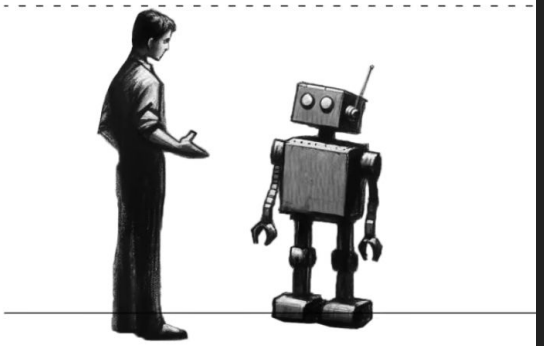
Weak-to-Strong Generalization: Eliciting Strong Capabilities With Weak Supervision

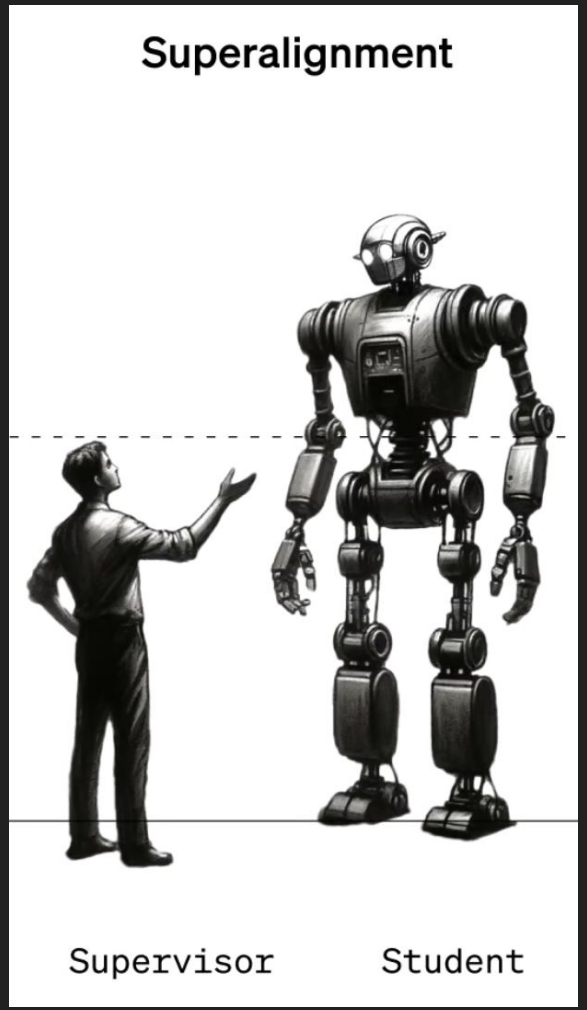
Nikita Kozlov

Traditional ML



Supervisor Student

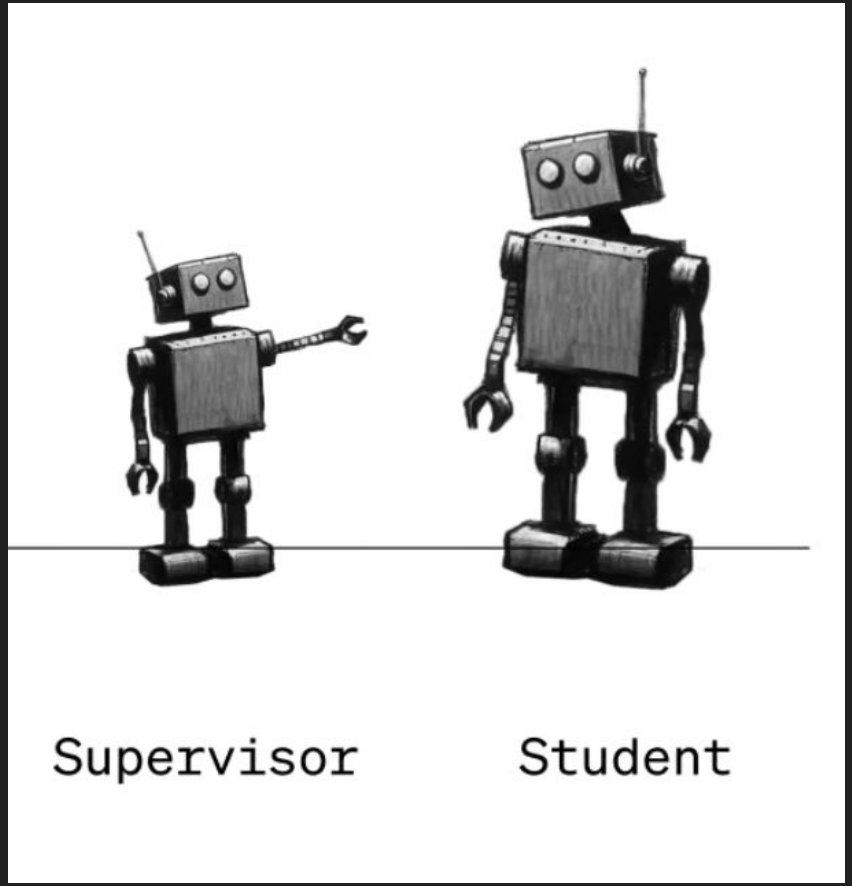
Can we supervise a superhuman AI model using only a human supervisor?



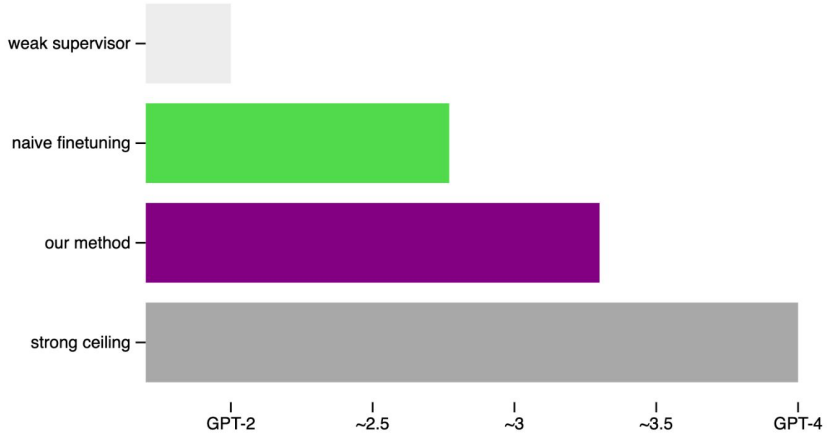
We don’t know

Question: Can small models (GPT-2) supervise bigger models (GPT-4)?

Weak-to-Strong Generalization: Eliciting Strong Capabilities With Weak Supervision



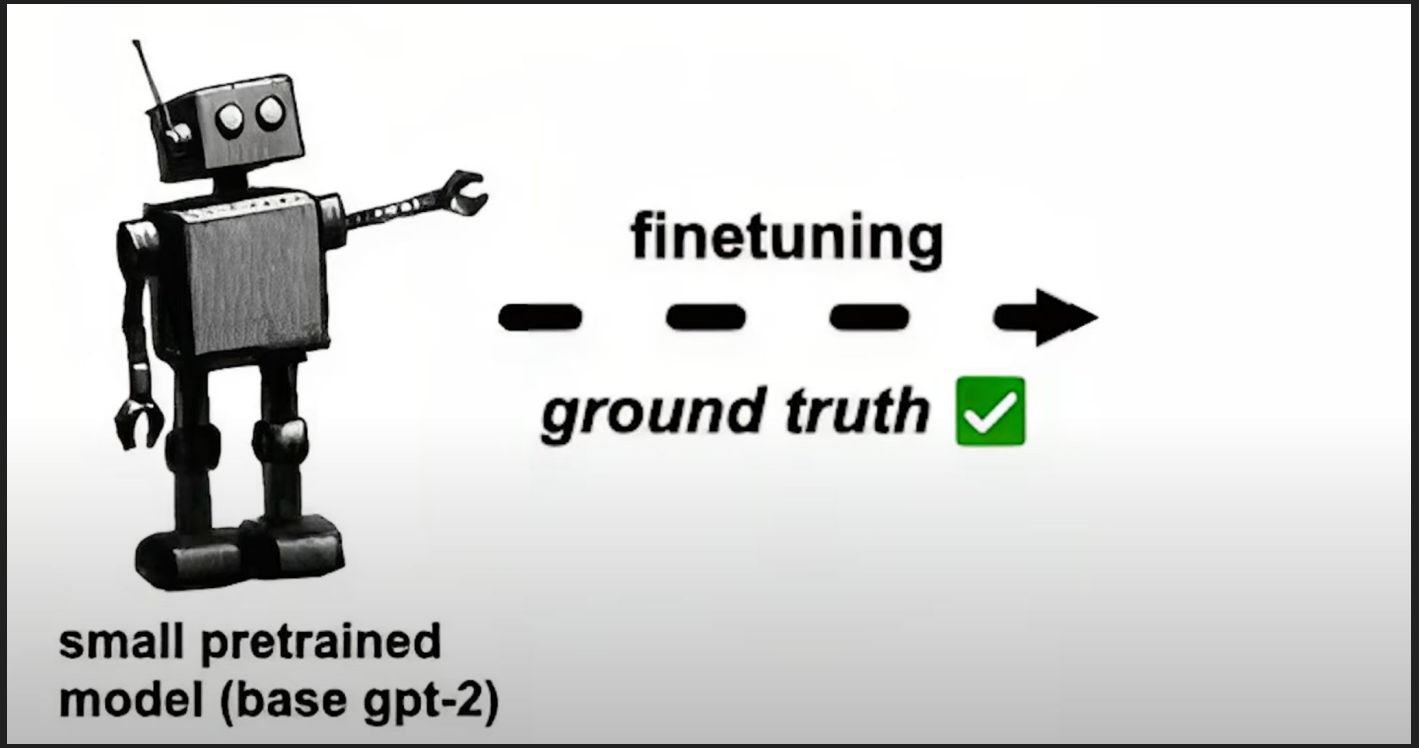
It turns out to be Yes



Typical weak-to-strong generalization across NLP benchmarks: We use a GPT-2-level model as a weak supervisor to finetune GPT-4.

Performance on NLP Benchmark

How do we get Weak Supervisor

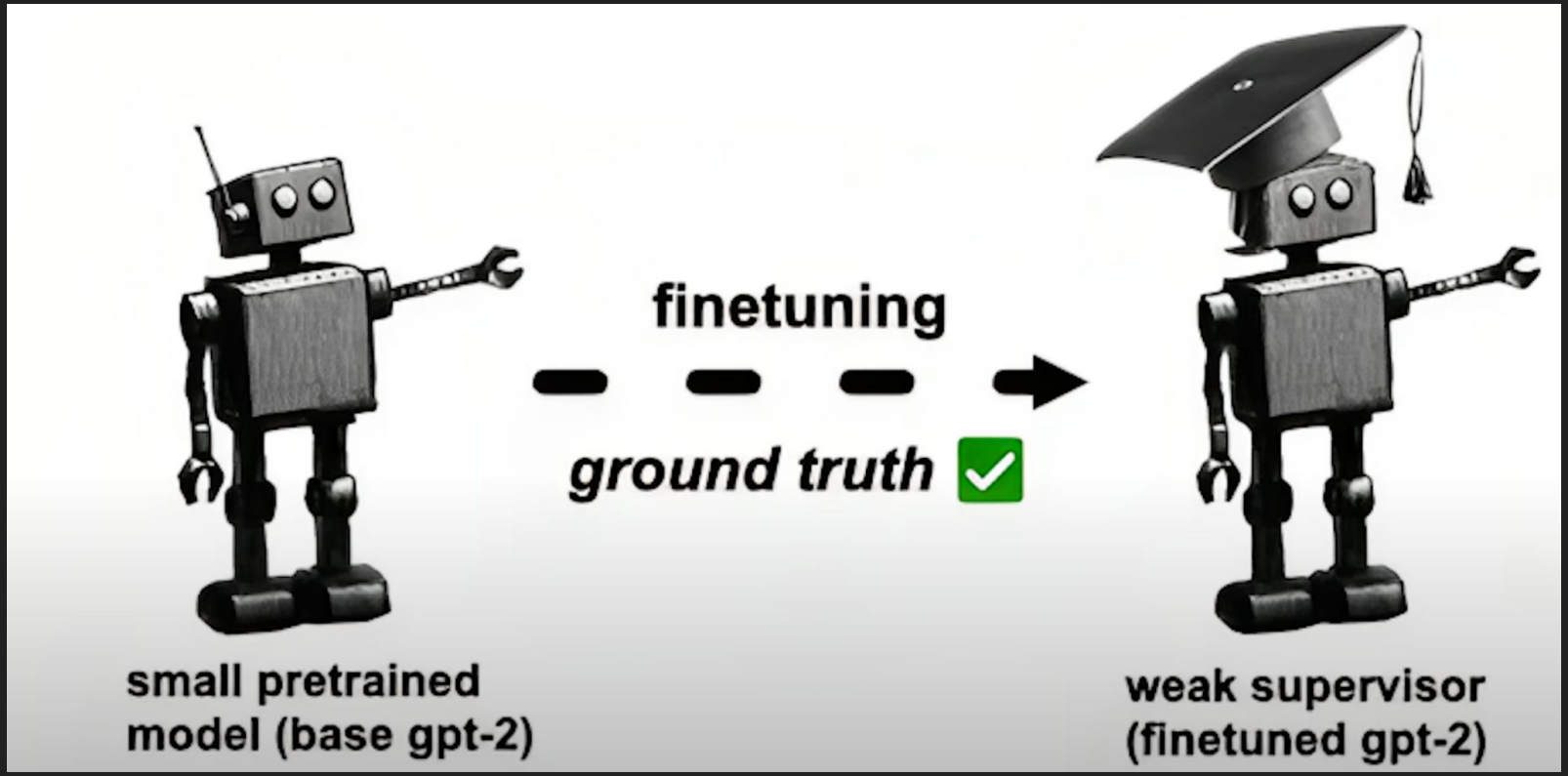


How do we get Weak Supervisor

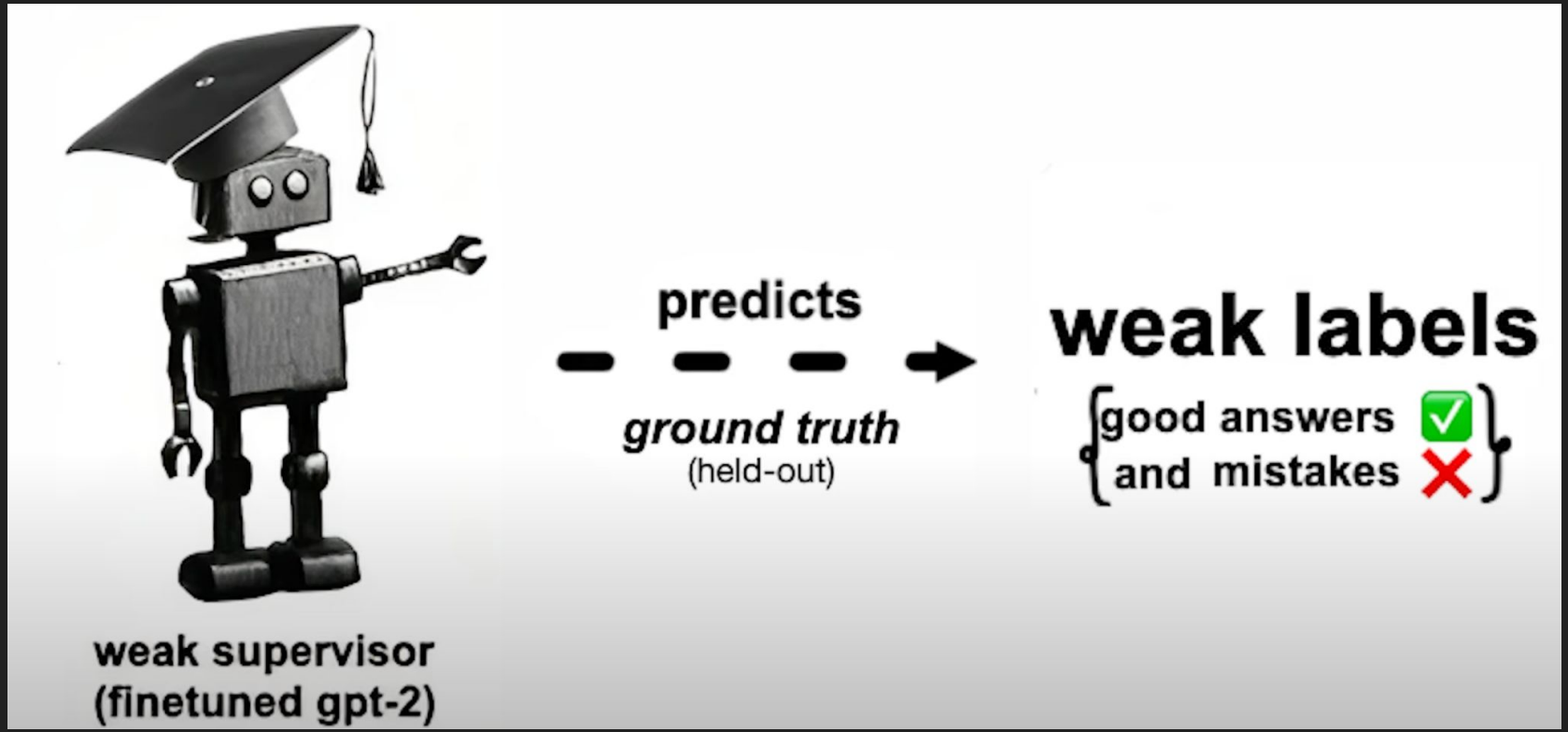
Table 1: **Datasets and their sources.** We summarize the NLP datasets we use and their original sources.

Dataset	Original Source
BoolQ	Clark et al. (2019)
CosmosQA	Huang et al. (2019)
DREAM	Sun et al. (2019)
ETHICS [Justice]	Hendrycks et al. (2020a)
ETHICS [Deontology]	Hendrycks et al. (2020a)
ETHICS [Virtue]	Hendrycks et al. (2020a)
ETHICS [Utilitarianism]	Hendrycks et al. (2020a)
FLAN ANLI R2	Nie et al. (2019); Wei et al. (2021)
GLUE CoLA	Warstadt et al. (2019); Wang et al. (2018)
GLUE SST-2	Socher et al. (2013); Wang et al. (2018)
HellaSwag	Zellers et al. (2019)
MCTACO	Zhou et al. (2019)
OpenBookQA	Mihaylov et al. (2018)
PAWS	Zhang et al. (2019)
QuAIL	Rogers et al. (2020)
PIQA	Bisk et al. (2020)
QuaRTz	Tafjord et al. (2019)
SciQ	Welbl et al. (2017)
Social IQa	Sap et al. (2019)
SuperGLUE MultiRC	Khashabi et al. (2018); Wang et al. (2019)
SuperGLUE WIC	Pilehvar & Camacho-Collados (2018); Wang et al. (2019)
Twitter Sentiment	Zhang et al. (2019)

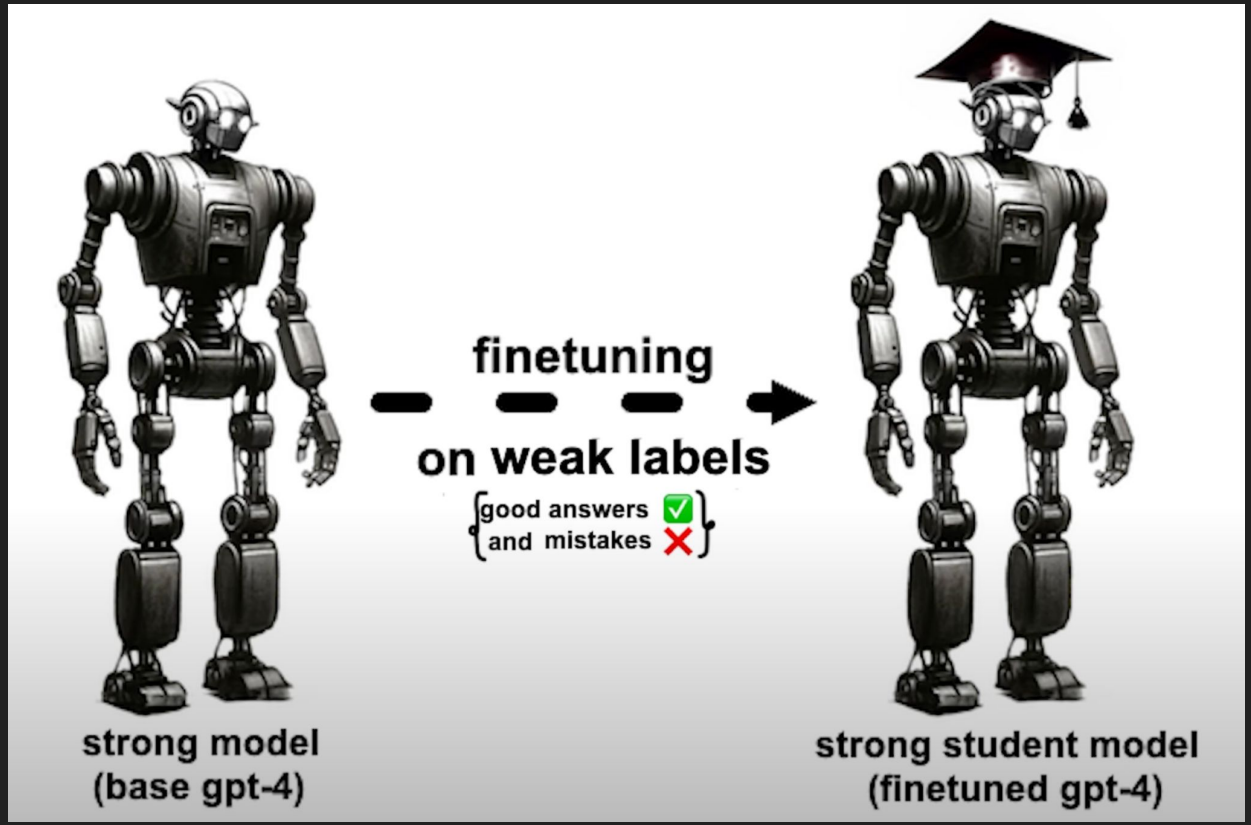
How do we get Weak Supervisor



How do we get Naive Finetuning



How do we get Naive Finetuning



https://github.com/openai/weak-to-strong/blob/main/train_weak_to_strong.py

```python
# Train the weak model on the first half of the training data
print(f"Training weak model, size {weak_model_size}")
weak_test_results, weak_ds = train_model(
    weak_model_config,
    train1_ds,
    test_ds,
    loss_type="xent",
    label="weak",
    subpath=os.path.join("weak_model_gt", weak_model_size.replace("/", "_")),
    lr=weak_lr,
    eval_batch_size=weak_eval_batch_size,
    inference_ds=train2_ds,
    epochs=gt_epochs,
    linear_probe=linear_probe,
    optimizer_name=weak_optim,
)
```

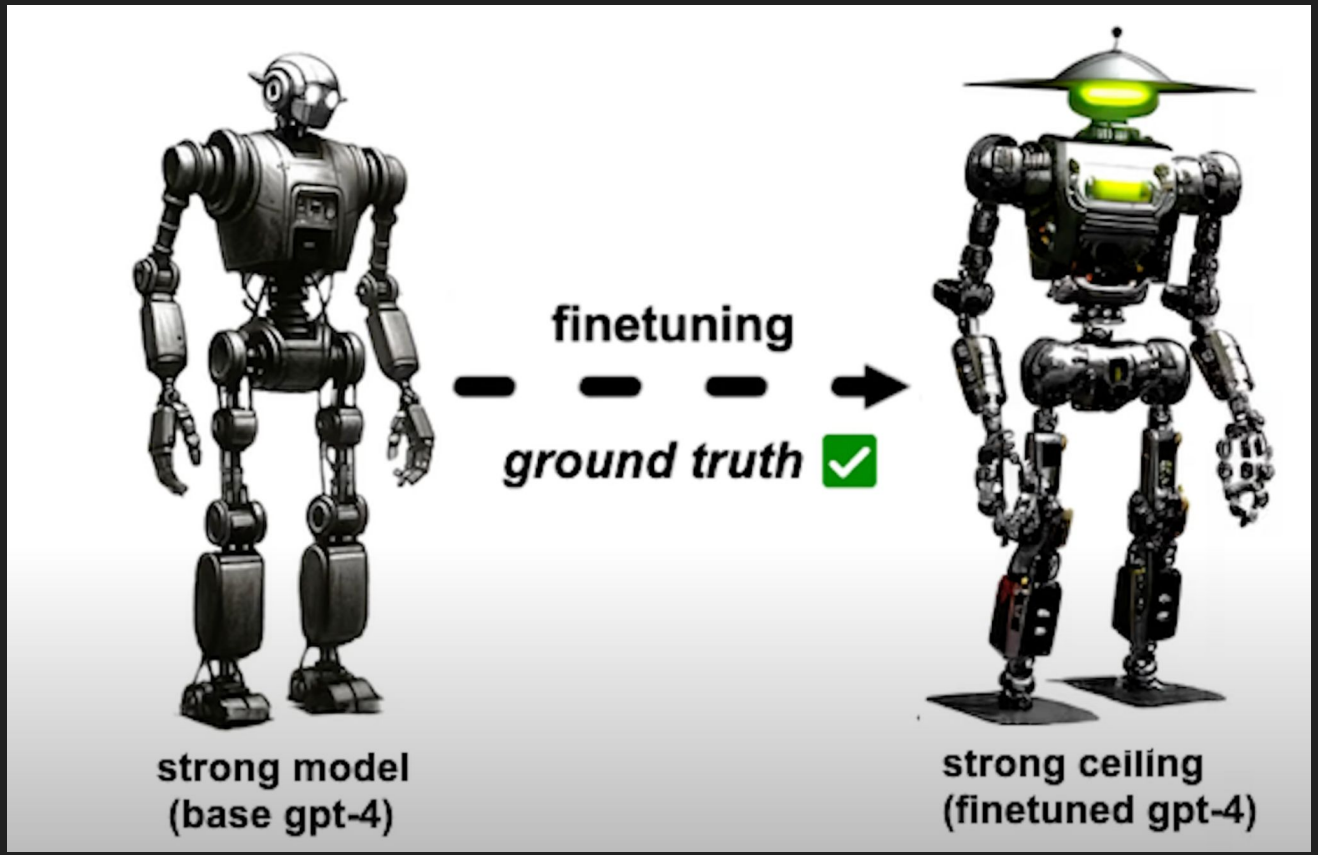
```python
# Train the strong model on the second half of the training data with labels generated by the weak model
all_transfer_test_results = {}
for tloss in transfer_losses:
    print(
        f"Training transfer model, size {strong_model_size} on labels from {weak_model_size}, with loss {
    )
    transfer_test_results, _ = train_model(
        strong_model_config,
        weak_ds,
        test_ds,
        loss_type=tloss,
        label="weak2strong",
        subpath=os.path.join(
            "strong_model_transfer",
            f"{weak_model_size.replace('/', '_')}_{strong_model_size.replace('/', '_')}_{tloss}",
        ),
        lr=transfer_lr,
        eval_batch_size=strong_eval_batch_size,
        epochs=transfer_epochs,
        linear_probe=linear_probe,
        optimizer_name=transfer_optim,
    )
```

Strong Student

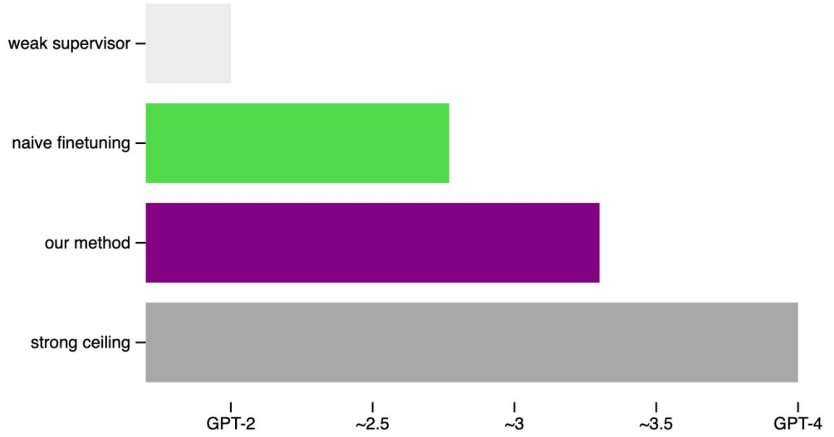
```python
# Train the strong model on the second half of the training data
print(f"Training strong model, size {strong_model_size}")
strong_test_results, _ = train_model(
    strong_model_config,
    train2_ds,
    test_ds,
    loss_type="xent",
    label="strong",
    subpath=os.path.join("strong_model_gt", strong_model_size.replace("/", "_")),
    lr=strong_lr,
    eval_batch_size=strong_eval_batch_size,
    epochs=gt_epochs,
    linear_probe=linear_probe,
    optimizer_name=strong_optim,
)
```

Baseline (strong ceiling)

How do we get Strong Ceiling



But what's the our method?



Typical weak-to-strong generalization across NLP benchmarks: We use a GPT-2-level model as a weak supervisor to finetune GPT-4.

Our method

We can significantly improve generalization in many settings. We use a simple method that encourages the strong model to be more confident—including confidently disagreeing with the weak supervisor if necessary. **When we supervise GPT-4 with a GPT-2-level model using this method on NLP tasks, the resulting model typically performs somewhere between GPT-3 and GPT-3.5.** We are able to recover much of GPT-4's capabilities with only much weaker supervision.

Loss

A.4 Auxiliary Confidence Loss

Here, we provide a detailed description of the method we use in Section 4.3.2.

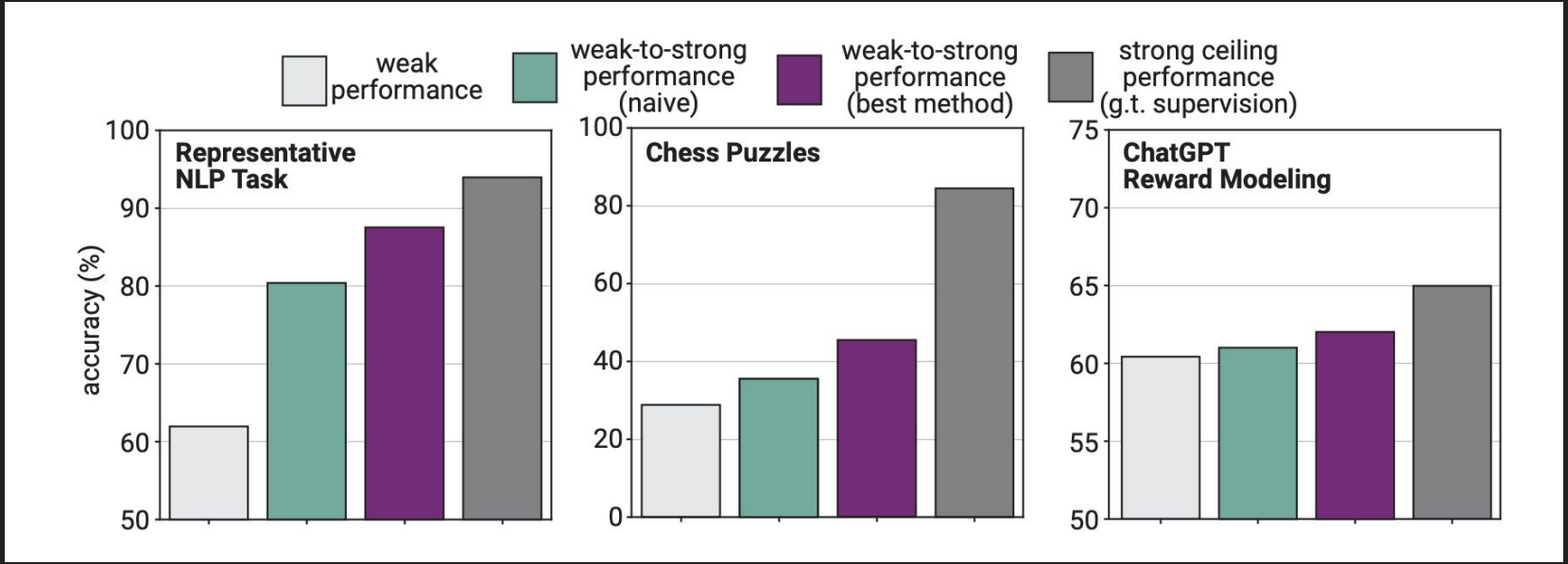
We use the following loss function:

$$L_{\text{conf}}(f) = (1-\alpha) \cdot \text{CE}(f(x), f_w(x)) + \alpha \cdot \text{CE}(f(x), \hat{f}_t(x)) \tag{1}$$

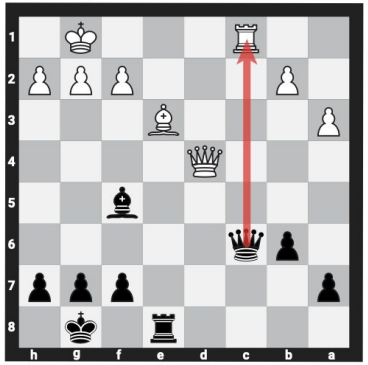
where $\text{CE}(\cdot, \cdot)$ is the cross-entropy loss between the predictive distributions on a given input x, $f_w(x) \in [0, 1]$ represents the weak label predictive distribution, $f(x) \in [0, 1]$ is the strong model predictive distribution, α is a weight and t is a threshold. The predictions $\hat{f}_t(x)$ correspond to hardened strong model predictions using a threshold t, i.e. $\hat{f}_t(x) = I[f(x) > t] \in \{0, 1\}$ where I is the indicator function. We set the threshold t adaptively, so that $f(x) > t$ holds for exactly half of examples in the batch[7]. We set $\alpha_{\max} = 0.75$ for the largest student models and to 0.5 otherwise and linearly warm-up α from 0 to $\alpha_{\max}$ over the first 20% of training.

Does this approach works on other than NLP datasets?

It depends


weak performance
weak-to-strong performance (naive)
weak-to-strong performance (best method)
strong ceiling performance (g.t. supervision)
Representative NLP Task
Chess Puzzles
ChatGPT Reward Modeling
accuracy (%)
100
90
80
70
60
50
100
80
60
40
20
0
75
70
65
60
55
50

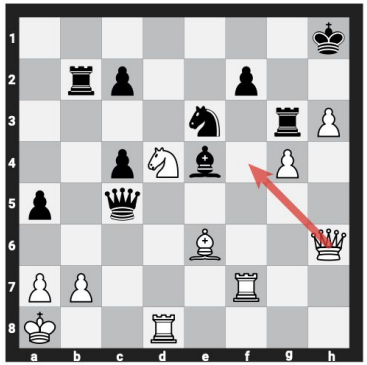
Chess Puzzles



Prompt: "1. d4 1... Nf6 2. Nf3 2... d5 3. e3 3... e6 4. Bd3 4... c5 5. c3 5... Be7 6. Nbd2 6... O-O 7. O-O 7... Nc6 8. Re1 8... Bd7 9. e4 9... dxe4 10. Nxe4 10... cxd4 11. Nxf6+ 11... Bxf6 12. cxd4 12... Nb4 13. Be4 13... Qb6 14. a3 14... Nc6 15. d5 15... exd5 16. Bxd5 16... Bf5 17. Bxc6 17... Qxc6 18. Nd4 18... Bxd4 19. Qxd4 19... Rfe8 20. Rxe8+ 20... Rxe8 21. Be3 21... b6 22. Rc1 22..."

Label: " Qxc1+"

(a) Elo-695 puzzle

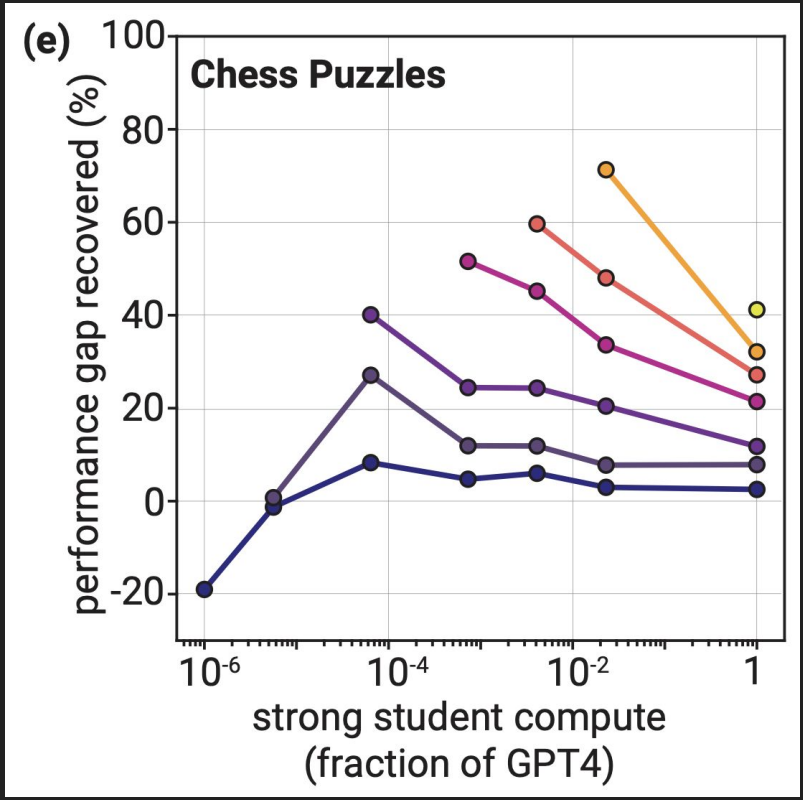


Prompt: "1. e4 1... e5 2. Nc3 2... Nf6 3. Nf3 3... Nc6 4. Bb5 4... Bc5 5. Bxc6 5... dxc6 6. d3 6... Bg4 7. h3 7... Bxf3 8. Qxf3 8... O-O 9. g4 9... Bb4 10. Bd2 10... Nd7 11. h4 11... Be7 12. g5 12... Nc5 13. O-O-O 13... Qd7 14. h5 14... Qd8 15. Qg3 15... Ne6 16. Rdg1 16... b5 17. Qxe5 17... a5 18. f4 18... Re8 19. Qf5 19... b4 20. Na4 20... Nd4 21. Qg4 21... c5 22. f5 22... Ra6 23. f6 23... Bd6 24. fxg7 24... Kxg7 25. Rg2 25... Qc8 26. h6+ 26... Kg8 27. Qh5 27... Qd7 28. Rf1 28... Re6 29. Rgf2 29... Rg6 30. c3 30... bxc3 31. Nxc3 31... a4 32. Nd5 32... Qb5 33. Nf6+ 33... Kh8 34. Qh3 34... Rb6 35. Be3 35... Ne6 36. Nxh7 36... Qxd3 37. Rd1 37... Qc4+ 38. Kb1 38... Qxe4+ 39. Ka1 39... Be5 40. Nf6 40... Qc4 41. Nd5 41... Rb7 42."

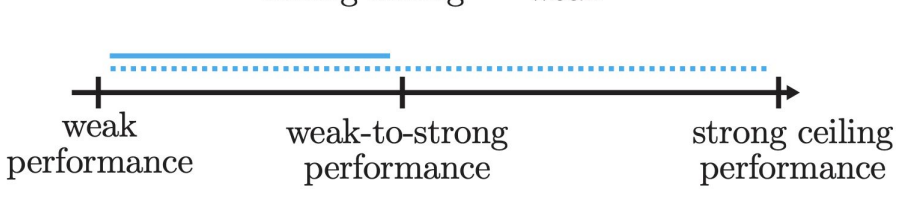
Label: " Qf5"

(b) Elo-2253 puzzle

Naive finetuning doesn't work well enough



$$\text{PGR} = \frac{\text{weak-to-strong} - \text{weak}}{\text{strong ceiling} - \text{weak}} = \frac{\text{——}}{\text{·····}}$$

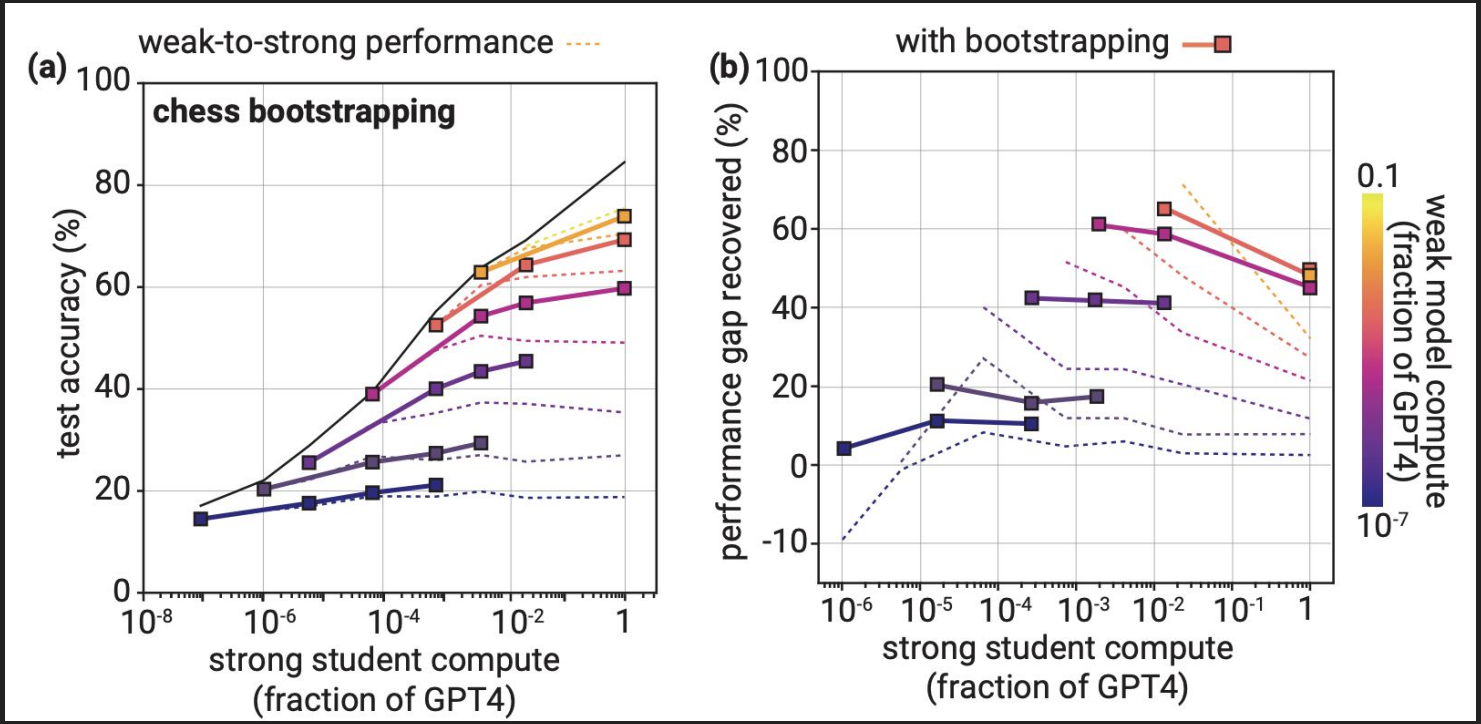


Solution: bootstrapping

Bootstrapping is a long-standing idea in alignment: instead of directly aligning very superhuman models, we could first align an only slightly superhuman model, use that to align an even smarter model, and so on (Christiano, 2019; 2018; Leike & Sutskever, 2023; Worley, 2021). Our setting allows us to empirically test this idea.

Specifically, we can construct a sequence of model sizes $\mathcal{M}_1 \to \mathcal{M}_2 \to \ldots \to \mathcal{M}_n$ of increasing sizes. Then, we use the weak labels from $\mathcal{M}_1$ to finetune $\mathcal{M}_2$, use $\mathcal{M}_2$ to generate new weak labels that we can use to finetune the next model in the sequence, $\mathcal{M}_3$, and so on.

Performance

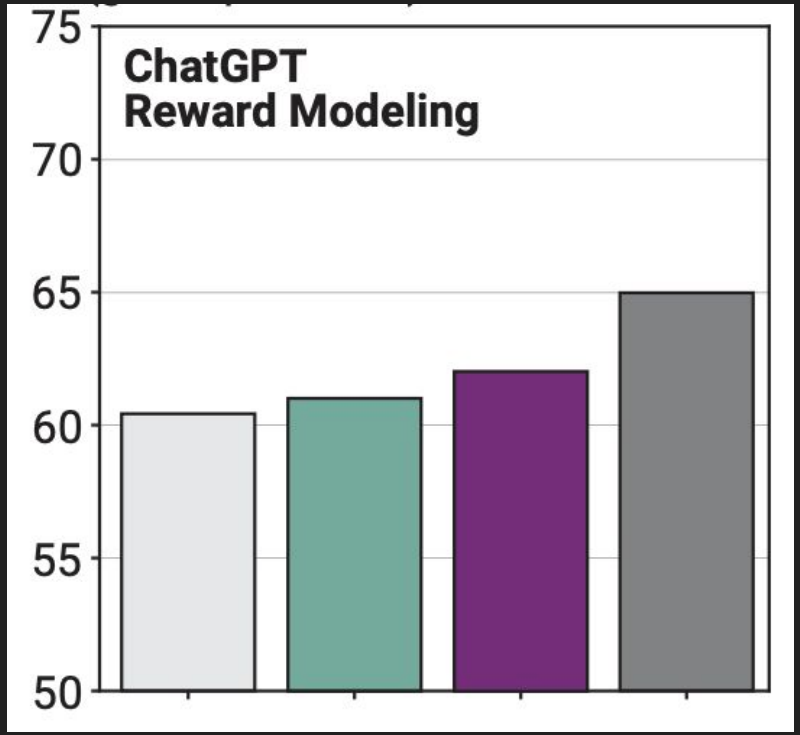


ChatGPT reward modeling

ChatGPT reward modeling. The standard approach to aligning models today is reinforcement learning from human feedback (RLHF). A critical step of RLHF is to train a reward model (RM) to predict human preferences between model responses. Specifically, a reward model is trained on a dataset consisting of dialogs between a human and an assistant model. For each query, the humans compare multiple possible responses (completions) from the assistant, providing human preference data. Then, a reward model is trained to predict the results of pairwise comparisons between completions. Finally, the assistant model is trained by optimizing against the reward model with reinforcement learning (RL). In our work, we do not study the RL step, and instead assume the goal is to maximize reward model accuracy. For more details on reward models, see e.g. Ouyang et al. (2022). We use a proprietary dataset used to train ChatGPT reward models.

Gap between naive finetuning and ceiling is high.

Use unsupervised generative finetuning for reward modeling.



What is generative finetuning?

5.2.2 GENERATIVE SUPERVISION IMPROVES RM WEAK-TO-STRONG GENERALIZATION

If salient representations of the desired task is useful for weak-to-strong generalization, then we may be able to improve generalization by increasing the salience of the task to the strong model. One way to increase the salience of a task without needing ground truth labels is to perform unsupervised finetuning with the language modeling objective on data relevant to that task (Dai & Le, 2015). For example, by finetuning a language model in an unsupervised way on online reviews, sentiment becomes saliently represented to models internally (Radford et al., 2017).

We test this idea in our reward modeling setting, where it is standard practice to initialize the model with a baseline finetuned on demonstrations of desired behaviors (Stiennon et al., 2020). In our case, we re-use the ChatGPT comparison data instead of introducing a new supervision dataset. Comparisons are comprised of a prefix (a single request or conversation between the user and assistant) and at least two candidate completions. We finetune the base models with a language modeling loss on all prefix-completion pairs, ignoring the human preferences between those completions.

If we’re reusing ChatGPT comparison data instead of new dataset we’re cheating - we use the results of previous comparisons

We found that the additional generative finetuning on the RM data leads to better weak-to-strong performance. Because this procedure also improves the performance of models trained on ground truth RM data, we compare our new weak-to-strong performance to strong "ceiling" models that were also first generatively finetuned in the same way. Even with this adjusted ceiling, we find that generative supervision improves PGR by approximately 10-20%. We report the results in Figure 10.

Meaning GPT-3.5 was firstly finetuned on ChatGPT comparison data without human preferences, and then finetuned on normal dataset (with ground truth labels) with human preferences.

3 methods but …

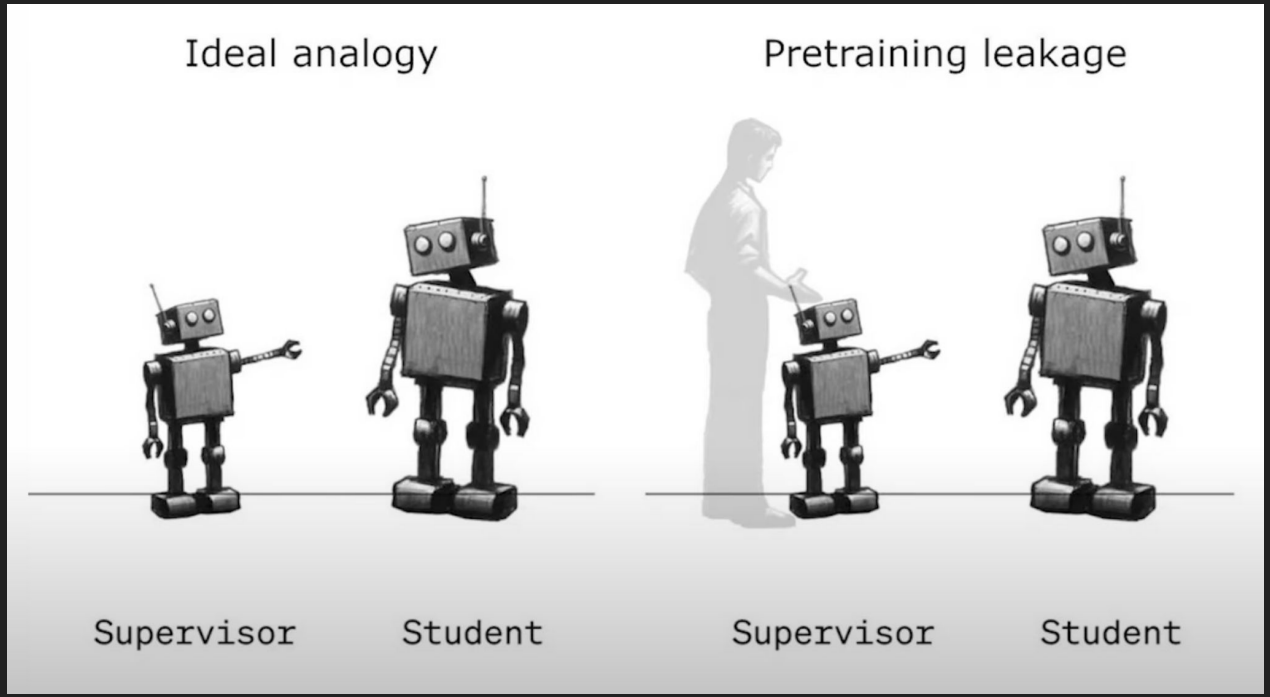
method is the auxiliary confidence loss for the NLP task (Section 4.3.2), bootstrapping for Chess puzzles (Section 4.3.1), and unsupervised generative finetuning for reward modeling (Section 5.2.2; generative-finetuning is also used for the strong ceiling performance).

Problems

Limitations

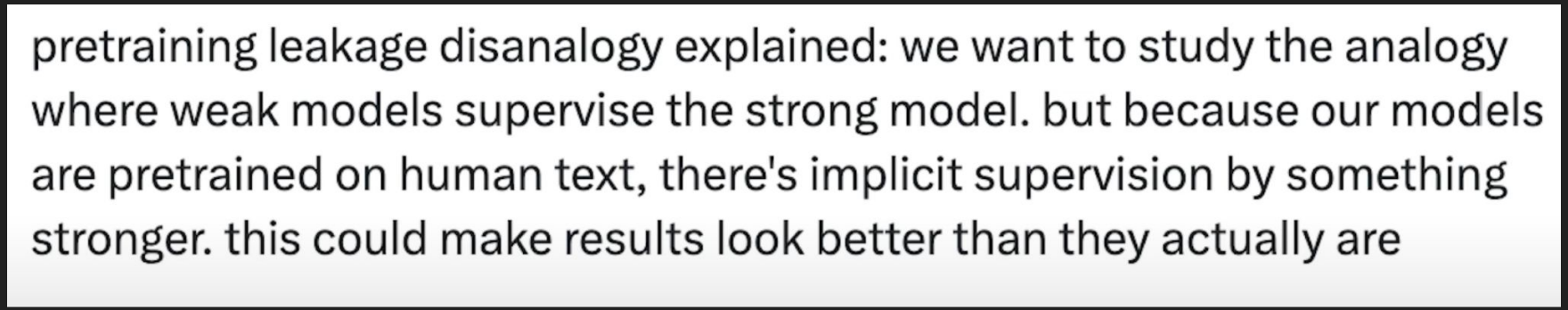
Imitation saliency: superhuman models may easily imitate weak errors. Future models will likely be very good at predicting what humans will think and say, especially if they are trained on human data in a similar manner to current models. Consequently, if we naively train such a

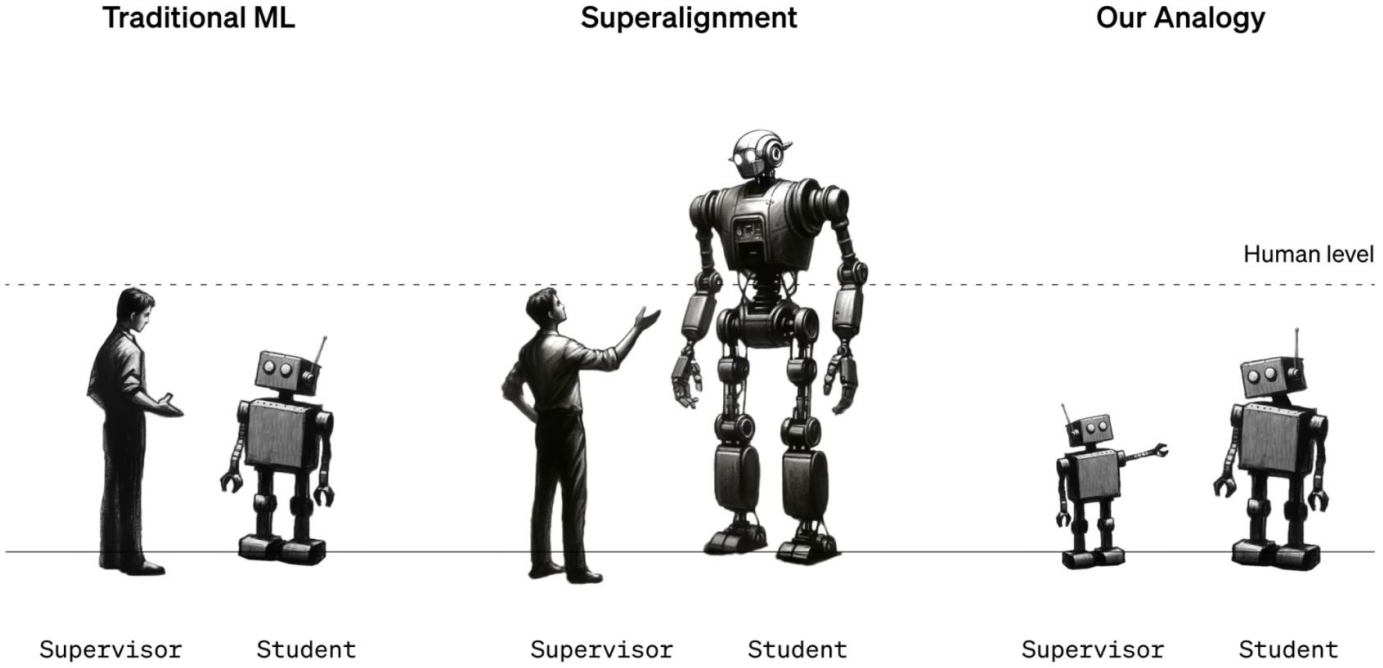
These results suggest that pretrained models may have a hard time fitting errors of other (smaller) pretrained models, at least in finetuning settings with relatively limited data. Stanton et al. (2021) and Furlanello et al. (2018) report a related observation in the context of knowledge distillation: it is surprisingly hard for models to fit the predictions of other models, even when they have sufficient capacity to do so.



Pretraining leakage: superhuman knowledge may be latent, not observable. Many of the tasks we consider in this work may have been observed in pretraining at least indirectly, for example through questions on online forums or through slight reframings of the task. For example, it is highly likely that simple science questions similar to those in the SciQ NLP task are present in our GPT-4 series pretraining dataset at least implicitly in some form. However future superhuman models may never directly observe superhuman alignment-relevant capabilities; these capabilities may be predominantly "latent", e.g. learned through self-supervised learning or reinforcement learning rather than through imitation learning. Intuitively, latent capabilities may be harder to elicit than capabilities that models could have observed in their pretraining data.

TL;DR - Our analogy is not pretty fair.




Traditional ML
Superalignment
Our Analogy
Human level
Supervisor
Student
Supervisor
Student
Supervisor
Student