OpenAI

Contrastive finetuning of generative language models for informal premise selection

Jesse Michael Han, Tao Xu, Stanislas Polu, Arvind Neelakantan, and Alec Radford

Premise selection / relevance filtering

- Premise selection:
  - Classic problem in automated theorem proving
  - Can we select the most relevant lemmas for proving a given theorem?
  - Usually attacked with neural methods in the *formal* setting

Premise selection / relevance filtering

- Informal premise selection:
    - Given a *natural language* theorem statement and a pool of *natural language* definitions/lemmas
    - Can we select the most relevant references for proving that theorem?

- Pro: more in-domain for existing NLP techniques
  Con: no algorithmic feedback from proof search

ProofWiki retrieval task

- **ProofWiki.** We download the public ProofWiki XML dump,[4] which contains a snapshot of all pages on ProofWiki. We filter pages according to manually designed rules (e.g. redirects, files, categories), and determine page type, title, contents, and references using each page's WikiMedia data structure.

Reference retrieval and generation. Each theorem $\mathbf{x}$ has a proof containing a sequence of references $\mathbf{y} = (\mathbf{r}_1, \ldots, \mathbf{r}_{|\mathbf{y}|})$, where each reference $\mathbf{r}_m \in \mathcal{R}$ is either a theorem, definition, or other statement (see §3). We consider two tasks: *retrieval* and *generation*.

Theorem

The number of primes is infinite.

Proof

Define a topology on the integers $\mathbb{Z}$ by declaring a subset $U \subseteq \mathbb{Z}$ to be an open set if and only if it is either:

the empty set $\varnothing$

or:

a union of sequences $S(a, b)$, where:

$$S(a, b) = \{an + b : n \in \mathbb{Z}\} = a\mathbb{Z} + b$$

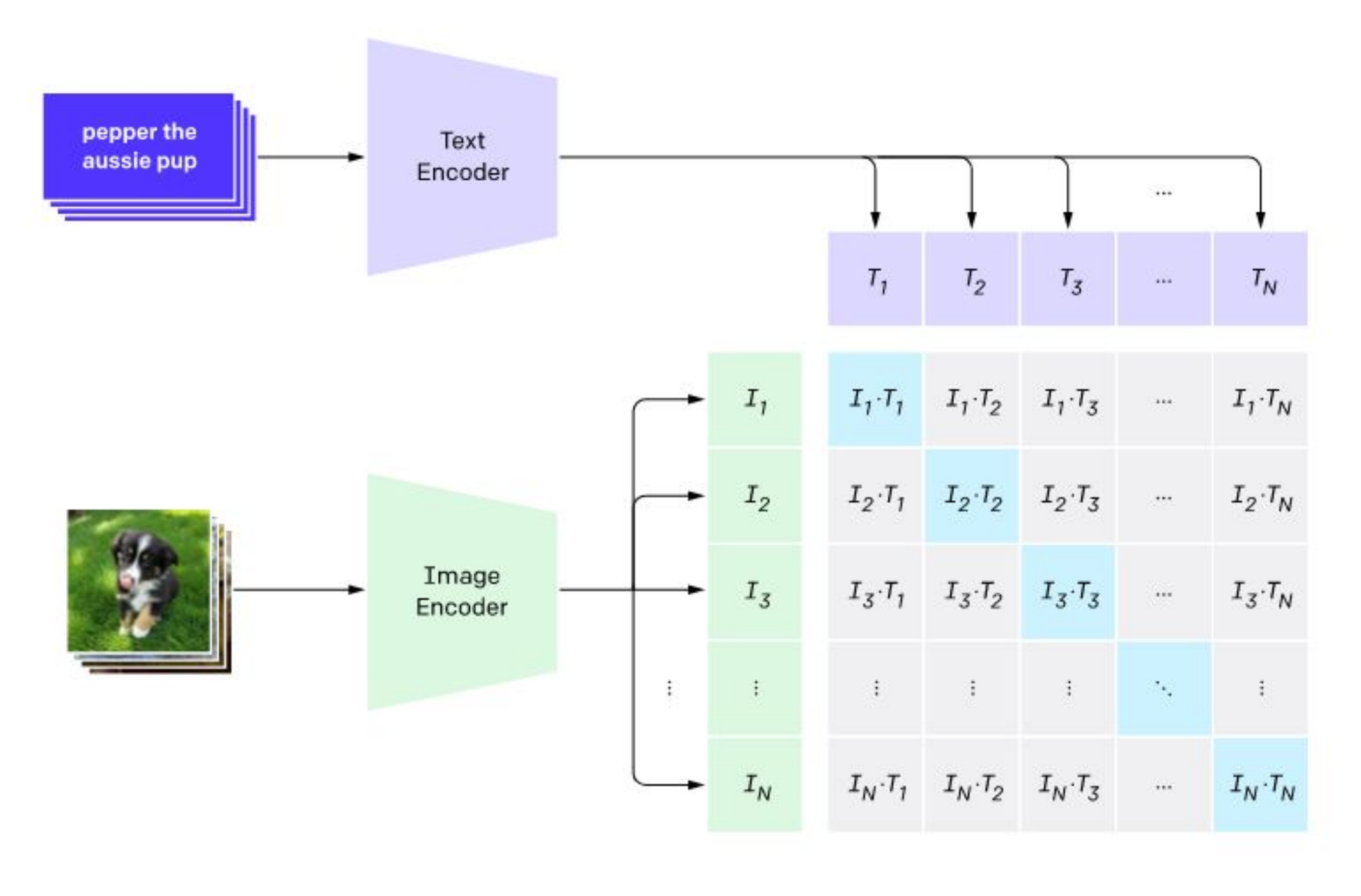
In other words, U is open if and only if every $x \in U$ admits some non-zero integer a such that $S(a, x) \subseteq U$.

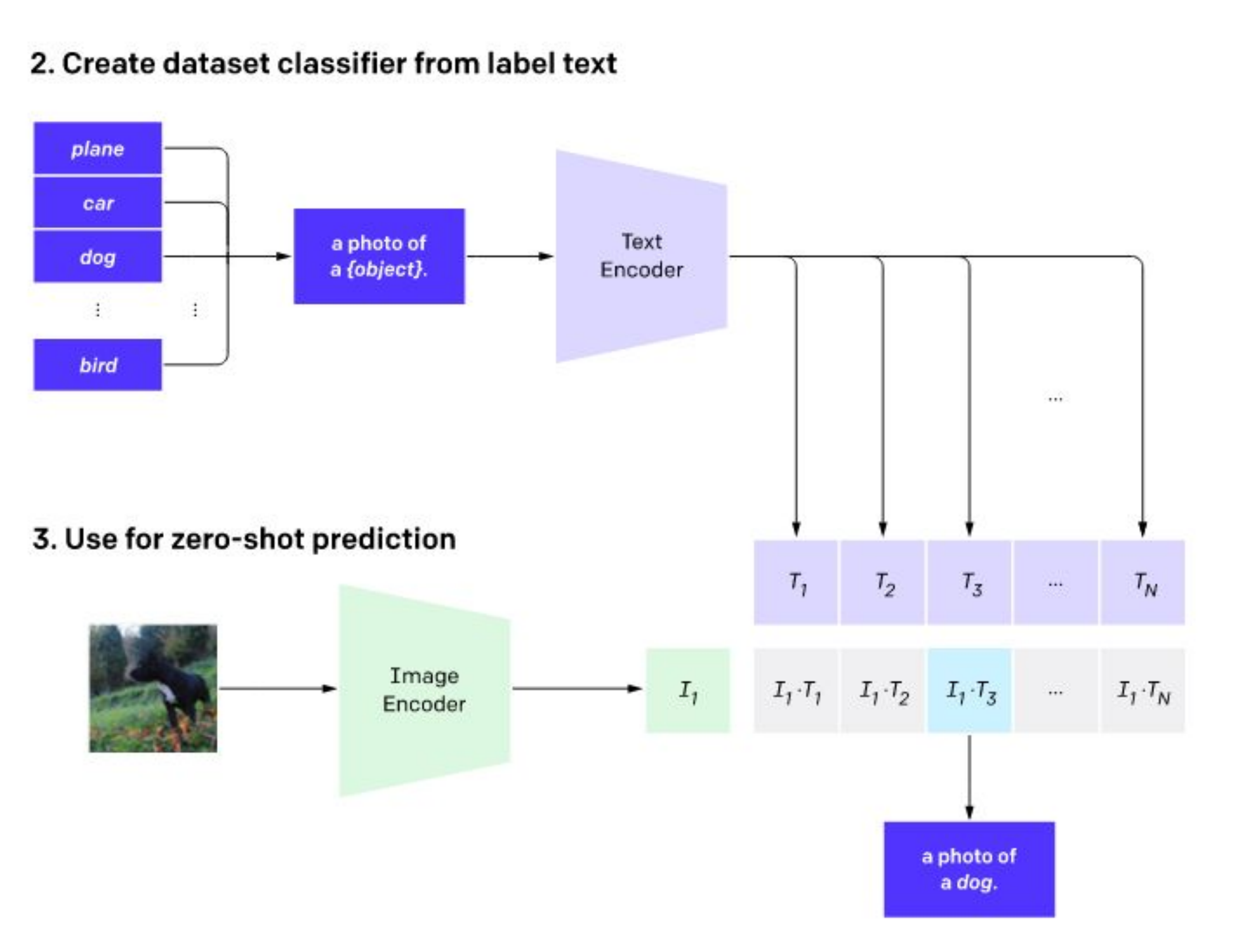
Contrastive finetuning of autoregressive decoder-only transformers

- Use the same technique as CLIP: contrastive loss using features from a decoder-only transformer

Learning Transferable Visual Models From Natural Language Supervision

Alec Radford [*1] Jong Wook Kim [*1] Chris Hallacy [1] Aditya Ramesh [1] Gabriel Goh [1] Sandhini Agarwal [1]
Girish Sastry [1] Amanda Askell [1] Pamela Mishkin [1] Jack Clark [1] Gretchen Krueger [1] Ilya Sutskever [1]


pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 ... T_N
I_1 I_2 I_3 ⋮ I_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
$I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ ... $I_2 \cdot T_N$
$I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ ... $I_3 \cdot T_N$
⋮ ⋮ ⋮ ⋱ ⋮
$I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ ... $I_N \cdot T_N$


2. Create dataset classifier from label text
plane
car
dog
bird
a photo of a {object}.
Text Encoder
3. Use for zero-shot prediction
Image Encoder
I_1
T_1 T_2 T_3 ... T_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
a photo of a dog.

FOOD101

guacamole (90.1%) Ranked 1 out of 101 labels

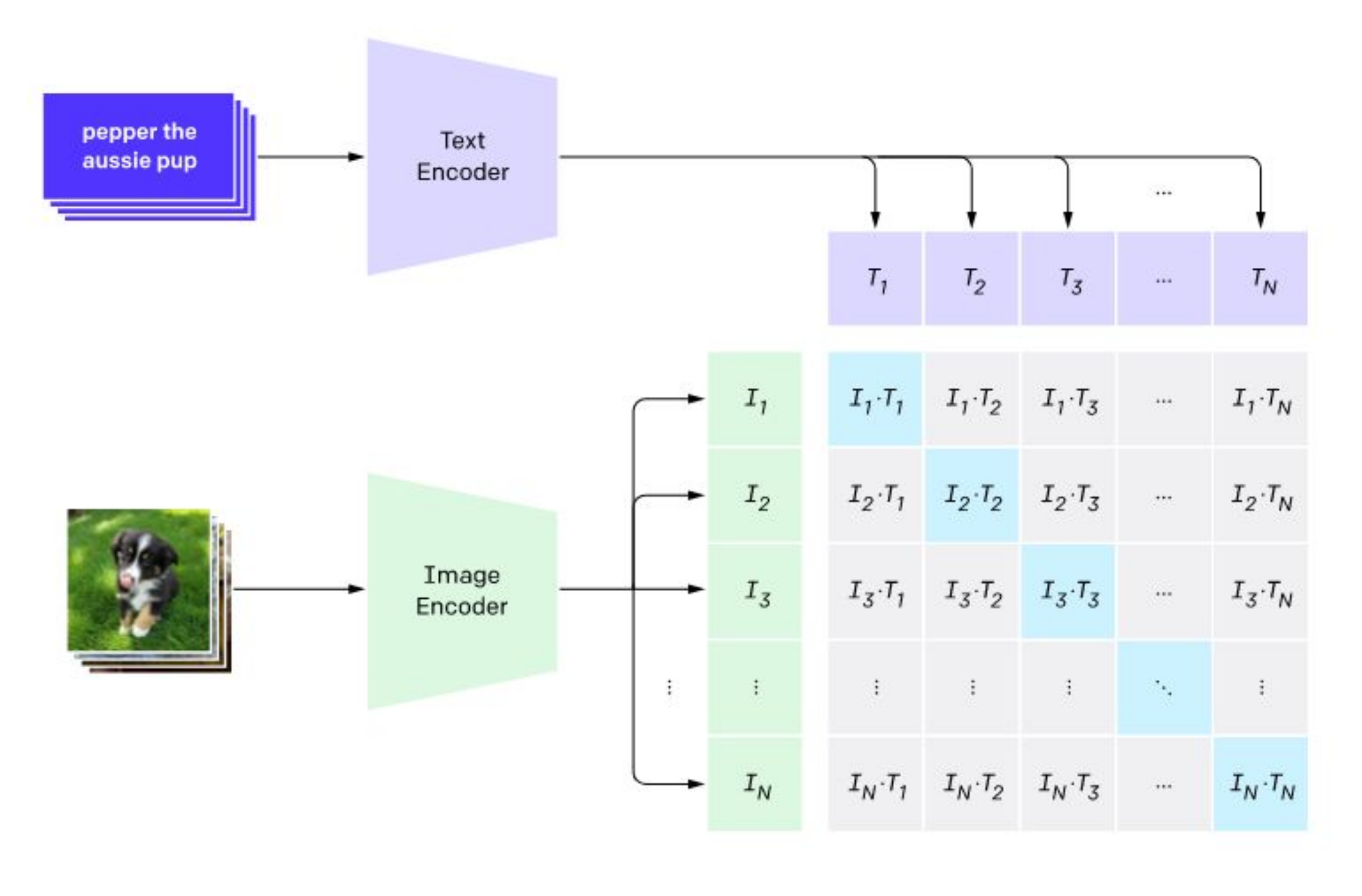
✓ a photo of **guacamole**, a type of food.

× a photo of **ceviche**, a type of food.

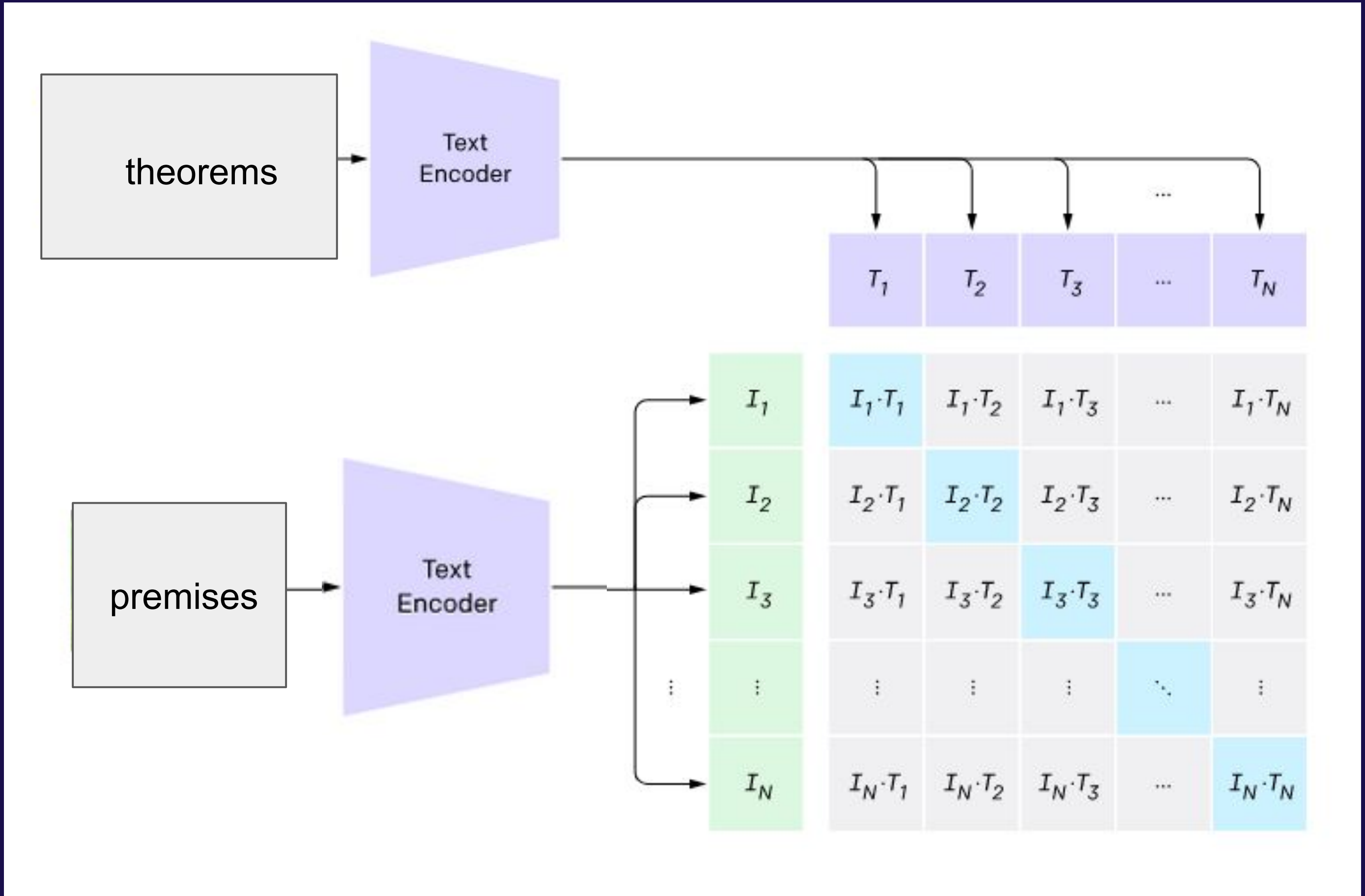
× a photo of **edamame**, a type of food.

× a photo of **tuna tartare**, a type of food.

× a photo of **hummus**, a type of food.



pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 ... T_N
I_1 I_2 I_3 ... I_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
$I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ ... $I_2 \cdot T_N$
$I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ ... $I_3 \cdot T_N$
$I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ ... $I_N \cdot T_N$



theorems
Text Encoder
premises
Text Encoder
T_1
T_2
T_3
...
T_N
I_1
I_2
I_3
I_N
$I_1 \cdot T_1$
$I_1 \cdot T_2$
$I_1 \cdot T_3$
$I_1 \cdot T_N$
$I_2 \cdot T_1$
$I_2 \cdot T_2$
$I_2 \cdot T_3$
$I_2 \cdot T_N$
$I_3 \cdot T_1$
$I_3 \cdot T_2$
$I_3 \cdot T_3$
$I_3 \cdot T_N$
$I_N \cdot T_1$
$I_N \cdot T_2$
$I_N \cdot T_3$
$I_N \cdot T_N$

```
# image_encoder - ResNet or Vision Transformer
# text_encoder  - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = np.linalg.norm(np.dot(I_f, W_i), axis=1)
T_e = np.linalg.norm(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

Figure 3. Numpy-like pseudocode for the core of an implementation of CLIP.

Generative pre-training is useful

Improving Language Understanding by Generative Pre-Training

Alec Radford
OpenAI
alec@openai.com

Karthik Narasimhan
OpenAI
karthikn@openai.com

Tim Salimans
OpenAI
tim@openai.com

Ilya Sutskever
OpenAI
ilyasu@openai.com

Generative pre-training is useful

Improving Language Understanding by Generative Pre-Training

Al

alec

Language Models are Unsupervised Multitask Learners

Alec Radford [*1] **Jeffrey Wu** [*1] **Rewon Child** [1] **David Luan** [1] **Dario Amodei** [**1] **Ilya Sutskever** [**1]

Generative pre-training is useful

Im

Alec R
Oper
alec@ope

ers

Sutskever ** 1

Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*

Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry

Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan

Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter

Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray

Benjamin Chess Jack Clark Christopher Berner

Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

OpenAI

Domain-specific generative pre-training is also useful

Table 1: Mix and source of data involved in the *WebMath* dataset.

Dataset	Size	Mix
Github	23 GB	33%
arXiv Math	10 GB	33%
Math StackExchange	2 GB	33%

Table 7: Performance for various model sizes and pre-training datasets.

Model	Performance	Perplexity	# Tokens
160m *from scratch*	28.96%	1.041	18B
160m *CommonCrawl*	32.34%	1.030	16B
160m *Github*	33.61%	1.030	16B
160m *WebMath*	34.79%	1.029	16B
700m *from scratch*	31.58%	1.040	18B
700m *CommonCrawl*	39.61%	1.026	15B
700m *Github*	41.55%	1.025	15B
700m *WebMath*	**42.56%**	**1.024**	15B

Domain-specific generative pre-training is also useful

Co-training (PACT)						
`WebMath > mix1 + tactic`	32B	18B	*0.08*		0.94	40.0%
`WebMath > mix1 + mix2 + tactic`	96B	71B	*0.09*	*0.09*	**0.91**	**48.4%**
Pre-training and co-training						
`WebMath > mix2 > mix1 + tactic`	32B	18B	*0.08*		0.93	46.9%

Figure 5. Comparison of pre-training and co-training on `mix-1` and `mix-2`. > denotes a pre-training step and + denotes a co-training. As an example, `WebMath > mix2 > mix1 + tactic` signifies a model successively pre-trained on `WebMath` then `mix2` and finally co-trained as a fine-tuning step on `mix1` and `tactic`. Columns `mix1`, `mix2`, `tactic` report the min validation loss achieved on these respective datasets.

Co-training					
`mix1 + tactic`	32B	27B	*0.11*		1.12
`mix1 + mix2 + tactic`	96B	71B	*0.10*	*0.11*	**1.07**
Pre-training and co-training					
`mix2 > mix1 + tactic`	32B	26B	*0.11*		1.09

Figure 6. Validation losses achieved in the pre-training and co-training setups without `WebMath` pre-training. See Figure 5 for a description of the columns and the models nomenclature used.

Domain-specific generative pre-training is also useful

Evaluating Large Language Models Trained on Code

Mark Chen [*1] **Jerry Tworek** [*1] **Heewoo Jun** [*1] **Qiming Yuan** [*1] **Henrique Ponde de Oliveira Pinto** [*1]

We introduce Codex, a GPT language model fine-tuned on publicly available code from GitHub, and study its Python code-writing capabilities. A distinct production version of Codex powers GitHub Copilot. On HumanEval, a new evaluation set we release to measure functional correctness for synthesizing programs from docstrings, our model solves 28.8% of the problems, while GPT-3 solves 0% and GPT-J solves 11.4%. Fur-

Strategy

- Generatively pre-train a language model
- Take activations for the end-of-text (EOT) token as embedding for theorems and references
- Finetune using the contrastive InfoNCE loss described above.

GPT-3 models

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

(Brown et al 2020)

GPT-3 models

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

(Brown et al 2020)

Use a single model to embed both theorems and references

Unlike CLIP [8] or the BERT-based model studied in NaturalProofs [12], we use the same encoder to embed both queries (theorems) and documents (premises). Since "X is useful to prove Y" is an asymmetric relation and we use a CLIP-style symmetric cross-entropy loss, the encoder must be allowed to distinguish between theorems and references. We do this by simply formatting the inputs to the transformer as

```
Theorem title: <title> <newline> Theorem statement: <statement>

Reference title: <title> <newline> Reference statement: <statement>.
```

Training details

- Use batch size of N=2048
- Sample N theorems from train set, then sample a reference from each of the theorems to create the batch
    - This way we don't contrast references from the same theorem
- Train for ~7000 steps using Adam, 0.2X the pre-training learning rate, using 32 V100 GPUs

How does generative pre-training affect retrieval performance?

- **No pretraining.** The model is randomly initialized and only learns theorem/premise representations through contrastive training.
- **GPT-3 style pretraining.** The model is pretrained for 300B tokens on the same data (a mix of filtered CommonCrawl, WebText, books, and Wikipedia) as GPT-3 [2].
- **WebMath pretraining.** Starting from the final snapshot of the previous model, we train for another 72B tokens on the WebMath dataset [7], comprising a mix of math arXiv, Python, Math StackExchange, Math Overflow, and PlanetMath.

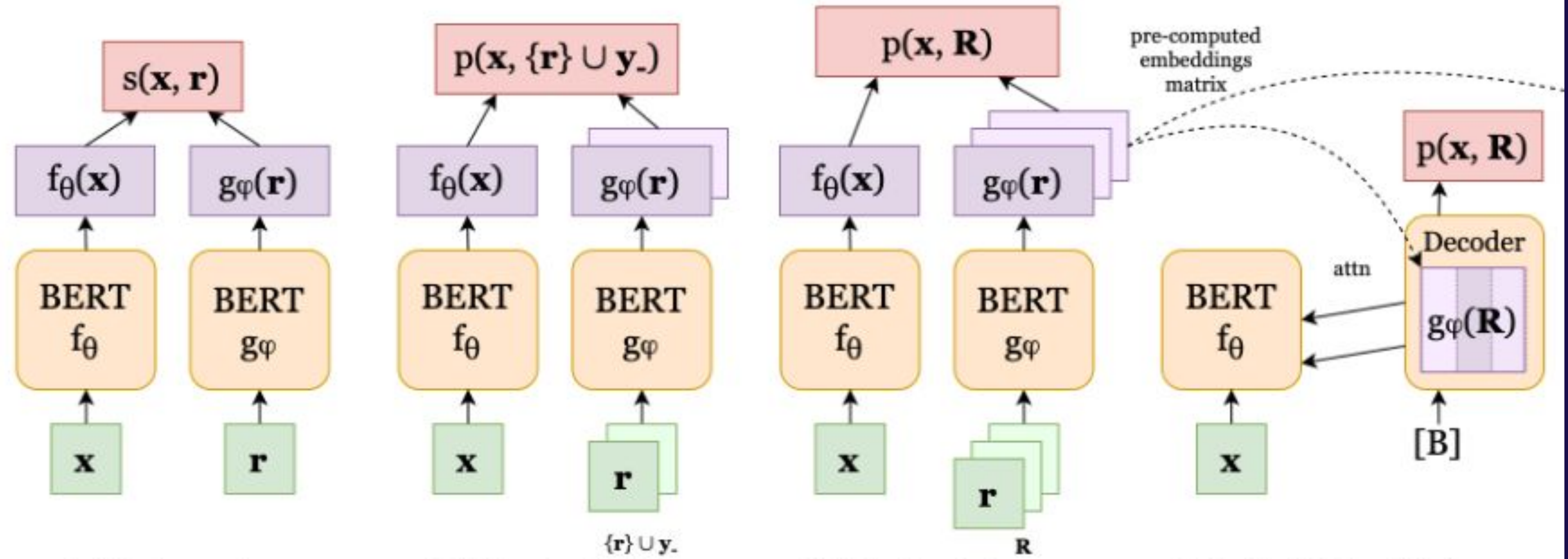
We refer to our methodology for informal premise selection as contrastive theorem-premise training (CTPT) and denote the three models above by `ctpt-no-pretrain`, `ctpt-webtext`, and `ctpt-webmath`.

	recall@10	recall@100	avgp@100	full@100	full@1K
BERT	**20.27**	**59.44**	**14.01**	**27.39**	**70.52**
ctpt-no-pretrain	23.76	54.01	11.91	23.75	56.32
ctpt-webtext	34.39	65.45	17.97	34.76	64.51
ctpt-webmath	**36.92**	**70.39**	**21.53**	**39.49**	**73.52**

Our main results are displayed in Table 1. The model `ctpt-webmath` outperforms the previous state-of-the-art on all metrics. Our models also utilize 43% fewer parameters since the BERT-based model embeds theorems and references with separate copies of `bert-base-cased` (110M params). It is possible that the `webtext` data contains ProofWiki, but WebMath does not and we consider the significant performance gap between `ctpt-webtext` and `ctpt-webmath` to be of primary interest. We speculate that the models studied in [12] are severely undertrained due to using only 200 randomly sampled negatives for each positive example.

		ProofWiki				
		mAP	**R@10**	**R@100**	**Full@10**	**Full@100**
Random		0.04	0.00	0.19	0.00	0.00
Frequency		3.38	5.90	24.30	0.44	2.29
TF-IDF		6.19	10.27	23.09	4.14	9.43
BERT (P+S)	**+pair**	13.54	20.10	58.75	6.17	31.28
	+joint	32.71	37.59	73.72	17.71	48.90
BERT (P/S)	**+pair**	16.82	23.73	63.75	7.31	38.50
	+joint	**36.75**	**42.45**	**75.90**	**20.35**	**50.22**

	recall@10	recall@100	avgp@100	full@100	full@1K
BERT	**20.27**	**59.44**	**14.01**	**27.39**	**70.52**
ctpt-no-pretrain	23.76	54.01	11.91	23.75	56.32
ctpt-webtext	34.39	65.45	17.97	34.76	64.51
ctpt-webmath	<u>**36.92**</u>	<u>**70.39**</u>	<u>**21.53**</u>	<u>**39.49**</u>	<u>**73.52**</u>



(a) Basic scoring (b) Pairwise: Training (c) Pairwise: Inference (d) Joint: Training & Inference

Future directions

- Retrieval-augmented language modeling of proofs
  - Can we improve informal (theorem, proof) perplexity when additionally conditioned on retrieved informal premises?
  - Can we improve formal (theorem, proof) perplexity when additionally conditioned on retrieved informal premises?
  - Can we improve formal theorem-proving pass-rate when conditioned on informal premises (either per-theorem or per-proofstep?)

- Re-ranking to address high-recall/low-precision behavior
  - Zero/few-shot re-ranking using full-size GPT-3
  - Zero/few-shot re-ranking using Webmath-finetuned GPT-3

- Scale?
  - Model size?
  - Batch size --- against current wisdom, doesn't seem to help too much

OpenAI

Q & A