. However, such a setting restricts the system to find a singular static set of prompt tokens that is effective for all the diverse samples of a given task.

Motivation. We hypothesize that to solve a complex language understanding task, the dependence of prompt on input X is a critical characteristic that aids in generalization over the unseen in and out-of-domain samples. In the parameter-efficient prompt tuning methods such as Lester et al. (2021),

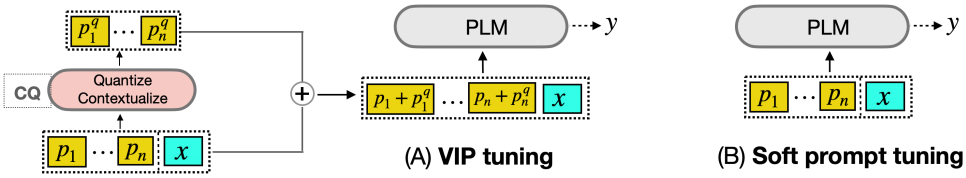

Figure 2: Comparison of VIP with baseline PT. A) **VIP** tuning denotes the proposed approach where CQ module contextualizes and quantizes the standard soft prompts. The quantized prompts are then added to the soft prompts through a skip connection. B) **Soft Prompt tuning** (PT) (Lester et al., 2021) is the soft prompt tuning which learns the vectors p_i through backpropagation.

the PLM is not allowed to tune its parameters and learn task-specific contextualization. This motivates us to learn compact prompt representations in VIP that can encode task-specific information and contextual information from each of its samples. We represent VIP tokens by the set of vectors $P^q = \{p_1^q, \ldots, p_n^q\} \in \mathbb{R}^{\mathrm{d}}$, which unlike static tokens P, holds the desirable property $\Pr(P^q|\mathrm{X}, T) \neq \Pr(P^q|T)$.

3 Methodology

In order to obtain input-dependent soft prompts P^q, we start with trainable static prompt tokens P similar to Lester et al. (2021). P along with the original input token sequence X is then fed to a parameterized submodule called CQ, which performs Contextualization and Quantization. Formally, $P^q = \mathrm{CQ}_\phi([P, \mathrm{X}])$ where ϕ is the parameter set of CQ module. Our final prompt token representation is obtained by combining the CQ module's output P^q and the static tokens P through a skip connection i.e. $P + P^q$. This we substitute in place of P in $\Pr_\theta(\mathrm{Y}|P, \mathrm{X})$ from Lester et al. (2021)

$$\Pr(Y|T, P, P^q, \mathrm{X}) \stackrel{\text{def}}{=} \mathrm{PLM}_\theta([\mathrm{T}, P + P^q, \mathrm{X}]). \quad (1)$$

Next, we elaborate on functioning of the CQ module, which performs **Contextualization** followed by **Quantization** of the soft prompts and is the primary contribution of this work.

3.1 Contextualization & Quantization (CQ)

CQ expects the input as a sequence of token embeddings, by prepending input X with the input-agnostic soft tokens P. As a first step, it performs the token contextualization described below:

3.1.1 Prompt Contextualization

To reduce the number of trainable parameters and perform meaningful contextualization that is coherent with the input space of PLM, we utilize non-tunable input embedding of PLM for mapping tokens in X to respective vectors. Then, as Figure 3 shows, $[P, \mathrm{X}]$ is passed through a trainable transformer-based sentence encoder (Vaswani et al., 2017). For a given task $\mathcal{T}$, P acts as a set of constant vectors. Through the attention mechanism, the encoder fuses P with the context information from X. The output of the sentence encoder is an input-contextualized set of prompt tokens P^c.

Sentence encoder. To keep the number of trainable parameters low, we perform contextualization in a lower dimensional space. We project $P \in \mathbb{R}^{n \times d}$ from d to d' dimensions (d'<d) to obtain $P_l \in \mathbb{R}^{n \times d'}$. P_l is subsequently fed into a two-layer transformer encoder layers having four attention heads and dense layers of dimension $\{d' \times 2d', 2d' \times d'\}$. The output of this transformer encoder is projected back from d' to d-dimension space to get the contextualized prompt token representation P^c. The input and output projections are also dense layers with trainable parameters. In our experiments, we have d=768 and d'=32. The sentence encoder is trained from scratch since it is significantly smaller (86K parameters) than standard PLMs.

3.1.2 Prompt Quantization

The continuous token embeddings P^c at the output of the sentence encoder suffer from high variance in their representations across diverse inputs for a given task, owing to their direct dependence on X. This high variance behaves as noise for the frozen PLM, resulting in unstable performance and poor generalization. Another potential limitation of directly utilizing the contextual tokens as prompts is representation collapse resulting in P^c becoming a constant embedding ignoring the input context We found that the posterior collapse issue persists unless specifically treated, particularly due to the downstream PLM being frozen.

This inspires us to learn a compact quantized representation P^q of the contextual prompt P^c, fol-

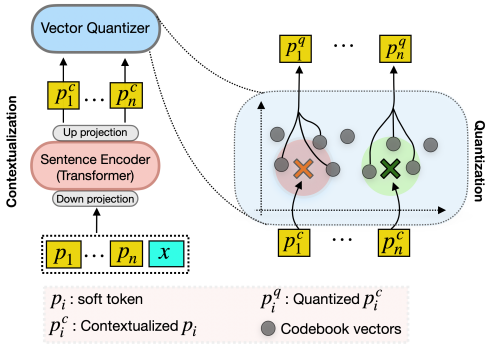

Figure 3: Contextualizer and quantizer in VIP.

lowing Angelidis et al. (2021); van den Oord et al. (2017). Vector quantization achieves a desirable discretization of the latent sentence representations which has been shown to be particularly suitable for language understanding tasks in (Roy et al., 2018; Roy and Grangier, 2019; van den Oord et al., 2017; Mercatali and Freitas, 2021) owing to the inherent discrete nature of text.

Vector quantizer. We maintain a set of K learnable codebook vectors $e \in \mathbb{R}^{K \times d}$ shared across all the contextualized soft tokens in P^c. The quantization is independently performed on each $p_i^c \in P^c$, by sampling a mixture of codebook entries from e that are closest in representation to p_i^c in order to obtain the corresponding quantized prompts p_i^q. For this, we first define logits l_i^k for the i_{th} token p_i^c as negative of its squared euclidean distance from the k_{th} codebook vector. We then sample, with replacement, m latent codes for each prompt token p_i^c, from a Multinomial distribution over the logits l_i, which results in the Soft-EM training followed by Angelidis et al. (2021); Roy et al. (2018).

$$l_i^k = -\frac{1}{\tau}||p_i^c - e_k||_2^2 \tag{2}$$

$$z_i^1, \ldots, z_i^m \sim \text{Multinomial}(l_i^1, \ldots, l_i^K) \tag{3}$$

The quantized p_i^c, i.e., p_i^q can be computed by averaging over the m samples of codebook entries obtained from the Multinomial sampling. p_i^q's are discrete latent variables that correspond to centroid of the cluster of the codebook entries closest to p_i^c.

$$p_i^q = \frac{1}{m}\sum_{j=1}^{m} e_{z_i^j} \tag{4}$$

The contextualization and quantization processes are shown in Figure 3. For experimental stability during the initial phase of training, we scale the logits by a temperature hyperparameter τ in order to encourage fair codebook usage when sampling using Equation (3). In our experiments τ is fixed to 100 across all tasks and datasets. It is chosen based on the norm of the codebook vectors in the initial training iterations, which is typically of order 10^2-10^3 for our initialization.

Learning a compact codebook (as described in 3.2) and setting the τ constant allows the quantizer to explicitly control and bottleneck the information capacity of prompt representations P^q before feeding it to the PLM, thus avoiding posterior collapse. Also, this sampling process ensures that for semantically similar inputs, the quantized prompts will also have similar representations, thus reducing its representation variance w.r.t the input. This consequently helps in stabilizing the frozen PLM's performance, as is evident from our empirical results. Essentially quantization serves as a trade-off between input-agnostic soft prompts and input-contextualized soft prompts, by sampling prompts from a limited set of learnable codebook vectors.

Skip connection over CQ. Roy and Grangier (2019) illustrated the importance of having a skip connection over the input and output of the quantization bottleneck network. Following this, we also add a skip connection over CQ module, combining its input and output i.e. $P + P^q$ in order to obtain the final Vector-quantized Input-contextualized Prompt tokens. It is worth noting that the skip connection allows us to effectively fuse the learned information from the input-dependent aspects captured by P^q with the static input-agnostic prompt P. Empirically we observe that the information bypass leads to more stable performance i.e. lower variance across random initializations of prompt embedding and sentence encoder parameters.

3.2 Training

The sentence encoder, soft tokens P, and codebook vector comprise the set of trainable parameters while we freeze the parameter set of PLM. Our training objective minimizes two losses: i) cross-entropy loss between the output generated by the frozen PLM and the ground truth sequence of tokens and ii) commitment loss which encourages an encoder output to commit to a set of closest codebook vectors, without fluctuating too much. This is important in order to ensure the volume of codebook embedding space does not grow arbitrarily.

$$\text{L} = \text{L}_{CE} + \beta \sum_{i=1}^{n} ||p_i^c - \text{sg}(p_i^q)||_2^2 \tag{5}$$

where L_{CE} denotes the cross-entropy loss, sg refers to the stop gradient operator such that $\mathrm{sg}(x) = x$ in the forward pass and zero on the backward pass. β is a hyperparameter referred to as the commitment cost, which regulates the extent to which the encoder output should remain close to the sampled codebook vectors.

On back-propagation, the loss term will update the parameters of the sentence encoder and the prompt embedding layer that originally generated P. However, a crucial part of the learning involves updating the codebook which we described below.

Learning codebook. Following works van den Oord et al. (2017); Roy et al. (2018); Angelidis et al. (2021) that use codebook-based quantization, we adopt the Exponential Moving Average (EMA) style update of the codebook vectors. This has been shown to result in more stable training than gradient-based updates. For each batch, we perform the following two steps:

- Step 1 counts the number of times j_{th} codebook vector is sampled and updates the count c_j.

$$c_j = \lambda c_j + (1-\lambda) \sum_{i \in [n], k \in [m]} \mathbb{1}[z_i^k = e_j] \quad (6)$$

- Step 2 updates the embedding of j_{th} codebook vector e_j by taking the mean of sentence encoder outputs for which that codebook vector was sampled during $\mathrm{Multinomial}$ sampling.

$$e_j = \lambda e_j + (1-\lambda) \sum_{i \in [n], k \in [m]} \frac{\mathbb{1}[z_i^k = e_j] p_i^c}{c_j} \quad (7)$$

where λ is decay parameter set to 0.99, $\mathbb{1}$ is the indicator function, [n] denotes set $\{1, \ldots, n\}$.

While we choose this online K-means clustering style of quantization of the latent space, there are other discretization techniques e.g. using Gumbel softmax to select the most appropriate codebook vector. We avoid that in our work due to its notorious sensitivity to its temperature hyperparameter.

4 Experiments

Tasks. We base our experiments on a wide range of language understanding tasks including natural language inference, question answering of extractive and multiple choice nature, relation classification, and named entity recognition.

- **SuperGLUE** (Wang et al., 2019). To gauge general language understanding capabilities of VIP against baselines, we evaluate on SuperGLUE benchmark using the prescribed evaluation metrics. It comprises of eight tasks—BoolQ, CB, COPA, MultiRC, ReCoRD, RTE, WiC, WSC.
- **QA** (Fisch et al., 2019). Here we focus on two types of QA tasks: i) *Extractive QA:* MRQA 2019 shared task consists of multiple question-answering datasets. Six datasets are dedicated each for training and testing while a small portion of training datasets is held-out for model selection. It focuses on evaluating the generalization of a QA system, i.e., how well it performs on out-of-domain examples. ii) *Multiple choice QA:* RACE-middle (Lai et al., 2017) task to predict the correct answer from four given candidates. Other datasets of this category, MultiRC, and ReCoRD, are incorporated in SuperGLUE.
- **Relation classification**. Relation classification aims to identify the semantic relation between two specified entities in a sentence. For this task, we use two datasets—SemEval 2010 task 8 (Hendrickx et al., 2010) and TACRED (Zhang et al., 2017). SemEval is a 19-way classification problem, while TACRED is a larger-scale dataset consisting of 41 relation types.
- **NER**. For named entity recognition, we use the English language dataset of CoNLL-2003 shared task (Tjong Kim Sang and De Meulder, 2003). We frame the same text-to-text format suitable for T5 as in Qin and Joty (2022).
- **NLI**. We use Adversarial NLI (ANLI) that is a large-scale benchmark dataset to gauge natural language inference capabilities of the system (Nie et al., 2019). We further evaluate generalization of the model trained on the challenging ANLI task to out-of-domain NLI datasets CB, AXB, AXG, RTE and SNLI.

Baselines. We consider the following baselines.

- **FT** (Raffel et al., 2019). It refers to the standard task-specific fine-tuning of PLM.
- **Adapter** (Houlsby et al., 2019). It inserts learnable modules between layers of a pretrained transformer to perform transfer learning from PLMs to the downstream tasks.
- **PT** (Lester et al., 2021). Soft prompt tuning (PT) is the primary baseline for our work. As shown in Figure 1, PT prepends a set of continuous vectors P to X and feeds it to PLM. P is learned via backpropagation through the frozen model. Thus P relies only on the task at hand and choice of PLM while being independent of X.
- **VIP-IDP**. Instance-Dependent Prompt Generation Method (IDPG) is a recent approach similar

Model	CB	COPA	WSC	RTE	WiC	BoolQ	MultiRC	ReCoRD	Avg
	$acc./F1$	$acc.$	$acc.$	$acc.$	$acc.$	$acc.$	$EM/F1_a$	$EM/F1$	
FT	$92.9_{\pm 3.69}$	$57.0_{\pm 1.0}$	$63.5_{\pm 0.1}$	$78.1_{\pm 1.74}$	$71.9_{\pm 0.7}$	$79.8_{\pm 0.23}$	$77.6_{\pm 0.31}$	$72.3_{\pm 0.32}$	$74.2_{\pm 0.99}$
Adapter	$92.7_{\pm 1.42}$	$56.7_{\pm 1.53}$	$65.6_{\pm 3.58}$	$79.1_{\pm 1.29}$	$68.9_{\pm 0.57}$	$79.3_{\pm 0.8}$	$77.7_{\pm 0.98}$	$70.2_{\pm 1.76}$	$73.8_{\pm 1.49}$
PT	$\mathbf{75.8}_{\pm 3.61}$	$59.0_{\pm 2.0}$	$66.7_{\pm 1.48}$	$70.8_{\pm 2.25}$	$67.4_{\pm 0.78}$	$69.3_{\pm 0.6}$	$69.6_{\pm 0.55}$	$63.3_{\pm 1.10}$	$67.7_{\pm 1.55}$
VIP-IDP	$37.6_{\pm 28.4}$	$53.0_{\pm 23.7}$	$66.7_{\pm 1.1}$	$49.8_{\pm 28.2}$	$61.2_{\pm 8.16}$	$62.6_{\pm 0.55}$	$52.8_{\pm 3.21}$	$18.2_{\pm 4.91}$	$57.0_{\pm 12.28}$
VIP-C	$73.4_{\pm 2.39}$	$57.0_{\pm 1.0}$	$65.1_{\pm 1.43}$	$70.9_{\pm 1.5}$	$65.4_{\pm 3.12}$	$69.2_{\pm 0.71}$	$69.7_{\pm 0.35}$	$63.7_{\pm 1.65}$	$66.7_{\pm 1.52}$
VIP	$75.5_{\pm 2.63}$	$\mathbf{62.7}_{\pm 0.57}$	$\mathbf{68.7}_{\pm 1.30}$	$\mathbf{72.1}_{\pm 0.87}$	$\mathbf{68.0}_{\pm 0.61}$	$\mathbf{69.6}_{\pm 1.07}$	$\mathbf{70.5}_{\pm 0.40}$	$\mathbf{65.2}_{\pm 0.81}$	$\mathbf{69.1}_{\pm 1.03}$

Table 1: Performance of VIP vs. baseline models on SuperGLUE development set. $acc.$, EM, $F1$, $F1_a$ and $acc./F1$ denote accuracy, exact match, macro and micro F1 scores and average of accuracy and macro F1 respectively. The numbers in subscript show the standard deviation across 3 random trials using different seeds.

Model	Rel. classification		NER	NLI
	SemEval	TACRED	CoNLL	ANLI
	$F1_a$	$F1_a$	$F1_a$	$F1$
FT	84.8	87.7	90.2	49.8
Adapter	84.4	85.8	89.9	47.7
PT	70.9	83.5	87.1	41.6
VIP-IDP	68.95	79	87.2	29.8
VIP-C	71.8	83.6	87	39.3
VIP	**72.4**	**84.4**	**87.4**	**43.2**

Table 2: Performance comparison on relation classification, NER and NLI tasks.

to our idea of making prompt tokens dependent on the input (Wu et al., 2022). IDPG cannot be directly treated as a baseline, since it uses RoBERTa-large as PLM and has only restricted applicability to text classification tasks, where it fine-tunes a classifier head over the frozen PLM. Instead, we adopt the IDPG-style prompt generation with our backbone T5-base model and reduce the trainable parameters to be comparable to ours. This version, named VIP-IDP, fetches the input representation from a frozen T5-base encoder and applies two nonlinear projections - first to a low dimension and then upscaling to a space of dimension 768*n to generate n prompt tokens. One limitation is that this network requires two forward passes through the T5-encoder.

- **VIP-C**. This is an ablation of our VIP architecture, obtained by removing the quantization network. The contextualized prompt tokens from the output of the sentence encoder, P^c, are directly augmented to X and fed to the frozen PLM.

- An important point to note is that while quantization is meaningful for input-dependent prompts, it cannot be applied to standard soft prompts where the prompt tokens have fixed embedding across all input instances of a task.

In results Tables 1 to 3, bold numbers represent the best performing *prompt*-based model, as FT and Adapter enjoy a much larger parameter size. For cross-domain QA, NLI, and multi-task settings we compare VIP against only primary baseline PT.

Experimental settings. To compare with PT, we base our experiments on the LM-adapted version of T5-base encoder-decoder. We refer to Lester et al. (2021) for finding the best prompt length and fix the prompt token length to $n = 100$ for PT, VIP, VIP-IDP. Our quantizer comprises 1000 codebook vectors and m in the multinomial sampling is fixed to 10. We select commitment cost $\beta = 0.1$ through grid search over $\{0.01, 0.1, 1\}$. We relegate precise details to Appendix A.4.

4.1 In-domain performance

In Table 1, we report the development set scores on SuperGLUE. Following Lester et al. (2021), we report results averaging across three random seeds. We find FT and Adapter with a large number of trainable parameters show significantly better performance than prompt-based methods on the three largest datasets, i.e., ReCoRD, BoolQ, and MultiRC. However, these models perform slightly worse than prompt-based methods on COPA which is a small-scale dataset with only 400 training samples. COPA is a difficult task that requires commonsense causal reasoning, hence reliable tuning a large number of parameters will require a large number of training samples. On the other hand large parameter-sized models can be tuned with less number of samples to solve simpler tasks such as textual entailment in CB.

VIP-IDP's performance ranks poorest amongst all models across most of the SuperGLUE tasks

Model	SQuAD	NewsQA	TriviaQA	SearchQA	HotpotQA	Natural QA	RACE-M	Parameter
FT	83.2	65.6	80.7	77	61.6	71.1	73.6	250 M
Adapter	81.4	63.9	79.3	75.1	60.1	71.6	69.9	2.5 M
PT	77.5	59.3	53.4	77.6	72.9	69.6	63.4	0.1 M
VIP-IDP	73.9	55.6	14.8	71.5	62.5	64.4	50.5	0.9 M
VIP-C	77.8	59.6	52.3	77.6	73.1	68.5	62.5	0.1 M
VIP	**78.7**	**61.1**	**53.9**	**77.9**	**73.5**	**69.7**	**65.9**	0.9 M

Table 3: Performance comparison (using F1 score) on Extractive QA from MRQA and multichoice QA RACE-M.

Train domain	Test domain Results in form of F1-Scores of {PT / VIP}						
	BioASQ	DROP	DuoRC	RACE	RE	TextbookQA	Micro-Avg.
SQuAD	54.3 / **54.7**	29.0 / **34.4**	**35.9** / 35.1	**42.6** / 40.4	78.7 / **80.0**	**20.8** / 16.6	48.9 / **49.3**
NewsQA	**50.9** / 49.4	24.9 / **25.5**	36.7 / **36.9**	39.4 / **39.6**	**73.7** / 73.0	**38** / 37.6	**48.8** / 48.4
TriviaQA	43.2 / **46.5**	17.1 / **18.1**	29.4 / **29.5**	27.6 / **29.3**	**55.5** / 54.1	27.7 / **29.7**	37.2 / **37.9**
SearchQA	42.9 / **48.3**	19.0 / **24.3**	**27.5** / 26.4	**21.9** / 17.8	**58.9** / 56.8	29.8 / **32.7**	38.1 / **39.2**
HotpotQA	**54.1** / 53.9	33.6 / **34.2**	33.7 / **35.0**	36.0 / **37.2**	79.1 / **80.0**	21.1 / **23.5**	49.0 / **50.0**
NaturalQA	**52.7** / 52.3	29.2 / **31.8**	29.8 / **30.5**	**35.3** / 32.8	75.8 / **77.5**	27.7 / **28.3**	47.4 / **48.3**

Table 4: Out-of-domain performance comparison (using F1 Score) of VIP vs. primary baseline PT on MRQA.

Model	CB	AXB	AXG	SNLI	RTE
	acc./F1	*acc.*	*acc.*	*F1*	*acc.*
PT	43.1	63.8	49.9	62.6	70.2
VIP	**59.7**	**66.1**	**51.9**	**66.7**	**71.8**

Table 5: ANLI out-of-domain evaluation.

while also suffering a very high variance across seeds. This is possibly due to straightaway feeding the input-dependent prompts into the frozen PLM. This variance issue is greatly alleviated in VIP by combining the input-agnostic representation P with the quantized representation P^q.

VIP outperforms the prompt-based baselines on seven out of eight SuperGLUE tasks, while also achieving the lowest variance across all prompt-tuning models. We also notice the drop in performance when the quantization bottleneck is removed (VIP-C). We posit that directly using the sentence encoder's output results in a high variance of the prompt representation w.r.t the input leading to training instability. The performance difference between VIP and VIP-C is higher for the tasks with lesser training samples such as CB, COPA, and WSC where underfitting is likely, leading to high performance variance.

Likewise in Table 2, we observe that VIP outperforms all other prompt-based baselines on all three tasks - relation classification, NER and NLI. Particularly on NLI, the improvement is very significant – 1.5% better than PT and 4-14% better than VIP-IDP and VIP-C, while on other tasks it improves by 1% over the prompt baselines.

We next compare our models on question answering. In MRQA shared task, since the test domain is different from the train and development domain, we perform in-domain testing of models on the development data. For model selection, we held out 10% from the training set. Table 3 compares performance on QA datasets from MRQA task and multi-choice based RACE-M. Across the board, VIP proves superior to all prompt baselines - outperforming PT and VIP-C by an average margin of 1% and VIP-IDP by over 8%. However, fine-tuning (FT) and Adapter consistently outperform all prompt models, owing to their large parameter size and due to using T5-base as PLM.

In Tables 1 to 3, by comparing VIP with contextualized soft prompt VIP-C and VIP-IDP on 20 datasets across 5 tasks, we empirically substantiate the necessity of a capacity bottleneck.

4.2 Out of domain performance

We next evaluate VIP and its primary baseline PT on out-of-domain (OOD) MRQA test set. In 5 out of the 6 training domains, the micro-average OOD test performance of VIP is better than PT, whereas, on NewsQA, both perform comparably. The more fine-grained results show that on all the training domains VIP performs better than PT in at least half of the OOD test datasets. Similarly on all the 6 test datasets VIP consistently outperforms PT in at least half of the training domains.

Next, in Table 5 we present the cross-domain performance of the ANLI model trained using VIP and PT with model selection done on ANLI devel-

Model	ANLI	RTE	TACRED	SemEval	ReCoRD	RACE-M	SQuAD	TriviaQA	NewsQA	SearchQA	HotpotQA	Nat. QA	Avg.
PT	27.5	60.3	**81.7**	72.6	53.3	57.7	**76.1**	**48.7**	57.0	73.8	69.1	64.7	61.9
VIP	30.6	62.5	80.9	**72.9**	**55.3**	57.9	**76.1**	48.4	**57.9**	**74.2**	69.7	64.8	62.6
VIP: NR	38.5	**66.4**	80.8	67.0	53.5	**59.3**	75.1	48.0	**57.9**	73.1	**69.8**	**65.0**	**62.9**
VIP: NR+DC	**39.1**	56.3	79.4	64.9	53.0	58.6	75.5	48.2	57.5	73.5	**69.8**	64.6	61.7

Table 6: Performance comparison of VIP and its ablations (from Section 4.4) against PT on multi-task setting.

opment set. The test domains consist of the test set of SNLI corpus (Bowman et al., 2015) and AXB and AXG and the combined train and development set of RTE from SuperGLUE. We observe that VIP achieves significantly better results on all the OOD NLI tasks as compared to PT. These results empirically justify that the quantized contextual prompts learned by VIP indeed have a richer and more robust representation than the static soft prompts.

4.3 Multi-task performance

We also perform multi-task experiments considering tasks from NLI (ANLI and RTE), extractive QA (MRQA train set), multi-choice QA (ReCoRD and RACE-M), and relation classification (SemEval and TACRED). For each task, we choose upto a maximum of 10K samples at random for both the training and validation set. For evaluation, we consider the complete standard test split of all the datasets. For SuperGLUE-based tasks RTE and ReCoRD, we consider the test on the validation set. We run the experiments for 50K steps, performing validation at every 2K step with an early stopping set to 5. The validation is done on each dataset separately and the model is chosen based on the best mean validation score. As shown in Table 6, VIP outperforms its primary baseline PT in 10 out of 12 tasks and achieves an overall 0.7% better mean performance. This indicates that our proposed CQ module is indeed able to learn more robust soft prompts. We posit that a lower score on TACRED and TriviaQA is due to the average validation performance-based model selection strategy.

4.4 Further Experiments

Dedicated codebook. In this ablation, instead of having a shared codebook across all prompt tokens, we assign a dedicated codebook matrix (of dimension $\frac{K}{n}$) to each of the n prompt tokens.

Noise resilience. We also employ noise resilience training of the sentence encoder (Gao et al., 2021) which outperforms on several tasks as discussed later. We pass the same input to the sentence encoder twice to obtain two sets of representations

Model	ANLI	TACRED	RACE-M	SQuAD	TriviaQA	Nat. QA
PT	41.6	83.5	63.4	77.5	53.4	69.6
VIP	43.2	84.4	65.9	78.7	53.9	69.7
VIP:NR	43.6	**85.0**	63.4	78.8	**54.5**	69.9
VIP: NR+DC	**44.5**	84.6	**66.0**	**79.4**	54.3	**70.2**

Table 7: Ablations of VIP with additional Noise Resilience (NR) & Dedicated Codebook (DC) training.

of the same batch. For a batch with B samples, we define noise as the similarity between two sentence encoder representations (p^c), by taking the negative of their Euclidean distance. With i and i^+ representing the two samples obtained by feeding an input instance i twice to the encoder, the sample-wise noise resilience loss l_i is defined to be

$$l_i = -\log\left(\frac{e^{\text{sim}(i,i^+)}}{\sum_{(i\in[B],j\in[B])} e^{\text{sim}(i,j)} + e^{\text{sim}(i,j^+)}}\right), \quad (8)$$

The average sample-wise noise resilience loss over the batch is added to the loss term in Equation (5). Table 6 and Table 7 show adding noise resilience and dedicated codebook boosts performance in ANLI, RACE-M, SQuAD, and Nat. QA, while TACRED, TriviaQA and multi-task settings perform better with only noise resilience-based training.

5 Discussions

Training-time and memory overhead. As compared to PT, we did not observe an adverse impact of VIP on training time even with more trainable parameters, and notice a similar convergence pattern. The number of training steps required on SuperGLUE datasets in order (Dataset, PT, VIP) are—(WSC, 180, 90), (CB, 320, 300), (COPA, 377, 442), (WiC, 4080, 1530), (RTE, 546, 760), (BoolQ, 5015, 2360), (RACE-M, 17500, 22000), (MultiRC, 5000, 6000), (ReCoRD, 13500, 19500), where per-iteration time-overhead of VIP is 0.07 seconds (1.96 (VIP) vs 1.89 (PT)). Thus, for WSC, CB, WiC, BoolQ, VIP converges faster. Regarding memory overhead, VIP (97.7MB) is slightly worse (3.3MB) than PT (94.4MB), with the main memory bottleneck being the common frozen T5-base PLM (892MB). While codebook in VIP increases the parameter size, it can be shared across multiple

tasks, thus reducing the parameter overhead. In our multi-task experiment, VIP with codebook shared across 3 tasks still outperforms the PT baseline.

Learning codebook for small datasets. Even on small SuperGLUE datasets such as CB, COPA, and WSC with only 200-400 training instances, VIP performs an average 1.8% better than PT. Furthermore for CB, the codebook visualization and interpretability analysis (Appendix A.3, Figure 4) shows that VIP indeed learns meaningful representations even for small scale datasets.

Comparison with L2 regularization. While L2-regularization can also limit the capacity of prompt space, quantization achieves a clustering over the contextualized prompts with the discrete latent variables (codebook vectors) being the clusters. This encourages the quantized prompts to have similar representations for semantically similar inputs, which can neither be achieved in VIP-C nor through L2-regularization. Experiments with L2-regularization did not show any performance gain, in fact, it degraded the performance for a large regularization constant. On SuperGLUE datasets CB and COPA, our pilot-study in the standard PT setting yielded these results in the form (L2-reg constant, Accuracy): For CB-(0,75.8), (1e-4,66.78), (1e-3,70.933), (1e-2,66.77), (1e-1,23.71). For COPA-(0,59.0), (1e-4,53.99), (1e-3,59.20), (1e-2,54.34), (1e-1,45.24). Lastly, learning the codebook can also lead to better interpretability of the prompt space (Appendix A.3, Figure 4).

Comparison with Adapters. Adapters fall under a different genre of parameter efficient tuning that require amending the PLM to add tunable intermediate layers. This makes them more complicated and expensive than prompt-tuning methods from practical implementation and usage perspective. In general, they are also much less parameter efficient than prompt tuning-based approaches, such as PT and VIP, and hence not directly comparable to them as one of the main baselines.

Comparing by making PT/VIP parameter-size more similar. Increasing the number of parameters in PT, i.e., prompt length > 100 shows adverse effects on model performance as mentioned by PT paper (Figure3-a of Lester et al. (2021)) and also observed in our experiments. In Appendix Table 8, we investigate different-sized VIP models by varying number of VIP-prompt-tokens and codebook-size, and each of these models still perform better than the best-setting of PT on SuperGLUE.

Impact of prompt quantization. We observed representation collapse empirically when the model ignores input during the prompt generation step, resulting in prompt representations collapsing to a single fixed embedding. We observed VIP-C (or removing quantization from VIP) is highly prone to this as the contextual prompt representations in the sentence encoder's output collapse to a constant embedding for several random seeds. VIP's Quantizer alleviates this by limiting the capacity of the prompt space, achieving a clustering over the contextual prompts through the codebook.

6 Related work

In this section, we briefly compare with recent prompt-based studies which particularly use dynamic prompts. Clive et al. (2021) proposes a variant of prefix-tuning (Li and Liang, 2021) which augments input-dependent activations to each layer, resulting in a much larger parameter size compared to us. Similarly Levine et al. (2022) also learns a cross-attention network of size 25M over a frozen T5-encoder to generate input-specific prompts, leading to 25x larger parameter size than VIP. (Wu et al., 2022) is another input-dependent prompt generator that we adapt to our setting as VIP-IDP and compare as a baseline. Jin et al. (2022) re-scales the soft prompt tokens based on their attention over input tokens, but also fine-tunes the PLM, hence they cannot be considered in a parameter-efficient model tuning regime.

7 Conclusion

We propose a novel extension of soft prompt tuning (PT) - Vector Quantized Input-Contextualized Prompt Tuning (VIP), designed to have two desirable characteristics - (i) contextualizing the soft prompt tokens w.r.t input text using a learnable sentence encoder (ii) discretizing the contextual prompts using a Vector Quantization network. On an extensive set of language understanding tasks - SuperGLUE, QA, NLI, NER, and Relation Classification, VIP outperforms PT baseline. Further, our generalization studies on out-of-domain evaluations of QA and NLI and multi-task settings over 4 tasks also show that VIP is able to learn richer and more robust prompt representations than PT.

8 Limitations

In this section, we point out the limitations of VIP and its potential future directions.

- **Pretraining prompt contextualizer**. The sentence encoder in VIP is trained from scratch for each downstream task. However, following the prompt pre-training proposed in Gu et al. (2021), a possible future work is to pretrain the prompt contextualizer in a task-agnostic way.

- **Larger parameter size**. VIP framework demands a larger parameter size than the baseline soft prompt tuning, owing mainly to the codebook. In Appendix A.2 we show that by reducing the number of VIP-prompt tokens and codebook-size, we can reduce the parameter size to one-third while compromising performance slightly on SuperGLUE. More extensive experimental analysis and better techniques for compressing the codebook, we leave as future work.

- **More hyperparameters**. Other than the standard hyperparameters of the sentence encoder, the quantizer introduces new hyperparameters - codebook-size, multinomial sample size, and the temperature constant τ to scale logits. While VIP needs additional hyperparameters, in all our experiments across 20 training datasets from 5 tasks, we fix all hyperparameters related to codebook and sentence-encoder. This shows that our model is indeed not sensitive to the hyperparameters and does not need very specific tuning for each task/setting.

- **Training challenges**. Learning the codebook requires an EMA style updating scheme instead of the standard gradient update. With the PLM being frozen, this needs more careful handling - for e.g. a critical hyperparameter is the value of the temperature constant τ. A very high value can lead to representation collapse of the codebook while very low values can lead to sparse codebook usage. However, as discussed above, τ is independent of the task and depends on the initial norm of codebook vectors.

- **Impact on small-scale datasets**. We posit that due to the larger parameter size of VIP, it performs worse than PT in tasks with lesser training data, e.g. scores on the CB dataset in Table 1. This is due to the larger parameter size of VIP. Indeed, by reducing the parameter size of VIP (in Appendix Table 8), we achieve much better performance on CB.

- **T5-base as backbone PLM**. Due to resource limitations, in all our experiments we use T5-base as the backbone. Following Lester et al. (2021) where larger PLMs are shown to improve prompt-tuning performance, we speculate VIP to showcase a similar effect. Also, though we use T5 as PLM in this work, our VIP architecture can be used in BERT or GPT style prediction or generation as well. However, a formal analysis of this is left as future work.

- **Data and model bias.** The language understanding tasks and datasets were predominantly in the English language, and thus limit our claims to the English language. Gender, age, race, and other socioeconomic biases may exist in these datasets, and models trained on these datasets may propagate these biases. It is likely that additional biases are also embedded within the T5-base PLM that was used as the backbone of VIP.

Acknowledgement

Soujanya Poria acknowledges the Ministry of Education, Singapore, under its AcRF Tier-2 grant (Project no. T2MOE2008, and Grantor reference no. MOET2EP20220-0017). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore.

References

Stefanos Angelidis, Reinald Kim Amplayo, Yoshihiko Suhara, Xiaolan Wang, and Mirella Lapata. 2021. Extractive opinion summarization in quantized transformer spaces. *Transactions of the Association for Computational Linguistics*, 9:277–293.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Jordan Clive, Kris Cao, and Marek Rei. 2021. Control prefixes for text generation. *arXiv preprint arXiv:2110.08329*.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, volume 23, pages 107–124.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *NAACL*.

Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Güney, Volkan Cirik, and Kyunghyun Cho. 2017. SearchQA: A new Q&A dataset augmented with context from a search engine. *arXiv:1704.05179*.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. Mrqa 2019 shared task: Evaluating generalization in reading comprehension. *arXiv preprint arXiv:1910.09753*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2021. Ppt: Pre-trained prompt tuning for few-shot learning. *arXiv preprint arXiv:2109.04332*.

R Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, volume 7.

Adi Haviv, Jonathan Berant, and Amir Globerson. 2021. Bertese: Learning to speak to bert. *arXiv preprint arXiv:2103.05327*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid 'O S'eaghdha, Sebastian Pad'o, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 33–38, Uppsala, Sweden. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Feihu Jin, Jinliang Lu, Jiajun Zhang, and Chengqing Zong. 2022. Instance-aware prompt learning for language understanding and generation. *arXiv preprint arXiv:2201.07126*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.

Aniruddha Kembhavi, Minjoon Seo, Dustin Schwenk, Jonghyun Choi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. In *CVPR*.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *TACL*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale reading comprehension dataset from examinations. In *EMNLP*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*.

Yoav Levine, Itay Dalmedigos, Ori Ram, Yoel Zeldes, Daniel Jannai, Dor Muhlgay, Yoni Osin, Opher Lieber, Barak Lenz, Shai Shalev-Shwartz, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2022. Standing on the shoulders of giant frozen language models.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *CoNLL*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Giangiacomo Mercatali and André Freitas. 2021. Disentangling generative factors in natural language with discrete variational autoencoders. In *EMNLP*.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2019. Adversarial nli: A new benchmark for natural language understanding. *arXiv preprint arXiv:1910.14599*.

Mohammad Taher Pilehvar and José Camacho-Collados. 2018. Wic: 10,000 example pairs for evaluating context-sensitive representations. *CoRR, abs/1808.09121*.

Chengwei Qin and Shafiq Joty. 2022. LFPT5: A unified framework for lifelong few-shot language learning based on prompt tuning of t5. In *International Conference on Learning Representations*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI spring symposium: logical formalizations of commonsense reasoning*, pages 90–95.

Aurko Roy and David Grangier. 2019. Unsupervised paraphrasing without translation. *arXiv preprint arXiv:1905.12752*.

Aurko Roy, Ashish Vaswani, Arvind Neelakantan, and Niki Parmar. 2018. Theory and experiments on vector quantized autoencoders. *arXiv preprint arXiv:1805.11063*.

Amrita Saha, Rahul Aralikatte, Mitesh M. Khapra, and Karthik Sankaranarayanan. 2018. DuoRC: Towards Complex Language Understanding with Paraphrased Reading Comprehension. In *ACL*.

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

Timo Schick and Hinrich Schütze. 2021. Few-shot text generation with natural language instructions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 390–402.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, et al. 2015. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC bioinformatics*, 16(1).

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural discrete representation learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6309–6318, Red Hook, NY, USA. Curran Associates Inc.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Zhuofeng Wu, Sinong Wang, Jiatao Gu, Rui Hou, Yuxiao Dong, VG Vydiswaran, and Hao Ma. 2022. Idpg: An instance-dependent prompt generation method. *arXiv preprint arXiv:2204.04497*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. Bartscore: Evaluating generated text as text generation. *Advances in Neural Information Processing Systems*, 34.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint arXiv:1810.12885*.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 35–45.

A Appendix

A.1 Dataset details

SuperGLUE (Wang et al., 2019). To gauge the general language understanding capabilities of VIP against baselines, we evaluate on SuperGLUE benchmark. It comprises of eight tasks—BoolQ (Clark et al., 2019), CB(De Marneffe et al., 2019), COPA (Roemmele et al., 2011), MultiRC(Khashabi et al., 2018), ReCoRD(Zhang et al., 2018), RTE(Haim et al., 2006), WiC (Pilehvar and Camacho-Collados, 2018), and WSC(Levesque et al., 2012).

MRQA (Fisch et al., 2019). MRQA comprises of multiple question answering datasets. Six datasets are dedicated each training—SQuAD (Rajpurkar et al., 2016), NewaQA (Trischler et al., 2017), TriviaQA (Joshi et al., 2017), SearchQA(Dunn et al., 2017), HotpotQA(Yang et al., 2018), NaturalQuestions(Kwiatkowski et al., 2019). Six separate datasets are kept for testing—BioASQ(Tsatsaronis et al., 2015), DROP(Dua et al., 2019), DuoRC(Saha et al., 2018), RACE(Lai et al., 2017), RelationExtraction (Levy et al., 2017), TextbookQA (Kembhavi et al., 2017). A small portion of training datasets is held-out for model selection.

NLI. We use Adversarial NLI (ANLI) which is a large-scale benchmark dataset to gauge the natural language inference capabilities of the system (Nie et al., 2019). We further evaluate the generalization of the model trained on the challenging ANLI task to out-of-domain NLI datasets CB, AXB, AXG, RTE (taken from SuperGLUE benchmark) and SNLI (Bowman et al., 2015).

A.2 Additional Analysis

Combining Soft and VIP Tokens for SuperGLUE Table 8 reports SuperGLUE development-set scores for different variants of the VIP model obtained by taking different combinations of soft prompt tokens and VIP tokens. We observe that for the configuration with all prompt tokens as VIP, i.e., input-contextualized and quantized, the best average performance is achieved while also having a very low variance across multiple runs.

Another notable observation is that the performance improves consistently as we increase the number of VIP tokens replacing static soft prompts. On the other hand, this also shows the trade-off between parameter size and performance Following this analysis, in our main experiments, we choose the configuration of VIP with 100 VIP-tokens and 0 soft prompt tokens.

A.3 Codebook Visualization

In this section we perform visualization and interpretability analysis on the learnt codebook, taking the CommitmentBank (CB) dataset (De Marneffe et al., 2019) as our setting. This is an NLI task where each sample is associated with one of 3 class labels ('neutral', 'entailment', 'contradiction'). For each instance in the CB development, we first compute the m sampled codebook vectors (i.e. $z_i^1, \ldots, z_i^m$ from Equation 3) that are used to construct each of the 100 VIP prompt tokens. Thus for each label across all instances in the development set, we can accumulate the information regarding which codebook vectors are sampled for instances of that label. With this, for each label we find the codebook vectors *dedicated* to that label i.e. the codebook vectors that are have high odds of getting assigned to that particular label than to any other label.

For each soft token, we thus select the three (i.e., number of classes) codebook vectors that are most *dedicated* to each of the 3 class labels. Note that this corresponds to codebook vectors with least-entropy label distribution and the mode of the distribution is that particular class label.

Figure 4 shows the t-SNE plot of 300 codebook vectors (3 labels $\times$ 100 prompt tokens) selected in this manner. Each codebook vector is color-coded by the corresponding class label it is *dedicated* to. We observe two characteristics of the learned codebooks:

1. codebook vectors of the same color are typically clustered together. These are label-specific codebook vectors that encode exclusive knowledge about that label. This clustering effect is naturally achieved through the EMA style learning of the codebook;

2. there are some clusters with codebook vectors that are dedicated to multiple labels. This we interpret as clusters of codes that capture general label-agnostic information coming from the input context or task description.

Model Selection in Multi-Task Setting Since the multi-task setting constructed by us involves sampling 10K instances randomly for the train and validation sets, we provide additional details on the

SuperGLUE	PT-(100,0)	VIP-(80,20)	VIP-(50,50)	VIP-(10,90)	VIP-(0,100)
# Tune param	76K	315K	546k	853K	930K
CB	$75.83_{\pm 3.61}$	$73.43_{\pm 0.50}$	$78.47_{\pm 3.89}$	$79.43_{\pm 4.44}$	$75.47_{\pm 2.63}$
COPA	$59.00_{\pm 2.00}$	$59.33_{\pm 0.58}$	$58.00_{\pm 2.65}$	$59.00_{\pm 1.73}$	$62.67_{\pm 0.57}$
WSC	$66.67_{\pm 1.48}$	$68.26_{\pm 0.95}$	$65.39_{\pm 0.01}$	$66.97_{\pm 1.16}$	$68.73_{\pm 1.30}$
RTE	$70.77_{\pm 2.25}$	$70.17_{\pm 1.46}$	$72.53_{\pm 1.29}$	$71.10_{\pm 0.70}$	$72.10_{\pm 0.87}$
WiC	$67.37_{\pm 0.78}$	$66.50_{\pm 1.37}$	$66.70_{\pm 1.51}$	$67.37_{\pm 0.25}$	$68.00_{\pm 0.61}$
BoolQ	$69.27_{\pm 0.60}$	$70.97_{\pm 0.71}$	$70.40_{\pm 0.44}$	$70.01_{\pm 0.38}$	$69.63_{\pm 1.07}$
MultiRC	$69.57_{\pm 0.55}$	$70.60_{\pm 1.01}$	$71.53_{\pm 0.29}$	$72.50_{\pm 0.87}$	$70.47_{\pm 0.40}$
ReCoRD	$63.33_{\pm 1.10}$	$65.47_{\pm 0.51}$	$64.37_{\pm 1.00}$	$64.70_{\pm 1.22}$	$65.23_{\pm 0.81}$
Avg	$67.72_{\pm 1.55}$	$68.09_{\pm 0.88}$	$68.42_{\pm 1.38}$	$68.89_{\pm 1.34}$	$69.04_{\pm 1.03}$

Table 8: SuperGLUE Performance of different VIP models obtained by combining u soft prompts and n VIP prompts denoted by (u, n) and codebook size of 10*n. Number of tunable parameters (denoted by # Tune params) varies across the models due to the different codebook sizes. Numbers in subscript show the standard deviation across 3 randomized trials over different seeds

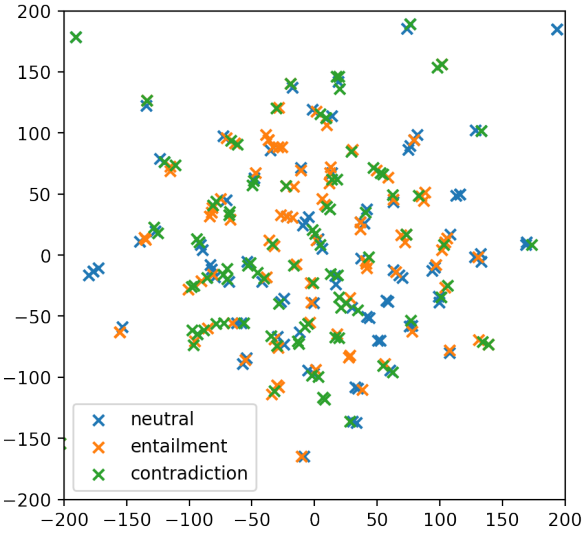

Figure 4: t-SNE map of the learnt codebook representation for NLI task in CB dataset

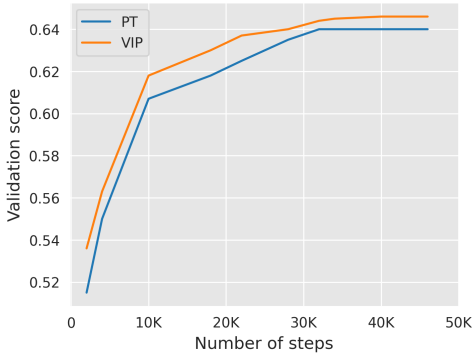

Figure 5: Plot of Validation performance on Multi-task dataset vs training iterations. Model selection in Section 4.3 and Table 6 is done based on this plot. VIP achieves more superior performance to baseline PT throughout all evaluation steps

model selection here. In Figure 5 we plot the average validation performance of the proposed VIP model and the baseline PT method, against the number of training steps, for the multi-task setting reported in Section 4.3. VIP model is observed to be consistently performing better on the validation dataset than the baseline PT method at all the evaluation steps.

A.4 Experimental Setup

Experimental settings. For a direct comparison with PT, we base our experiments on the LM-adapted version of T5-base encoder-decoder PLM[2]. We refer to the study of Lester et al. (2021) to find the best prompt length and fix the number of prompt tokens to $n = 100$ for PT, VIP and VIP-IDP. The sentence encoder settings are previously discussed in Section 3.1.1. For the classification tasks such as relation classification and NLI, we prepend the set of label tokens to $[P^q, \mathrm{X}]$ along with the task description (Equation (1)) as non-trainable hard prompt tokens. For VIP-IDP setting, we keep the weight as learnable parameters in the feedforward generator network. This is to enforce the model to rely its predictions on soft tokens which are generated from the input X[3]. FT feeds X to the PLM and fine-tunes the

[2] T5-lm-adapt

[3] experimentally, we found the systems ignore information from X and rely only on the learned bias terms

model parameters. Following the same setting as FT, Adapter learns adapter layer weights as proposed by (Houlsby et al., 2019) while keeping the PLM parameters frozen. For all the experiments, standard cross-entropy is used as prediction loss. For optimization, we use Adafactor (Shazeer and Stern, 2018) with constant learning rates (LR), selected via grid search over values $\{0.0001, 0.0005, 0.001\}$ for each of the models, CQ, VIP-IDP, and Adapter. The parameters of the soft prompt (P) embedding layer, however, need a significantly different LR, which we set to 0.3, following Lester et al. (2021). Batch size is set to 128 for the multi-task setup and to 32 for all remaining experiments following Lester et al. (2021). The quantizer comprises 1000 codebook vectors as parameters learned through EMA. We select commitment cost $\beta = 0.1$ through grid search over $\{0.01, 0.1, 1\}$. All our experiments are run for 30K steps, except multi-task setting which is run for 50K steps. We employ different evaluation step sizes for different datasets including epoch-wise evaluation, at 500 steps, and at 2K steps. We tune early stopping according to the evaluation step size and the number of train data samples.