

# SUPPLEMENTARY MATERIAL

## CONTEXT-AWARE PROMPT TUNING: ADVANCING IN-CONTEXT LEARNING WITH ADVERSARIAL METHODS

**Anonymous authors**

Paper under double-blind review

### A STANDARD DEVIATION

**Table 1: Standard Deviation Analysis** Standard deviations (STD) corresponding to Table 1. Each experiment shows three STD values separated by a backslash: (1) STD over 30 experiments with 10 random templates and 3 seeds, (2) mean STD over templates, and (3) mean STD over seeds.

Dataset	Method	BLOOM 1.7B						Model GPT-j 6B						Llama3 8B					
		2	4	6	2	4	6	2	4	6	2	4	6	2	4	6	2	4	6
SST-2	Prefix Tuning	00.5/00.4/00.1	03.1/02.7/01.9	03.1/02.6/02.1	00.8/00.6/00.2	02.3/01.5/00.7	05.8/04.3/03.9	—	—	—	—	—	—	—	—	—	—	—	—
	ICL	04.3/04.0/01.5	12.6/08.6/09.4	14.9/10.6/09.7	05.5/04.0/03.0	14.1/12.6/08.2	13.1/09.9/09.9	13.2/12.7/06.1	15.7/11.9/10.2	13.7/12.2/06.5	—	—	—	—	—	—	—	—	—
	PT†	07.6/07.6/00.4	08.2/08.2/00.7	08.1/08.1/00.6	06.8/06.5/02.3	07.8/06.9/04.3	09.5/09.0/04.9	16.6/16.5/01.0	17.1/17.1/01.6	12.7/10.9/05.7	—	—	—	—	—	—	—	—	—
	PT	08.6/08.5/01.7	08.9/08.6/02.4	08.7/08.4/02.5	07.7/07.6/01.3	07.0/07.0/00.9	07.4/07.4/01.0	06.4/06.1/03.1	06.8/06.6/02.8	07.0/06.5/03.7	—	—	—	—	—	—	—	—	—
	IP†	12.3/12.3/03.4	14.1/11.4/09.8	15.3/10.1/12.6	05.8/05.1/02.5	11.1/08.0/08.3	12.3/12.3/02.0	08.7/06.8/05.6	12.5/11.2/07.3	02.7/02.6/01.6	—	—	—	—	—	—	—	—	—
	IP†	02.0/01.5/00.4	11.6/08.7/08.2	14.8/09.0/11.8	00.7/00.4/00.2	13.0/08.1/10.0	07.2/05.7/04.7	13.7/13.7/03.9	10.6/09.6/05.5	12.0/10.3/05.3	—	—	—	—	—	—	—	—	—
	LoRA	06.9/06.9/00.2	06.5/06.5/00.5	06.4/06.3/00.5	09.5/09.5/00.8	10.2/10.2/01.2	10.0/09.9/01.4	11.4/11.4/01.4	15.0/14.6/07.6	12.1/11.9/07.2	—	—	—	—	—	—	—	—	—
	CPT†	12.3/12.3/03.6	13.8/09.8/10.5	15.4/14.2/09.6	07.6/06.7/03.2	11.1/09.1/07.0	07.0/06.2/04.1	05.0/04.3/02.5	04.1/03.0/02.3	02.0/01.7/01.2	—	—	—	—	—	—	—	—	—
AG News	CPT	07.7/05.3/02.9	12.0/11.1/06.9	12.9/10.5/09.7	05.5/03.9/02.9	12.7/12.0/05.9	10.7/08.9/05.1	13.2/10.9/08.5	01.6/01.5/01.0	01.3/01.2/01.0	—	—	—	—	—	—	—	—	—
	Prefix Tuning	01.7/01.7/00.6	05.8/02.9/05.2	05.3/03.9/03.9	05.9/05.9/00.5	06.2/06.1/01.4	12.7/09.4/08.2	—	—	—	—	—	—	—	—	—	—	—	—
	ICL	10.5/06.8/08.7	11.7/10.1/06.4	12.2/11.1/06.0	10.0/08.9/05.2	13.3/10.6/09.2	10.4/09.4/05.0	08.8/03.2/08.1	03.2/03.0/02.3	03.1/02.7/02.4	—	—	—	—	—	—	—	—	—
	PT†	05.2/04.1/04.1	06.0/04.3/04.4	10.9/10.5/07.3	16.2/16.1/00.9	13.7/11.5/09.5	13.6/11.9/07.8	11.9/11.8/06.1	10.9/10.3/05.8	10.5/09.0/06.5	—	—	—	—	—	—	—	—	—
	PT	07.9/04.6/06.9	08.4/06.8/05.7	09.9/09.3/05.2	12.3/11.5/05.3	10.8/06.7/08.6	07.0/02.5/06.8	15.0/15.0/01.5	15.4/15.2/02.7	12.4/12.3/02.2	—	—	—	—	—	—	—	—	—
	IP†	11.7/09.2/08.4	07.5/05.5/05.0	15.0/09.4/12.4	11.0/08.3/07.9	07.1/03.1/06.5	07.6/03.5/06.8	02.7/02.5/01.6	03.5/02.8/02.2	03.4/02.9/02.5	—	—	—	—	—	—	—	—	—
	IP†	12.0/08.6/08.9	10.6/09.1/07.2	12.0/09.4/07.9	11.1/10.0/06.3	08.8/04.6/07.6	07.4/04.6/05.7	03.6/03.1/01.8	08.1/03.2/07.6	05.2/02.2/04.7	—	—	—	—	—	—	—	—	—
	LoRA	03.2/03.2/00.5	06.8/03.3/06.1	04.6/04.4/02.7	09.0/09.0/01.3	09.4/09.3/01.8	08.8/05.3/07.3	15.0/15.0/01.5	15.4/15.2/02.7	12.4/12.3/02.2	—	—	—	—	—	—	—	—	—
DBpedia	CPT†	07.2/05.7/05.4	06.0/03.4/04.8	11.9/08.5/09.2	09.6/07.2/07.1	09.0/03.6/08.5	09.2/04.4/08.1	03.1/02.5/02.2	02.9/02.6/02.4	03.4/01.8/03.2	—	—	—	—	—	—	—	—	—
	CPT	12.8/08.7/10.3	12.2/07.8/10.3	11.3/08.9/07.1	08.6/05.4/07.4	09.1/03.8/08.5	07.2/03.7/06.2	03.3/02.6/02.3	04.3/03.6/02.8	02.9/02.3/02.3	—	—	—	—	—	—	—	—	—
	Prefix Tuning	03.5/03.4/01.9	06.4/03.1/05.7	08.7/03.8/08.1	02.3/02.3/01.7	04.5/02.7/03.6	07.9/04.9/06.2	—	—	—	—	—	—	—	—	—	—	—	—
	ICL	24.1/23.3/06.9	25.8/23.5/08.9	23.9/23.6/06.0	16.6/16.3/05.9	15.7/13.1/08.1	06.7/05.8/04.0	07.7/06.4/06.2	06.8/02.6/06.5	04.2/02.3/04.0	—	—	—	—	—	—	—	—	—
	PT†	15.6/08.5/13.2	09.7/09.6/01.7	07.1/04.6/05.8	10.3/10.3/00.9	06.3/05.8/04.2	06.3/05.8/04.2	19.5/17.3/11.0	15.7/12.7/09.0	15.4/13.7/08.2	—	—	—	—	—	—	—	—	—
	PT	11.2/11.1/04.2	10.8/10.8/01.3	13.0/12.4/05.9	09.9/08.2/06.4	11.3/07.1/09.1	08.9/05.0/07.5	11.9/11.7/04.2	15.7/15.3/02.9	13.5/13.4/01.5	—	—	—	—	—	—	—	—	—
	IP†	25.6/21.2/14.7	24.3/22.6/08.9	27.3/26.2/08.5	16.4/15.5/05.7	11.0/09.7/06.2	06.8/05.3/05.2	05.3/04.3/03.0	04.5/03.8/02.7	04.5/04.1/01.9	—	—	—	—	—	—	—	—	—
	IP†	26.2/25.1/07.5	25.0/20.3/11.9	07.6/07.0/02.9	12.2/11.2/06.0	09.6/06.0/07.2	05.4/03.7/04.1	09.7/08.4/04.6	06.0/03.6/05.1	05.6/03.1/05.3	—	—	—	—	—	—	—	—	—
TREC	LoRA	11.4/11.0/03.1	11.6/11.6/00.3	11.7/11.7/00.4	11.6/10.9/04.9	13.0/06.0/11.6	09.8/06.0/07.9	13.1/13.0/01.7	14.3/14.2/02.2	13.7/13.7/01.5	—	—	—	—	—	—	—	—	—
	CPT†	23.2/14.5/18.0	12.0/10.1/07.1	22.1/20.1/10.5	15.6/08.6/14.3	06.5/05.0/04.5	06.0/03.2/05.2	06.2/05.6/02.7	03.8/03.4/02.4	02.3/02.2/01.7	—	—	—	—	—	—	—	—	—
	CPT	15.5/13.4/06.5	11.8/08.8/07.1	04.7/04.0/02.5	10.9/08.2/05.5	05.0/03.8/03.5	03.9/03.0/03.0	06.1/05.0/04.8	04.3/03.5/03.0	04.3/02.6/03.5	—	—	—	—	—	—	—	—	—
	Prefix Tuning	06.7/00.8/06.6	06.4/02.9/06.0	07.0/04.6/06.0	03.1/02.0/02.5	05.9/02.6/05.1	03.7/03.6/00.8	—	—	—	—	—	—	—	—	—	—	—	—
	ICL	11.0/07.2/08.2	10.5/08.2/06.8	13.8/09.0/09.1	08.9/05.9/06.8	11.0/08.2/07.8	12.6/08.3/09.4	08.6/05.6/06.3	14.2/07.9/12.0	13.2/08.3/10.7	—	—	—	—	—	—	—	—	—
	PT†	05.9/04.4/03.7	07.5/06.7/04.5	11.2/08.7/07.8	05.5/04.3/03.4	06.2/06.0/04.3	13.5/08.2/11.7	09.5/05.7/08.3	11.3/06.1/09.4	10.3/08.4/08.1	—	—	—	—	—	—	—	—	—
	PT	03.8/03.4/01.5	08.2/07.3/06.7	11.2/08.6/09.1	04.0/04.0/00.9	08.1/07.5/05.7	09.7/08.2/07.8	05.0/05.0/01.5	04.5/04.5/02.5	03.8/03.8/02.0	—	—	—	—	—	—	—	—	—
	IP†	05.5/03.6/04.0	10.1/09.1/07.1	16.8/08.1/15.5	06.8/05.3/04.2	07.7/05.1/05.9	14.0/07.8/11.9	13.6/06.3/12.3	09.7/05.9/08.6	07.2/05.4/02.0	—	—	—	—	—	—	—	—	—
CPT	IP†	10.5/07.1/07.8	06.1/05.9/05.0	14.1/07.1/12.8	09.7/05.4/08.0	09.3/05.7/07.5	13.0/03.8/12.5	12.1/08.5/07.8	11.5/09.1/08.0	14.0/05.4/13.3	—	—	—	—	—	—	—	—	—
	LoRA	03.9/03.9/01.0	04.0/04.0/00.3	04.1/04.1/00.4	02.5/02.5/00.4	03.6/03.5/02.2	11.9/07.2/10.4	03.3/03.3/01.0	03.5/02.7/02.5	16.5/08.1/15.4	—	—	—	—	—	—	—	—	—
	CPT†	08.0/05.8/05.4	07.7/06.9/06.3	09.9/07.0/07.9	08.5/05.7/06.4	12.9/08.3/10.6	11.2/08.2/09.0	13.1/06.9/11.6	09.8/03.7/09.2	05.0/04.1/03.4	—	—	—	—	—	—	—	—	—
	CPT	09.1/05.2/07.3	07.9/07.2/05.6	12.9/07.0/10.8	07.4/04.1/06.1	08.7/06.2/06.9	08.6/05.5/07.3	16.8/08.4/14.5	07.4/06.5/05.8	07.9/05.6/06.5	—	—	—	—	—	—	—	—	—

In table 1 we present the standard deviations (STD) corresponding to the main results shown in Table 1. For each experiment, we display three STD values, separated by a backslash. These values represent the variability in the results across different configurations:

1. The first value shows the standard deviation over 30 experiments, which includes 10 random templates and 3 seeds that determine the training examples.
2. The second value provides the mean of the standard deviation over the templates, the standard deviation across 10 templates, and the mean of the standard deviation across 3 seeds.
3. The third value presents the mean standard deviation over the seeds, the standard deviation over 3 seeds, and the mean over 10 templates.

This detailed breakdown of standard deviations allows for a more thorough understanding of the variability in model performance across different templates and seeds.

## B EVALUATION DETAILS

All the graphs and ablation studies were conducted and evaluated using the DBPedia dataset with the GPT-J model. This setup was chosen due to the diversity of the DBPedia dataset, which includes a broad range of categories and entities, making it an ideal candidate for comprehensive evaluation. The use of GPT-J, a powerful generative model, ensures that the results are reflective of state-of-the-art performance in language modeling tasks. The combination of DBPedia and GPT-J allows us to thoroughly investigate the behavior of the model across various ablation settings, ensuring robust insights into the performance of different methods and configurations.

### B.1 PRUNING FOR CLASSIFICATION

In our evaluation setup, we use pruning for classification by focusing only on the first token of the label, which is unique across all datasets. A common approach in the in-context learning setup is to iterate over all possible labels for each test sample and select the label with the highest probability according to the language model (LM). However, this approach can become computationally expensive, especially in cases where there are a large number of classes.

Similarly to Ratner et al. (2022), and given that the first token in each dataset is unique, we predict only the first token of the label and perform classification based on this value. While this approach deviates slightly from the common practice of iterating over all possible labels, the effect on the results should be minor.

### B.2 TEST SET SIZE

For our experiments, we used a varying number of test examples depending on the dataset. Specifically, we used 100 test examples for the SST-2 dataset, and for datasets with a larger number of classes, the number of test examples was scaled linearly with the number of classes. For example, in the DBpedia dataset, which has 7 times more classes than SST-2, we used 700 test examples to ensure that the evaluation is proportional to the number of classes. This scaling helps to maintain a balanced evaluation across datasets with differing complexities, ensuring robust performance metrics for each method.

## C INSTRUCTION DETAILS

In some of the experiments, we use specific instructions to guide the model in performing the classification tasks. Below in table 2 that shows the instructions used for each dataset across all relevant methods:

Table 2: Instructions used for relevant datasets in the experiments.

Dataset	Instruction
SST2	Classify the sentiment of the following text as positive or negative:
AG News	Classify the following text into one of the following categories: World, Sports, Business, Technology
DBpedia	Classify the following text into one of the following categories: Company, Educational Institution, Artist, Athlete, Office Holder, Mean Of Transportation, Building, Natural Place, Village, Animal, Plant, Album, Film, Written Work
TREC	Classify the following text into one of the following categories: Description, Entity, Expression, Human, Location, Number

Table 3: **Dataset Overview** These are the datasets used, representing a range of different types of classification tasks, including SST-2, AG News, DBpedia, and TREC. Each dataset has a varying number of classes (denoted by  $|C|$ ).

Dataset	Task	$ C $
SST-2	Sentiment analysis (movie)	2
AG News	News classification (topic)	4
DBpedia	Ontology classification	14
TREC	Question classification (answer type)	6

## D DATASET DETAILS

In our experiments, we used four different datasets, each representing a unique classification task. Table 3 provides an overview of the datasets and their respective tasks. Each dataset has a varying number of classes, denoted by  $|C|$ , which are detailed below:

- **SST-2:** This dataset is used for *sentiment analysis*, where the task is to classify movie reviews as either positive or negative. It contains 2 distinct classes.
- **AG News:** The AG News dataset is used for *news classification*. The task is to classify news articles into one of four categories: World, Sports, Business, and Technology. This dataset contains 4 classes.
- **DBpedia:** The DBpedia dataset is focused on *ontology classification*. The task involves classifying textual content into one of 14 distinct categories, which include entities such as Company, Artist, Village, and more.
- **TREC:** This dataset is used for *question classification*, where the goal is to classify questions into one of 6 answer types, including Description, Entity, Human, and Location.

Each dataset contains a specific number of examples based on its classification task, allowing us to evaluate the model’s performance across a diverse range of challenges.

## E TEMPLATE DETAILS

Table 4: **Template Options for Various Datasets** We provide various template options for different datasets. Each dataset include both input and output templates, and also includes intra-separators between inputs and labels, as well as inter-separators between examples.

Dataset	Input Template	Intra-Separator	Output Template	Inter-Separator
SST-2	"input: {}", "text: ",	" "	"output: {}", "target: {}", "label: {}", "emotion: {}", "sentiment: {}", "A {} one.", "It was {}.", "All in all {}.", "A {} piece."	" "
AG News	"sentence: {}",	"\n"	"output: {}", "target: {}", "label: {}",	"\n"
DBpedia	"{}"		"Topic: {}.", "Subject: {}.",	"\n\n"
TREC			"This is about {}.", "It is about {}."	

In our experiments, we use randomly selected templates from the options provided in table 4, suggested in Voronov et al. (2024). Each dataset is associated with both input and output templates, which are used to format the input data and the expected output during few-shot learning tasks.

- **Input Template:** As shown, this column lists the different templates for formatting the input data. For example, the SST-2 dataset uses "input: " and "text: " as input templates to introduce the input text.
- **Intra-Separator:** This separator is used between components (input and output) within a single example. For instance, AG News uses "\n" as an intra-separator between the input sentence and the output label.
- **Output Template:** The output template defines how the expected output is structured. For example, SST-2 employs formats like "output: , target: , label: " to guide the model in generating structured output.
- **Inter-Separator:** This column represents the separator used between multiple examples during training. In datasets like AG News and DBpedia, "\n\n" is used to separate examples.

We randomly select templates from the ones listed in table 4 for each experiment. This randomness in selecting templates introduces variability in the prompts, making the evaluation more robust and testing the model’s ability to generalize across different input-output structures.

## F IMPLEMENTATION DETAILS

### F.1 HYPERPARAMETER DETAILS

In table 5 we present the hyperparameters used in our experiments across different models and datasets. The table provides the specific learning rates ( $\text{'lr'}$ ), epsilon values ( $\text{'}\epsilon\text{'}$ ), and format settings for the various methods applied to each dataset. The experiments were conducted using multiple model architectures, including **BLOOM 1.7B**, **GPT-j 6B**, and **Llama3 8B**, and we selected the best hyperparameters for each experiment: 2, 4, and 6 shots. Below is an overview of the key hyperparameters:

Table 5: **Hyperparameters** Hyperparameters used for each experiment across 2, 4, and 6 shots for different models, including BLOOM 1.7B, GPT-j 6B, and Llama3 8B. The table shows learning rates (lr), epsilon values for input and format, and other parameters for methods such as Prefix Tuning, Prompt Tuning, IPT, LoRA, and CPT. The experiments were conducted on datasets like SST-2, AG News, DBpedia, and TREC.

Dataset	Method	Parameter	BLOOM 1.7B			GPT-j 6B			Llama3 8B		
			2	4	6	2	4	6	2	4	6
SST-2	Prefix Tuning	lr	1e-3	1e-3	1e-3	1e-5	1e-4	1e-3	-	-	-
	PT†	lr	1e-5	1e-5	1e-5	1e-4	1e-3	1e-3	1e-5	1e-5	1e-5
	PT	lr	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5
	IPT†	lr	1e-5	1e-4	1e-4	1e-5	1e-3	1e-4	1e-5	1e-5	1e-4
	IPT	lr	1e-5	1e-5	1e-5	1e-5	1e-4	1e-4	1e-5	1e-5	1e-5
	LoRA	lr	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5	1e-4	1e-4
	CPT†	lr	1e-5	1e-3	1e-4	1e-5	1e-4	1e-3	1e-5	1e-5	1e-5
		Input $\epsilon$	1e-3	1e-0	1e-0	1e-3	1e-1	1e-1	1e-1	1e-1	1e-0
		Format $\epsilon$	1e-3	1e-3	1e-3	1e-3	1e-2	1e-3	1e-2	1e-1	1e-0
	CPT	lr	1e-3	1e-3	1e-4	1e-5	1e-4	1e-4	1e-3	1e-4	1e-4
		Input $\epsilon$	1e-2	1e-0	1e-0	1e-3	1e-0	1e-0	1e-2	1e-0	1e-2
		Format $\epsilon$	1e-2	1e-2	1e-3	1e-3	1e-3	1e-2	1e-3	1e-3	1e-3
AG News	Prefix Tuning	lr	1e-4	1e-3	1e-3	1e-5	1e-5	1e-3	-	-	-
	PT†	lr	1e-3	1e-3	1e-3	1e-5	1e-3	1e-3	1e-4	1e-4	1e-4
	PT	lr	1e-3	1e-3	1e-3	1e-4	1e-3	1e-3	1e-4	1e-5	1e-4
	IPT†	lr	1e-3	1e-3	1e-3	1e-5	1e-4	1e-4	1e-4	1e-5	1e-5
	IPT	lr	1e-4	1e-3	1e-4	1e-5	1e-5	1e-4	1e-5	1e-5	1e-5
	LoRA	lr	1e-5	1e-4	1e-3	1e-5	1e-5	1e-4	1e-5	1e-5	1e-5
	CPT†	lr	1e-4	1e-3	1e-3	1e-4	1e-4	1e-4	1e-5	1e-5	1e-5
		Input $\epsilon$	1e-2	1e-0	1e-0	1e-1	1e-1	1e-2	1e-1	1e-3	1e-3
		Format $\epsilon$	1e-1	1e-2	1e-0	1e-1	1e-3	1e-0	1e-1	1e-2	1e-3
	CPT	lr	1e-4	1e-4	1e-3	1e-3	1e-4	1e-4	1e-3	1e-4	1e-3
		Input $\epsilon$	1e-2	1e-0	1e-0	1e-2	1e-0	1e-0	1e-2	1e-3	1e-3
		Format $\epsilon$	1e-2	1e-0	1e-0	1e-3	1e-3	1e-0	1e-3	1e-3	1e-3
DBpedia	Prefix Tuning	lr	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	-	-	-
	PT†	lr	1e-3	1e-5	1e-3	1e-5	1e-3	1e-3	1e-4	1e-4	1e-4
	PT	lr	1e-4	1e-5	1e-4	1e-3	1e-3	1e-3	1e-4	1e-5	1e-5
	IPT†	lr	1e-4	1e-5	1e-5	1e-5	1e-4	1e-5	1e-5	1e-5	1e-5
	IPT	lr	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5
	LoRA	lr	1e-4	1e-5	1e-5	1e-4	1e-4	1e-4	1e-5	1e-5	1e-5
	CPT†	lr	1e-5	1e-5	1e-5	1e-4	1e-5	1e-5	1e-5	1e-5	1e-5
		Input $\epsilon$	1e-2	1e-2	1e-1	1e-0	1e-1	1e-1	1e-0	1e-1	1e-1
		Format $\epsilon$	1e-1	1e-0	1e-1	1e-3	1e-0	1e-1	1e-1	1e-0	1e-1
	CPT	lr	1e-4	1e-4	1e-5	1e-4	1e-4	1e-4	1e-5	1e-5	1e-5
		Input $\epsilon$	1e-0	1e-2	1e-0	1e-0	1e-0	1e-0	1e-2	1e-0	1e-3
		Format $\epsilon$	1e-0	1e-0	1e-0	1e-0	1e-3	1e-3	1e-2	1e-3	1e-2
TREC	Prefix Tuning	lr	1e-3	1e-3	1e-3	1e-3	1e-3	1e-5	-	-	-
	PT†	lr	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-4	1e-4	1e-4
	PT	lr	1e-5	1e-3	1e-3	1e-5	1e-3	1e-3	1e-5	1e-5	1e-5
	IPT†	lr	1e-3	1e-3	1e-3	1e-4	1e-3	1e-4	1e-4	1e-4	1e-5
	IPT	lr	1e-5	1e-3	1e-3	1e-4	1e-4	1e-4	1e-5	1e-5	1e-5
	LoRA	lr	1e-4	1e-5	1e-5	1e-5	1e-5	1e-4	1e-5	1e-5	1e-4
	CPT†	lr	1e-3	1e-3	1e-3	1e-4	1e-4	1e-x	1e-4	1e-5	1e-5
		Input $\epsilon$	1e-0	1e-0	1e-0	1e-1	1e-0	1e-0	1e-1	1e-1	1e-1
		Format $\epsilon$	1e-3	1e-1	1e-2	1e-1	1e-0	1e-2	1e-3	1e-0	1e-0
	CPT	lr	1e-3	1e-3	1e-4	1e-3	1e-3	1e-3	1e-4	1e-4	1e-4
		Input $\epsilon$	1e-0	1e-0	1e-0	1e-0	1e-0	1e-3	1e-0	1e-0	1e-0
		Format $\epsilon$	1e-0	1e-0	1e-3	1e-2	1e-2	1e-0	1e-2	1e-3	1e-0

- **Learning Rate ('lr')**: The table provides the learning rates used for each method and dataset combination. For methods like *Prefix Tuning (PT)*, *Prompt Tuning (PT)*, *IPT*, and *LoRA*, learning rates vary from **1e-5** to **1e-3**, depending on the specific model and dataset.
- **CPT Hyperparameters**: For *CPT*, we also report epsilon values (' $\epsilon$ ') for both the *input* and the *format* components. These epsilon values control the magnitude of the perturbations applied during optimization. The values of epsilon vary across different models and datasets, generally ranging from **1e-2** to **1e-0** for both input and format components.
- **Model Variability**: The table reflects variability in hyperparameter choices depending on the model size and architecture. For instance, *GPT-3 6B* typically requires higher learning rates compared to *BLOOM 1.7B*, as seen with *CPT* and other methods. The hyperparameters are carefully tuned to optimize performance on tasks such as SST-2, AG News, DBpedia, and TREC.

These hyperparameters are critical for achieving optimal performance in few-shot learning settings. They control the learning process, model updates, and how much the model is allowed to adapt to new data. The values in table 5 are based on extensive experimentation and fine-tuning to ensure the best results for each method and dataset.

## F.2 METHODS IMPLEMENTATION DETAILS

In our experiments, we utilized existing implementations for several methods and implemented IPT ourselves. Specifically, we used the implementations provided by the *Parameter-Efficient Fine-Tuning* Mangrulkar et al. (2022) (PEFT) library<sup>1</sup> for methods such as **LoRA**, **Prefix Tuning**, and **Prompt Tuning (PT)**. For IPT, we built our implementation based on the PEFT framework.

For all experiments, we used the recommended parameters:

- For LoRA, we set  $\alpha = 16$  and the rank  $r = 8$ .
- For Prompt Tuning, Prefix Tuning, and IPT we used 8 learnable tokens.

By using the PEFT framework, we ensure that our fine-tuning processes for LoRA, Prefix Tuning, and PT are aligned with current standards, while our custom IPT implementation extends the framework to allow for additional flexibility in parameter-efficient training.

## F.3 TRAINING DETAILS

We utilized the 'Fine-tune a pretrained model' package from Wolf et al. (2020), which provides a comprehensive framework for training and evaluating models<sup>2</sup>. For all baselines, we employed the default parameters provided by the trainer, ensuring consistency across experiments. Each model was trained for 25 epochs, allowing sufficient time for convergence while maintaining uniform training conditions across methods.

<sup>1</sup><https://huggingface.co/docs/peft/en/index>

<sup>2</sup><https://huggingface.co/docs/transformers/en/training>

## G INPUT PREPARATION

In this section, we provide a detailed explanation of how the input is constructed for different methods, including Prompt Tuning (PT), Instruction Prompt Tuning (IPT), and Context-Aware Prompt Tuning (CPT), both with and without the  $\dagger$  variant. To clarify the differences, we use SST-2 as an example with the instruction: "Classify the sentiment of the following text as positive or negative."

Each example is constructed using a template that includes `input:` and `output:`, where the `input` corresponds to the actual text of the example, and the `output` corresponds to its label. For instance:

- **Example 1:** The input is "the greatest musicians", and the output is "positive".
- **Example 2:** The input is "the action is stilted", and the output is "negative".

Using the template, these examples are represented as:

- **Example 1:** `input: the greatest musicians output: positive`
- **Example 2:** `input: the action is stilted output: negative`

This template-based construction ensures consistency across the methods, allowing us to clearly define how the input and output are represented in different approaches, such as PT, IPT, and CPT.

The table below outlines the construction of the prefix for each method and highlights which parts are updated during training.

Table 6: Input Construction for PT, IPT, and CPT (with and without  $\dagger$ ) using SST-2. The updated text during training is marked in red.

Method	Prefix Construction
<b>PT</b>	In this part, we use only random embedding initialization.
<b>PT<math>\dagger</math></b>	Classify the sentiment of the following text as positive or negative.
<b>IPT</b>	In this part, we use only random embedding initialization. <code>input: the greatest musicians output: positive. input: the action is stilted output: negative.</code>
<b>IPT<math>\dagger</math></b>	Classify the sentiment of the following text as positive or negative. <code>input: the greatest musicians output: positive. input: the action is stilted output: negative.</code>
<b>CPT</b>	<code>input: the greatest musicians output: positive. input: the action is stilted output: negative.</code>
<b>CPT<math>\dagger</math></b>	Classify the sentiment of the following text as positive or negative. <code>input: the greatest musicians output: positive. input: the action is stilted output: negative.</code>

## H PROJECTED GRADIENT DESCENT (PGD) ALGORITHM

In our method, we initialize the context tokens, denoted as  $x_i$ , using the training examples, with each token  $x_i$  associated with a vector  $\delta_i$ , which is initially set to zero. For simplicity, we use  $x_i$  and  $\delta_i$  to denote these components only in this part of the explanation.

During the optimization process, the tokens  $x_i$  remain fixed, while the  $\delta_i$  vectors are updated iteratively. After each optimizer update, we perform a post-processing step where each  $\delta_i$  is projected to ensure that its L2 norm does not exceed a predefined limit,  $\epsilon$ . It is important to note that this projection step is independent of the optimizer and serves as an additional operation to control the extent of change for each context token.

```

1: Initialize each  $\delta_i \leftarrow 0$ 
2: Initialize  $x_i \leftarrow \text{training\_examples\_tokens}$ 
3: for  $j \leftarrow 1$  to  $\text{num\_of\_training\_steps}$  do
4:    $\delta_i \leftarrow \delta_i - \alpha \nabla \text{Loss}(f(x_i + \delta_i), y_i)$  ▷ Gradient descent step
5:    $n_i \leftarrow \|\delta_i\|$  ▷ Compute the L2 norm of  $\delta_i$ 
6:    $\delta_i \leftarrow \delta_i \times \text{clip}(n_i, \epsilon)/n_i$  ▷ Project  $\delta_i$  to ensure L2 norm  $\leq \epsilon$ 
7: end for

```

This ensures that the updates to  $\delta_i$  remain constrained, preventing excessive modifications to the context tokens and maintaining a balance between optimization and regularization. The process allows the model to adapt while ensuring that changes to the context tokens remain meaningful and controlled.

## I EVALUATING THE IMPACT OF PROJECTED GRADIENT DESCENT (PGD)

Our method use the same optimizer used for all baselines. However, our method incorporates an additional step after each parameter update: we project each token, restricting its allowed change. The allowed change is determined by the hyperparameters `Input  $\epsilon$`  and `Format  $\epsilon$` , which define the L2 norm limit for each token’s modification.

To ensure that PGD Madry et al. (2017) is not the sole reason for our method’s improvement, we conducted two types of experiments. First, we compared our method without PGD to PT and IPT. Second, we added a PGD step to PT and IPT for comparison.

For the first experiment, we compared CPT (without PGD) to PT and IPT on the DBpedia dataset. The results for 2, 4, and 6 shots are presented in Table 7.

Table 7: Performance Comparison Without PGD (DBpedia), using GPT-j.

Method	2 Shots	4 Shots	6 Shots
PT	23.39	29.69	40.53
IPT	52.86	67.27	70.73
CPT (No PGD)	68.28	74.17	77.52

For the second experiment, we compared  $\text{CPT}^\dagger$  to  $\text{PT}^\dagger$  and  $\text{IPT}^\dagger$  (with and without PGD) on the DBpedia dataset. To ensure a fair comparison, we performed hyperparameter tuning (HPT) over  $\epsilon$  and the learning rate for both PT and IPT. The results for 2, 4, and 6 shots are presented in Table 8.

Table 8: Performance Comparison With and Without PGD (DBpedia), using GPT-j.

Method	2 Shots	4 Shots	6 Shots
$\text{PT}^\dagger$	12.96	22.12	37.44
$\text{PT}^\dagger + \text{PGD}$	12.80	22.02	38.69
$\text{IPT}^\dagger$	47.10	66.37	75.09
$\text{IPT}^\dagger + \text{PGD}$	47.10	66.40	75.09
$\text{CPT}^\dagger + \text{PGD}$	52.87	77.30	81.00

The results clearly demonstrate that, in both experiments, our method consistently outperforms PT and IPT. Furthermore, it is evident that other methods do not necessarily benefit from the addition of PGD. While we cannot definitively explain this, we hypothesize that it may be due to the highly effective way in which we employ PGD, leveraging prior knowledge about the structure of the input, format, and labels within the context. Our approach allows us to apply distinct projections to different components of the context, which we believe significantly contributes to the superior performance of our method.

## REFERENCES

- Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.
- Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. <https://github.com/huggingface/peft>, 2022.
- Nir Ratner, Yoav Levine, Yonatan Belinkov, Ori Ram, Inbal Magar, Omri Abend, Ehud Karpas, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. Parallel context windows for large language models. *arXiv preprint arXiv:2212.10947*, 2022.
- Anton Voronov, Lena Wolf, and Max Ryabinin. Mind your format: Towards consistent evaluation of in-context learning improvements. *arXiv preprint arXiv:2401.06766*, 2024.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing, 10 2020. URL <https://github.com/huggingface/transformers>.