🔪 KNIFE: DISTILLING META-REASONING KNOWLEDGE WITH FREE-TEXT RATIONALES

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent works have explored using free-text rationales (FTRs)—*i.e.*, natural language explanations of a task output—to teach language models (LMs) how to solve NLP tasks. In these works, the LM is often finetuned or prompted to *jointly generate* the FTR and task output. However, this approach either involves *finetuning* LMs on possibly conflicting objectives or *prompting* prohibitively large LMs. To address this, we propose KNIFE, which guides LM reasoning via FTR knowledge distillation, instead of via FTR generation. KNIFE first finetunes an FTR-augmented teacher LM to predict the task output, then finetunes a student LM so that its hidden states are aligned with the teacher's. As a result, the student LM learns general reasoning knowledge from the FTRs and can be used for inference, without FTR generation or large LMs. On two question answering datasets, we show that KNIFE outperforms various baselines in both fully-supervised and low-resource settings. Also, using two more datasets, we analyze KNIFE's failure modes and identify FTR quality as critical to KNIFE performance.[1]

1 INTRODUCTION

Whereas conventional supervised learning only gives feedback on a language model's (LM's) task output correctness, *explanation-based learning* (EBL) aims to improve LM generalization by additionally explaining the correct reasoning process behind a given correct output (Hase & Bansal, 2021; Narang et al., 2020; Joshi et al., 2022). In particular, there is growing interest in learning from *free-text rationales* (FTRs), which use natural language to explain the reasoning process for solving a given task instance (Narang et al., 2020; Rajani et al., 2019; Camburu et al., 2018; Wei et al., 2022).

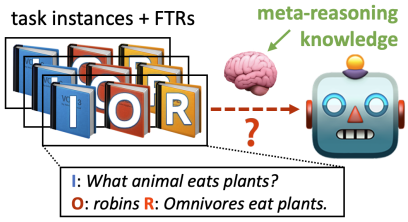


Figure 1: **FTR Knowledge Injection.** A free-text rationale (FTR) explains how to solve a given task instance. Collectively, FTR-augmented task instances (*i.e.,* **I**nput, **O**utput, FT**R**) can provide meta-reasoning knowledge for solving unseen instances. Still, it remains unclear how to effectively inject this knowledge into LMs.

Prior works have considered three paradigms for FTR learning (Fig. 2). In the *input augmentation* paradigm, the LM is finetuned to generate the task output given both the task input and an FTR (Sun et al., 2022; Wang et al., 2022; Wiegreffe et al., 2021; Hase et al., 2020). In the *self-rationalization* paradigm, the LM is finetuned or prompted to generate both the task output and an FTR (Narang et al., 2020; Brahman et al., 2021; Li et al., 2022; Wei et al., 2022; Marasović et al., 2022; Zelikman et al., 2022; Liu et al., 2018). In the *pipeline rationalization* paradigm, a finetuned rationalizing LM first generates the FTR, which is then used as input for a finetuned reasoning LM to predict the task output (Kumar & Talukdar, 2020; Rajani et al., 2019; Wiegreffe et al., 2021; Hase et al., 2020).

However, these FTR learning paradigms either struggle to improve the LM's task performance (input augmentation, finetuned self-rationalization, pipeline rationalization) or require prohibitively large LMs (*i.e.,* >100B) to work well (prompted self-rationalization). While an individual FTR provides instance-level reasoning knowledge, a set of FTRs can collectively convey task-level *meta-reasoning knowledge* that generalizes to many unseen task instances (Fig. 1). With this in mind, our goal is to

[1]Code and data will be publicly released after the review period.

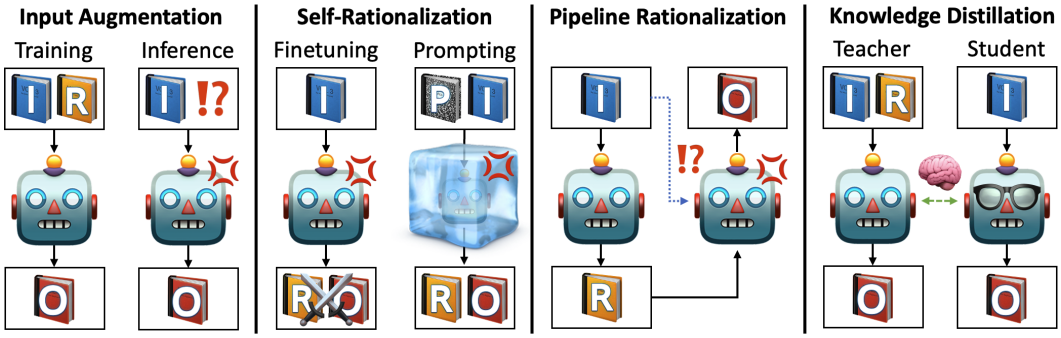


Figure 2: **FTR Learning Methods** aim to transfer meta-reasoning knowledge from FTR-augmented task instances (*i.e.,* **I**nput, **O**utput, FT**R**) to LMs. Existing methods follow *input augmentation*, *self-rationalization*, or *pipeline rationalization*. However, input augmentation causes input distribution shift (training *vs.* inference); self-rationalization has conflicting losses (finetuning) or requires large LMs (prompting); and pipeline rationalization predicts task output without direct access to task input. We propose KNIFE, which avoids these issues by using *knowledge distillation* to transfer meta-reasoning knowledge from an FTR-augmented teacher LM to a student LM.

extract meta-reasoning knowledge from a set of FTRs and effectively inject this knowledge into a small-scale (*i.e.,* <1B) LM, in order to improve its generalization ability.

We propose **KN**owledge D**I**stillation with **F**ree-Text Rational**E**s (🔪 **KNIFE**). Instead of simply inserting FTRs into the LM's input or target output, KNIFE guides the LM's reasoning by distilling meta-reasoning knowledge from the FTRs to the LM's hidden states. First, given both the task input and an FTR, a teacher LM is finetuned to predict the task output. Second, given only the task input, a student LM is finetuned so that its task input/output (hidden) states align with the teacher's. Since the student does not take FTR inputs, the teacher's forward computation includes a *bottleneck* stage where its FTR (hidden) states are masked out. Beyond this bottleneck, the teacher must use only its task input/output states to predict the task output, thus routing knowledge from the FTR to the teacher's task input/output states.

Unlike input augmentation, KNIFE does not need FTRs for inference or cause an input distribution shift, since the student is given only the task input during both training and inference. Unlike finetuned self-rationalization, KNIFE does not involve jointly optimizing task and FTR generation losses (which may conflict), since the student is only trained with KNIFE's distillation losses. Unlike prompted self-rationalization, KNIFE does not require large LMs, since the teacher and student are finetuned instead of relying only on their pretrained knowledge. Unlike pipeline rationalization, KNIFE does not require multiple inference LMs or create a non-differentiable path between them, since only the student is used for inference. Plus, KNIFE generally has lower inference-time costs, since the student does not process additional FTR inputs or generate additional FTR tokens.

On two question answering (QA) datasets (OBQA, StrategyQA), we show that KNIFE outperforms various baselines on both fully-supervised and low-resource settings, using either gold FTRs or generated FTRs (§4.5, §A.4). Furthermore, we validate our KNIFE design choices via extensive ablation studies (§A.5). Finally, using two additional datasets (ECQA, QuaRTz), we analyze KNIFE's failure modes and identify FTR quality as a critical factor in KNIFE performance (§A.6).

2 BACKGROUND

Problem Definition Let $\mathcal{F}$ denote an LM for text classification. Given task input $\mathbf{x}$ and class labels $Y = \{\mathbf{y}_i\}$, $\mathcal{F}$'s goal is to predict a confidence score $\rho(\mathbf{x}, \mathbf{y}_i)$ for each $(\mathbf{x}, \mathbf{y}_i)$ pair, so the predicted label $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}_i \in Y} \rho(\mathbf{x}, \mathbf{y}_i)$ matches the gold (*i.e.,* correct) label $\mathbf{y}^* \in Y$. We consider multi-choice (Y can vary across instances) and closed-set (Y is fixed) text classification.

Free-Text Rationales Given a $(\mathbf{x}, \mathbf{y}_i)$ pair, a *free-text rationale* (FTR) $\mathbf{r}$ is a natural language text sequence that explains the reasoning process for predicting label $\mathbf{y}_i$ (Camburu et al., 2018; Rajani et al., 2019). Compared to extractive rationales (Sundararajan et al., 2017; Li et al., 2016; Chan et al., 2022b), FTRs may be more intuitive to humans, can reference things beyond the task input, and support high flexibility in content, style, and length (Wiegreffe et al., 2021; Chan et al., 2022a). As a result, recent works have explored generating FTRs to explain LM behavior (Camburu et al., 2018;

Narang et al., 2020; Wei et al., 2022) and utilizing FTRs to improve LM decision-making (Sun et al., 2022; Wang et al., 2022; Li et al., 2022).

3 KNIFE

Existing works aim to learn from FTRs by using approaches like input augmentation, self-rationalization, or pipeline rationalization. However, such approaches may hurt task performance or require prohibitively large LMs to work well. To address this, we propose KNIFE, a knowledge distillation (KD) approach for injecting FTR-based meta-reasoning knowledge into LMs (Fig. 3).

3.1 LM DESIGNS

KNIFE consists of a teacher LM and a student LM. Below, we discuss the design of each LM type.

Basic LM Design Following prior works, we use text-to-text (*i.e.,* encoder-decoder) Transformer (Vaswani et al., 2017) architectures for both teacher and student LMs (Raffel et al., 2020; Narang et al., 2020). Building upon §2, let the task input be denoted as n_x-token sequence $\mathbf{x} = [x^{(1)}, x^{(2)}, ..., x^{(n_x)}]$, while each label is denoted as n_{y_i}-token sequence $\mathbf{y}_i = [y_i^{(1)}, y_i^{(2)}, ..., y_i^{(n_{y_i})}] \in Y$. Given an LM $\mathcal{F}$, we can decouple $\mathcal{F}$ as encoder $\mathcal{F}_{\text{enc}}$ and decoder $\mathcal{F}_{\text{dec}}$. By default, $\mathcal{F}_{\text{enc}}$ outputs hidden states $\mathcal{F}_{\text{enc}}(\mathbf{x}) = [\mathbf{e}_x^{(1)}, \mathbf{e}_x^{(2)}, ...\mathbf{e}_x^{(n_x)}]$.

Typically, decoding is done by taking only a special start token as $\mathcal{F}_{\text{dec}}$'s input, then using greedy search to sequentially generate the remaining tokens until a special end token is generated (Raffel et al., 2020; Vaswani et al., 2017). However, when using this decoding strategy for classification, it is possible to generate an output sequence that does not match any of the labels in Y. To avoid this issue, we instead produce a dedicated decoder output for each label sequence $\mathbf{y}_i$, by separately teacher-forcing each $\mathbf{y}_i$ to be $\mathcal{F}_{\text{dec}}$'s input (Wang et al., 2022). This gives us a conditional probability $P(y_i^{(j)} \,|\, y_i^{(1)}, ..., y_i^{(j-1)}, \mathbf{x})$ for each token $y_i^{(j)}$ in $\mathbf{y}_i$. Then, following Wang et al. (2022) and Shwartz et al. (2020), we compute $\rho_i = \rho(\mathbf{x}, \mathbf{y}_i)$, the confidence score for $\mathbf{y}_i$, by aggregating these token probabilities as: $\rho_i = \frac{1}{n_{y_i}} \sum_{j=1}^{n_{y_i}} \log P(y_i^{(j)} \,|\, y_i^{(1)}, ..., y_i^{(j-1)}, \mathbf{x})$. Finally, we use the softmax function to normalize ρ_i as label confidence probability $P(\mathbf{y}_i \,|\, \mathbf{x}) = e^{\rho_i} / \sum_{j=1}^{|Y|} e^{\rho_j}$. Given gold label $\mathbf{y}^*$, the goal of the downstream classification task is to train $\mathcal{F}$ such that $P(\mathbf{y}^* \,|\, \mathbf{x})$ is maximized.

Teacher LM Design KNIFE first finetunes an FTR-augmented teacher LM to solve the task, using both the task input $\mathbf{x}$ and an FTR $\mathbf{r}$ as input. We assume that the FTR expresses a sufficiently correct reasoning process corresponding to the gold label $\mathbf{y}^*$. By default, $\mathbf{r}$ is a gold FTR, which is human-annotated to support $\mathbf{y}^*$. Let $\mathcal{T}$ denote the teacher LM, which has encoder $\mathcal{T}_{\text{enc}}$ and decoder $\mathcal{T}_{\text{dec}}$. Let the FTR be denoted as n_r-token sequence $\mathbf{r} = [r^{(1)}, r^{(2)}, ..., r^{(n_r)}]$. Hence, $\mathcal{T}$'s input is denoted as $[\mathbf{x}, \mathbf{r}]$, where $|[\mathbf{x}, \mathbf{r}]| = n_x + n_r$. As a result, $\mathcal{T}_{\text{enc}}$ outputs hidden states $\mathbf{e}_{\text{T}} = \mathcal{T}_{\text{enc}}([\mathbf{x}, \mathbf{r}]) = [\mathbf{e}_{\text{T}x}, \mathbf{e}_{\text{T}r}]$. In particular, we refer to $\mathbf{e}_{\text{T}x}$ as the *task input states* and $\mathbf{e}_{\text{T}r}$ as the *FTR states*.

Unlike the basic LM design described earlier, $\mathcal{T}$ has a *bottleneck* in the cross-attention between $\mathcal{T}_{\text{enc}}$ and $\mathcal{T}_{\text{dec}}$. Here, $\mathbf{e}_{\text{T}r}$ is masked out so that $\mathcal{T}_{\text{dec}}$ only has access to $\mathbf{e}_{\text{T}x}$ during decoding. This means all FTR knowledge must be funneled to $\mathcal{T}_{\text{dec}}$ through $\mathbf{e}_{\text{T}x}$. Thus, for each label $\mathbf{y}_i$, $\mathcal{T}_{\text{dec}}$ produces hidden states $\mathbf{d}_{\text{T}y_i} = \mathcal{T}_{\text{dec}}(\mathbf{y}_i, \mathbf{e}_{\text{T}x}) = [\mathbf{d}_{\text{T}y_i}^{(1)}, \mathbf{d}_{\text{T}y_i}^{(2)}, ..., \mathbf{d}_{\text{T}y_i}^{(n_{y_i})}]$, which we call the *task output states*. The bottleneck is designed to move knowledge from the FTR to $\mathbf{e}_{\text{T}x}$ and $\mathbf{d}_{\text{T}y_i}$, through which knowledge is distilled to the student (which has no $\mathbf{e}_{\text{T}r}$). Note that the same $\mathbf{r}$ (which supports gold label $\mathbf{y}^*$) is used for decoding each label $\mathbf{y}_i$. We can interpret $\mathcal{T}_{\text{dec}}$'s decoding process as predicting the compatibility between $\mathbf{y}_i$ and the correct reasoning process expressed by $\mathbf{r}$. We assume that, among these labels, only $\mathbf{y}^*$ is compatible with $\mathbf{r}$.

Student LM Design After finetuning $\mathcal{T}$, KNIFE finetunes a student LM based on KD objectives, given only $\mathbf{x}$. Let $\mathcal{S}$ be the student LM, with encoder $\mathcal{S}_{\text{enc}}$ and decoder $\mathcal{S}_{\text{dec}}$. Since $\mathcal{S}$ does not have access to $\mathbf{r}$, $\mathcal{S}_{\text{enc}}$ only outputs task input states $\mathbf{e}_{\text{S}} = \mathcal{S}_{\text{enc}}(\mathbf{x}) = [\mathbf{e}_{\text{S}x}^{(1)}, \mathbf{e}_{\text{S}x}^{(2)}, ..., \mathbf{e}_{\text{S}x}^{(n_x)}]$ and can be used for inference. Hence, $\mathcal{S}$ does not have a bottleneck. For each label $\mathbf{y}_i$, $\mathcal{S}_{\text{dec}}$ produces task output states $\mathbf{d}_{\text{S}y_i} = \mathcal{S}_{\text{dec}}(\mathbf{y}_i, \mathbf{e}_S) = [\mathbf{d}_{\text{S}y_i}^{(1)}, \mathbf{d}_{\text{S}y_i}^{(2)}, ..., \mathbf{d}_{\text{S}y_i}^{(n_{y_i})}]$.

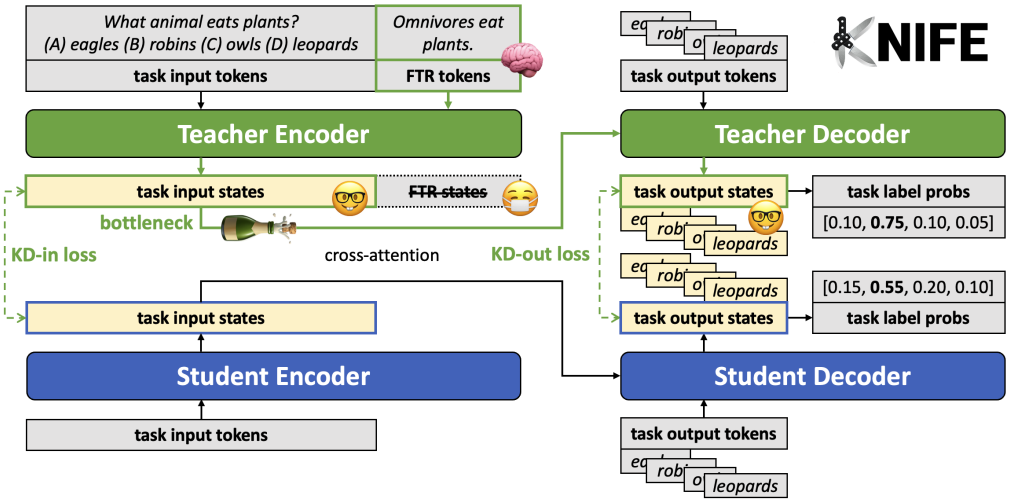


Figure 3: **KNIFE Framework.** KNIFE distills meta-reasoning knowledge from an FTR-augmented teacher LM (given task input and FTR) to a student LM (given task input) that is used for inference. The teacher has a *bottleneck*, which masks out all FTR states during cross-attention. As a result, the teacher's decoder must use only its task input states to compute its task output states, thus routing FTR knowledge to the task input/output states. Finally, knowledge is distilled to the student by training the student so its task input/output states to align with the teacher's.

3.2 Knowledge Distillation

Recall that $\mathcal{T}$ is trained to predict the compatibility of each $(\mathbf{y}_i, \mathbf{r})$ pair. $\mathcal{T}$'s bottleneck design causes the FTR knowledge to be routed to $\mathbf{e}_{\text{T}x}$ and $\mathbf{d}_{\text{T}y_i}$. First, since $\mathcal{T}_{\text{dec}}$ can only use $\mathbf{e}_{\text{T}x}$ as context, $\mathcal{T}_{\text{enc}}$ must be trained to distill sufficient knowledge from $\mathbf{r}$ into the $\mathbf{e}_{\text{T}x}$. Second, since $\mathcal{T}_{\text{dec}}$ is the component that actually performs the reasoning process to compute the final compatibility scores, $\mathbf{d}_{\text{T}y_i}$ must also contain useful knowledge. Thus, in KNIFE, KD is done by training the student LM $\mathcal{S}$ so that its task input states and/or task output states are aligned with the teacher LM's. Besides the KD losses, we can also train $\mathcal{S}$ with the same task loss used to train $\mathcal{T}$. However, we show in our experiments that $\mathcal{S}$ actually performs better when trained with only KD losses (§A.5).

Now, we formally define these learning objectives. Given predicted label distribution P and target label distribution Q, we first define the task loss $\mathcal{L}_{\text{task}}$ as cross-entropy loss $\mathcal{L}_{\text{task}} = -\sum_{\mathbf{y}_i \in Y} Q(\mathbf{y}_i \,|\, \mathbf{x}) \log P(\mathbf{y}_i \,|\, \mathbf{x})$, where $Q(\mathbf{y}_i \,|\, \mathbf{x})$ is 1 if $\mathbf{y}_i = \mathbf{y}^*$ and 0 otherwise. Next, let dist denote an arbitrary distance function, *e.g.,* mean squared error (MSE). Let $\mathcal{L}_{\text{KD-in}}$ denote KNIFE's task input states based KD loss, which pushes the student LM's task input states ($\mathbf{e}^{(j)}_{\text{S}x}$) to be closer to the teacher LM's ($\mathbf{e}^{(j)}_{\text{T}x}$): $\mathcal{L}_{\text{KD-in}} = \frac{1}{n_x} \sum_{j=1}^{n_x} \text{dist}(\mathbf{e}^{(j)}_{\text{S}x}, \mathbf{e}^{(j)}_{\text{T}x})$. Similarly, let $\mathcal{L}_{\text{KD-out}}$ denote KNIFE's task output states based KD loss, which pushes the student LM's task output states ($\mathbf{e}^{(j)}_{\text{S}y_i}$) to be closer to the teacher LM's ($\mathbf{e}^{(j)}_{\text{T}y_i}$): $\mathcal{L}_{\text{KD-out}} = \frac{1}{|Y|\, n_{y_i}} \sum_{y_i \in Y} \sum_{j=1}^{n_{y_i}} \text{dist}(\mathbf{e}^{(j)}_{\text{S}y_i}, \mathbf{e}^{(j)}_{\text{T}y_i})$. Finally, let $\mathcal{L}$ denote the total loss, defined as $\mathcal{L} = \lambda_{\text{task}}\mathcal{L}_{\text{task}} + \lambda_{\text{KD-in}}\mathcal{L}_{\text{KD-in}} + \lambda_{\text{KD-out}}\mathcal{L}_{\text{KD-out}}$, with loss weights λ_{task}, $\lambda_{\text{KD-in}}$, and $\lambda_{\text{KD-out}}$. The teacher is trained only with $\mathcal{L}_{\text{task}}$, whereas the student may or may not use $\mathcal{L}_{\text{task}}$.

4 Experiments

This section (along with the appendix) presents experiments showing KNIFE's effectiveness in FTR learning. First, in both fully-supervised and low-resource settings, we show that KNIFE outperforms various baselines, using either gold FTRs or generated FTRs (§4.5, §A.4). Second, we validate our KNIFE design choices via extensive ablation studies (§A.5). Third, we analyze KNIFE's failure modes and identify FTR quality as critical to KNIFE performance (§A.6).

4.1 Evaluation Protocol

Datasets Since FTRs are commonly annotated for QA datasets (Wiegreffe & Marasovic, 2021), we focus on two QA datasets: OBQA and StrategyQA. OBQA (Mihaylov et al., 2018) is a multi-choice (*i.e.,* four-choice) QA dataset that simulates science exams. StrategyQA (Geva et al., 2021) is a

closed-set (*i.e.,* binary yes/no) QA dataset that tests open-domain knowledge. Since StrategyQA does not provide public test set labels, we use the data split from Wang et al. (2022).

Presentation of Results For all results, we report mean and standard deviation accuracy over three random seeds (`mean` $\pm$ `std`). For each table, we use horizontal lines to partition the table into sub-tables. Each sub-table contains results for methods that have comparable settings, so `Result1` should only be compared to `Result2` if there is no horizontal line between them. In each sub-table, we highlight the best performing method in red and the second-best performing method in blue.

4.2 Baselines

Standard Finetuning does not use FTRs or KD. **FT (I→O)** finetunes an LM to generate the task output, given the task input. This is equivalent to training KNIFE's student LM without KD.

Input Augmentation appends an FTR to a finetuned LM's input. **FT (IR→O)** finetunes an LM to generate the task output, given the task input and an FTR. However, for fair comparison, the FTR input is omitted during inference. This is equivalent to training KNIFE's teacher LM without the bottleneck. **FT Dropout (IR→O)** mitigates FT (IR→O)'s input distribution shift by randomly dropping out the FTR input during training.

Finetuned Self-Rationalization appends an FTR to a finetuned LM's target output. **FT (I→OR)** finetunes an LM to generate the task output followed by the FTR, given the task input. **FT (I→RO)** finetunes an LM to generate the FTR followed by the task output, given the task input.

Prompted Self-Rationalization uses chain-of-thought (CoT) prompting (Wei et al., 2022). **CoT (I→RO)** prompts an LM to generate the FTR followed by the task output, given the task input.

Pipeline Rationalization finetunes two LMs as a pipeline. For **FT (I→R→O)**, the first LM is finetuned to generate the FTR given the task input, while the second LMis finetuned to generate the task output given the first LM's generated FTR. Since FT (I→R→O) is known to not perform well (Wiegreffe et al., 2021; Wang et al., 2022), we only consider FT (I→R→O) in a limited set of settings. In these settings, we simply present the results reported in Wang et al. (2022).

FT Teacher Init. (I→O) modifies FT (I→O) by initializing the LM with the KNIFE teacher's parameters, which is equivalent to training KNIFE's student with task loss only. This is used to verify that the KNIFE student's gains come from KD, not from approximating the teacher's parameters.

4.3 KNIFE Variants

We consider three KNIFE variants, each with a different combination of the $\mathcal{L}_{\text{KD-in}}$ and $\mathcal{L}_{\text{KD-out}}$ losses (§3.2). **KNIFE (In)** trains the student LM only using $\mathcal{L}_{\text{KD-in}}$. **KNIFE (Out)** trains the student LM only using $\mathcal{L}_{\text{KD-out}}$. **KNIFE (In+Out)** trains the student LM using both $\mathcal{L}_{\text{KD-in}}$ and $\mathcal{L}_{\text{KD-out}}$. By default, we use KNIFE (In+Out) and do not train the student with task loss $\mathcal{L}_{\text{task}}$ (§A.5).

4.4 Implementation Details

Following prior works (Narang et al., 2020; Sun et al., 2022; Wiegreffe et al., 2021), we use T5-Base and T5-Large (Raffel et al., 2020). For KD methods, **T5-Base→T5-Base** means teacher and student use T5-Base, **T5-Large→T5-Large** means teacher and student use T5-Large, and **T5-Large→T5-Base** means T5-Large teacher and T5-Base student. For non-KD methods, T5-Base→T5-Base, T5-Large→T5-Large, and T5-Large→T5-Base mean the LM uses T5-Base, T5-Large, and T5-Base, respectively. For KNIFE, the student uses the teacher's language modeling head. Also, for KNIFE, if the student and teacher have the same architecture, the student's parameters are initialized as the teacher's. See §A.2-§A.3 for more about hyperparameters and implementation details, respectively.

4.5 Main Results

Table 1 presents our main results. Here, LMs are finetuned on the entire training set, using gold FTRs if applicable. We observe that KNIFE (In+Out) consistently outperforms all baselines. Unlike FT (I→O), KNIFE (In+Out) benefits from training with FTR knowledge. Unlike FT (I→OR) and FT

(I→RO), KNIFE (In+Out) does not have an FTR generation loss to compete with the task loss. FT (I→RO) performs especially poorly because generating the FTR before the task output makes the LM more likely to skip the task output. Unlike FT (I→R→O), KNIFE (In+Out)'s LM has direct access to the task input when predicting the task output. Unlike FT (IR→O) and FT Dropout (I→RO), KNIFE (In+Out) avoids input distribution shift since its student only takes task input in training and inference. Unlike FT Teacher Init. (I→O), KNIFE (In+Out) explicitly trains the student to follow the teacher's reasoning process.

Also, we report CoT (I→RO) results for GPT-NeoX (20B) (Black et al., 2022) and GPT-3 (text-davinci-003, 175B) (Brown et al., 2020) Since GPT-NeoX and GPT-3 are much larger than T5-Base (220M) and T5-Large (770M), it is unfair to expect other methods to perform as well as CoT (I→RO). Even so, we find that KNIFE (In+Out) greatly outperforms GPT-NeoX on all settings, while KNIFE (In+Out) with T5-Large achieves similar performance to GPT-3 on StrategyQA. See §A.4 for results with GPT-NeoX generated FTRs and low-resource learning.

Architecture	**Method**	**Accuracy** (↑)	
		OBQA	StrategyQA
T5-Base→T5-Base	FT (I→O)	57.93 (±1.15)	59.05 (±0.23)
	FT (I→OR)	53.93 (±1.33)	51.84 (±1.45)
	FT (I→RO)	39.40 (±1.20)	0.87 (±0.50)
	FT (I→R→O)	56.65	57.11
	FT (IR→O)	53.73 (±2.31)	49.97 (±2.92)
	FT Dropout (IR→O)	58.27 (±1.33)	59.39 (±1.33)
	FT Teacher Init. (I→O)	58.33 (±0.90)	57.25 (±2.22)
	KNIFE (In+Out)	61.53 (±0.76)	60.45 (±0.31)
T5-Large→T5-Large	FT (I→O)	65.60 (±0.40)	57.58 (±0.70)
	FT (I→OR)	61.93 (±1.97)	57.58 (±0.12)
	FT (I→RO)	44.00 (±2.00)	0.80 (±0.35)
	FT (IR→O)	61.27 (±2.16)	53.24 (±2.54)
	FT Dropout (IR→O)	66.67 (±0.83)	59.25 (±4.59)
	FT Teacher Init. (I→O)	65.67 (±2.25)	61.72 (±2.36)
	KNIFE (In+Out)	68.73 (±1.36)	63.79 (±0.64)
T5-Large→T5-Base	Best T5-Base→T5-Base	58.33 (±0.90)	59.39 (±1.33)
	KNIFE (In+Out)	60.93 (±0.12)	61.12 (±2.03)
GPT-NeoX	CoT (I→RO)	33.80	55.31
GPT-3 (text-davinci-003)	CoT (I→RO)	86.40	66.53

Table 1: **KNIFE Main Results**

5 Related Work

Learning from Free-Text Rationales There are three main FTR learning paradigms: input augmentation, self-rationalization, and pipeline rationalization. In input augmentation, the LM is finetuned to generate the task output given both the task input and an FTR (Sun et al., 2022; Wang et al., 2022; Wiegreffe et al., 2021; Hase et al., 2020). Still, this either assumes access to gold (or large-LM-generated) FTRs during inference, or introduces an input distribution shift between training and inference when FTRs are unavailable during inference. In self-rationalization, the LM is finetuned or prompted to generate both the task output and an FTR (Narang et al., 2020; Brahman et al., 2021; Li et al., 2022; Wei et al., 2022; Marasović et al., 2022; Zelikman et al., 2022; Liu et al., 2018; Majumder et al., 2022). However, for self-rationalization, finetuning may create conflict between the task and FTR objectives, while prompting requires very large LMs to work well. In pipeline rationalization, a finetuned rationalizing LM first generates the FTR, which is then used as input for a finetuned reasoning LM to predict the task output (Kumar & Talukdar, 2020; Rajani et al., 2019; Wiegreffe et al., 2021; Hase et al., 2020). Here, the generated FTR forms a non-differentiable path between the two LMs, which complicates end-to-end training and can hurt task performance. Unlike prior works, KNIFE distills FTR knowledge from an FTR-augmented teacher LM to a student LM that does not have FTR inputs, in order to improve the student's generalization.

Knowledge Distillation Knowledge distillation has been widely used to transfer knowledge from a larger teacher to a smaller student model. (Hinton et al.; Sanh et al., 2019; Jiao et al., 2020; Mirzadeh et al., 2020, etc). Instead of aiming for this typical goal, KNIFE distills the FTR knowledge learned by a teacher model to a student model, which has no direct access to FTRs. Similar to the line of work that incorporates knowledge distillation with privileged information (Lopez-Paz et al., 2015; Vapnik et al., 2015; Fukuda et al., 2017; Wang et al., 2018), where student models benefit from privilege information, the student model in KNIFE essentially gains additional knowledge from FTRs rather than relying on the larger teacher model capacity. Snell et al. (2022) propose to internalize the in-context learning ability such that the performance gains can keep without context tokens. It does not directly distill the knowledge from FTRs and requires dedicated prompt designs. Shridhar et al. (2022); Magister et al. (2022); Ho et al. (2022) propose to distill reasoning abilities from larger language models to smaller models. They require large-scale language models with such abilities, while KNIFE can work well with small models.

REFERENCES

Shourya Aggarwal, Divyanshu Mandowara, Vishwajeet Agrawal, Dinesh Khandelwal, Parag Singla, and Dinesh Garg. Explanations for commonsenseqa: New dataset and models. In *ACL*, 2021.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*, 2022.

Faeze Brahman, Vered Shwartz, Rachel Rudinger, and Yejin Choi. Learning to rationalize for nonmonotonic reasoning with distant supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 12592–12601, 2021.

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.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. e-snli: Natural language inference with natural language explanations. *Advances in Neural Information Processing Systems*, 31, 2018.

Aaron Chan, Shaoliang Nie, Liang Tan, Xiaochang Peng, Hamed Firooz, Maziar Sanjabi, and Xiang Ren. Frame: Evaluating rationale-label consistency metrics for free-text rationales. *arXiv preprint arXiv:2207.00779*, 2022a.

Aaron Chan, Maziar Sanjabi, Lambert Mathias, Liang Tan, Shaoliang Nie, Xiaochang Peng, Xiang Ren, and Hamed Firooz. Unirex: A unified learning framework for language model rationale extraction. In *International Conference on Machine Learning*, pp. 2867–2889. PMLR, 2022b.

Takashi Fukuda, Masayuki Suzuki, Gakuto Kurata, Samuel Thomas, Jia Cui, and Bhuvana Ramabhadran. Efficient knowledge distillation from an ensemble of teachers. In *Interspeech*, pp. 3697–3701, 2017.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021.

Peter Hase and Mohit Bansal. When can models learn from explanations? a formal framework for understanding the roles of explanation data. *arXiv preprint arXiv:2102.02201*, 2021.

Peter Hase, Shiyue Zhang, Harry Xie, and Mohit Bansal. Leakage-adjusted simulatability: Can models generate non-trivial explanations of their behavior in natural language? In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4351–4367, 2020.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network.

Namgyu Ho, Laura Schmid, and Se-Young Yun. Large language models are reasoning teachers. *arXiv preprint arXiv:2212.10071*, 2022.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*, 2022.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4163–4174, 2020.

Brihi Joshi, Aaron Chan, Ziyi Liu, Shaoliang Nie, Maziar Sanjabi, Hamed Firooz, and Xiang Ren. Er-test: Evaluating explanation regularization methods for language models. *arXiv preprint arXiv:2205.12542*, 2022.

Sawan Kumar and Partha Talukdar. Nile: Natural language inference with faithful natural language explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8730–8742, 2020.

Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*, 2016.

Shiyang Li, Jianshu Chen, Yelong Shen, Zhiyu Chen, Xinlu Zhang, Zekun Li, Hong Wang, Jing Qian, Baolin Peng, Yi Mao, et al. Explanations from large language models make small reasoners better. *arXiv preprint arXiv:2210.06726*, 2022.

Frederick Liu and Besim Avci. Incorporating priors with feature attribution on text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6274–6283, 2019.

Hui Liu, Qingyu Yin, and William Yang Wang. Towards explainable nlp: A generative explanation framework for text classification. *arXiv preprint arXiv:1811.00196*, 2018.

David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643*, 2015.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. *arXiv preprint arXiv:2212.08410*, 2022.

Bodhisattwa Prasad Majumder, Oana Camburu, Thomas Lukasiewicz, and Julian Mcauley. Knowledge-grounded self-rationalization via extractive and natural language explanations. In *International Conference on Machine Learning*, pp. 14786–14801. PMLR, 2022.

Ana Marasović, Iz Beltagy, Doug Downey, and Matthew E Peters. Few-shot self-rationalization with natural language prompts. *NAACL*, 2022.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 5191–5198, 2020.

Sharan Narang, Colin Raffel, Katherine Lee, Adam Roberts, Noah Fiedel, and Karishma Malkan. Wt5?! training text-to-text models to explain their predictions. *arXiv preprint arXiv:2004.14546*, 2020.

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

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4932–4942, 2019.

Laura Rieger, Chandan Singh, William Murdoch, and Bin Yu. Interpretations are useful: penalizing explanations to align neural networks with prior knowledge. In *International conference on machine learning*, pp. 8116–8126. PMLR, 2020.

Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. Right for the right reasons: training differentiable models by constraining their explanations. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 2662–2670, 2017.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. Distilling multi-step reasoning capabilities of large language models into smaller models via semantic decompositions. *arXiv preprint arXiv:2212.00193*, 2022.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Unsupervised commonsense question answering with self-talk. *arXiv preprint arXiv:2004.05483*, 2020.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. *arXiv preprint arXiv:2209.15189*, 2022.

Jiao Sun, Swabha Swayamdipta, Jonathan May, and Xuezhe Ma. Investigating the benefits of free-form rationales. *arXiv preprint arXiv:2206.11083*, 2022.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pp. 3319–3328. PMLR, 2017.

Oyvind Tafjord, Matt Gardner, Kevin Lin, and Peter Clark. Quartz: An open-domain dataset of qualitative relationship questions. In *EMNLP*, 2019.

Vladimir Vapnik, Rauf Izmailov, et al. Learning using privileged information: similarity control and knowledge transfer. *J. Mach. Learn. Res.*, 16(1):2023–2049, 2015.

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

Peifeng Wang, Aaron Chan, Filip Ilievski, Muhao Chen, and Xiang Ren. Pinto: Faithful language reasoning using prompt-generated rationales. *arXiv preprint arXiv:2211.01562*, 2022.

Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. Kdgan: Knowledge distillation with generative adversarial networks. *Advances in neural information processing systems*, 31, 2018.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Sarah Wiegreffe and Ana Marasovic. Teach me to explain: A review of datasets for explainable natural language processing. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

Sarah Wiegreffe, Ana Marasović, and Noah A Smith. Measuring association between labels and free-text rationales. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 10266–10284, 2021.

Eric Zelikman, Yuhuai Wu, and Noah D Goodman. Star: Bootstrapping reasoning with reasoning. *arXiv preprint arXiv:2203.14465*, 2022.

A APPENDIX

A.1 BACKGROUND (EXTENDED)

Problem Definition Let $\mathcal{F}$ denote an LM for text classification. Given a task input $\mathbf{x}$ and a set of class labels $Y = \{\mathbf{y}_i\}$, $\mathcal{F}$'s goal is to predict a confidence score $\rho(\mathbf{x}, \mathbf{y}_i)$ for each $(\mathbf{x}, \mathbf{y}_i)$ pair, so that the predicted label $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}_i \in Y} \rho(\mathbf{x}, \mathbf{y}_i)$ matches the gold (*i.e.,* correct) label $\mathbf{y}^* \in Y$. In this work, we consider both multi-choice (Y can vary across task instances) and closed-set (Y is the same for all task instances) text classification.

Free-Text Rationales Given some $(\mathbf{x}, \mathbf{y}_i)$ pair, a *free-text rationale* (FTR) $\mathbf{r}$ is a natural language text sequence that explains the reasoning process for predicting label $\mathbf{y}_i$ (Camburu et al., 2018; Rajani et al., 2019) Most explainability works have focused on extractive rationales (Sundararajan et al., 2017; Li et al., 2016; Chan et al., 2022b), which highlight important features (*e.g.,* tokens) in x, but there is growing interest in FTRs (Narang et al., 2020; Kumar & Talukdar, 2020; Wei et al., 2022). First, compared to extractive rationales, FTRs may be more intuitive and understandable to humans, since they follow how humans communicate in natural language (Camburu et al., 2018; Wiegreffe et al., 2021). Second, in contrast to extractive rationales, FTRs can reference things beyond the task input (Rajani et al., 2019; Wiegreffe et al., 2021). Third, FTRs support high flexibility in content, style, and length (Chan et al., 2022a). Hence, FTRs can potentially provide much richer knowledge than extractive rationales. As a result, a number of recent works have explored generating FTRs to explain LM behavior (Narang et al., 2020; Rajani et al., 2019; Camburu et al., 2018) and utilizing FTRs to improve LM decision-making (Sun et al., 2022; Wang et al., 2022; Li et al., 2022).

Learning from Free-Text Rationales Since extractive rationales assign an importance score to each input feature, there is a one-to-one correspondence between the features and scores (Sundararajan et al., 2017; Li et al., 2016; Chan et al., 2022b; Joshi et al., 2022). Given this structure, prior works regularize LM behavior by aligning the LM's extractive rationales with human-annotated extractive rationales (Joshi et al., 2022; Ross et al., 2017; Rieger et al., 2020; Liu & Avci, 2019). However, unlike extractive rationales, FTRs do not have such a feature-score correspondence to leverage. Without this kind of structure, it becomes less straightforward to effectively learn from FTRs.

The FTR learning literature revolves around three main paradigms: input augmentation (Sun et al., 2022; Wang et al., 2022; Wiegreffe et al., 2021), self-rationalization (Narang et al., 2020; Wei et al., 2022; Marasović et al., 2022), and pipeline rationalization (Kumar & Talukdar, 2020; Rajani et al., 2019; Hase et al., 2020). These paradigms explicitly add the FTR to the LM's input or target output, but this general approach has key limitations. First, adding FTRs to the LM's input assumes access to sufficiently "correct" (*i.e.,* explains correct reasoning process) FTRs during inference. Yet, this is unrealistic because having correct FTRs would already be tantamount to solving the task. Meanwhile, simply omitting the FTR only during inference introduces an input distribution shift between training and inference, thus hurting inference performance. Second, adding FTRs to the LM's target output assumes that task prediction and FTR generation are complementary objectives. In practice, this is often not the case, especially if the generated FTR does not support the LM's predicted label. Plus, if the LM is trained to generate the predicted label and FTR as a single output sequence (Narang et al., 2020; Wiegreffe et al., 2021), then it is not uncommon for the LM to generate a predicted label that does not match any valid label candidates or even omit the predicted label entirely.

A.2 HYPERPARAMETERS

For KD, we used sweeps of $\lambda_{\text{task}} = [0, 1]$, $\lambda_{\text{KD-in}} = [0, 1]$, and $\lambda_{\text{KD-out}} = [0, 1]$. We always use AdamW (Loshchilov & Hutter, 2017) as the optimizer. For early stopping, we stop training when the model performance on the development set has not improved for five epochs. The maximum number of epochs is 10. For OBQA with T5-Base, we train the teacher model with learning rate of $1\text{e}{-4}$ and batch size of 64. We train the student model (by KD) with batch size of 64. For OBQA with T5-Large, we train the teacher model with learning rate of $5\text{e}{-5}$ and batch size of 64. We train the student model with batch size of 48. For the KD training student model, we always search the learning rate in $\{1\text{e}{-4}, 2\text{e}{-4}, 3\text{e}{-4}, 4\text{e}{-4}, 5\text{e}{-4}\}$.

Architecture	Method	Accuracy (↑)	
		OBQA	StrategyQA
T5-Base→T5-Base	FT (I→O)	57.93 (±1.15)	59.05 (±0.23)
	FT (I→OR)	50.60 (±1.25)	53.04 (±1.36)
	FT (I→RO)	21.40 (±1.31)	30.39 (±2.32)
	FT (IR→O)	48.40 (±1.71)	49.37 (±2.52)
	FT Dropout (IR→O)	58.53 (±1.14)	60.39 (±1.17)
	FT Teacher Init. (I→O)	59.80 (±1.64)	59.25 (±0.42)
	KNIFE (In+Out)	61.53 (±0.76)	61.92 (±1.04)
T5-Large→T5-Large	FT (I→O)	65.60 (±0.40)	57.58 (±0.70)
	FT (I→OR)	59.20 (±1.56)	59.79 (±2.20)
	FT (I→RO)	26.13 (±1.55)	28.52 (±1.82)
	FT (IR→O)	53.13 (±1.94)	49.70 (±3.03)
	FT Dropout (IR→O)	66.87 (±0.31)	59.85 (±1.80)
	FT Teacher Init. (I→O)	66.87 (±1.10)	59.99 (±1.01)
	KNIFE (In+Out)	68.73 (±1.55)	63.99 (±0.81)
T5-Large→T5-Base	Best T5-Base→T5-Base	59.80 (±1.64)	60.39 (±1.17)
	KNIFE (In+Out)	60.47 (±0.81)	62.39 (±0.42)
GPT-NeoX	CoT (I→RO)	33.80	55.31
GPT-3 (text-davinci-003)	CoT (I→RO)	86.40	66.53

Table 2: **KNIFE Main Results (GPT-NeoX FTRs)**

For StrategyQA, we always set the warmup rate as 0.06. For T5-Base, we train the teacher model with learning rate of $3\text{e}{-4}$ and batch size of 16. For T5-Large, we train the teacher model with learning rate of $5\text{e}{-5}$ and batch size of 16. For the KD training student model, the batch size is always 16, and we always search the learning rate in $\{1\text{e}{-4}, 2\text{e}{-4}, 3\text{e}{-4}, 4\text{e}{-4}, 5\text{e}{-4}\}$.

FT (I→O), FT (I→OR), FT (I→RO), and FT (IR→O) use the same hyperparameters as the teacher models in the same settings do, except that we always search the learning rate in $\{1\text{e}{-5}, 2\text{e}{-5}, 5\text{e}{-5}\ 1\text{e}{-4}, 2\text{e}{-4}, 3\text{e}{-4}, 4\text{e}{-4}, 5\text{e}{-4}\}$. FT Dropout (IR→O) uses the same hyperparameters as FT (IR→O) does, and we search the dropout rate in $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. FT Teacher Init. (I→O) uses the same hyperparameters as FT (I→O) does.

A.3 Implementation Details

For T5-Base, the parameter number of a single backbone model is around 220M. As we have two backbone models for the teacher and student model, the total number is around 440M. For T5-Large, the parameter number of a single backbone model is around 770M, and the total number is around 1.5B. GPT-NeoX has 20B parameters. We use NVIDIA Quadro RTX8000 GPUs for all experiments, which take around 700 GPU hours in total. We implement the models using Hugging Face Transformers, PyTorch, and Lightning.

FT (I→OR) and FT (I→RO) use greedy decoding (§3.1) because, unlike task labels, there do not exist multiple FTR choices (besides gold FTR $\mathbf{y}^*$) to separately teacher-force into the decoder. When using greedy decoding for classification, it is possible for the LM to generate an output sequence that does not match any of the labels in Y. Thus, the LM may perform much worse than random chance. For T5-Large→T5-Base, the teacher and student have different embedding dimensionalities. Thus, for the task input states and task output states, we train a linear projection layer transforming the student LM's states to have the same dimensionality as the teacher LM's. Note that only the mean performance is available for FT (I→R→O), since these results were obtained from Wang et al. (2022). All CoT (I→RO) results were obtained from Wang et al. (2022), except GPT-3 on OBQA, which was obtained from Huang et al. (2022).

A.4 Main Results (Extended)

In Table 2, we repeat these experiments for GPT-NeoX generated FTRs (instead of gold FTRs) and obtain the same conclusions. This makes sense since GPT-NeoX's FTRs are known to fluently convey useful knowledge (Black et al., 2022). Interestingly, KNIFE with GPT-NeoX FTRs still considerably

Method	OBQA Acc. (↑)
FT (I→O)	44.87 (±0.23)
FT (I→OR)	38.07 (±1.14)
FT (I→RO)	28.93 (±1.81)
FT (IR→O)	44.80 (±2.23)
FT Dropout (IR→O)	47.00 (±1.00)
FT Teacher Init. (I→O)	44.20 (±2.95)
KNIFE (In)	47.20 (±1.40)
KNIFE (Out)	48.13 (±2.12)
KNIFE (In+Out)	47.47 (±1.96)

Table 3: **KNIFE Low-Resource Learning Results**

Architecture	Method	Accuracy (↑)	
		OBQA	StrategyQA
T5-Base→T5-Base	KNIFE (In) + Gold	60.00 (±0.40)	61.39 (±1.90)
	KNIFE (Out) + Gold	62.27 (±1.01)	59.05 (±1.22)
	KNIFE (In+Out) + Gold	61.53 (±0.76)	60.45 (±0.31)
	KNIFE (In) + GPT-NeoX	61.07 (±0.12)	61.92 (±1.74)
	KNIFE (Out) + GPT-NeoX	61.60 (±0.53)	60.72 (±0.20)
	KNIFE (In+Out) + GPT-NeoX	61.53 (±0.76)	61.92 (±1.04)
T5-Large→T5-Large	KNIFE (In) + Gold	66.20 (±0.53)	62.66 (±3.38)
	KNIFE (Out) + Gold	68.07 (±1.50)	64.40 (±1.22)
	KNIFE (In+Out) + Gold	68.73 (±1.36)	63.79 (±0.64)
	KNIFE (In) + GPT-NeoX	67.20 (±0.40)	62.32 (±1.84)
	KNIFE (Out) + GPT-NeoX	68.53 (±1.89)	62.26 (±0.64)
	KNIFE (In+Out) + GPT-NeoX	68.73 (±1.55)	63.99 (±0.81)
T5-Large→T5-Base	KNIFE (In) + Gold	31.13 (±2.87)	53.77 (±0.46)
	KNIFE (Out) + Gold	55.60 (±2.42)	61.12 (±0.53)
	KNIFE (In+Out) + Gold	60.93 (±0.12)	61.12 (±2.03)
	KNIFE (In) + GPT-NeoX	30.07 (±2.97)	53.91 (±0.69)
	KNIFE (Out) + GPT-NeoX	55.60 (±2.03)	60.59 (±0.61)
	KNIFE (In+Out) + GPT-NeoX	60.47 (±0.81)	62.39 (±0.42)

Table 4: **KNIFE Variants**

outperforms CoT (I→RO) with GPT-NeoX, despite KNIFE using much smaller LMs. Furthermore, in Table 3, we consider a low-resource setting where LMs are finetuned on only 10% of the training data, using T5-Base→T5-Base on OBQA. We find that KNIFE beats all baselines in the low-resource setting, showing that KNIFE more efficiently leverages the FTR learning signal.

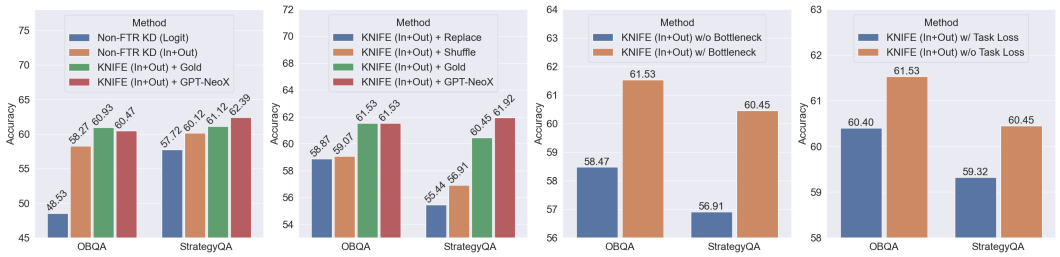
A.5 Ablation Studies

To justify design choices made for KNIFE and understand why it works, we present five KNIFE ablation studies, analyzing the impacts of KD objective, FTR usage, FTR quality, teacher bottleneck, and student task loss.

KD Objectives Table 4 compares the performance of KNIFE (In), KNIFE (Out), and KNIFE (In+Out). For both gold FTRs and GPT-NeoX FTRs, we find that KNIFE (In+Out) generally achieves the highest performance. This suggests that useful FTR knowledge can be distilled via both task input states and task output states, so it is best to use both. Although KNIFE (In) and KNIFE (Out) perform similarly for T5-Base→T5-Base and T5-Large→T5-Large, KNIFE (In) performs much worse for T5-Large→T5-Base. Since KNIFE (In) only performs KD via the task input states, the student's decoder cannot be trained without task loss. Yet, for T5-Large→T5-Base, the student cannot be initialized with the teacher's parameters, leaving the decoder with T5-Base's pretrained parameters. While this could be addressed by training the student with task loss, it shows a major disadvantage of KNIFE (In) compared to other KNIFE variants.

FTR Usage This experiment verifies the importance of FTR usage in KD by comparing KNIFE to non-FTR KD baselines, where the teacher is FT (I→O). **Non-FTR KD (Logit)** finetunes the student so its logit distribution aligns with the teacher's. **Non-FTR KD (In+Out)** finetunes the student so its task input states and task output states aligns with the teacher's. Since non-FTR KD generally requires the teacher to be larger than the student, we only use T5-Large→T5-Base here. In Fig. 4a, both KNIFE (In+Out) + Gold and KNIFE (In+Out) + GPT-NeoX outperform the non-FTR KD baselines, showing that FTR-based KD is helpful. Plus, Non-FTR KD (Logit) performs much worse than Non-FTR KD (In+Out), which further validates KNIFE's use of representation-based KD.

FTR Quality By default, KNIFE uses gold FTRs to train the teacher, and we have also explored using GPT-NeoX FTRs. Using T5-Base→T5-Base, we investigate the relationship between KNIFE performance and FTR quality by considering KNIFE variants that train the teacher on noisy FTRs: **Replace** creates noisy FTRs by replacing each gold FTR token with a random token from the token vocabulary. **Shuffle** creates noisy FTRs by shuffling gold FTRs across all training instances in the dataset. In Fig. 4b, KNIFE (In+Out)'s performance with Gold and GPT-NeoX is much higher than with Replace and Shuffle. Plus, Replace performs worse than Shuffle since Replace's token-level randomization fully corrupts the FTR. This suggests that KNIFE's performance is positively correlated with FTR quality.



(a) FTR Usage (b) FTR Type (c) Teacher Bottleneck (d) Student Task Loss

Figure 4: **KNIFE Ablation Studies**

Teacher Bottleneck We validate KNIFE's teacher bottleneck by considering a KNIFE variant with no bottleneck. In Fig. 4c, for T5-Base→T5-Base, KNIFE (In+Out) w/ Bottleneck greatly outperforms KNIFE (In+Out) w/o Bottleneck. This demonstrates the bottleneck's effectiveness in routing FTR knowledge to the teacher's task input/output states, through which FTR knowledge can be distilled to the student. In Table 8, we show that the bottleneck is also very important for KNIFE (In), but not as critical for KNIFE (Out).

Student Task Loss By default, KNIFE trains the student with only KD losses. We justify this design choice by comparing to KNIFE variants where the student is also trained on the task loss. In Fig. 4d, we see that omitting the task loss consistently yields higher performance, which indicates that KD losses are not always compatible with task loss. If the KD and task losses produce conflicting gradients during optimization, the student may get confused and learn a suboptimal "in-between" reasoning process leading to even worse generalization.

A.6 Failure Analysis

Although KNIFE performs well on OBQA and StrategyQA, it yield negative results on other QA datasets like ECQA (Aggarwal et al., 2021) and QuaRTz (Tafjord et al., 2019). Using T5-Base→T5-Base, we compare the performance of KNIFE and most of the baselines considered in the main results. In Table 5, we see that KNIFE generally outperforms all FTR-based baselines, sometimes by a very large margin. Still, none of the FTR-based methods (including all KNIFE variants) are able to significantly outperform FT (I→O).

Since KNIFE distills FTR knowledge to the student LM, the student's performance is expected to depend on the amount and quality of meta-reasoning knowledge stored in the FTRs. Thus, to investigate these negative results, we conducted a case study to qualitatively analyze the gold FTRs in

Architecture	Method	Accuracy (↑)	
		ECQA	QuaRTz
T5-Base→T5-Base	FT (I→O)	62.02 (±0.48)	68.20 (±0.52)
	FT (I→OR)	56.09 (±0.47)	57.19 (±0.58)
	FT (I→RO)	13.31 (±1.12)	29.38 (±1.85)
	FT (IR→O)	41.02 (±1.57)	66.41 (±0.90)
	KNIFE (In)	55.12 (±2.19)	68.45 (±0.83)
	KNIFE (Out)	57.26 (±2.68)	68.45 (±0.52)
	KNIFE (In+Out)	56.12 (±1.91)	68.41 (±0.99)

Table 5: **Negative Results**

Method	Accuracy (↑)	
	OBQA	StrategyQA
Non-FTR KD (Logit)	48.53 (±4.06)	57.72 (±2.12)
Non-FTR KD (In)	31.87 (±2.10)	53.77 (±0.46)
KNIFE (In) + Gold	31.13 (±2.87)	53.77 (±0.46)
KNIFE (In) + GPT-NeoX	30.07 (±2.97)	53.91 (±0.69)
Non-FTR KD (Out)	55.60 (±2.99)	59.99 (±3.53)
KNIFE (Out) + Gold	55.60 (±2.42)	61.12 (±0.53)
KNIFE (Out) + GPT-NeoX	55.60 (±2.03)	60.59 (±0.61)
Non-FTR KD (In+Out)	58.27 (±1.01)	60.12 (±1.40)
KNIFE (In+Out) + Gold	60.93 (±0.12)	61.12 (±2.03)
KNIFE (In+Out) + GPT-NeoX	60.47 (±0.81)	62.39 (±0.42)

Table 6: **Ablation Study on FTR Usage.**

OBQA, StrategyQA, ECQA, and QuaRTz. Overall, we found that FTRs in OBQA and StrategyQA are more informative than those in ECQA and QuaRTz. For OBQA and StrategyQA, we find that gold FTRs tend to have the following properties. First, they describe a logically sufficient reasoning process for getting from the question (input) to the answer (output). Second, they provide general and self-contained knowledge that goes beyond the information given in the question and answer. This means they do not simply rephrase the question and/or answer. Meanwhile, FTRs from ECQA and QuaRTz tend to exhibit the opposite properties.

To illustrate, we give a representative example of a good OBQA FTR: "***Question**: There is most likely going to be fog around:* ***Answer Choices**: (A) a marsh, (B) a tundra, (C) the plains (D) a desert.* ***Gold FTR**: fog is formed by water vapor condensing in the air.*"

We also give a representative example of a bad ECQA FTR: "***Question**: What might a person see at the scene of a brutal killing?* ***Answer Choices**: (A) bloody mess, (B) pleasure, (C) being imprisoned, (D) feeling of guilt, (E) cake.* ***Gold FTR**: Bloody mess is covered or stained with blood. A person might see a bloody mess at the scene of a brutal killing.*"

A.7 Ablation Studies (Extended)

We present the full results of ablation studies in the Appendix. Table 6 shows the full results of ablation studies on FTR Usage. Table 7 shows the full results of ablation studies on FTR Quality. Table 8 shows the full results of ablation studies on teacher bottleneck. Table 9 shows the full results of ablation studies on student task loss. The details of ablation studies are in A.5. In the last three tables, we always use T5-Base→T5-Base.

FTR Type	KNIFE Variant	Accuracy (↑)	
		OBQA	StrategyQA
Replace	KNIFE (In)	57.47 (±0.42)	56.65 (±1.75)
Shuffle	KNIFE (In)	57.73 (±1.01)	56.65 (±2.71)
Gold	KNIFE (In)	60.00 (±0.40)	61.39 (±1.90)
GPT-NeoX	KNIFE (In)	61.07 (±0.12)	61.92 (±1.74)
Replace	KNIFE (Out)	58.67 (±1.10)	54.31 (±2.84)
Shuffle	KNIFE (Out)	58.87 (±1.50)	57.11 (±1.64)
Gold	KNIFE (Out)	62.27 (±1.01)	59.05 (±1.22)
GPT-NeoX	KNIFE (Out)	61.60 (±0.53)	60.72 (±0.20)
Replace	KNIFE (In+Out)	58.87 (±1.30)	55.44 (±4.43)
Shuffle	KNIFE (In+Out)	59.07 (±0.31)	56.91 (±1.59)
Gold	KNIFE (In+Out)	61.53 (±0.76)	60.45 (±0.31)
GPT-NeoX	KNIFE (In+Out)	61.53 (±0.76)	61.92 (±1.04)
Replace	KNIFE Teacher	57.73 (±0.61)	55.31 (±3.03)
Shuffle	KNIFE Teacher	56.40 (±1.20)	56.05 (±2.93)
Gold	KNIFE Teacher	73.80 (±0.60)	66.20 (±1.10)
GPT-NeoX	KNIFE Teacher	74.33 (±0.46)	64.93 (±1.40)

Table 7: **Ablation Study on FTR Quality.**

Bottleneck	KNIFE Variant	Accuracy (↑)	
		OBQA	StrategyQA
No	KNIFE (In)	58.67 (±0.70)	49.77 (±2.91)
Yes	KNIFE (In)	60.00 (±0.40)	61.39 (±1.90)
No	KNIFE (Out)	62.20 (±0.72)	59.92 (±0.87)
Yes	KNIFE (Out)	62.27 (±1.01)	59.05 (±1.22)
No	KNIFE (In+Out)	58.47 (±0.83)	56.91 (±2.31)
Yes	KNIFE (In+Out)	61.53 (±0.76)	60.45 (±0.31)
No	KNIFE Teacher	73.40 (±1.51)	67.47 (±0.42)
Yes	KNIFE Teacher	73.80 (±0.60)	66.20 (±1.10)

Table 8: **Ablation Study on Teacher Bottleneck.**

Task Loss	KNIFE Variant	Accuracy (↑)	
		OBQA	StrategyQA
Yes	KNIFE (In)	59.73 (±1.10)	56.31 (±1.00)
No	KNIFE (In)	60.00 (±0.40)	61.39 (±1.90)
Yes	KNIFE (Out)	58.53 (±1.47)	58.65 (±2.02)
No	KNIFE (Out)	62.27 (±1.01)	59.05 (±1.22)
Yes	KNIFE (In+Out)	60.40 (±1.04)	59.32 (±0.35)
No	KNIFE (In+Out)	61.53 (±0.76)	60.45 (±0.31)

Table 9: **Ablation Study on Student Task Loss.**