Prompt Learning with Extended Kalman Filter for Pre-trained Language Models

Quan Li[1,4] , **Xike Xie**[2,4] *, **Chao Wang**[1,3] * and **S. Kevin Zhou**[2,4,5]
[1]School of Computer Science and Technology, University of Science and Technology of China, China
[2]School of Biomedical Engineering, University of Science and Technology of China (USTC), China
[3]School of Software Engineering, USTC, China
[4]Data Darkness Lab, MIRACLE Center, Suzhou Institute for Advanced Research, USTC, China
[5]Key Laboratory of Precision and Intelligent Chemistry, USTC, Hefei Anhui, 230026, China
SA21011062@mail.ustc.edu.cn, {xkxie, cswang, skevinzhou}@ustc.edu.cn

Abstract

Prompt learning has gained popularity as a means to leverage the knowledge embedded in pre-trained language models (PLMs) for NLP tasks while using a limited number of trainable parameters. While it has shown promise in tasks like sentiment classification and natural language inference, generating suitable prompts for PLMs, as opposed to human prompts, remains a challenge. In this paper, we introduce an abstraction of the prompt learning process using an Extended Kalman Filter. Our approach, called Conditional Extended Kalman Filter based on Neural Networks (CEKFNN), effectively infers more appropriate prompt tokens by enhancing the classic Extended Kalman Filter with PLM's contextual representation power. Specifically, CEKFNN learns transition and emission functions from PLM embeddings of input sentences to infer latent prompt tokens. We refine CEKFNN using an alternate-training approach, retraining a PLM's emission function with prompt tokens inferred by prompt models (PMs), as well as the initial and transition functions. PLM's output labels assist in PMs' training. When updating the pre-trained language model (PLM), we use an adapter approach with few trainable parameters, leaving PLM parameters frozen. We evaluate CEKFNN across open-source PLMs, demonstrating performance improvements over state-of-the-art methods while using a limited number of trainable parameters. It shows that CEKFNN performs on-par or better than fine-tuning, which requires updating all parameters in the PLM.

1 Introduction

PLMs (Pre-trained Language Models) have proven successful in numerous natural language processing tasks [Brown *et al.*, 2020]. Evidence indicates that during pre-training, these models not only learn contextualized representations but also acquire knowledge of grammar [Clark *et al.*, 2019b], syntax [Hewitt and Manning, 2019], and commonsense [Davison *et al.*, 2019], representing some of the more abstract and intangible aspects of human cognition. Stemmed from the extensive pre-training on diverse datasets, PLMs have become foundational in advancing the field of natural language understanding and generation, driving innovation in areas ranging from automated text summarization to sophisticated conversational agents. From the perspective of adapting PLMs to specific tasks, fine-tuning and prompt learning stand out as two principal approaches [Liu *et al.*, 2023a].

Fine-tuning is a process where a PLM is further trained on a specific task-oriented dataset, leading to more specialized and accurate performance in that domain. The primary advantage of fine-tuning is its ability to tailor the model precisely to the task, resulting in high accuracy and efficiency. However, fine-tuning is resource-intensive, requiring significant computational power and specific training data. Moreover, the fine-tuned model retains the original model's parameter count, which, although a minor issue for smaller models like BERT [Devlin *et al.*, 2019] or GPT-2 [Radford *et al.*,], poses a substantial challenge for larger models such as T5 [Raffel *et al.*, 2020], GPT-3 [Brown *et al.*, 2020], and other PLMs with billions of parameters [Hu *et al.*, 2022].

Prompt learning, in contrast, relies on creating and employing specific prompts to direct a pre-trained model to produce the desired output without further training. This method is agile and resource-efficient, as it leverages the existing capabilities of the model. However, it introduces technical challenges in crafting effective prompts and can sometimes be less precise than fine-tuning.

For example, recent research has shown that PLMs' performance in prompt-based tasks can vary greatly, with minor changes in prompts causing large fluctuations in performance ([Arora *et al.*, 2023]; [Liu *et al.*, 2023b]). Such variability or inconsistency is influenced by factors such as the type of PLM used [Ouyang *et al.*, 2022] and its size [Lampinen *et al.*, 2022]. Efforts to boost the dependability of prompt learning include crafting ideal prompts or combining several of them. Methods such as proposed in [Shin *et al.*, 2020] focus on automating the search for effective discrete prompts, while [Liu *et al.*, 2023b] suggests the use of trainable continuous tokens alongside these discrete prompts to improve the performance. [Arora *et al.*, 2023] investigates the effectiveness of aggregat-

*Corresponding authors

ing predictions from different prompts. However, a common limitation in existing methods is the repeated use of the same prompt tokens across various sentences, without considering the specific information of the sentence involved or the contextual relevance.

This paper introduces a cutting-edge method to improve the accuracy of prompt learning while maintaining its agility. Building on the idea that a sentence's information can be treated as a latent variable shaped by its semantic content and the preceding sentence's latent variable ([Li *et al.*, 2021] and [Revach *et al.*, 2022]), we present a novel Conditional Extended Kalman Filter based Neural Network (CEKFNN). This technique refines the training and inference processes through PLM embeddings, enhancing initial, transition, and emission function predictions. CEKFNN not only broadens the context understanding of Extended Kalman Filter beyond traditional Markov model limitations, but also adapts prompt tokens dynamically, depending on whether input sentences are related or independent. In scenarios where input sentences are interconnected, the prompt tokens for a sentence are derived from the preceding sentence's tokens. Conversely, in cases without such a connection, the prompt tokens are determined independently using the sentence's embeddings and initial prompt tokens.

Further, we integrate CEKFNN with a supervised mode using an alternating training method. This method involves retraining the PLM with prompt tokens generated by prompt models (PMs). By leveraging the pre-trained knowledge contained in PLM, this process aims to retrain the PLM to obtain true labels. The retrained PLM serves as an emission function, whose outputs are combined with true labels for the next round of PMs training. The alternate-training method trains PMs (initial function of hidden state and transition function) and PLM (emission function) alternately until the result is optimized. When updating the PLM, we use an adapter method with few trainable parameters to simulate the PLM update. Figure 1 shows the architecture of CEKFNN framework. An Initial model generates initial prompt tokens. Given the initial prompt tokens, we can get different prompt tokens for different input sentences by using transition model and embeddings of current input sentence. When training the whole model, we use the alternative learning strategy to update the trainable parameters of PMs and PLM. The contributions of this paper include:

- We conceptualize the process of prompt learning within the CEKFNN framework, where prompt tokens are viewed as time-variant latent variables and task labels are considered observed variables influenced by these latent variables. CEKFNN is designed with minimal trainable parameters, effectively adapting to downstream tasks through initial, transition, and emission functions specifically for prompt tokens.
- We introduce an alternating training methodology that sequentially trains PMs for the initial and transition functions, and the PLM for the emission function. Each iteration uses the outputs of one to inform the training of the other in successive iterations.
- Our extensive evaluation covers a range of open-source PLM families, including BERT, GPT-2, and T5, and various model sizes, ranging from 110M to 3B parameters. It shows enhanced performance compared to existing methods, with the added advantage of requiring a small set of trainable parameters. Our method achieves results comparable or superior to those of comprehensive fine-tuning, which necessitates updating all parameters in the PLM. We have made the code for all tasks involving CEKFNN available in PyTorch at https://anonymous.4open.science/r/GAP-888F.

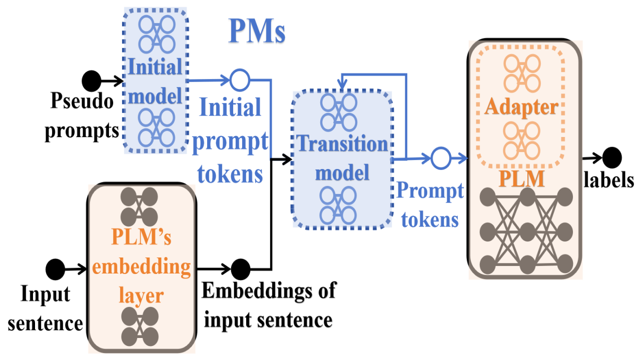

Figure 1: An architecture of CEKFNN framework. The models in the blue region represent PMs, while the model in the orange region represents PLM. Within all models, parameters within hollow circles are trainable, while those within solid circles remain fixed. Regarding the architecture's variables, solid circles are observed elements; hollow circles are hidden elements.

2 Related Works

Pre-trained Language Models. The recent breakthrough in self-supervised pre-trained language models [Dong *et al.*, 2019] has significantly advanced the development of natural language processing. Decoder-based language models such as GPT [Radford *et al.*,] leverage the Transformer architecture to pre-train on large-scale web texts. Meanwhile, encoder-based models like BERT [Devlin *et al.*, 2019] introduce masked language modeling and establish the pre-training/fine-tuning paradigm. Subsequently, various language models have emerged, including encoder-decoder-based models such as T5 [Raffel *et al.*, 2020], which aim to unify language understanding and generation. Training larger language models generally leads to improved performance and remains an active area of research [Hu *et al.*, 2022].

Prompt Learning and Fine-Tuning. While a Large Language Model can adapt its behavior with just a few additional training examples, the result depends heavily on the input prompt [Brown *et al.*, 2020]. This necessitates an empirical art of composing and formatting the prompt to maximize a model's performance on a specific task, a practice known as prompt Learning [Liu *et al.*, 2023a]. Fine-tuning involves retraining a PLM on general domains for specific tasks [Devlin *et al.*, 2019]. Variants of fine-tuning include learning only a subset of the parameters [Devlin *et al.*, 2019]. However, practitioners often choose to retrain all parameters to maximize downstream performance.

Nevertheless, performing fine-tuning in the usual way for a PLM such as T5 [Raffel *et al.*, 2020] presents challenges. This is due to the large checkpoint it produces and the high hardware barrier to entry, as it incurs the same memory overhead as the pre-training process.

Parameter-Efficient Adaptation. Some methods propose inserting adapter layers between existing layers of a neural network ([Zhang *et al.*, 2023]; [Hu *et al.*, 2022]). These methods aim to reduce memory requirements by utilizing a limited number of trainable parameters, while keeping the full model's parameters fixed. Given the minimal memory overhead of adapters, recent approaches [Dettmers *et al.*, 2023] suggest using additional adapters to enhance performance without substantially increasing the total memory usage.

Neuralizing Hidden Markov Models and the Extended Kalman Filter. The key distinction between the Extended Kalman Filter (EKF) and Hidden Markov Models (HMMs) lies in their modeling of latent variables. The former accommodates continuous variables, while the latter limits its latent variables to discrete ones. Some studies aim to neuralize HMM and EKF to relax the Markov assumption while preserving their generative properties [Kim *et al.*, 2018]. For instance, [Li *et al.*, 2021] explore learning discrete structures for text or label generation using neuralized HMMs. Meanwhile, [Revach *et al.*, 2022] focus on learning continuous variables for safety-critical applications such as self-driving vehicles and signal processing using neuralized EKFs. We hypothesize that prompt learning can also be viewed as Bayesian inference with some continuous trainable prompt tokens, leading to our proposed CEKFNN approach.

3 Method

In this section, we present the framework of CEKFNN. CEKFNN formulates the prompt learning as Bayesian inference process, involving gradient updates that include trainable prompt tokens and adapter parameters. This training method mirrors that used in generative adversarial networks, applied across all trainable parameters, leading to the designation of this model as CEKFNN.

3.1 Preliminaries

We analyze models pre-trained on masked language modeling (MLM) objectives. $\mathcal{X}$ denotes a finite vocabulary of input tokens.

Pre-training and Downstream Task. Consider the MLM, denoted as $G(x) = \langle G_1(x), G_2(x), ...\rangle$, which is responsible for predicting some vectors associated with input sentence x. Additionally, let $\langle G(x^1), G(x^2), ...\rangle$ represent the MLM that predicts some vectors for each time step in the input sequence x^i. Each component G_i calculates the embedding of the i-th token, denoted as X_i, conditioned on all other tokens. The main task involves labeled examples in the form of pairs $\langle x, H^*(x)\rangle \in \mathcal{X} \times \mathcal{Y}$. Here,. $H^* : \mathcal{X} \rightarrow \mathcal{Y}$ serves as the source of ground-truth labels for downstream tasks, and $\mathcal{Y}$ represents a discrete set of classification labels.

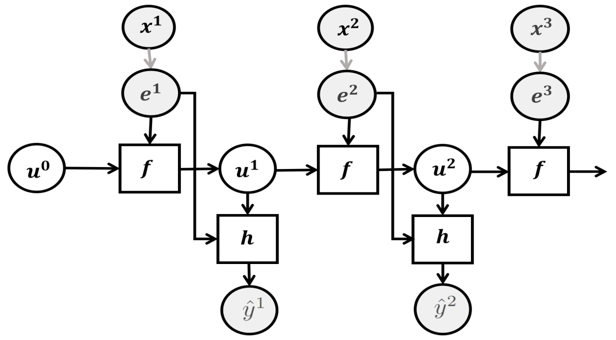

Figure 2: An illustration of CEKFNN's architecture. Shaded circles are observed elements; white circles are hidden elements; rectangles are functions. The arrows between x^t and e^t denote the context representation ability of PLM.

Prompt Learning. Prompt learning requires viewing the pre-trained model G as a function of the token embedding. We denote this model as $\overline{G}$. If we represent the token embeddings of input x as $e(x) = \langle e(x_1), ..., e(x_m)\rangle$, then $\overline{G}(e(x)) = G(x)$. m is the length of sentence x.

In prompt learning, a trainable prompt u is concatenated with the token embeddings to produce the model's output, which can be expressed as $\overline{G}((u, e(x)))$. It's important to note that "u" is considered an unobserved variable since we don't have direct knowledge of the information incorporated by the model from the input. In some research, "u" is referred to as an unobserved concept [Xie *et al.*, 2022].

Other Notations. Let $\triangle^d$ denote the space of d-dimensional probability vectors. $Pr(V = v) \in [0, 1]$ denotes the probability that V assumes the specific value v. For a sequence $v = \langle v^1, \ldots, v^t\rangle$, we use the notation $v^{\langle i:j\rangle}$ for $i \le j$ to denote $\langle v^i, \ldots, v^j\rangle$, and v^{-i} to denote $\langle v^{1:i-1}, v^{i+1:t}\rangle$.

3.2 CEKFNN Prompt Learning

In this section, we formulate prompt learning for linking pre-training and downstream tasks using the Conditional Extended Kalman Filter. Consider a sequence of input sentences containing T sentences, denoted as $X^{\langle 1:T\rangle}$. A downstream task can be formulated as a sequence labeling task that assigns a label to each sentence. Suppose we have some prompt tokens consisting of T parts, labeled as $U^{\langle 1:T\rangle}$, where $U^{\langle 0:T\rangle} = u^{\langle 0:T\rangle}, u^t = \langle u^t_1, \ldots, u^t_k\rangle$, and $u^t_i \in \mathbb{R}^d$. Here, k is the length of the sentence u^t and d is the dimension of the continuous vector u^t_i. For the input sentences $X^{\langle 1:T\rangle}$, we use $Y^{\langle 1:T\rangle}$ to represent the true labels and $E^{\langle 1:T\rangle}$ to represent the sentences' embedding. Each $e^t_i|_{i\in\{1,2,\ldots,m\}}$ has the same dimension as $u^t_j|_{j\in\{1,2,\ldots,k\}}$.

Defining a relation between pre-training and downstream tasks with prompt learning is the primary challenge. We propose to establish this connection via prompt tokens, which can be considered as latent variable assumptions about the input. In supervised downstream tasks with prompt learning, the goal is to find the appropriate underlying sequence of prompt tokens $\hat{U}^{\langle 1:T\rangle}$ given $E^{\langle 1:T\rangle}, Y^{\langle 1:T\rangle}$. We use $U = \langle U^0, U^1, \ldots, U^T\rangle \in \mathcal{U}^*$ to denote the sequence of hidden

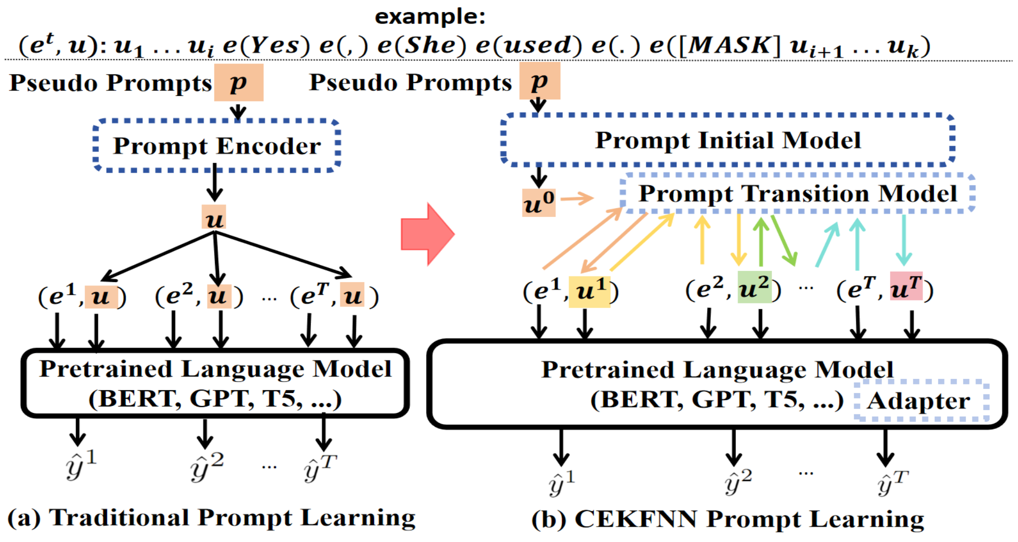
example:
(e^t, u): $u_1 \dots u_i\ e(Yes)\ e(,)\ e(She)\ e(used)\ e(.)\ e([MASK]\ u_{i+1} \dots u_k)$
Pseudo Prompts p
Prompt Encoder
u
(e^1, u) (e^2, u) ... (e^T, u)
Pretrained Language Model (BERT, GPT, T5, ...)
$\hat{y}^1$ $\hat{y}^2$ ... $\hat{y}^T$
(a) Traditional Prompt Learning
Pseudo Prompts p
Prompt Initial Model
u^0
Prompt Transition Model
(e^1, u^1) (e^2, u^2) ... (e^T, u^T)
Pretrained Language Model (BERT, GPT, T5, ...)
Adapter
$\hat{y}^1$ $\hat{y}^2$ ... $\hat{y}^T$
(b) CEKFNN Prompt Learning

Figure 3: The example of prompts for "Yes, she used.[MASK] ". Given the context, the colorful zone refer to the prompt tokens. These prompt tokens can be fixed prompt tokens. (e.g., "The sentence ... is grammatically ... ", "The sentence's knowledge ... is ... "), unified trainable tokens (e,g., "u_1 ... u_k ... "), or time-related trainable tokens (e.g., "u_1^t ... u_k^t ... ", "u_1^{t+1} ... u_k^{t+1} ... ") and their positions are flexible and adjustable. The models within the dashed box in figure are trainable. In (a), the prompt encoder only generates unified prompt tokens across different e^t; on the contrary, in (b) the prompt initial model and prompt transition model can generate time-related trainable prompt tokens by optimizing trainable model in a differentiable way.

states. For all timesteps $i \geq 1$, the transition probabilities are time-invariant, i.e., $P(U^i|U^{i-1}, E^i)$. For each timestep $i \geq 1$, U^i is emitted following some time-invariant probability: $P(Y^i|U^i, E^i)$.

Figure 2 shows the schematic architecture of CEKFNN. Here, $u^t|_{t\in\{0,1,\cdots,T\}}$ represents the continuous hidden states of CEKFNN, which correspond to the underlying appropriate prompt tokens. The transition function $f(\cdot)$ is implemented using a single layer of a transformer encoder, where

$$u^t = f(u^{t-1}, e^t) \tag{1}$$

describes the transition from prompt tokens u^{t-1} to u^t at time step t. The emission function $h(\cdot)$ is implemented using an adapter-based PLM. A emission function is utilized to obtain $\hat{y}^t$, which is formalized as:

$$\hat{y}^t = h(u^t, e^t) \tag{2}$$

To further elucidate the implementation and details of CEKFNN, we illustrate with a specific example. Given a PLM denoted as G, a sequence of input tokens $x^{\langle 1:T\rangle} = \langle x^1, x^2, \dots, x^T\rangle$ is mapped to input embeddings $\langle e^1, e^2, \dots, e^T\rangle$ through a PLM's embedding layer. Here, $x^t = \langle x_1^t, x_2^t, \dots, x_m^t\rangle$ and $e^t = \langle e_1^t, e_2^t, \dots, e_m^t\rangle$, where m represents the length of a sentence x^t. The purpose of a prompt is to organize the context x^t and itself into a template T^t. For instance, in a task involving sentence correctness prediction, a template might be expressed as:

$$\langle u_1^t, ..., u_i^t, e(x^t), u_{i+1}^t, ..., u_k^t\rangle \tag{3}$$

where $e(x^t)$ is equivalent to e^t. Figure 3 complements the architecture with the integration of PLMs and some examples, providing further details about CEKFNN.

3.3 Alternate Training Procedure

The alternate training method involves training two components of the CEKFNN, based on a deep neural network: a set of PMs and a PLM, sequentially using the output of each other. The PMs generate all prompt tokens $U^t|_{t\in\{0,1,\dots,T\}}$, while the PLM refines them leveraging its language modeling capabilities acquired during pre-training. This training process is divided into two distinct phases.

- In **phase I**, PMs receive $E^{\langle 1:T\rangle} = e^{\langle 1:T\rangle}$ and output $U^{\langle 0:T\rangle} = u^{\langle 0:T\rangle}$, which are then utilized to retrain the PLM.
- In **phase II**, PMs and PLM engage in mutual enhancement through several loops. Each loop begins with training the PMs with $e^{\langle 1:T\rangle}$, $Y^{\langle 1:T\rangle} = y^{\langle 1:T\rangle}$, and $\hat{Y}^{\langle 1:T\rangle} = \hat{y}^{\langle 1:T\rangle}$ provided by the fixed PLM. Then, the predictions prompt tokens $u^{\langle 0:T\rangle}$ are used to retrain the PLM with keeping a small set of trainable parameters.

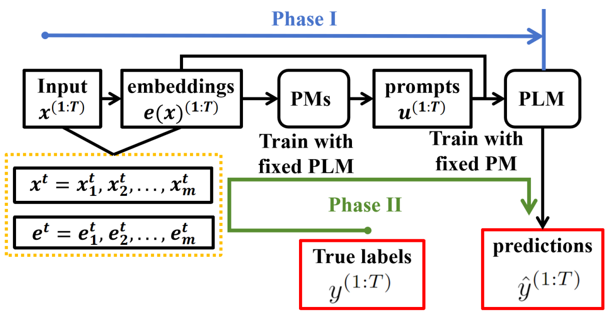
Phase I
Input $x^{(1:T)}$
embeddings $e(x)^{(1:T)}$
PMs
prompts $u^{(1:T)}$
PLM
Train with fixed PLM
Train with fixed PM
$x^t = x_1^t, x_2^t, \dots, x_m^t$
$e^t = e_1^t, e_2^t, \dots, e_m^t$
Phase II
True labels $y^{(1:T)}$
predictions $\hat{y}^{(1:T)}$

Figure 4: The illustration of the alternate training method.

Model & Method	# Trainable Parameters	GLUE									
		WNLI	RTE	CoLA	MRPC	STS-B	SST-2	QNLI	QQP	MNLI-m/-mm	AX-m/-mm
BERT$_{base}$ (FT)	110M	56.5	**65.0**	59.6	**85.1**	**81.9**	**92.4**	90.7	**89.8**	82.7/83.5	73.3/**74.5**
BERT$_{base}$ (P-Tuning)	11.8M	61.9	61.0	47.5	61.1	73.8	90.8	80.5	77.2	65.2/67.9	46.8/51.6
BERT$_{base}$ (LoRA)	0.6M	62.3	62.2	57.5	81.0	78.51	91.9	90.5	88.0	82.5/83.2	73.3/73.8
BERT$_{base}$ (AdaLoRA)	0.4M	66.1	61.5	50.5	72.0	77.5	91.1	89.7	86.7	80.5/81.4	69.7/70.9
BERT$_{base}$ (CEKFNN)	12.4M	**72.9**	63.7	**60.5**	82.0	80.4	**92.3**	**91.0**	88.7	**83.5/83.7**	**73.5**/74.0

		SuperGLUE				
		BoolQ	CB	COPA	WiC	WSC
BERT$_{base}$ (FT)	110M	**74.1**	**92.7**	57.8	**65.7**	58.5
BERT$_{base}$ (P-Tuning)	11.8M	63.0	71.8	51.5	62.7	60.9
BERT$_{base}$ (LoRA)	0.6M	68.4	71.3	52.3	64.4	58.5
BERT$_{base}$ (AdaLoRA)	0.4M	66.0	69.8	56.3	59.1	62.2
BERT$_{base}$ (CEKFNN)	12.4M	68.6	76.0	**61.7**	65.1	**64.1**
BERT$_{large}$ (FT)	340M	62.3	**82.8**	61.7	50.0	59.4
BERT$_{large}$ (P-Tuning)	21.1M	62.5	71.3	63.2	59.8	61.7
BERT$_{large}$ (LoRA)	1.6M	65.5	69.8	53.9	66.4	58.6
BERT$_{large}$ (AdaLoRA)	1.2M	63.2	67.7	54.7	63.2	57.8
BERT$_{large}$ (CEKFNN)	22.6M	**69.9**	78.6	**70.3**	**66.9**	**63.3**

Table 1: BERT$_{base}$ and BERT$_{large}$ with different methods on the GLUE and SuperGLUE benchmarks. We report the overall (matched and mismatched) accuracy for MNLI and AX, Matthew's correlation for CoLA, Pearson correlation for STS-B, and accuracy for other tasks. For all metrics, higher is better. CEKFNN outperforms several baselines.

Figure 4 illustrates the alternate-training method. Overall, the PMs generate increasingly suitable prompt tokens for the PLM, which in turn uses these prompts to more effectively adapt to downstream tasks.This synergy, amplified by the alternate training approach, significantly enhances the overall performance.

3.4 Model Training

CEKFNN undergoes supervised training using the available labeled data $y^t|_{t\in\{1,2,\ldots,T\}}$. We employ an integrated model to generate both the estimate $\hat{y}^t|_{t\in\{1,2,\ldots,T\}}$ and $u^t|_{t\in\{0,1,\ldots,T\}}$, allowing us to train CEKFNN end-to-end. In other words, we calculate the loss function $\mathcal{L}$ based on the discrete set $\mathcal{Y}$, in which $\hat{y}^t$ takes values. The joint distribution of the hidden states and the observed labels for a single sequence, denoted as $p(u^{\langle 0:T\rangle}, y^{\langle 1:T\rangle}|\theta)$, can be expressed as a factorized form:

$$\begin{aligned} p(u^{\langle 0:T\rangle}, y^{\langle 1:T\rangle}) &= p(u^0)p(y^{\langle 1:T\rangle}|u^{\langle 1:T\rangle}) \\ &= p(u^0)\prod_{t=1}^{T} p(u^t|u^{t-1}, e^t)\prod_{t=1}^{T} p(y^t|u^{t-1}, e^t) \end{aligned} \tag{4}$$

where θ represents all the trainable parameters. The Extended Kalman Filter is typically trained with both predictions and update phases.

The first step predicts the current a-priori statistic based on the previous a posteriori estimates. Specifically, we compute $u^t|_{t\in\{0,1,\ldots,T\}}$ and $\hat{y}^t|_{t\in\{1,2,\ldots,T\}}$ using the transition function $f(\cdot)$ and the emission function $h(\cdot)$ as shown in Equations (1) and (2). During the first time step, we determine u^1 through the expression $u^1 = f(u^0, e^1)$, with u^0 being derived from an initial function constructed using [Liu *et al.*, 2023b].

In the update step, utilizing the downstream loss function $\mathcal{L}$, we can differentially enhance the continuous prompt tokens u^t (where $0 \le t \le T$) by adjusting the trainable parameters of the entire model $\mathcal{M}$ as follows:

$$\hat{\theta} = \underset{\hat{\theta}}{argmin}\mathcal{L}(\mathcal{M}(x^{\langle 1:T\rangle}, y^{\langle 1:T\rangle})) \tag{5}$$

Note that when updating parameters in PLM, the presence of large number of parameters leads to a significant training overhead. In order to tackle this challenge, we employ the adapter method, which involves introducing a small number of parameters into the PLM while keeping all other PLM parameters frozen. In this study, we utilize the Lora adapter method [Hu *et al.*, 2022] to simulate the PLM updates.

The key insight is that the entire process of CEKFNN can be viewed as Extended Kalman Filter (EKF). Consequently, we must determine the appropriate initial probability distribution for the hidden state, the transition function, and the emission function, employing an alternate training strategy.

4 Experiments

We evaluate the downstream task performance of CEKFNN first on BERT [Devlin *et al.*, 2019] and GPT-2 [Radford *et al.*,], before scaling up to T5-3B [Raffel *et al.*, 2020]. To validate the effectiveness of our approach, we select three mainstream language model architectures: BERT, a encoder-based model; GPT-2, a decoder-based model; and T5, an encoder-decoder based model. Our experiments cover a wide range of tasks, including those with single-sentence inputs, semantic similarity detection, question answering, and natural language inference. These tasks are part of the GLUE [Wang *et al.*, 2019b] and SuperGLUE [Wang *et al.*, 2019a] benchmarks. Specifically, we evaluate BERT and GPT-2 on the GLUE benchmark, following their setup, and conduct large-scale experiments on T5 using the SuperGLUE benchmark. All experiments are performed with NVIDIA A100s.

4.1 Datasets

To evaluate CEKFNN, we perform experiments on a total 15 natural language understanding tasks. These tasks cover a wide spectrum of challenges and can be categorized as follows: single-sentence inputs, including Corpus of Linguistic Acceptability (CoLA; [Warstadt *et al.*, 2019]), Stanford Sentiment Treebank (SST-2; [Socher *et al.*, 2013]), and Winograd Schema Challenge (WSC; [Levesque *et al.*, 2012a]);

Model & Method	# Trainable Parameters	GLUE									
		WNLI	RTE	CoLA	MRPC	STS-B	SST-2	QNLI	QQP	MNLI-m/-mm	AX-m/-mm
GPT-2 M (FT)	355M	56.5	**66.6**	50.1	81.0	**83.6**	94.0	89.9	**89.7**	**84.7/85.0**	**75.8/79.0**
GPT-2 M (P-Tuning)	21.1M	63.9	51.1	24.8	51.0	46.8	77.9	80.3	77.2	63.4/56.1	51.8/53.2
GPT-2 M (LoRA)	1.6M	52.9	52.8	44.5	72.5	80.1	92.1	88.2	85.9	80.8/81.3	69.9/70.8
GPT-2 M (AdaLoRA)	1.2M	57.6	56.5	39.0	71.2	77.7	93.8	89.6	86.7	80.9/81.8	70.1/71.8
GPT-2 M (CEKFNN)	22.6M	**70.8**	58.4	**51.2**	**81.9**	83.1	**94.1**	**90.5**	87.6	83.8/84.2	74.4/74.9

		SuperGLUE				
		BoolQ	CB	COPA	WiC	WSC
GPT-2 M (FT)	355M	64.3	65.6	54.5	66.6	61.6
GPT-2 M (P-Tuning)	21.1M	62.4	67.1	48.4	56.4	58.5
GPT-2 M (LoRA)	1.6M	69.4	45.3	52.3	65.0	57.0
GPT-2 M (AdaLoRA)	1.2M	64.2	68.8	60.1	64.2	59.4
GPT-2 M (CEKFNN)	22.6M	**70.0**	**69.3**	**62.5**	**66.9**	**61.8**
GPT-2 L (FT)	774M	67.5	**89.6**	53.9	**69.6**	60.2
GPT-2 L (P-Tuning)	32.8M	63.9	72.4	59.4	57.9	62.5
GPT-2 L (LoRA)	3.0M	68.5	67.7	53.1	67.7	62.5
GPT-2 L (AdaLoRA)	2.2M	65.5	65.6	51.8	60.8	61.6
GPT-2 L (CEKFNN)	35.8M	**72.3**	86.5	**62.5**	69.1	**64.1**

Table 2: GPT-2 medium (M) and GPT-2 large (L) with different methods on the GLUE and SuperGLUE benchmarks. We report the overall(matched and mismatched) accuracy for MNLI and AX, matthew's correlation for CoLA, Pearson correlation for STS-B, and accuracy for other tasks. For all metrics, higher is better. CEKFNN outperforms several baselines.

four tasks involve detecting semantic similarity, such as Microsoft Research Paraphrase Corpus (MRPC; [Dolan and Brockett, 2005];), Quora Question Pairs (QQP; [Alishahi *et al.*, 2019]), Semantic Textual Similarity Benchmark (STS-B; [Cer *et al.*, 2017]), and Word-in-Context (WiC; [Pilehvar and Camacho-Collados, 2019]); one question-answering task, Boolean Question (BoolQ; [Clark *et al.*, 2019a]); and five tasks structured as natural language inference problems, including Multi-Genre NLI corpus (MNLI; [Williams *et al.*, 2018]), Recognizing Textual Entailment (RTE; [Bentivogli *et al.*, 2009]), SQuAD [Rajpurkar *et al.*, 2016], Winograd Schema Challenge (WNLI; [Levesque *et al.*, 2012b]), and Choice of Plausible Alternatives (COPA; [Roemmele *et al.*, 2011]). The individual datasets are released under different permissive licenses.

4.2 Baselines

To facilitate a broad comparison with other baseline methods, we seek to replicate the setups employed by prior research whenever feasible.

Fine-Tuning (FT) is common approach for adaptation. During fine-tuning, the model is initialized to the pre-trained weights and biases, and all pre-trained model parameters undergo gradient updates. A simplified variant involves updating only specific layers while keeping others frozen.

P-Tuning [Liu *et al.*, 2023b] involves inserting special tokens among the input tokens. These special tokens have trainable word embeddings and generally do not exist in the model's vocabulary. The placement of these tokens can significantly impact performance.

LoRA introduces trainable pairs of rank decomposition matrices in parallel to the existing weight matrices. One of the most popular adapter approaches is low-rank adaptation (LoRA), which leverages the insight of decomposing the adapter weights into the product of two low-rank matrices. LoRA has claimed comparable performance to full fine-tuning while utilizing significantly fewer trainable parameters.

AdaLoRA [Zhang *et al.*, 2023] parametrizes the incremental updates of weight matrices using singular value decomposition and dynamically allocates the parameter budget among incremental matrices based on a novel importance metric. AdaLoRA has claimed to achieve performance on par with or surpassing LoRA while employing fewer trainable parameters.

4.3 Evaluation across Models

BERT Base/Large. While BERT [Devlin *et al.*, 2019] has been surpassed by much larger models on NLP leaderboards, such as the GLUE benchmark, in recent years, it remains a competitive and popular pre-trained model among practitioners due to its manageable size. We utilize the pre-trained BERT base (110M) and BERT large (340M) models from the HuggingFace Transformers library [Wolf *et al.*, 2020] to evaluate their performance using various efficient learning approaches on tasks from both the GLUE and SuperGLUE benchmarks. We also replicate the experimental setups of [Devlin *et al.*, 2019], [Liu *et al.*, 2023b], [Hu *et al.*, 2022], and [Zhang *et al.*, 2023]. A consistent batch size is maintained across all tasks, and a sequence length of 512 is employed to align with the baselines. The results can be found in Table 1. For detailed information regarding the hyperparameters used, please refer to Section A.1 in the supplementray materials.

GPT-2 Medium/Large. We obtain the pre-trained GPT-2 [Radford *et al.*,] medium (355M) and GPT-2 large (774M) models from the HuggingFace Transformers library and evaluated their performance using various efficient learning approaches on tasks from the GLUE and SuperGLUE benchmarks. Additionally, we replicate the experimental setups of [Radford *et al.*,], [Liu *et al.*, 2023b], [Hu *et al.*, 2022], and [Zhang *et al.*, 2023]. We maintain a consistent batch size for all tasks and utilize a sequence length of 512 to align with the

Model & Method	# Trainable Parameters	SuperGLUE					
		BoolQ	CB	COPA	WiC	WSC	RTE
T5 (FT)	3B	**88.2**	**89.3**	53.8	**71.4**	**63.5**	**88.6**
T5 (P-Tuning)	21.1M	62.2	50.0	54.5	52.8	58.6	54.8
T5 (LoRA)	11.8M	83.9	65.6	57.0	61.6	58.6	60.6
T5 (AdaLoRA)	8.9M	85.5	60.9	58.0	66.5	58.6	68.6
T5 (CEKFNN)	32.8M	87.4	71.9	**60.9**	69.9	60.9	69.0

Table 3: T5-3b with different methods on the SuperGLUE benchmarks.

Method	# Trainable Parameters	RTE	CoLA	STS-B
PA	22.6M	53.4	1.1	12.4
CEKFNN	22.6M	**58.4**	**51.2**	**83.1**

Table 4: CEKFNN and PA methods on the three different NLP tasks for evaluation. We report the Matthew's correlation for CoLA, Pearson correlation for SST-B, and accuracy for RTE.

Method	# Trainable Parameters	CB	COPA	WSC
LoRA	1.6M	45.3	52.3	57.0
AdaLoRA	1.2M	68.8	60.1	59.4
CEKFNN	22.6M	69.3	62.5	**61.8**
CEKFNN (AdaLoRA)	22.2M	**98.4**	**68.0**	61.7

Table 5: GPT-2 M with CEKFNN uses different adapters on the three tasks. we report accuracy for all tasks.

baselines. The results are presented in Table 2. For detailed information regarding the hyperparameters used, please refer to Section A.2 in the supplementray materials.

Scaling up to T5-3B. As a final stress test for CEKFNN, we scale up to T5 [Raffel *et al.*, 2020], an encoder-decoder based model with 3 billion parameters. Due to the high training cost, we only report partial tasks in SuperGLUE. We utilize the pre-trained T5-3B model from the HuggingFace Transformers library and replicated the setups from [Raffel *et al.*, 2020], [Liu *et al.*, 2023b], [Hu *et al.*, 2022], and [Zhang *et al.*, 2023]. We maintain a consistent batch size across all tasks and use a sequence length of 512 to match the baselines. The results are presented in Table 3. For further details on the hyperparameters used, please refer to Section A.3 in the supplementray materials.

5 Understanding the CEKFNN

CEKFNN achieves either the best or second-best performance in all experiments, thereby substantiating the efficacy of conceptualizing prompt learning through the lens of a Conditional Extended Kalman Filter. Considering the empirical advantage and theoretical analysis of CEKFNN, we aim to further elucidate the properties of CEKFNN when applied to downstream tasks. In this section, our focus is primarily on the widely used GPT-2 medium model, where we have achieved a reduction in trainable parameters while maintaining superior performance compared to some state-of-the-art (SOTA) methods.

We conduct a series of empirical studies to address the following questions: 1) Is the alternative learning method proposed in CEKFNN truly necessary when we have the same number of CEKFNN trainable parameters? 2) How does CEKFNN perform with various adapters?

We believe that our answers to these questions will provide valuable insights into the fundamental principles of using pre-trained language models for downstream tasks, a critical topic in the field of natural language processing (NLP).

5.1 Is the Alternate Learning Procedure Necessary?

To address this question, we compare the performance of CEKFNN with that of PA, which updates all trainable parameters of the models in each loop. We select three different NLP tasks: Single-Sentence Task (CoLA), Similarity and Paraphrase Task (STS-B), and Inference Task (RTE) for evaluation. The results are presented in Table 4. Based on these results, we can observe a significant decline in performance without the alternate learning strategy. This further illustrates the necessity of the CEKFNN method.

5.2 How Does CEKFNN Fare With Various Adapters?

To address this question, we implement CEKFNN using the AdaLoRA adapter and select several tasks where AdaLoRA outperforms Lora. We evaluat the effectiveness of CEKFNN on these tasks, and the results are presented in Table 5. Based on the results, it is evident that CEKFNN (AdaLoRA) achieves better results than AdaLoRA and outperforms most performances compared to CEKFNN with the Lora adapter. These results further demonstrate the generality and effectiveness of the CEKFNN framework.

6 Conclusions

In this work, we study how to improve the accuracy of prompt learning by generating high-quality prompt tokens. To this end, we propose the CEKFNN framework, which models the prompt learning with Extended Kalman Filter to integrate traditional prompt learning methods and adapter methods. The CEKFNN framework enables the generation of time-variant prompt tokens and the updating of the initial, transition, and emission functions with only a small set of trainable parameters. We conduct extensive experiments with diversified model families (BERT, GPT-2, and T5) and model sizes (117M-3B). It shows that the CEKFNN framework achieves comparable or superior performance in comparison to comprehensive fine-tuning methods. In future, we will study how to extend the applications of the CEKFNN framework from NLP tasks to other tasks, such as computer vision and signal processing.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61772492, Grant 62072428, Grant 62271465, in part by the Open Fund Project of Guangdong Academy of Medical Sciences, China under Grant YKY-KF202206, in part by the Suzhou Basic Research Program under Grant SYG202338, and in part by OPPO Research Fund. Xike Xie and Chao Wang are both corresponding authors.

References

[Alishahi *et al.*, 2019] Afra Alishahi, Grzegorz Chrupała, and Tal Linzen. Analyzing and interpreting neural networks for nlp: A report on the first blackboxnlp workshop. *Natural Language Engineering*, 25(4):543–557, 2019.

[Arora *et al.*, 2023] Simran Arora, Avanika Narayan, Mayee F. Chen, Laurel J. Orr, Neel Guha, Kush Bhatia, Ines Chami, and Christopher Ré. Ask me anything: A simple strategy for prompting language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[Bentivogli *et al.*, 2009] Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth pascal recognizing textual entailment challenge. *TAC*, 7:8, 2009.

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

[Cer *et al.*, 2017] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

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

[Clark *et al.*, 2019b] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. What does bert look at? an analysis of bert's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, 2019.

[Davison *et al.*, 2019] Joe Davison, Joshua Feldman, and Alexander M Rush. Commonsense knowledge mining from pretrained models. 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 1173–1178, 2019.

[Dettmers *et al.*, 2023] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[Dolan and Brockett, 2005] Bill Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[Dong *et al.*, 2019] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13042–13054, 2019.

[Hewitt and Manning, 2019] John Hewitt and Christopher D Manning. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, 2019.

[Hu *et al.*, 2022] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[Kim *et al.*, 2018] Yoon Kim, Sam Wiseman, and Alexander M Rush. A tutorial on deep latent variable models of natural language. *arXiv preprint arXiv:1812.06834*, 2018.

[Lampinen *et al.*, 2022] Andrew Lampinen, Ishita Dasgupta, Stephanie Chan, Kory Mathewson, Mh Tessler, Antonia Creswell, James McClelland, Jane Wang, and Felix Hill. Can language models learn from explanations in context? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 537–563, 2022.

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

[Levesque *et al.*, 2012b] Hector Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In

Thirteenth international conference on the principles of knowledge representation and reasoning, 2012.

[Li *et al.*, 2021] Yinghao Li, Pranav Shetty, Lucas Liu, Chao Zhang, and Le Song. Bertifying the hidden markov model for multi-source weakly supervised named entity recognition. 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 6178–6190, 2021.

[Liu *et al.*, 2023a] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

[Liu *et al.*, 2023b] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. *AI Open*, 2023.

[Ouyang *et al.*, 2022] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[Pilehvar and Camacho-Collados, 2019] Mohammad Taher Pilehvar and Jose Camacho-Collados. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, 2019.

[Radford *et al.*,] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners.

[Raffel *et al.*, 2020] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[Rajpurkar *et al.*, 2016] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, 2016.

[Revach *et al.*, 2022] Guy Revach, Nir Shlezinger, Xiaoyong Ni, Adria Lopez Escoriza, Ruud JG Van Sloun, and Yonina C Eldar. Kalmannet: Neural network aided kalman filtering for partially known dynamics. *IEEE Transactions on Signal Processing*, 70:1532–1547, 2022.

[Roemmele *et al.*, 2011] Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*, 2011.

[Shin *et al.*, 2020] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, 2020.

[Socher *et al.*, 2013] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[Wang *et al.*, 2019a] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3261–3275, 2019.

[Wang *et al.*, 2019b] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[Warstadt *et al.*, 2019] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

[Williams *et al.*, 2018] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. 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 1112–1122, 2018.

[Wolf *et al.*, 2020] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45, 2020.

[Xie *et al.*, 2022] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[Zhang *et al.*, 2023] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.