Towards Personalized Intelligence at Scale

Anonymous ACL submission

Abstract

Personalized Intelligence (PI) is the problem of providing customized AI experiences tailored to each individual user. In many applications, PI is preferred or even required (Martinez et al., 2017; Rudovic et al., 2018). Existing personalization approaches involve fine-tuning pre-trained models to create new customized models. However, these approaches require a significant amount of computation to train, scaling with model size and the number of users, inhibiting PI to be realized widely. In this work, we introduce a novel model architecture and training/inference framework to enable *Personalized Intelligence* at scale. We achieve this by attaching a *Personalization Head (PH)* and freezing the base pre-trained LM. Since only the parameters in PH are updated during training, this results in a model much smaller than the conventional fine-tuned LM when scaled across users. We evaluate on academia and industry-focused datasets and show that this is much more scalable than traditional fine-tuning and outperforms zeroshot baseline in F1 score. We identify key factors required for effective PH design and training.

1 Introduction

As AI becomes ubiquitous in our lives, the experience remains largely homogeneous across users. Companies invest a tremendous amount of data and time into training one or more sets of models which are then served to all users (Wiggers, 2021). However, in many cases, these "one-size-fits-all" models provide suboptimal experience on an individual level because of the high degree of heterogeneity in the user population (Bellec, 2020; Measures, 2021). This calls for *Personalized Intelligence* enabled by AI models that can continuously learn and improve with user feedback and are tailored to each user.

The major obstacle in realizing *Personalized Intelligence* is the cost of training and storing individualized models in production (Sharir et al., 2020; Strubell et al., 2019). State-of-the-art deep learning models are larger than ever and require significant computation cycles to train. Therefore, the brute-force approach of fine-tuning pre-trained models per individual user is not feasible due to the high compute and storage complexity which scales with the number of users.

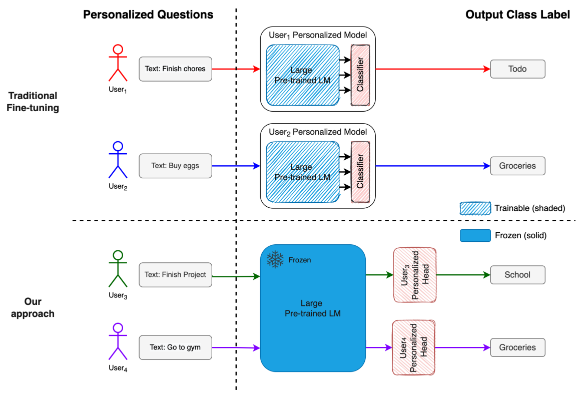

Figure 1: Example of individually personalized experience.

Zeroshot models have been studied recently as a method for applying pre-trained models to solve new problems domains without any additional training (Brown et al., 2020a; Raffel et al., 2020; Sanh et al., 2021). While promising, zeroshot models' performance is still ways off from the level of accuracy that is needed for production usage (Halder et al., 2020; Wenpeng Yin and Roth, 2019). Thus, the question remains as to how to enable *Personalized Intelligence* at a production scale for millions of users and beyond.

In this work, we propose a novel model training and inference framework for *Personalized Intelligence* at scale. In order to enable individually personalized models for a large user population, we investigate the approach of attaching to a pre-trained transformer-based encoder, a small module called *Personalization Head (PH)*. In our framework, we train only the *PHs* while keeping the base models frozen, as shown in Figure 1. Our insight here is that the language representation from the state-of-the-art encoders (e.g., BERT) captures high-level language features and can be "translated" to solve personalized problems. We address the following key questions in this paper:

1. Can the PHs effectively leverage output pre-trained LM despite not being jointly trained?
2. How should PHs be designed to scale to millions of users and beyond?
3. How should the PH design change based on its target personalization task?

Transformer architectures have been shown to be effective in a wide range of translation tasks (Vaswani et al., 2017; Liu et al., 2020; Gheini et al., 2021). Leveraging this insight, we design our *PH* to constitute of a single transformer encoder block with multi-head attention, followed by a linear classification layer. Because the large-scale transformer base is frozen during training, this base can be used to serve across users and only a small *PH* needs to be fine-tuned and stored individually. We explore the design space of *PH* by experimenting with a spectrum of designs with varying model sizes and training costs.

For evaluation, we explore the efficacy of our *PHs* on academia and industry-oriented datasets, comparing with the performance of zeroshot models and traditional fine-tuning approach. We show that the *PH*s can effectively leverage the output from the frozen encoder to achieve comparable state-of-the-art performance. Specifically, we show that current zeroshot models are not effective in this context and the *PHs* outperform the zeroshot baselines by up to XX. Furthermore, we show that our personalization framework requires orders of magnitude less training cycles and model storage cost while maintaining comparable performance when compared to fine-tuning the entire model, drastically improving the applicability of these LMs for production usage. We make the following specific contributions in this paper:

- We introduce a novel training and inference framework for *Personalized Intelligence* where only a small *Personalized Head* needs to be trained and stored for each user instead of the traditional approach of fine-tuning a language model.

- We design the *Personalization Heads (PH)*, a transformer-based "head" module that effectively adapts its frozen LM base outputs to specific personalized problems without having to fine-tune the base LM. The implementation of the proposed framework and PH architecture are open-sourced.[1]

- We evaluate the proposed PH on academia and industry-oriented datasets and show it outperforms current approaches in accuracy and scalability. We present a series of insights and design considerations for delivering personalized intelligence at scale.

2 Fine-tuning at Scale

Let M be a pre-trained language model (LM) and Θ_M be the set of parameters of M. When applying M to a downstream task T with labelled training data D_T, a linear output layer L with parameters Θ_L is attached to M. M and L are trained jointly:

$$\Theta\prime_M, \Theta\prime_L \leftarrow \arg\min_{\Theta_M,\Theta_L} L_T(D_T; \Theta_M, \Theta_L) \quad (1)$$

where L is the loss function and $\Theta\prime_M$ and $\Theta\prime_L$ are the fine-tuned parameters of the language model M and linear layer L, which are distinct from Θ_M. Let $\Omega(\Theta)$ be the computation complexity required to train a set of parameters Θ. We then define the training complexity of the above fine-tuning operation as:

$$\Omega(\Theta\prime_M) + \Omega(\Theta\prime_L) \quad (2)$$

Consider the scenario of fine-tuning for an individual user to create a personalized model. We define $U = \{(D_1, L_1), (D_2, L_2), ..., (D_N, L_N)\}$ as the collection of personalization tasks where $i \in [1, ..., N]$ for N users. D_i is the unique data for user i and L_i is the loss function. The problem of fine-tuning to personalize for user i becomes

$$\Theta\prime_{M,i}, \Theta\prime_{L,i} \leftarrow \arg\min_{\Theta_M,\Theta_L} L_i(D_i; \Theta_M, \Theta_L) \quad (3)$$

The aggregated training complexity is

$$\sum_1^N \Omega(\Theta\prime_{M,i}) + \sum_1^N \Omega(\Theta\prime_{L,i}) \quad (4)$$

The collection of model parameters to be stored is:

$$\sum_1^N |\Theta\prime_{M,i}| + \sum_1^N |\Theta\prime_{L,i}| \quad (5)$$

[1] https://datasets.code URL obfuscated for blind review.

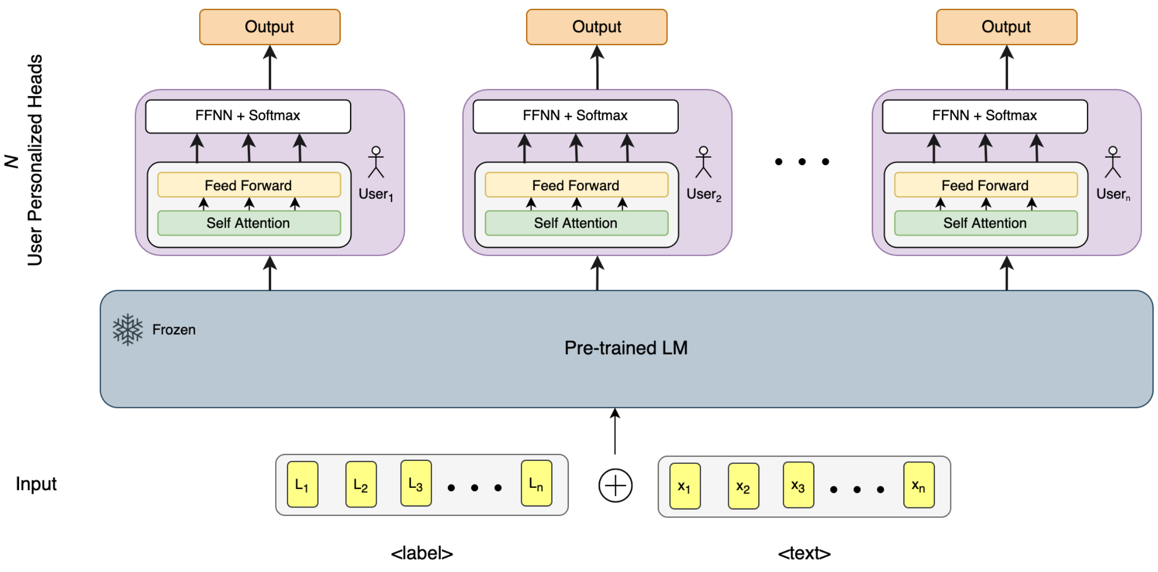

Figure 2: Overview of the proposed approach for personalization at scale with Personalization Head (PH).

3 Scalable Personalized Intelligence

Large-scale pre-trained language models can be adapted and personalized for each individual through the fine-tuning process defined above. The shortcoming is LMs (M) have millions to billions of parameters (Θ_M) (Brown et al., 2020b). The training complexity ($\Omega(\Theta_M)$) is high for a given task and it scales with the number of users, as defined in Equation 4.

Inspired by the effectiveness of adapting pre-trained LMs to new tasks, we hypothesis that by adding a lightweight transformer module between the LM and the linear output layer, we can eliminate the need for fine-tuning the LMs and keep it constant across all users. The intuition is that the LMs learn to capture generalized language features and a small transformer can leverage that for new tasks. To this end, we introduce *Personalization Head (PH)*. Figure 2 provides an overview of the proposed approach.

3.1 Fine-tuning with PH

We define PH_i as the Personalization Head for user i. When fine-tuning for a given user, we keep the LM parameters frozen and only train the PH and the linear output layer:

$$\Theta\prime_{PH}, \Theta\prime_L \leftarrow \underset{\Theta_{PH}, \Theta_L}{\arg\min}\, L_T(D_T; \Theta_M, \Theta_P H, \Theta_L) \tag{6}$$

Note that, compared to the traditional fine-tuning defined in Equation 1, no $\Theta\prime_M$ is generated. The aggregated training complexity for fine-tuning with PH is:

$$\sum_1^N \Omega(\Theta\prime_{PH,i}) + \sum_1^N \Omega(\Theta\prime_{L,i}) \tag{7}$$

and the total parameters is:

$$\Theta_M + \sum_1^N |\Theta\prime_{PH,i}| + \sum_1^N |\Theta\prime_{L,i}| \tag{8}$$

PH's model architecture is much smaller than a state-of-the-art language model. Therefore the model size and training cost is significantly lower:

$$|\Theta_{PH}| << |\Theta_M|,\ \Omega(\Theta\prime_{PH}) << \Omega(\Theta\prime_M) \tag{9}$$

As a result, the training complexity and model size are significantly reduced when fine-tuning with PH.

3.2 Universal Binary Classification

We aim to design a personalization framework that is generalizable to arbitrary classification tasks without requiring modification to the model architecture. To that end, we draw inspiration from (Halder et al., 2020) and formulate the multi-class classification problem as a series of binary classification tasks:

$$f(label(y_i), x) = P(True|y_i, x)\, \forall Y \tag{10}$$

We provide the model with both the class label $label(y_i)$ and the input text x and the output layer generates a binary $True/False$ prediction with a confidence score P. The class with the most

Dataset	Description	# Classes	# Train	# Test
SNIPS	Smart assistants questions	7	13,034	1,442
Clinc150	Production VA tasks	150	15,100	1,500

Table 1: Dataset Statistics

confident $True$ prediction is selected as the final prediction:

$$y = \underset{i \in \{1...M\}}{\arg\max}\ f(label(y_i), x) \tag{11}$$

where M is the number of classes.

3.3 PH Architecture

The functionality of the PH is converting the feature representation from pre-trained language models to new representations suited for a new problem unique to an individual. This is akin to a language translation problem, at which Transformer models have been proven to perform well (Raganato et al., 2018; Wang et al., 2019; Lu et al., 2021). We draw inspiration from these works and base our PH design on the Transformer architecture.

Figure 2 shows an overview of the PH, which consists of a single Transformer encoder layer. We follow the Transformer architecture defined in the original transformer paper (Vaswani et al., 2017). Each PH has a multi-head self-attention layer and two fully-connected layers, followed by layer normalization (Ba et al., 2016). Dropout (Srivastava et al., 2014) is applied to the output of the fully-connected layers.

It is important to understand how to configure the encoder block to design an effective PH. To help us explore the design space, we parameterize the following configuration: the size of the hidden dimension of the feedforward network in the encoder and the number of attention heads in the attention layer. We select these parameters to study because they have a significant effect on the size and capability of a Transformer encoder. We investigate the impact of these factors on the performance of PH in detail in Section 5.

4 Experiments

4.1 Datasets

We evaluate the proposed approach by applying PHs to adapt to a pre-trained LM to the SNIPS (Coucke et al., 2018) dataset and Clinc-150 (Larson et al., 2019) dataset. We select SNIPS because its classes feature intents that cover many popular user daily interaction topics and it is a representative dataset widely studied in the literature. We select Clinc-150 for its focus on production use cases. It has 150 intents and features intent and sentences inspired by real virtual assistants in production. An overview of the datasets is shown in Table 1. Due to the large number of testing examples in Clinc's original test set, we randomly sample 10 examples (out of 30) per class to construct a representative test set. This test set is used across all experiments, including baselines and PHs.

We generate $< class, text >$ pair as training and testing input and $True/False$ label as output for each training example in the dataset. We decide consciously to include only the True examples and leave out the False examples to optimize training cost. Training with True and False examples increases the size of training data by a factor equal to the number of classes, incurring significant additional training cost (Halder et al., 2020). This issue is further exacerbated in production use cases, where there can exist hundreds of classes, as in the Clinc-150 dataset.

4.2 Experimental Settings

We implement our model using the Flair NLP framework (Akbik et al., 2019) with an underlying pytorch runtime (Paszke et al., 2019). We use the uncased BERT encoder as the base LM (Devlin et al., 2018). We train for 50 epochs (unless noted otherwise) and report F1 scores on the test set. For hyper-parameters, we use a batch size of 16 and a learning rate of 0.02, following the standard in (Halder et al., 2020).

5 Results

We compare PHs to fine-tuning both LM and linear output layer, fine-tuning just the linear layer, and applying the model zeroshot.

5.1 Understanding zero-shot efficacy

We first investigate the efficacy of the zeroshot approach. We use the TARS classifier from (Halder et al., 2020) to predict the test examples without additional training and measure the F1 score, shown in the first row of table 2. The TARS classifier uses BERT as the underlying language model and is pre-trained on a suite of datasets including classification datasets such as AGNews and DBPedia (Zhang et al., 2015). It achieves an F1 score of 35.27 and 23.98 on SNIPS and Clinc, respectively, much

Model			# Params / User	Size / User	SNIPS F1 / Acc.	Clinc F1 / Acc.
Zero-shot	TARS (BERT)		-	-	35.27 / 26.70	23.98 / 23.67
Fine-tuning LM + Linear Layer	BERT		109M	417MB	98.61 / 98.61	93.79 / 95.70
	TARS		109M	417MB	98.13 / 98.06	91.18 / 94.27
Fine-tuning Linear Layer Only	BERT		1.5K	7KB	68.70 / 58.67	52.43 / 50.27
	TARS		1.5K	7KB	71.12 / 63.11	33.27 / 33.20
Personalization Head (PH) w/ frozen LM	**Hidden Dim**	**# Attn. Heads**				
	2048	8	5.52M	21MB	96.52 / 96.12	76.57 / 77.00
		4			97.18 / 96.74	76.46 / 76.47
		2			97.33 / 97.16	76.36 / 75.93
	1024	8	3.94M	15MB	97.09 / 97.09	76.37 / 74.40
		4			96.76 / 96.39	75.82 / 75.73
		2			96.77 / 96.67	76.61 / 76.60
	512	8	3.15M	12MB	97.05 / 97.02	75.95 / 76.33
		4			96.26 / 95.49	70.79 / 71.20
		2			96.94 / 96.74	75.29 / 75.27
	256	8	2.76M	11MB	95.90 / 95.70	66.99 / 67.53
		4			95.64 / 95.15	68.64 / 67.87
		2			97.06 / 96.32	67.68 / 67.13
	128	8	2.76M	11MB	95.32 / 95.28	63.43 / 64.53
		4			96.32 / 96.32	62.70 / 63.67
		2			96.36 / 96.36	63.82 / 64.60

Table 2: F1 score and accuracy of the proposed PHs and baselines of zeroshot, fine-tuning the LM + linear head and fine-tuning the linear head only, evaluated on SNIPS and Clinc-150 datasets. We also show the number of parameters that are required to be fine-tuned for each approach, as well as the size of the personalized model to be managed for each user as a representation of the scalability of each approach.

Hidden Dims	# Params	Size	epoch=50 # data/class=100	epoch=100 # data/class=100	epoch=50 # data/class=200	epoch=100 # data/class=200
2048	2.7M	10.5MB	65.92	70.43 (+4.50)	78.12 (+12.20)	92.99 (+27.70)
1024	3.2M	12.0MB	57.42	65.39 (+7.97)	80.17 (+22.75)	93.72 (+36.30)
512	3.9M	15.0MB	61.75	67.01 (+5.26)	83.72 (+21.97)	93.99 (+32.24)
256	5.5M	21.0MB	57.40	63.58 (+6.18)	79.85 (+22.45)	94.58 (+37.18)

Table 3: F1 score (and differential) with increasing training data and/or training for more epochs

lower than the reported state-of-the-art results. This shows that zeroshot approach requires significant improvement to reach the level of performance required for adapting to new tasks in production.

5.2 PH performance

Table 2 shows the performance of the Personalization Head(PH) when fully trained and evaluated on SNIPS and Clinc. We experiment with a wide range of PH configurations by varying the hidden dimension of the feedforward network and the number of attention heads. For brevity, we include results for configurations with hidden dimensions from 128 to 2048 and # attention heads of 2, 4, and 8. Results for additional configurations are included in the Appendix. We compare to the current approach of fine-tuning both LM and the linear output layer and fine-tuning only the linear output layer while keeping the base LM frozen. We also include in Table 2 the # training parameters and model size required per user for PHs and the fine-tuning baselines.

We observe that PHs, across all configurations, significantly outperform fine-tuning just the linear layer on both datasets. When comparing to the baseline of fine-tuning the entire model stack of the base language model and the linear output layer, PH achieves similar results for the SNIPS dataset, while requiring orders of magnitude less training parameters. Fine-tuning both LM and linear layer achieves the better F1-score on Clinc but it requires, for each user, the training of 109 million parameters which generates a 417MB model, while each PH only incurs for each user 2-5 million additional parameters to learn and 11 - 21MB model to store. The large computation cycles and storage capacity required for the fine-tuning approach renders it inapplicable for production applications that require scaling to millions of users and beyond. We analyze the scalability impact of PHs in more detail in Section 6.3

6 Analysis

We conduct experiments aimed at understanding the learning behavior of PHs and gain insights into

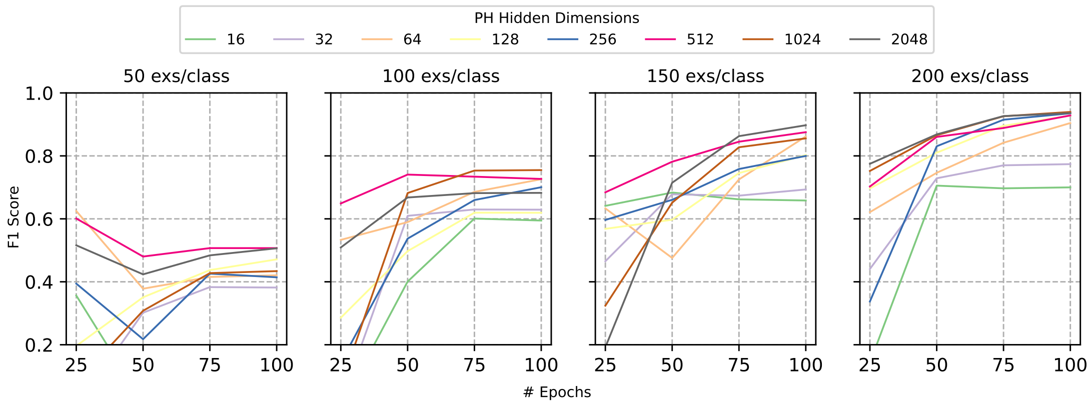

Figure 3: F1 Score w.r.t # training epochs, for fixed amounts of data.

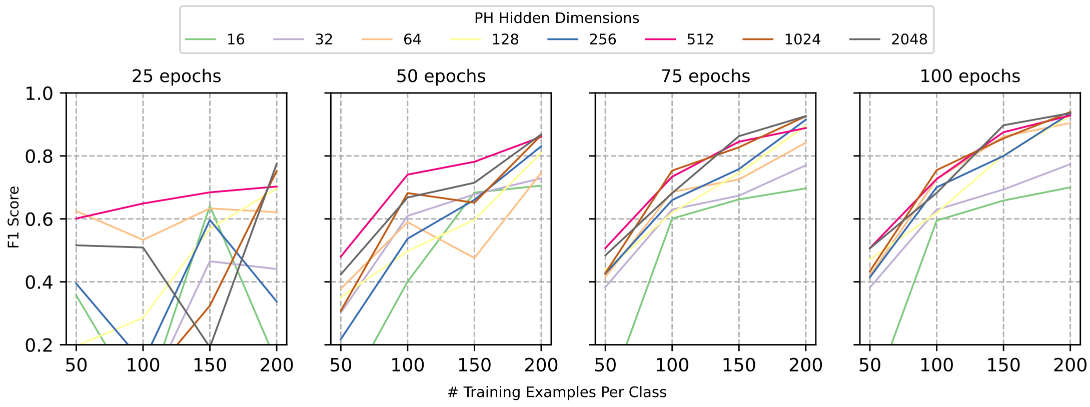

Figure 4: F1 Score w.r.t # training data, for fixed amounts of epochs.

how to design and deploy an effective PH for real-world use cases. We aim to answer the following questions: 1) how to effectively train PHs in production? 2) how does the PH configuration affect its learning behavior? 3) does larger PH achieve better performance? is there a sweet spot of PH design that is the most compute and data effective?

6.1 Impact of data vs epoch on training PHs

We study the impact of training data scale and training epochs on the PH performance. In real-time training in production, there is often limited training data and training cycles available. Therefore it is imperative to understand which training methods are effective. To this end, we construct a SNIPS sub-dataset by random sampling of 100 training samples per class (1400 samples in total). We keep the full SNIPS test set for F1-score measurement. We train the spectrum of PH designs for 50 epochs and then 50 more epochs (100 epochs in total) and record the F1 score at both points. We then select another 100 training samples per class to add to the training set and repeat the same experiment. Table 3 shows the average F1 scores, as well as improvement gained by increasing training data, training epochs, and both.

We observe that increasing the training data from 100 per class to 200 per class provides a significantly higher F1 score increase (+19.85 on average), compared to training for more epochs (+5.98 average). This behavior is consistent across the PH configurations. This is intuitive because 100 training samples/class represents only 5.3% of the full SNIPS training set and does not provide robust coverage of the problem space. Therefore, training data scale should be the priority over more training iterations in the early stages of applying a PH to a personalized problem.

6.2 PH Design Analysis

Two main design choices for PH are the hidden dimension size of the encoder block and the number of attention heads in the multi-attention layer. We study how these design choices impact the learning behavior of the PH. To that end, we conduct a set of experiments where we gradually increase the amount of training data or epochs and measure the F1 score at each stopping point. This is to simulate a training setup in production, where the model gradually gets exposed to more training data as the applications collect more data from the users.

Hidden Dimension Size Figure 3 shows the F1 score of PHs with different hidden dimensions as they are trained with more epochs on the same amount of training data. We experiment with 50, 100,150, and 200 training examples per class for 25, 50, 75, and 100 epochs. Conversely, Figure 4 shows the F1 score of the same suite of PHs as they are trained with more training examples for the same number of training epochs. We make several observations from the result.

First, we observe that when the model's exposure to data is limited in the early stages of training, PH training can exhibit unpredictable behavior. This is shown in the leftmost graphs of Figure 3 and 4, where the model performance is not improving with additional training data or more training epochs.

Furthermore, larger PHs perform better than smaller PHs but with diminishing returns at higher ends, indicating a sweet spot of PH design. We observe 512 to be the sweet spot of PH design for SNIPS as it performs better or similar to the other configuration across all experiments. This finding is corroborated with results on the Clinc-150 dataset. Figure 5 shows the F1 score of PHs with varying hidden dimensions on both SNIPS and Clinc-150. We observe similar trends for diminishing return in performance for Clinc as the PH design gets larger. Similarly, 512 is the inflection point of F1 score improvement for Clinc, making it the sweet spot PH design for Clinc. Furthermore, when comparing Clinc against SNIPS, we observe a slower rate of F1 score growth with respect to hidden dimension sizes in Clinc than SNIPS. This makes sense because, as described in Section 4.1, Clinc is a more diverse and challenging task with significantly more classes than SNIPS.

# Attention Heads We study the impact of attention heads on PHs performance. Figure 6 shows F1 score w.r.t hidden dimensions per attention head. We follow the description in (Vaswani et al., 2017) that when changing the number of attention heads, the hidden dimensions of the feedforward layer are effectively distributed evenly among the available attention heads. We observe that PHs achieve better performance with higher hidden dimensions per head but eventually see diminishing returns at higher dimensionality. This is aligned with our findings on the impact of total hidden dimensions on PH performance.

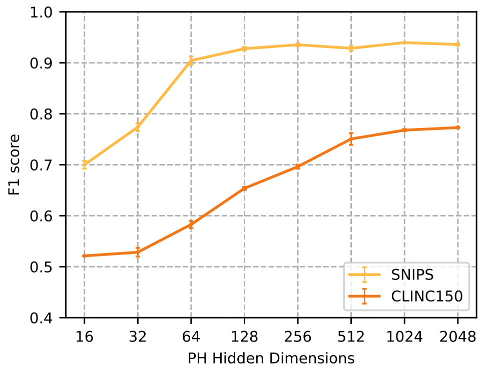

Figure 5: F1 score of PH w.r.t. hidden dimension size

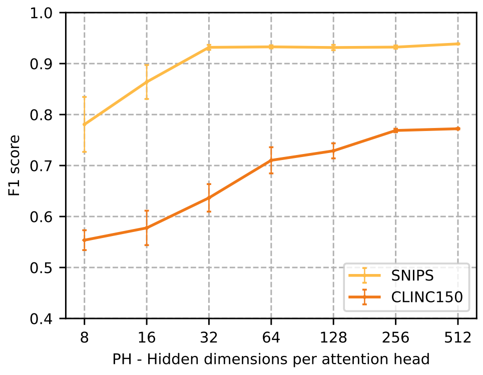

Figure 6: F1 score of PH w.r.t. hidden dims per head

6.3 Scalability

We study the scalability of the PH approach and its impact on production deployment. Because of the complexity of personalization in production, we first introduce a new metric, Personalization Efficiency:

$$PE = \frac{F - score^2}{Training\ Cost \times Model\ Size} \qquad (12)$$

to holistically evaluate a personalization approach. Figure 7 shows the efficiency of 4 PH configurations normalized to the fine-tuning BERT baseline. We show that PHs achieve efficiency up to 155X compared to the fine-tuning baseline. Furthermore, for SNIPS we observe smaller heads generally measure higher in efficiency than larger heads but see a diminishing return. For Clinc, 512 achieves the highest efficiency of the PHs tested. This corroborates our recommendation earlier that 512 is the PH design sweet spot.

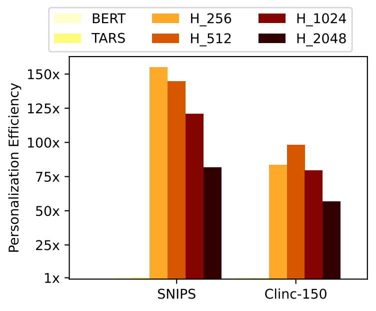

Figure 7: Personalization Efficiency

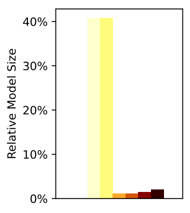

Figure 8: Data Scalability

We also quantify the potential storage overhead required for personalized models. Figure 8 shows the additional storage overhead required by the personalized models per individual relative to the existing user data in production. We use Gmail as an example application. We calculate approximately the current per-user data usage based on a report that Gmail user creates up to 1.4MB of data per day and 3 years is the average account lifetime (ZDNet, 2012). Figure 8 shows that proposed PHs constitute 1% - 1.5% of additional storage overhead across all 4 sizes, while the fine-tuning baselines would incur around 40% additional storage overhead per user.

7 Related Work

The most common approach to NLU problems today is to leverage large pre-trained transformer-based language models (Devlin et al., 2018; Liu et al., 2019) typically trained on language understanding objectives such as Masked Language Modeling and Next Sentence Prediction. These language models is then fine-tuned on a specific target task and have been shown to produce state-of-the-art results. This transfer of learning to the task of interest is achieved by tuning all of the model weights on that singular task. The performance of these LMs have shown to scale with model size (Kaplan et al., 2020) resulting in massive models consisting of billions of parameters (Brown et al., 2020a; Raffel et al., 2020; Sanh et al., 2021). While undeniably packed with knowledge, when applied to the online setting of personalization where data arrives in a stream the applicability of these language models is severely constrained as it results in a dedicated model of each user. This section explores existing works improving the applicability of transformer models at scale.

7.1 Zero-shot Learning

Zero-shot learning approaches aim to provide out-of-the-box generalizable performance on a range of language-based tasks without needing additional training steps as required by traditional transfer learning approaches. Recent approaches to this problem frame this as a text-to-text generation task (Brown et al., 2020a; Raffel et al., 2020; Sanh et al., 2021) giving rise heavy focus on prompt design (Perez et al., 2021; Khashabi et al., 2020). While shown to be fairly proficient in tasks such as QA and Summarization, when applied text classification tasks out-of-the-box zero-shot perform subpar (Halder et al., 2020; Wenpeng Yin and Roth, 2019). This is further shown in our zero-shot experimental results. Halder et al. (2020) in their work explores the shortcomings of the existing transfer learning mechanisms for text classification, proposing the formalization of text classification as a general binary classification problem. We utilize this formalization as a basis of our text classification experimental architecture.

7.2 Adapter Networks

Another method used to finetune transformers is adapter networks (Houlsby et al., 2019; Pfeiffer et al., 2021). Adapters are new modules that add a fully connected residual block for each unique downstream task and finetune the layer norm parameters. Our approach is similar in nature but instead applies a small trainable module allowing us to keep the base language model completely frozen during training.

8 Conclusion

Today's AI experience remains largely homogeneous across all users. This is because they are often served with the same pre-trained models. Many applications prefer or even require AI capability personalized at the individual level. In this work, we investigate achieving personalized intelligence at scale. We introduce a novel model training and inference framework, where a small personalization head is added to adapt large-scale pre-trained LMs. We only train the small PH and keep the base LM frozen, thus significantly reducing the computation and storage cost compared to the current fine-tuning approach. We show that the PHs outperform zeroshot in accuracy and scales far better than traditional fine-tuning approach. We then perform analysis on various PH design factors.

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. Flair: An easy-to-use framework for state-of-the-art nlp. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Anne-Claire Bellec. 2020. Why ai makes the human touch even more important in personalization.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020a. Language models are few-shot learners.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020b. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Mozhdeh Gheini, Xiang Ren, and Jonathan May. 2021. Cross-attention is all you need: Adapting pretrained Transformers for machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1754–1765, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Kishaloy Halder, Alan Akbik, Josip Krapac, and Roland Vollgraf. 2020. Task-aware representation of sentences for generic text classification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3202–3213.

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.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *CoRR*, abs/2001.08361.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. Unifiedqa: Crossing format boundaries with a single qa system.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1311–1316, Hong Kong, China. Association for Computational Linguistics.

Xiaodong Liu, Kevin Duh, Liyuan Liu, and Jianfeng Gao. 2020. Very deep transformers for neural machine translation.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Yu Lu, Jiali Zeng, Jiajun Zhang, Shuangzhi Wu, and Mu Li. 2021. Attention calibration for transformer in neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1288–1298, Online. Association for Computational Linguistics.

Daniel Lopez Martinez, Ognjen Rudovic, and Rosalind Picard. 2017. Personalized automatic estimation of self-reported pain intensity from facial expressions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2318–2327. IEEE.

Chris Measures. 2021. Overcoming the pitfalls to smart and successful ai personalization.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. Adapterfusion: Non-destructive task composition for transfer learning.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer.

Alessandro Raganato, Jörg Tiedemann, et al. 2018. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. The Association for Computational Linguistics.

Ognjen Rudovic, Jaeryoung Lee, Miles Dai, Björn Schuller, and Rosalind W Picard. 2018. Personalized machine learning for robot perception of affect and engagement in autism therapy. *Science Robotics*, 3(19).

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Stella Biderman, Leo Gao, Tali Bers, Thomas Wolf, and Alexander M. Rush. 2021. Multitask prompted training enables zero-shot task generalization.

Or Sharir, Barak Peleg, and Yoav Shoham. 2020. The cost of training nlp models: A concise overview.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. 2019. Learning deep transformer models for machine translation. *arXiv preprint arXiv:1906.01787*.

Jamaal Hay Wenpeng Yin and Dan Roth. 2019. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. In *EMNLP*.

Kyle Wiggers. 2021. Ai model training costs on the rise, highlighting need for new solutions.

Jack Schofield ZDNet. 2012. Is your Gmail really worth $3,600? Backup now! `https://www.zdnet.com/article/is-your-gmail-really-worth-3600-backup-now/`. [Online; accessed 16-Jan-2022].

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*.

A Appendix

A.1 More PH configurations

hidden dim	# attn. heads	SNIPS	Clinc
64	8	95.89	57.85
	4	96.32	54.41
	2	96.03	56.60
32	8	96.32	51.37
	4	96.24	50.83
	2	95.15	46.92
16	8	94.87	49.92
	4	94.43	49.99
	2	94.94	50.28
8	8	83.09	52.58
	4	80.86	52.66
	2	86.34	53.29

Table 4: F1 score of PHs with smaller sizes.

Table 4 shows the F1 score of PHs with 64, 32, 16, and 8 hidden dimensions, on SNIPS and Clinc datasets. This is an extension to the result shown in Table 2. We observe similar trends carry over to this set of even smaller PHs. This shows that even a tiny PH can adapt LM well to the SNIPS task. On the other hand, the smaller PHs are not as effective for the more challenging Clinc datasets.

A.2 Training for more epochs on Clinc

Table 5 shows the F1 scores of PHs of 4 different sizes on Clinc when training for an additional 50 epochs (100 epochs in total). This shows that PHs performance continues to improve with more training iterations, indicating that continuing training for more iterations could still be beneficial in improving PH performance after training on a decent amount of training data and epochs.

A.3 Analyzing # of attention heads

Figure 9 and 10 analyzes the impact of # attention heads on the performance of PHs. We conduct experiments similar to that in Section 6.1. We gradually increase the amount of training data while

hidden dim	# attn. heads	Clinc
2048	2	78.29
	4	77.25
	8	77.78
512	2	75.82
	4	75.05
	8	74.98
128	2	65.29
	4	65.72
	8	63.63
32	2	51.35
	4	53.24
	8	52.84

Table 5: F1 score of PHs trained on clinc150 dataset for an additional 50 epochs (100 epochs in total)

holding the training epochs fixed and measure the F1 score at each stopping point, and vice versa. We observe that, compared to hidden dimension sizes, # of attention heads has less effect on the learning behavior and capacity of the PHs.

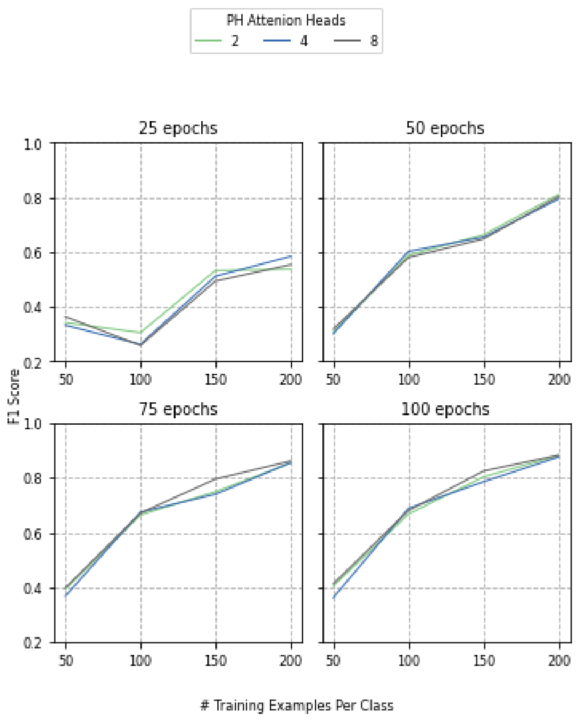

Figure 9: With the same number of training epochs, the impact of training data on performance.

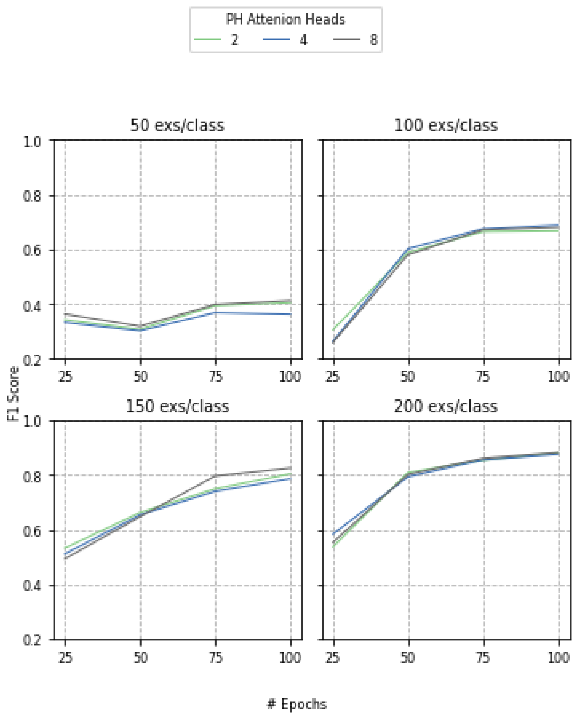

Figure 10: With the same amount of training examples per class, the impact of epochs on performance.