

# Pre-training + Fine-tuning Paradigm

Ling 575j: Deep Learning for NLP

C.M. Downey

Spring 2023

# Announcements

- HW5 ref code available
- HW6: PyTorch directly, no more edugrad (but same API)
- A note on runtime and scalability of LSTMs
  - Definitely fast on small data, but doesn't scale
  - From `hw6.model.LSTM.forward`:
  - hw7 will use torch native LSTMs, a bit faster

```
# [batch_size, hidden_dim]
hidden, memory = self.init_hidden_and_memory(batch_size)
hidden_states = []
for timestep in embeddings:
    # apply the recurrence
    # [batch_size, hidden_dim]
    hidden, memory = self.cell(hidden, memory, timestep)
    # store the hidden state
    hidden_states.append(hidden)
# [seq_len, batch_size, hidden_dim]
return torch.stack(hidden_states)
```

# Some Python/PyTorch Notes

- Training loop in HW6: saves best model based on dev loss, evaluates that
- `model.eval()`, `model.train()`
  - Sets the “mode” of a model in PyTorch
  - `.eval()` turns off things like Dropout, which are only used in training; `.train()` turns them back on
- `with torch.no_grad():``
  - For *inference/test* mode: doesn't allocate memory for gradients on Tensors, saves lots of memory
- `RNNClassifier.forward`: fancy-ish tensor operations to extract the “last” hidden state from a batch of variable-length sequences

# Note on Transformer Architecture

---

## Do Transformer Modifications Transfer Across Implementations and Applications?

---

Sharan Narang\* Hyung Won Chung Yi Tay William Fedus  
Thibault Fevry† Michael Matena † Karishma Malkan† Noah Fiedel  
Noam Shazeer Zhenzhong Lan† Yanqi Zhou Wei Li  
Nan Ding Jake Marcus Adam Roberts Colin Raffel

Google Research

### Abstract

The research community has proposed copious modifications to the Transformer architecture since it was introduced over three years ago, relatively few of which have seen widespread adoption. In this paper, we comprehensively evaluate many of these modifications in a shared experimental setting that covers most of the common uses of the Transformer in natural language processing. Surprisingly, we find that most modifications do not meaningfully improve performance. Furthermore, most of the Transformer

will yield equal-or-better performance on any task that the pipeline is applicable to. For example, residual connections in convolutional networks (He et al., 2016) are designed to ideally improve performance on any task where these models are applicable (image classification, semantic segmentation, etc.). In practice, when proposing a new improvement, it is impossible to test it on every applicable downstream task, so researchers must select a few representative tasks to evaluate it on. However, the proposals that are ultimately adopted by the research community and practitioners tend to be those that reliably improve performance across a wide variety of tasks “in



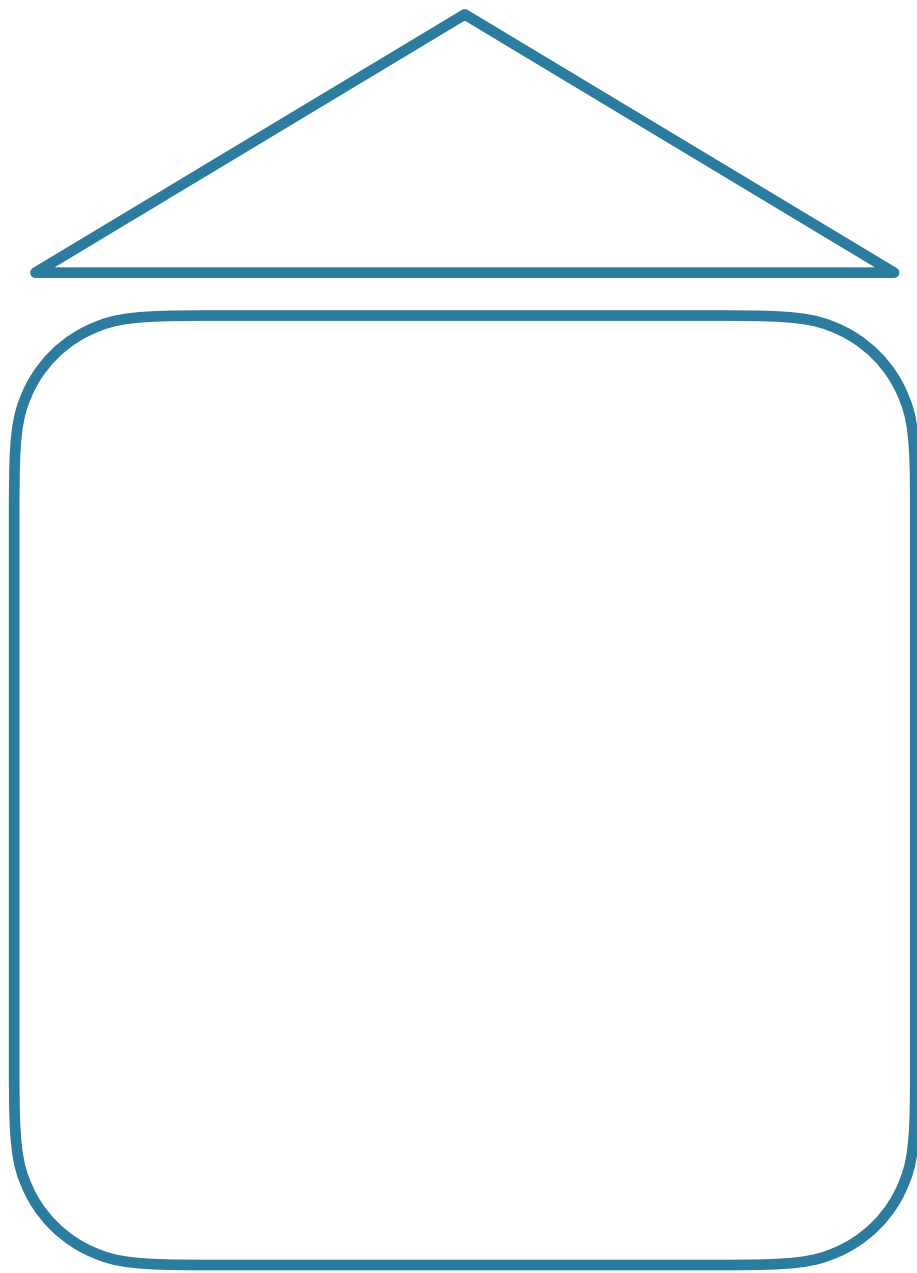
# Today's Plan

- Transfer learning in general
- Language model pre-training: initial steps
- Transformer-based pre-training
  - Encoder only
  - Decoder only
  - Encoder-Decoder
- Some limitations

# Transfer Learning

# Standard Learning

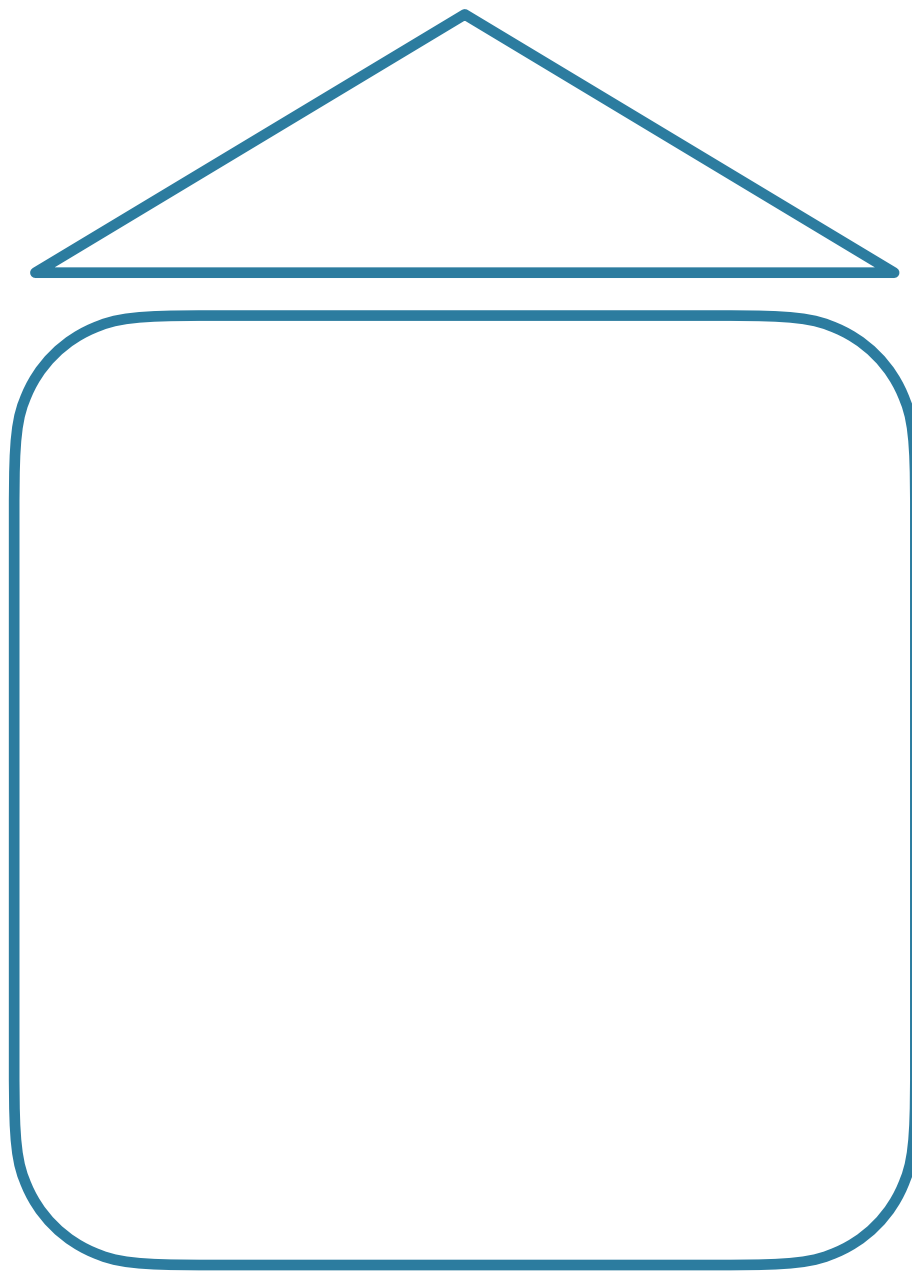
Task 1 outputs



Task 1 inputs

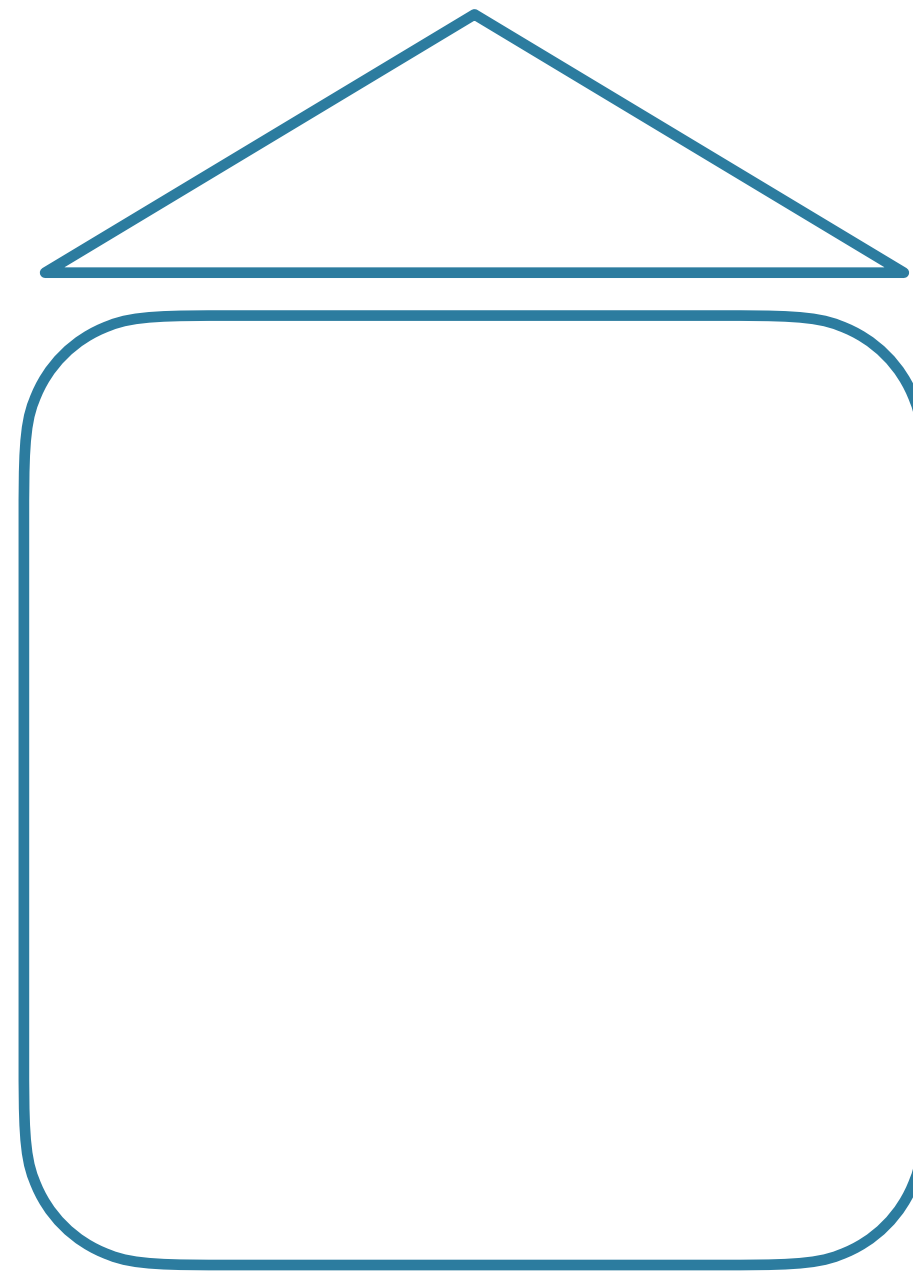
# Standard Learning

Task 1 outputs



Task 1 inputs

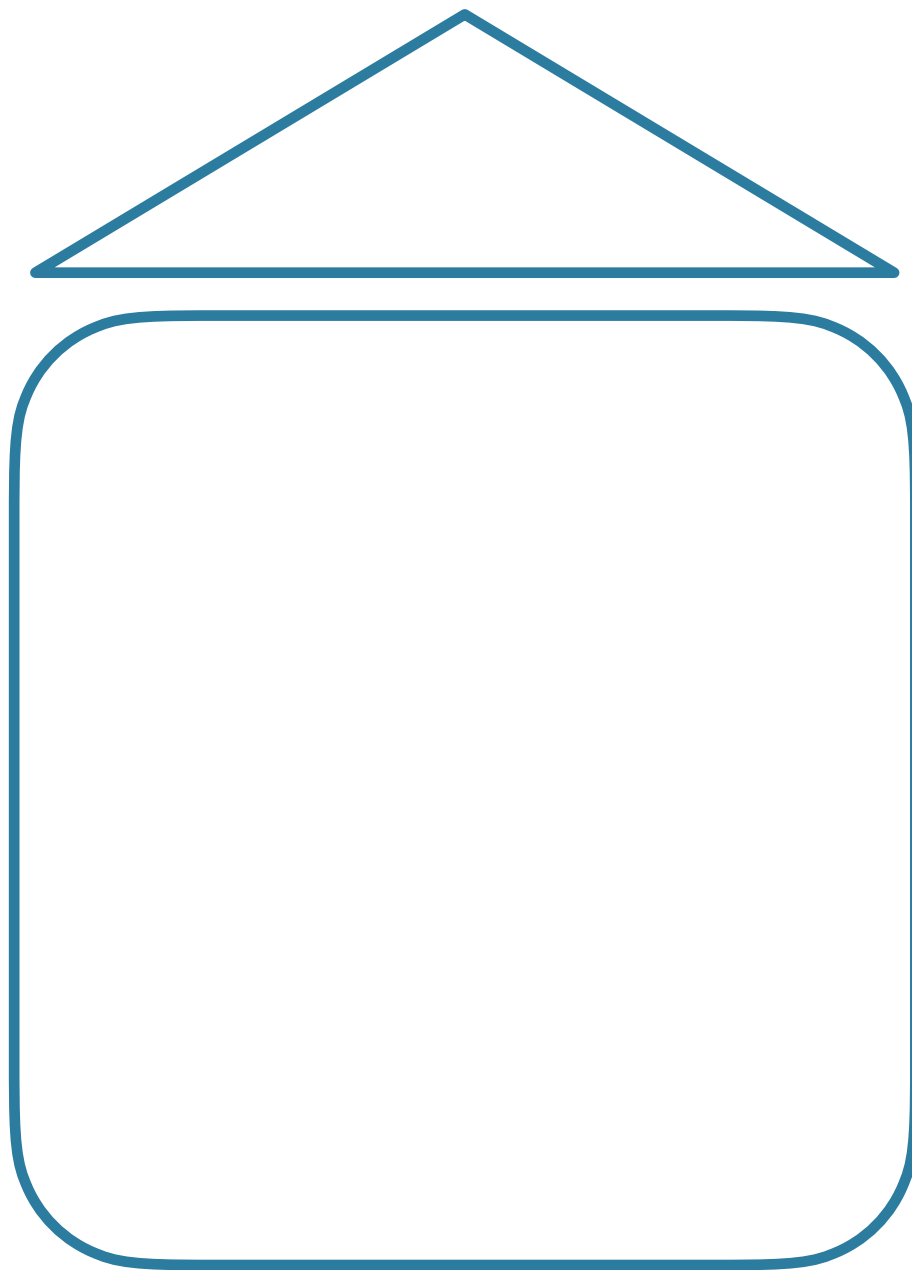
Task 2 outputs



Task 2 inputs

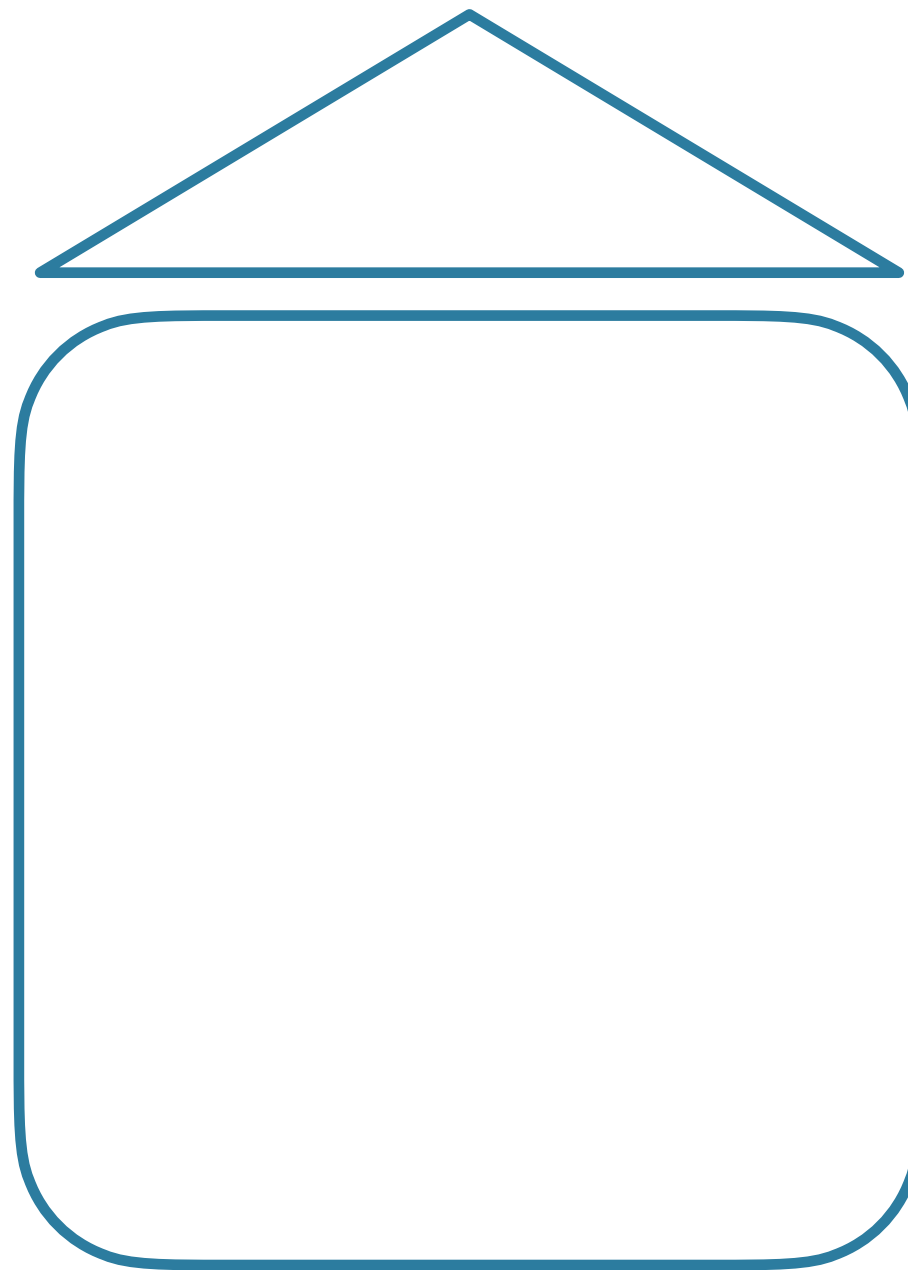
# Standard Learning

Task 1 outputs



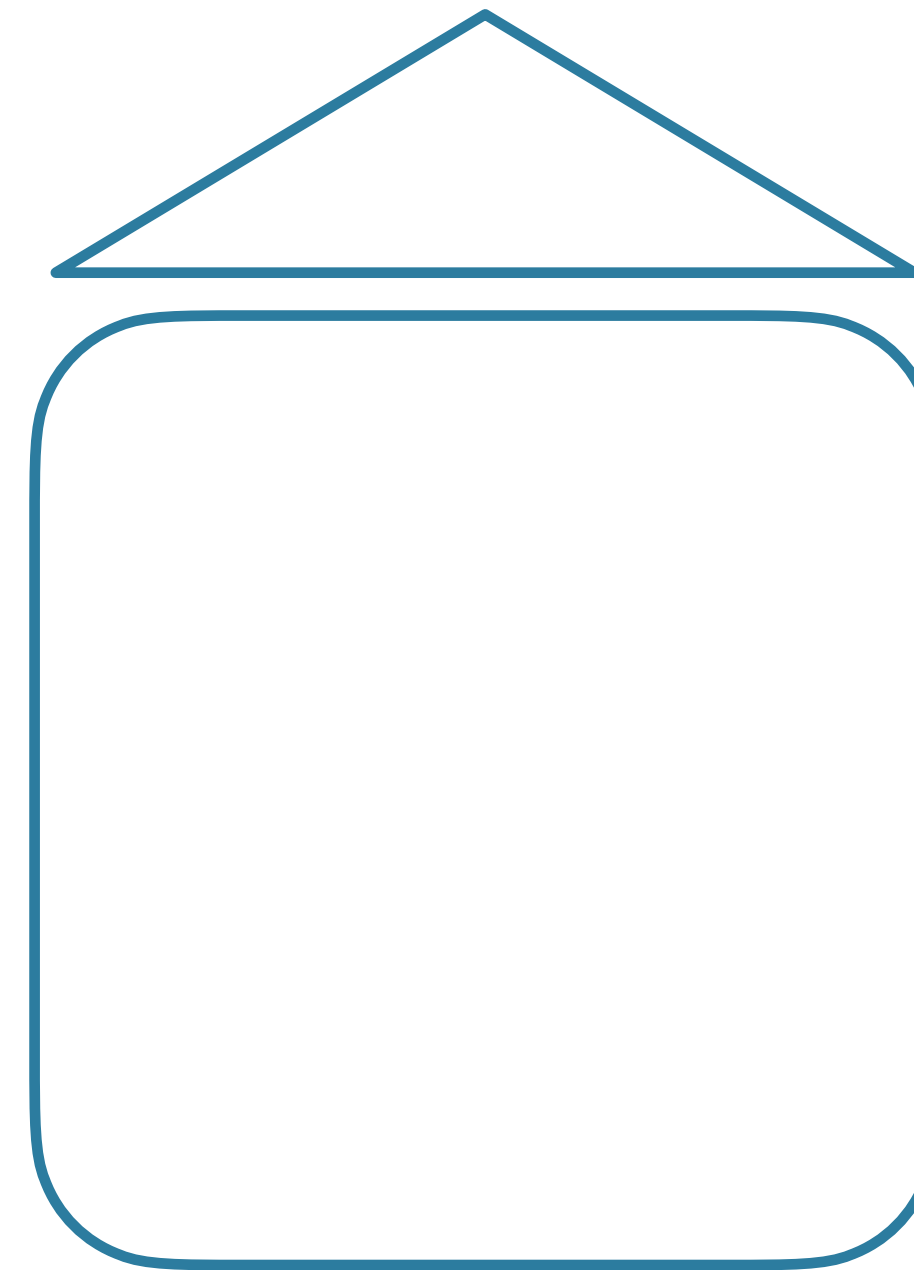
Task 1 inputs

Task 2 outputs



Task 2 inputs

Task 3 outputs

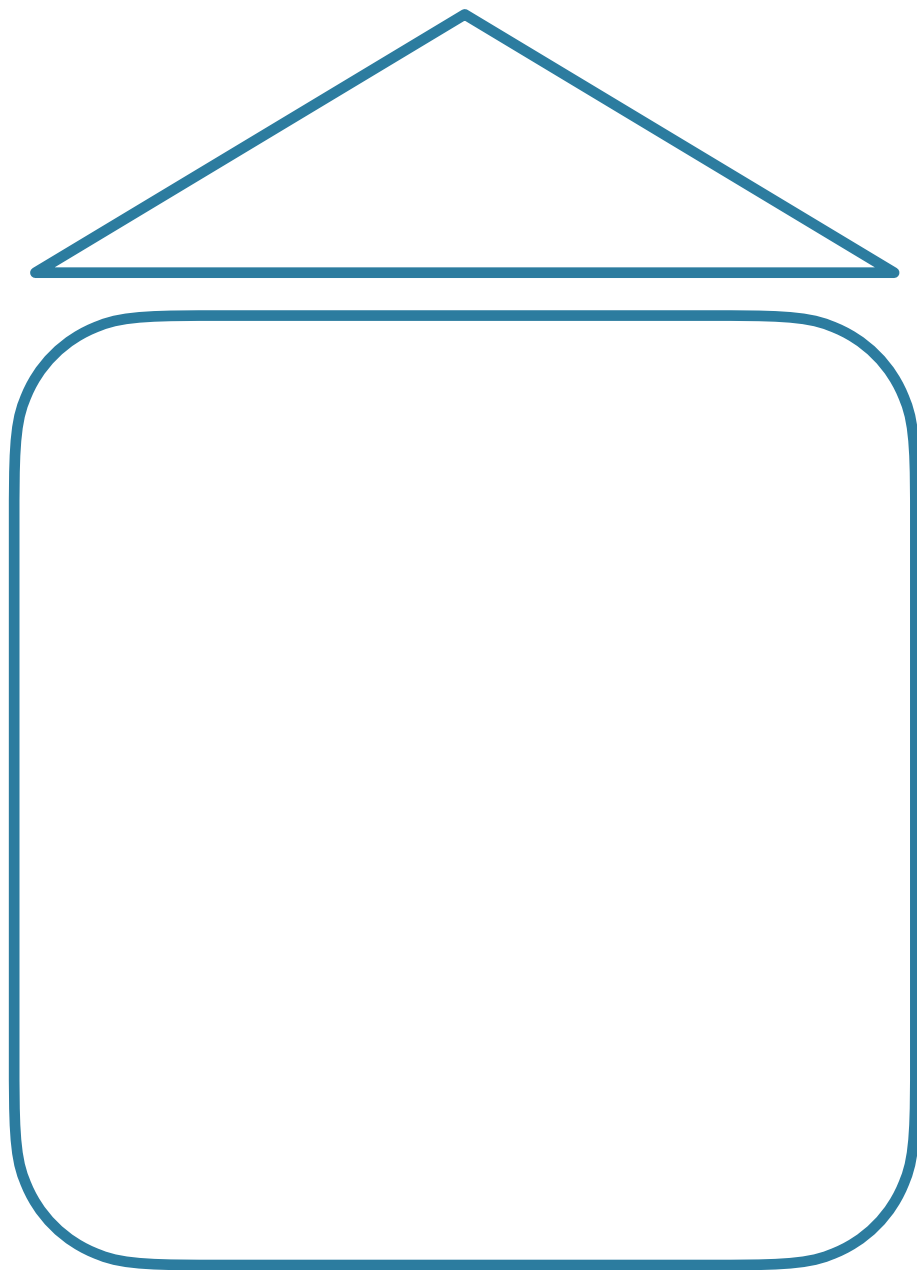


Task 3 inputs

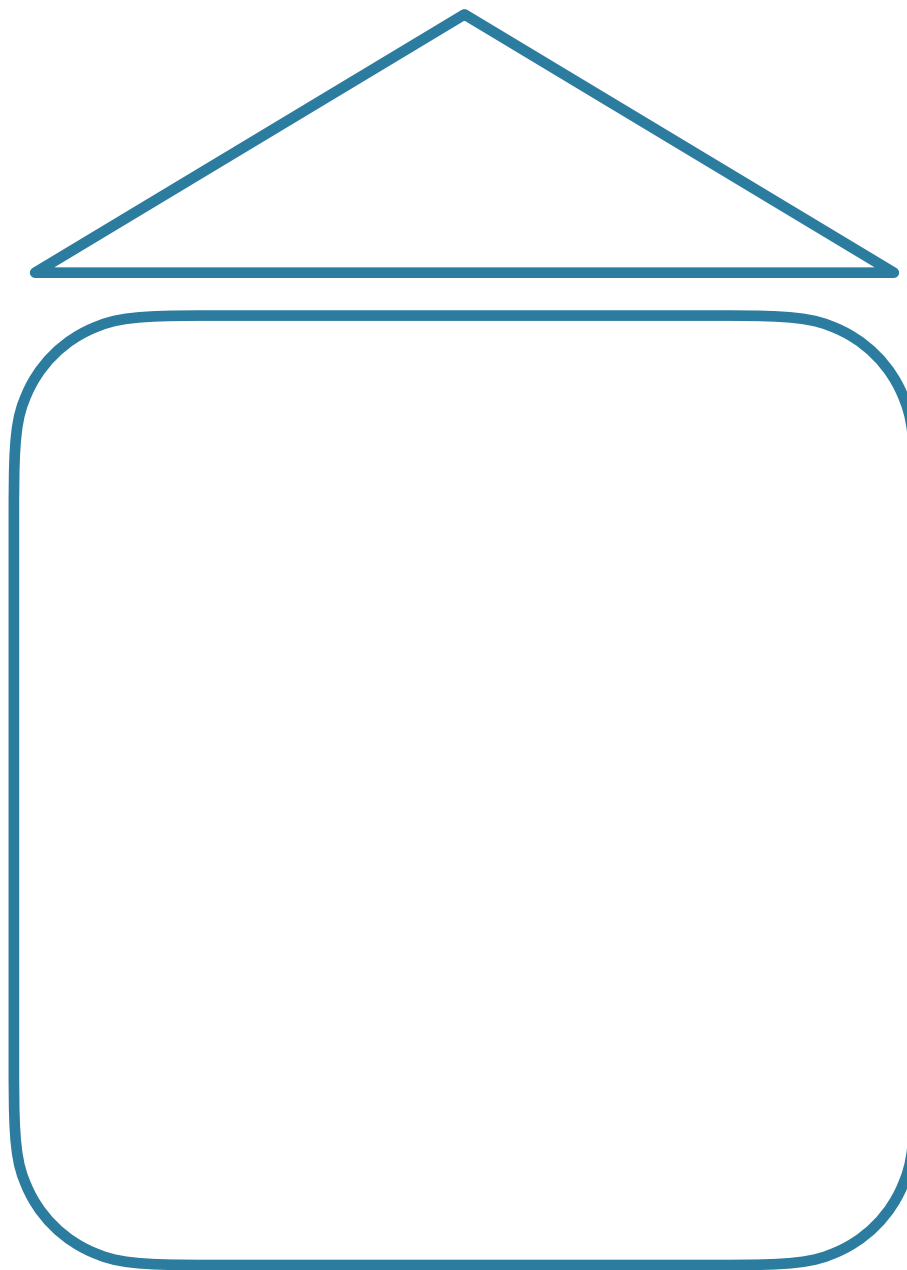


# Standard Learning

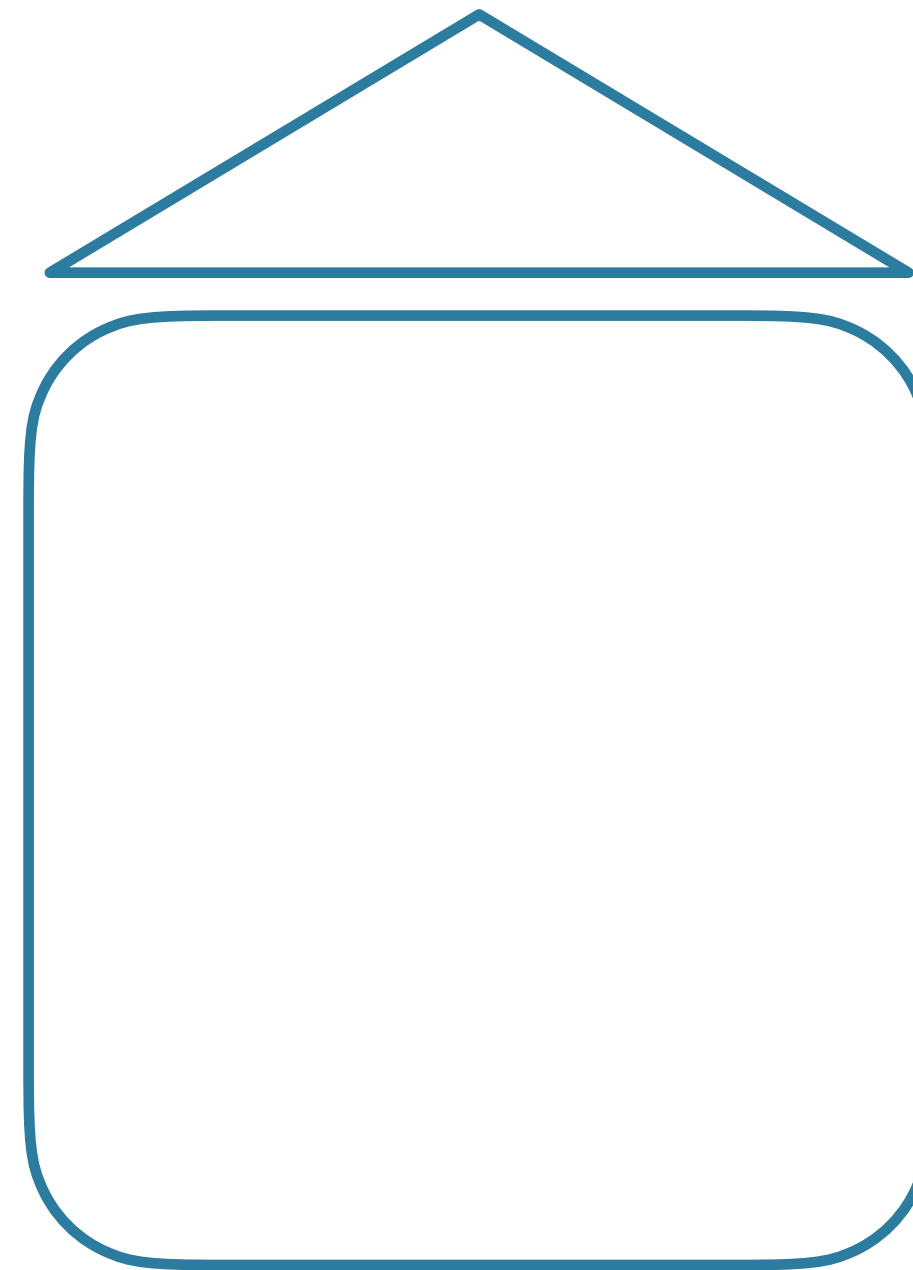
Task 1 outputs



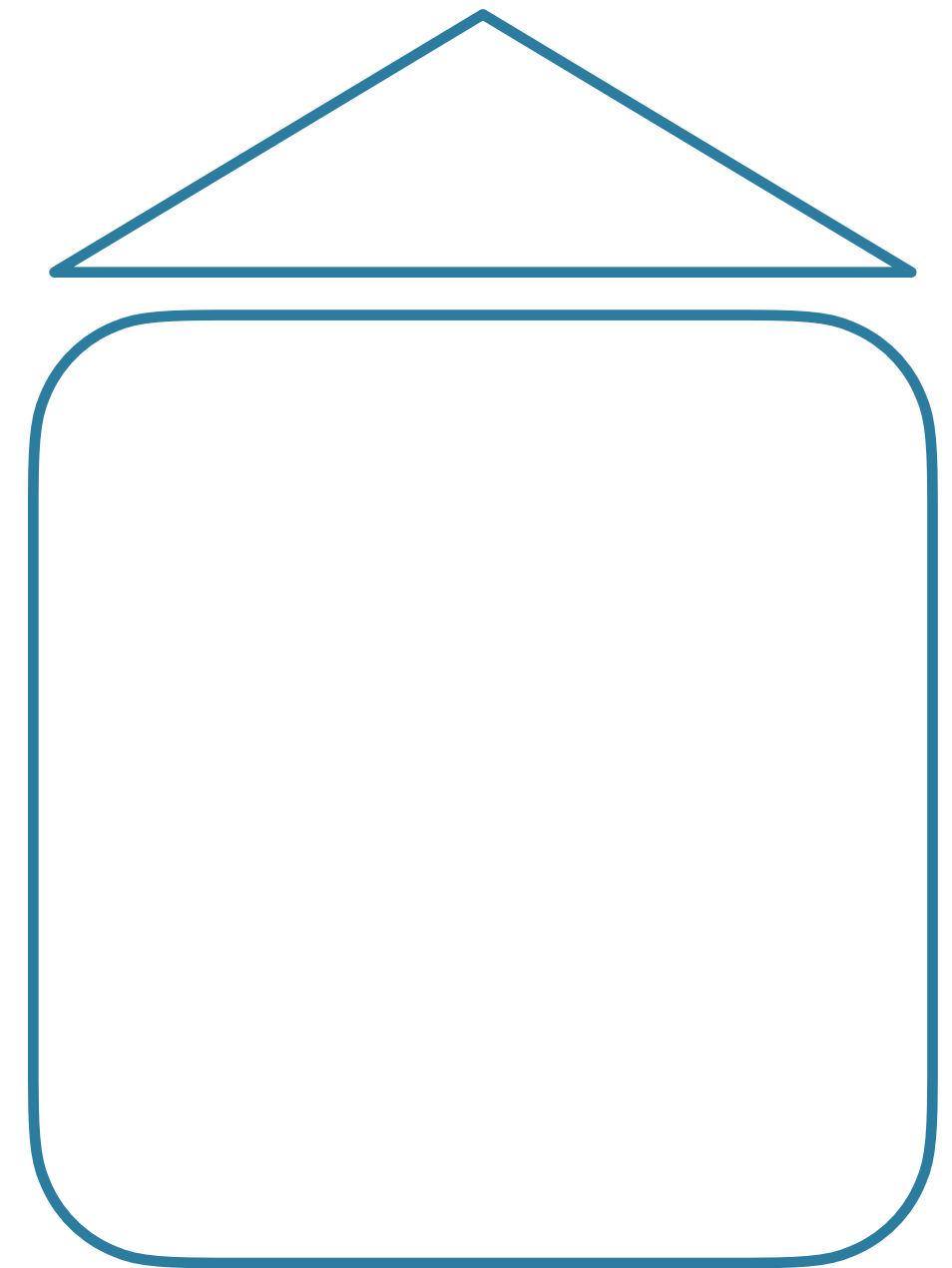
Task 2 outputs



Task 3 outputs



Task 4 outputs



Task 1 inputs

Task 2 inputs

Task 3 inputs

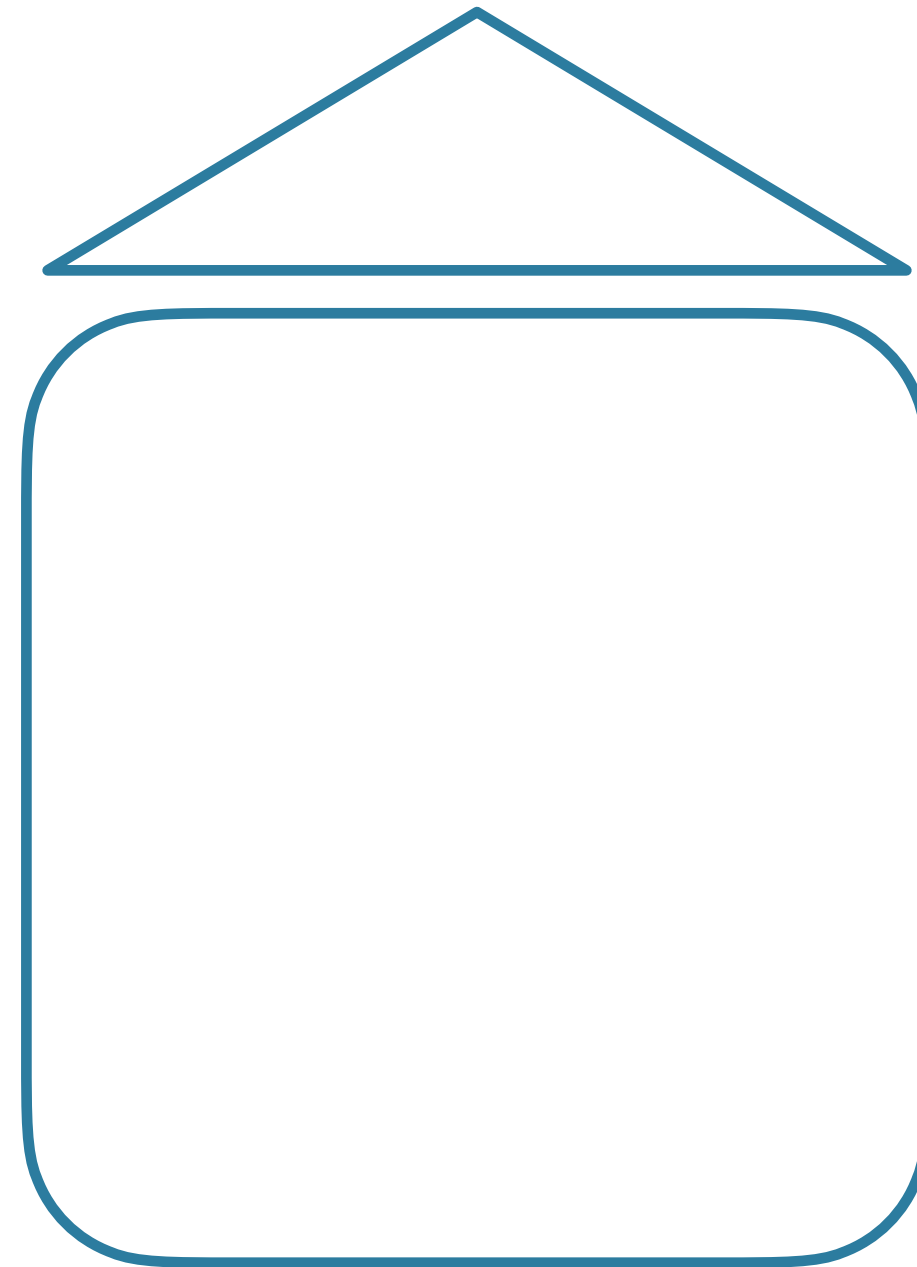
Task 4 inputs

# Standard Learning

- New task = new model
- Expensive!
  - Training time
  - Storage space
  - Data availability
    - Can be impossible in low-data regimes

# Transfer Learning

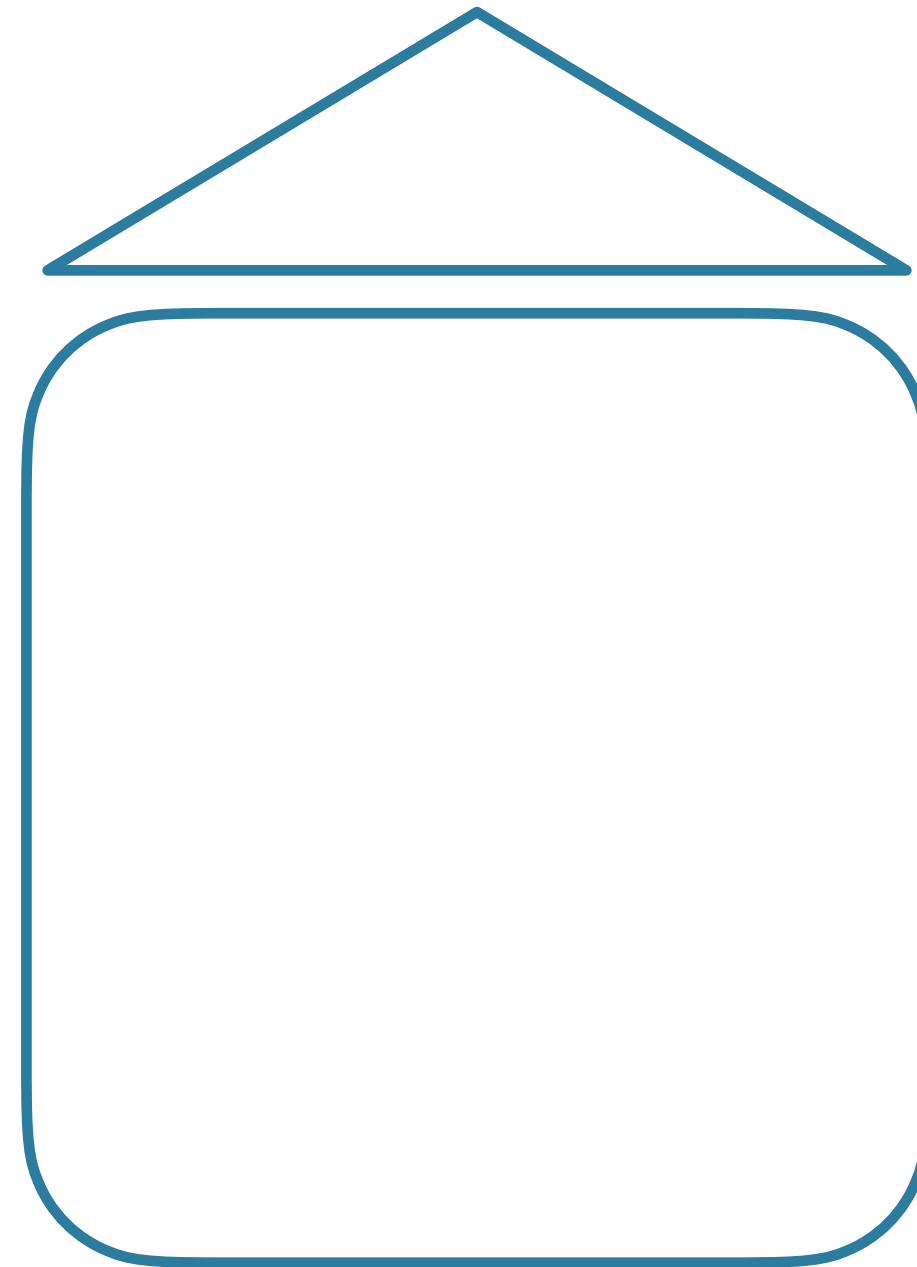
“pre-training” task outputs



“pre-training” task inputs

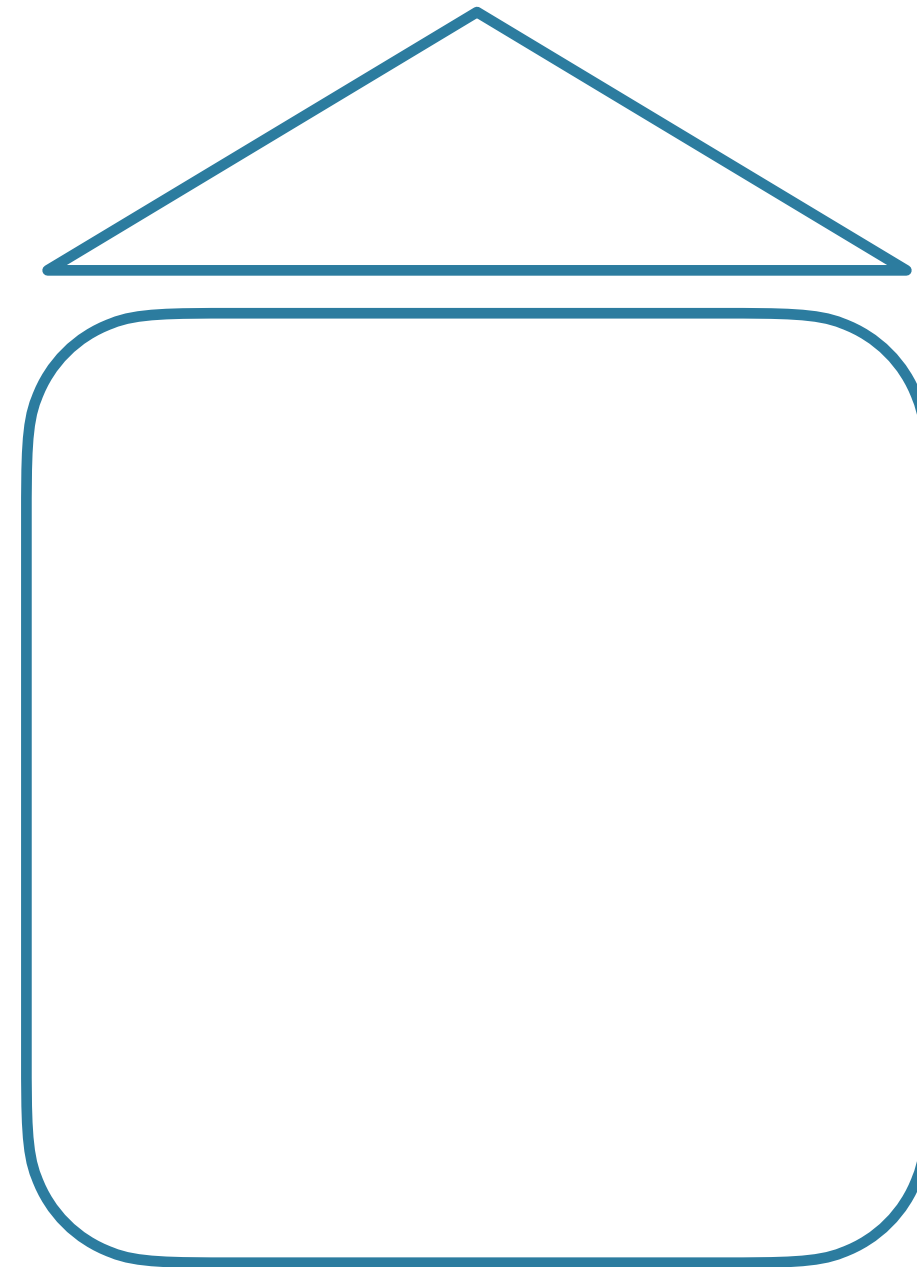
# Transfer Learning

“pre-training” task outputs



# Transfer Learning

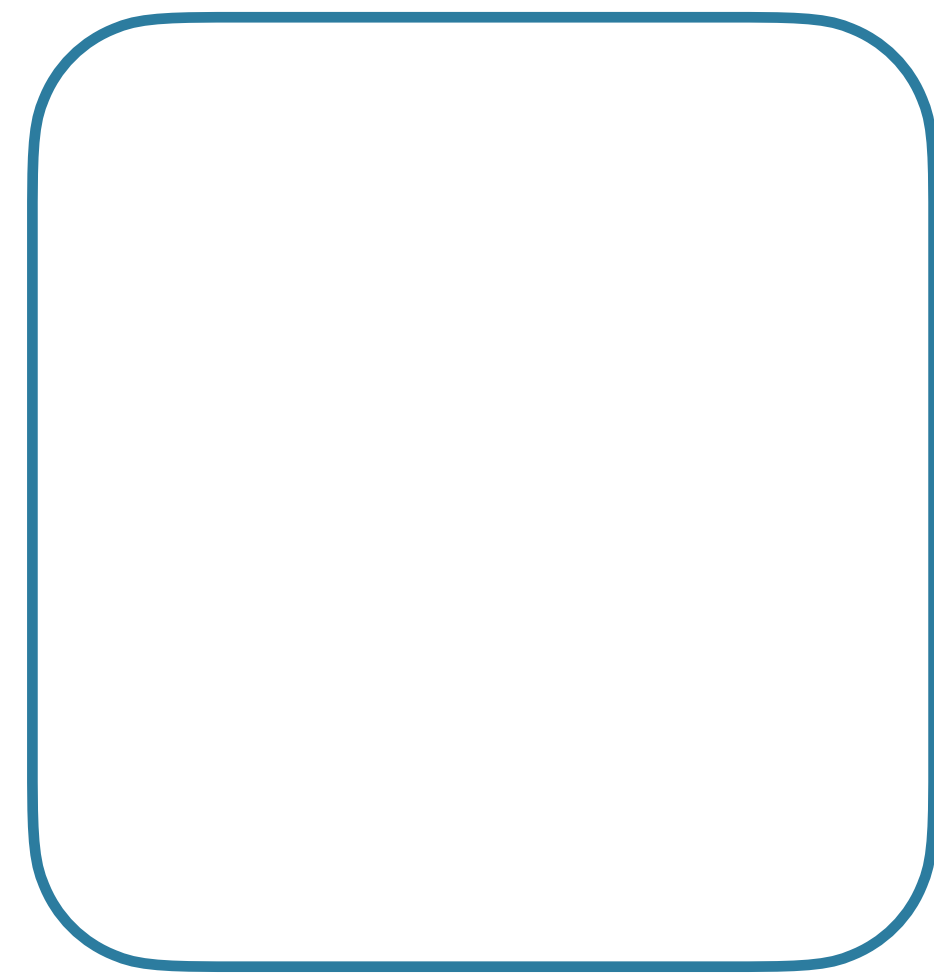
“pre-training” task outputs



Task I inputs

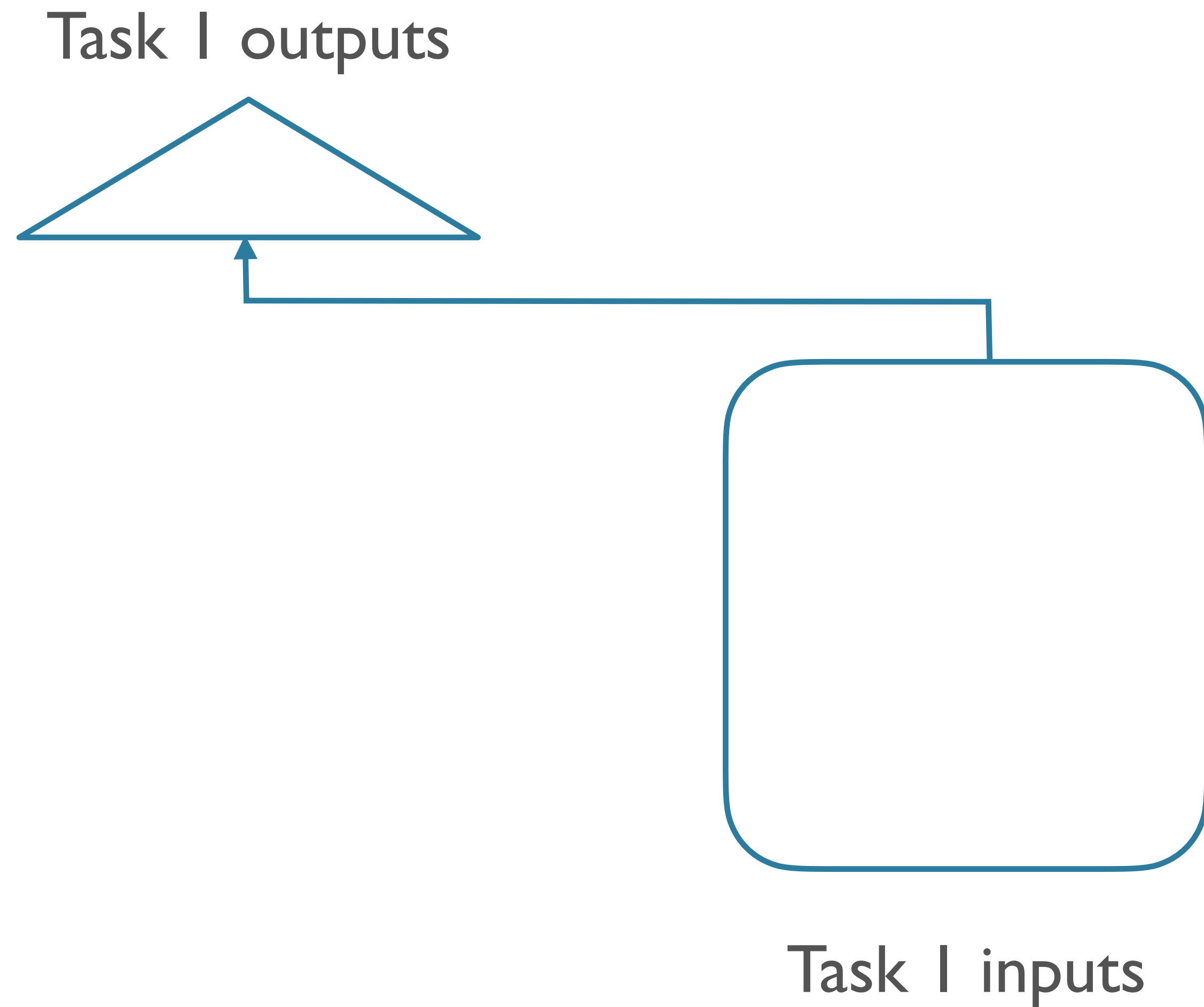


# Transfer Learning

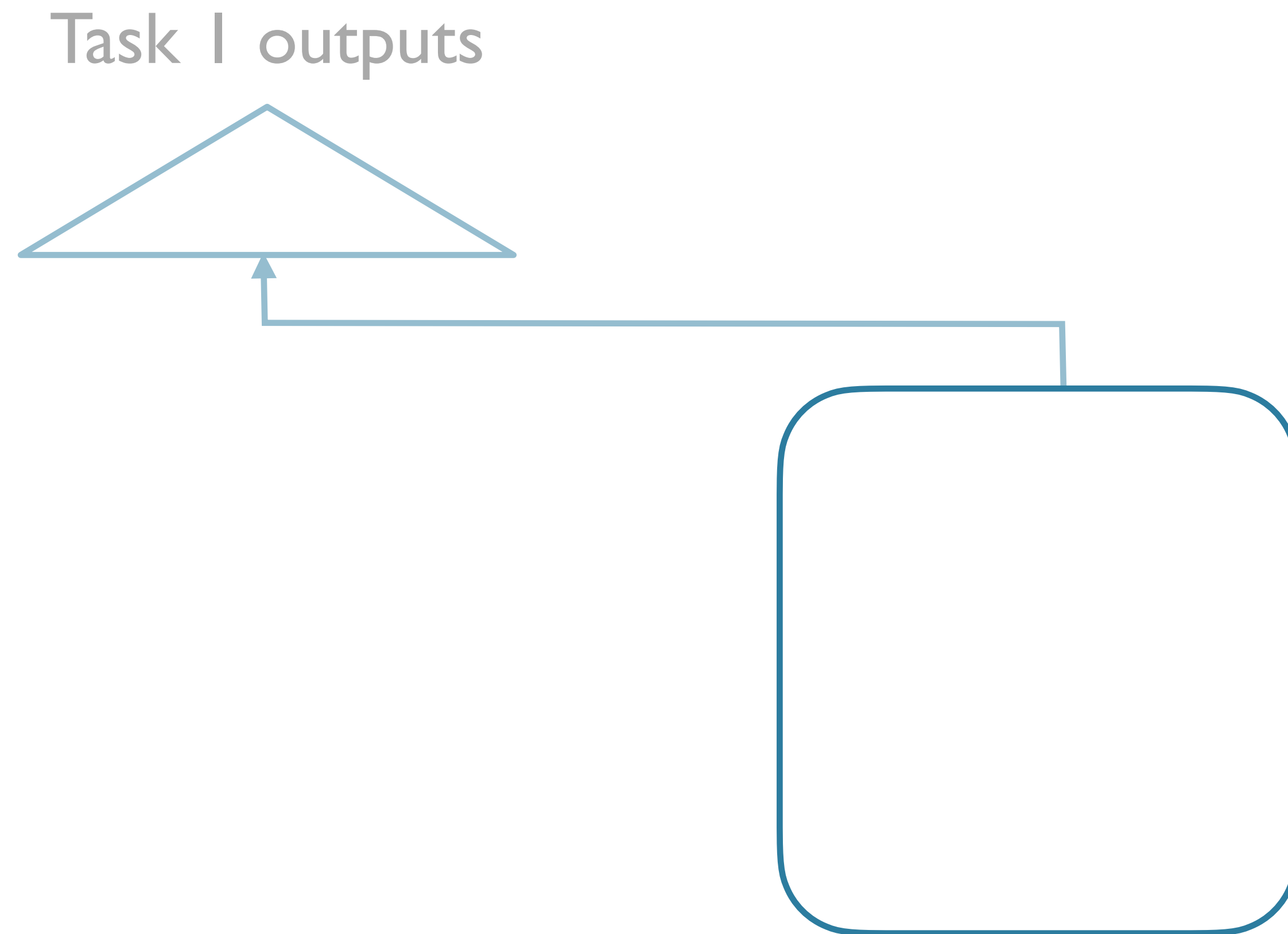


Task I inputs

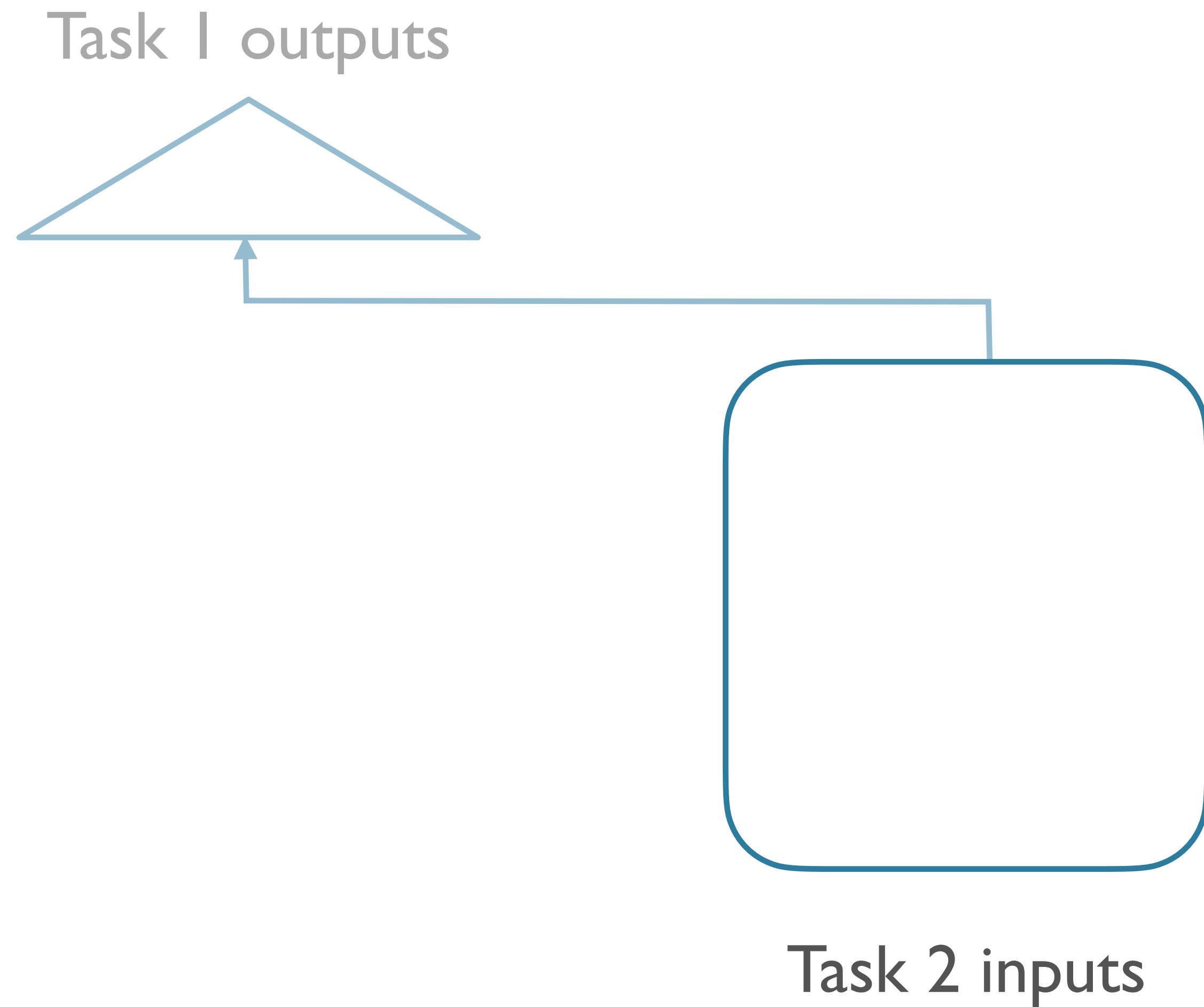
# Transfer Learning



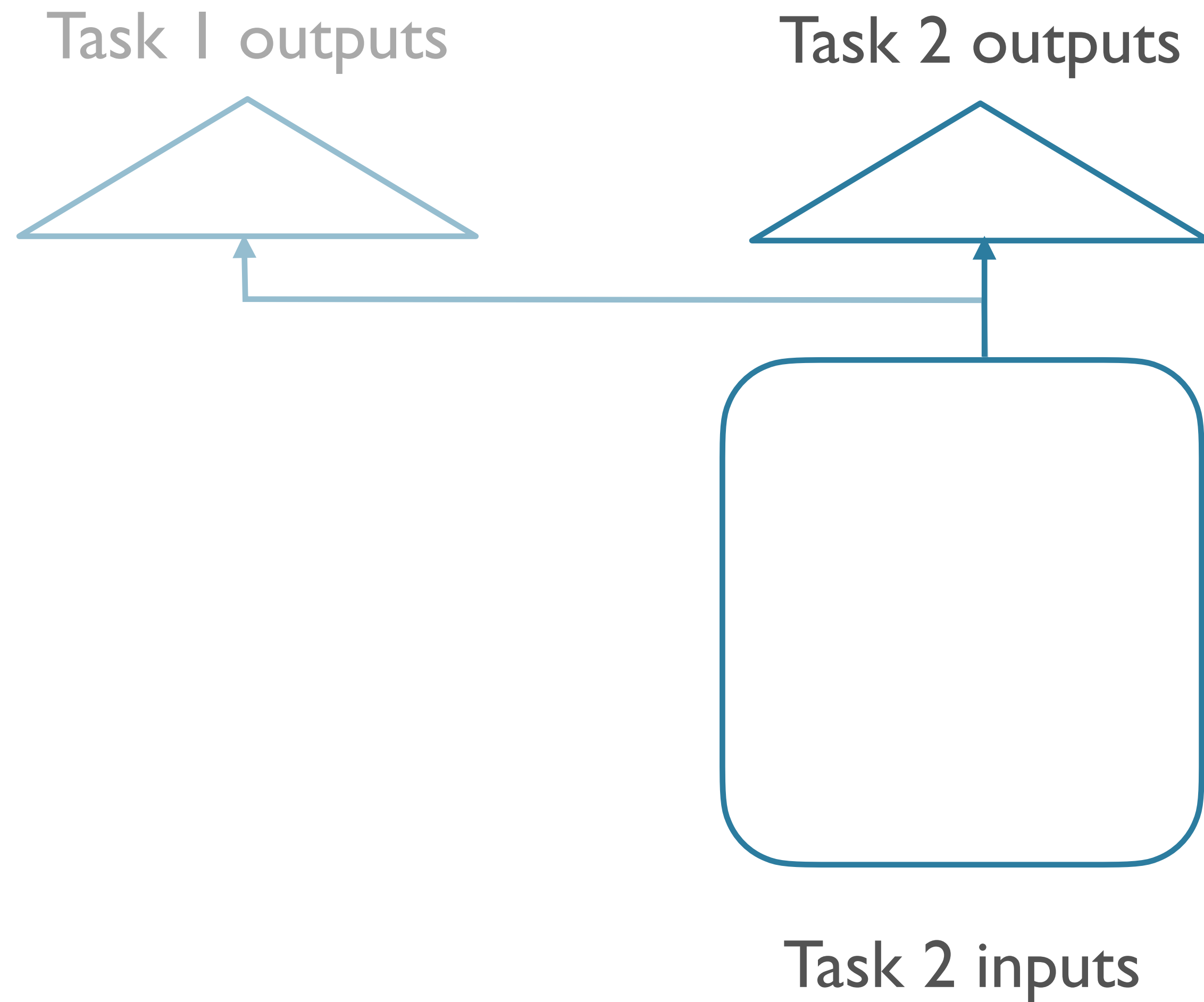
# Transfer Learning



# Transfer Learning

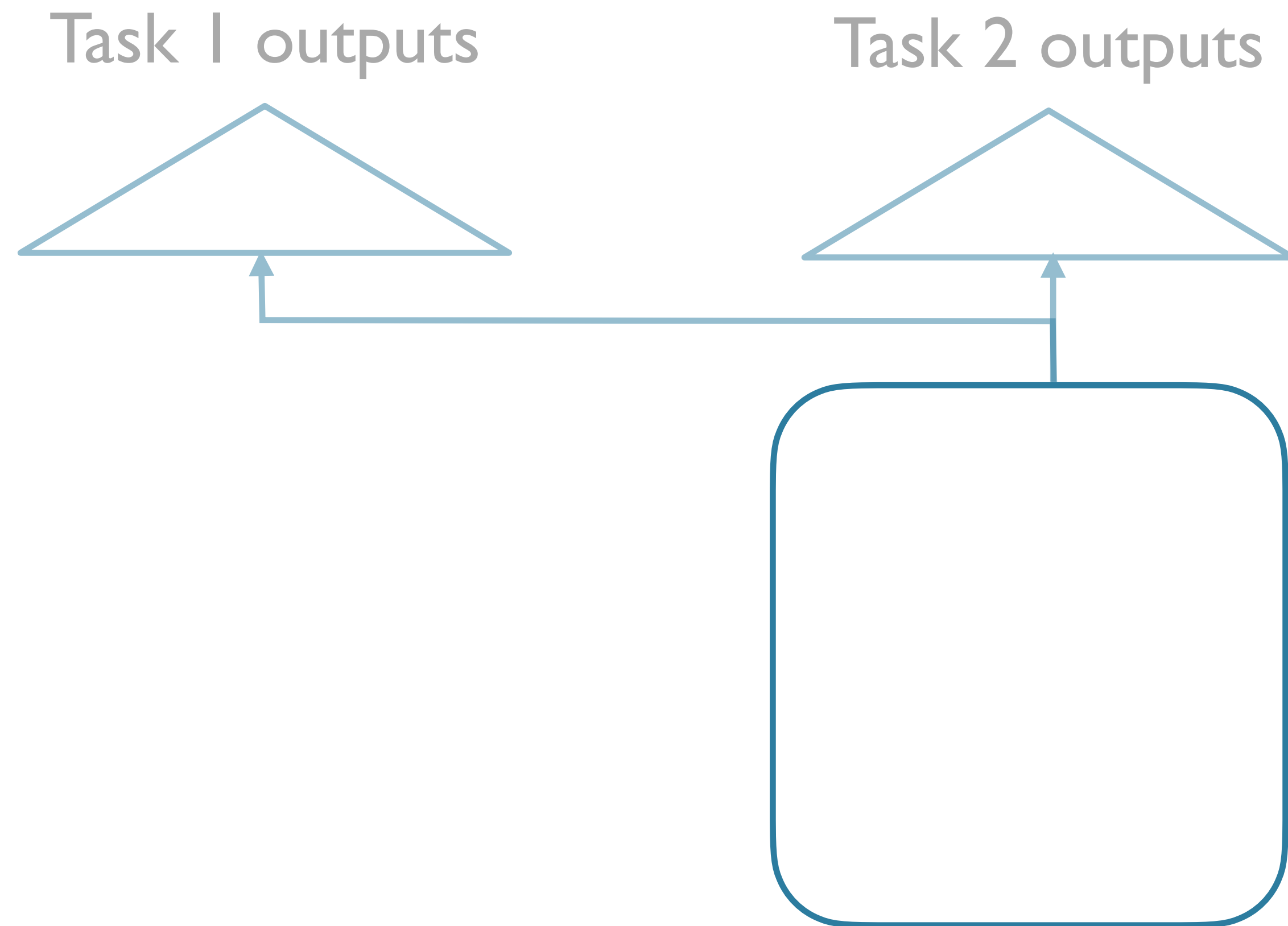


# Transfer Learning

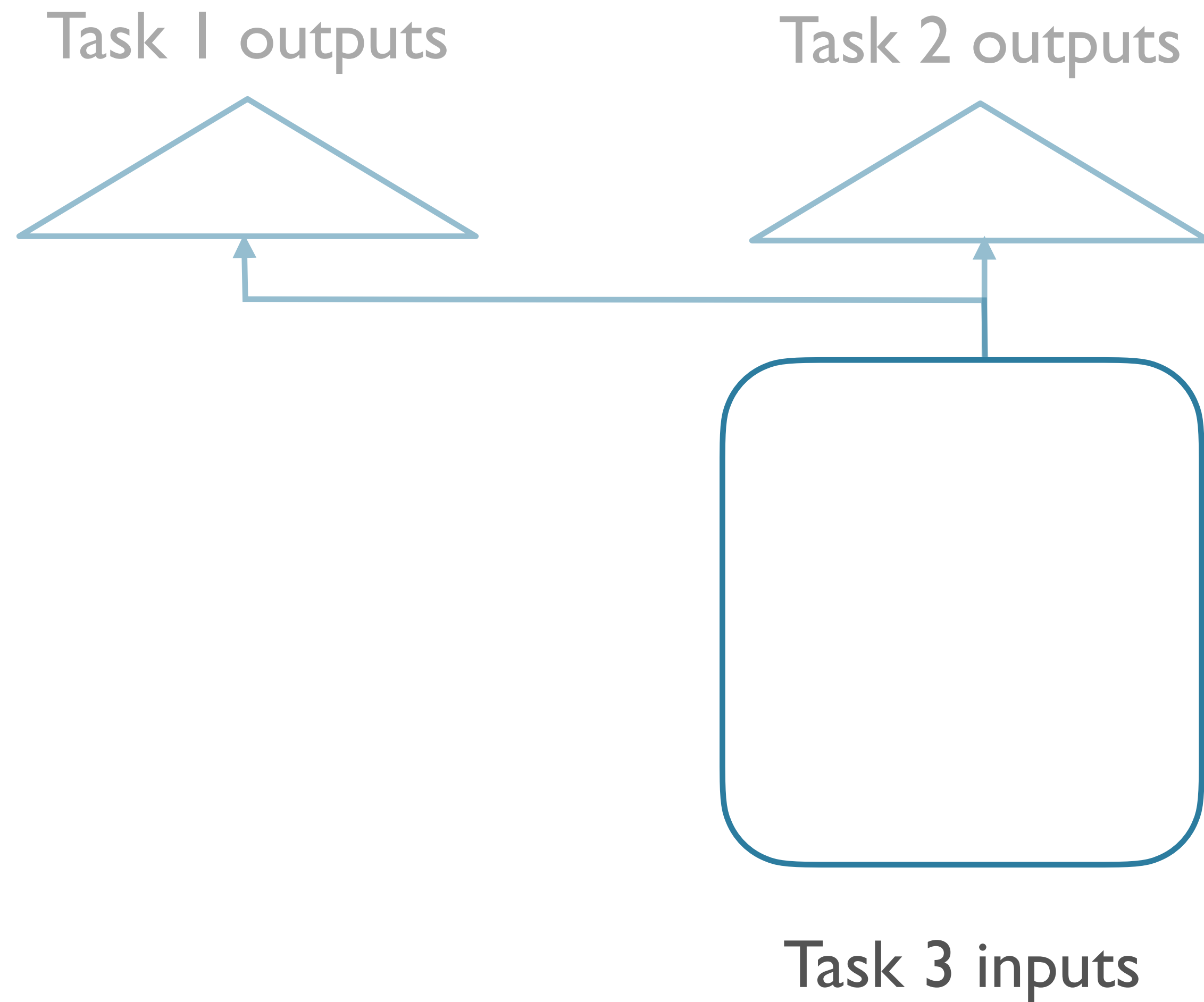




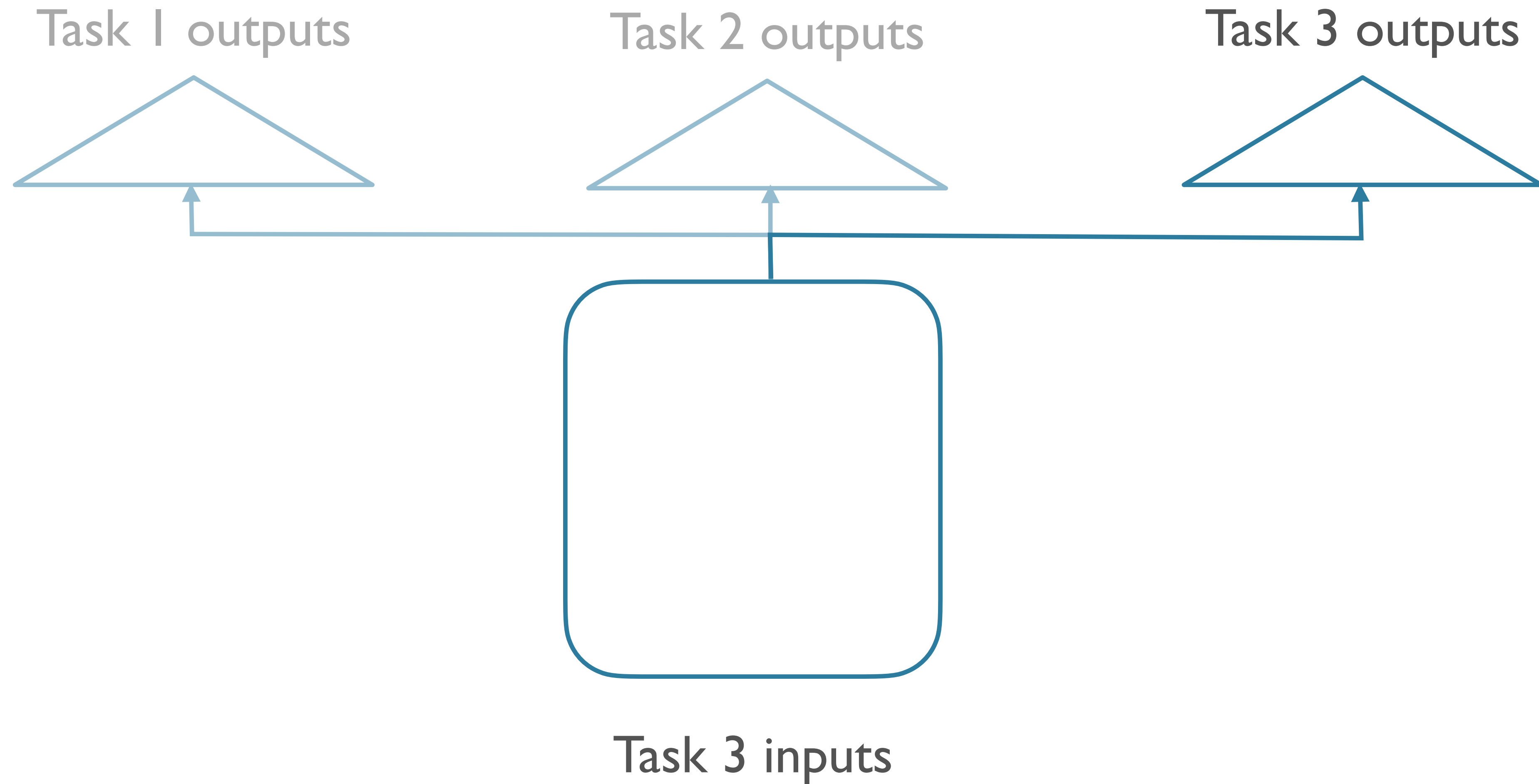
# Transfer Learning



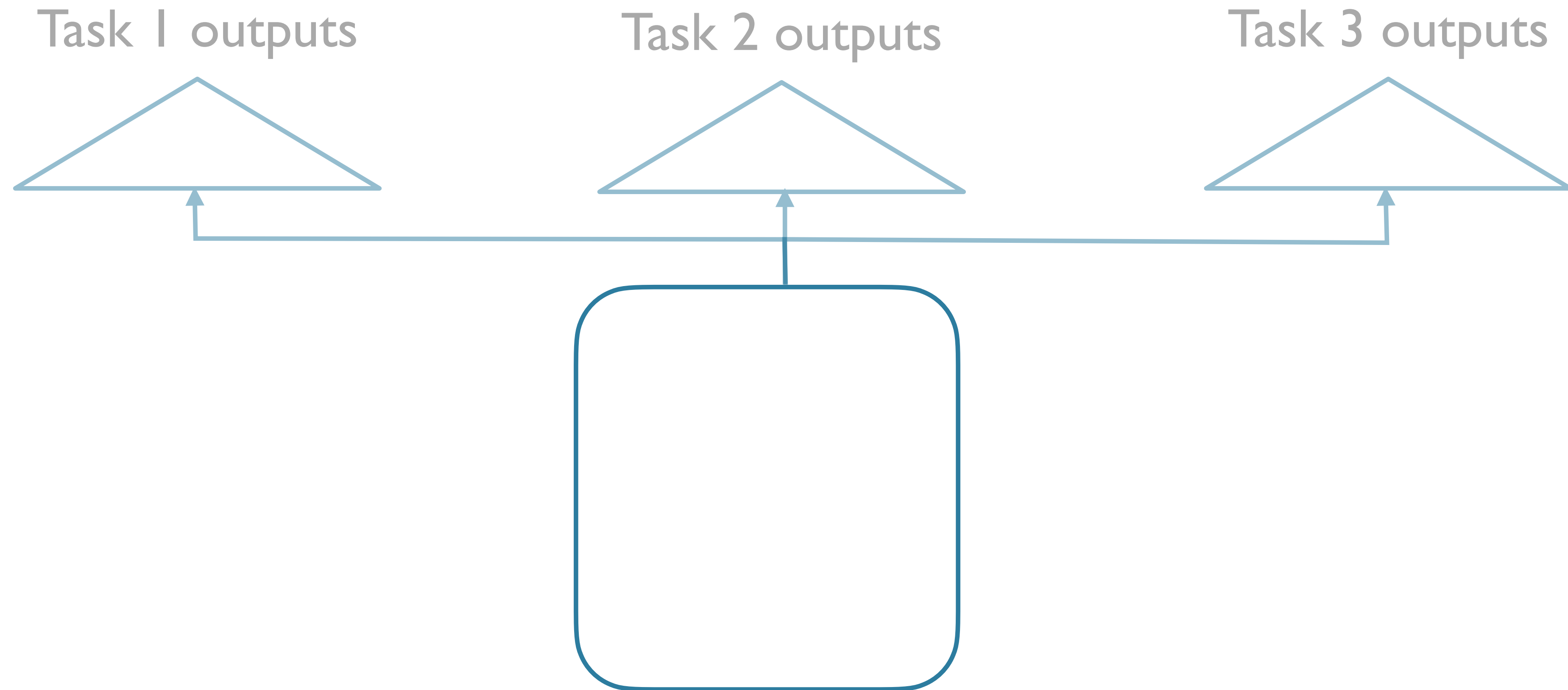
# Transfer Learning



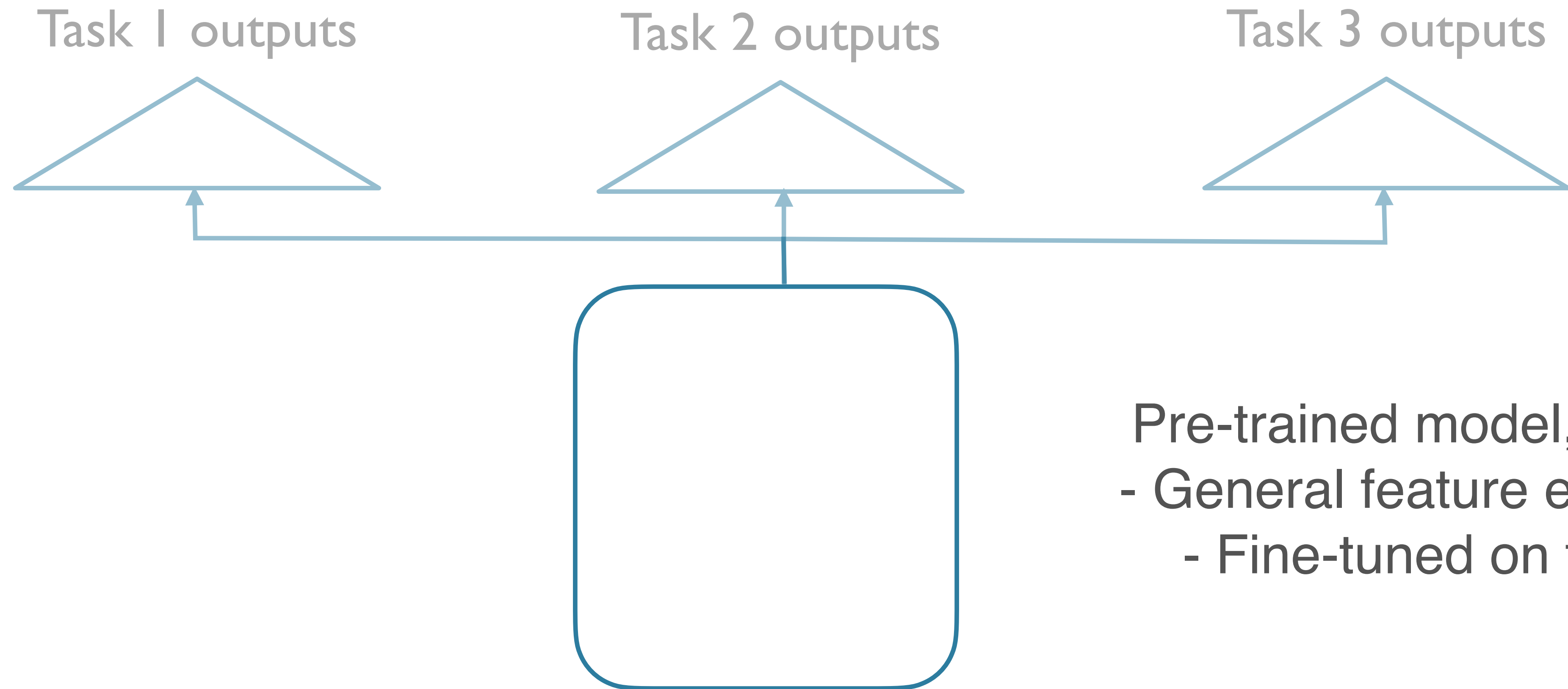
# Transfer Learning



# Transfer Learning



# Transfer Learning



Pre-trained model, either:  
- General feature extractor  
- Fine-tuned on tasks



# Pre-training + Fine-tuning

- Step 1: *pre-train* a model on a “general” task
  - Questions: which task for pre-training? More in a minute.
  - Goal: produce general-purpose representations of the input (“representation learning”), that will be useful when “transferred” to a more specific task.
- Step 2: *fine-tune* that model on the main task
  - Replace the “head” of the model with some task-specific layers
  - Run supervised training with the resulting model

# Transfer Learning in Computer Vision

“We use features extracted from the `OverFeat` network as a generic image representation to tackle the diverse range of recognition tasks of object image classification, scene recognition, fine grained recognition, attribute detection and image retrieval applied to a diverse set of datasets. We selected these tasks and datasets as they gradually move further away from the original task and data the `OverFeat` network was trained to solve [cf. ImageNet]. Astonishingly, we report consistent superior results compared to the highly tuned state-of-the-art systems in all the visual classification tasks on various datasets”

## **CNN Features off-the-shelf: an Astounding Baseline for Recognition**

Ali Sharif Razavian Hossein Azizpour Josephine Sullivan Stefan Carlsson  
CVAP, KTH (Royal Institute of Technology)  
Stockholm, Sweden

`{razavian, azizpour, sullivan, stefanc}@csc.kth.se`

# Current Benchmarks

Leaderboard Version: 2.0

Rank	Name	Model	URL	Score	BoolQ	CB	COPA	MultiRC	ReCoRD	RTE	WIC	WSC	AX-g	AX-b
+	1	Zirui Wang	T5 + Meena, Single Model (Meena Team - Google Brain)	90.4	91.4	95.8/97.6	98.0	88.3/63.0	94.2/93.5	93.0	77.9	96.6	92.7/91.9	69.1
+	2	DeBERTa Team - Microsoft	DeBERTa / TuringNLRv4	90.3	90.4	95.7/97.6	98.4	88.2/63.7	94.5/94.1	93.2	77.5	95.9	93.3/93.8	66.7
	3	SuperGLUE Human Baselines	SuperGLUE Human Baselines	89.8	89.0	95.8/98.9	100.0	81.8/51.9	91.7/91.3	93.6	80.0	100.0	99.3/99.7	76.6
+	4	T5 Team - Google	T5	89.3	91.2	93.9/96.8	94.8	88.1/63.3	94.1/93.4	92.5	76.9	93.8	92.7/91.9	65.6
+	5	Huawei Noah's Ark Lab	NEZHA-Plus	86.7	87.8	94.4/96.0	93.6	84.6/55.1	90.1/89.6	89.1	74.6	93.2	87.1/74.4	58.0
+	6	Alibaba PAI&ICBU	PAI Albert	86.1	88.1	92.4/96.4	91.8	84.6/54.7	89.0/88.3	88.8	74.1	93.2	98.3/99.2	75.6
+	7	Infosys : DAWN : AI Research	RoBERTa-iCETS	86.0	88.5	93.2/95.2	91.2	86.4/58.2	89.9/89.3	89.9	72.9	89.0	88.8/81.5	61.8
+	8	Tencent Jarvis Lab	RoBERTa (ensemble)	85.9	88.2	92.5/95.6	90.8	84.4/53.4	91.5/91.0	87.9	74.1	91.8	89.3/75.6	57.6
	9	Zhuiyi Technology	RoBERTa-mtl-adv	85.7	87.1	92.4/95.6	91.2	85.1/54.3	91.7/91.3	88.1	72.1	91.8	91.0/78.1	58.5
	10	Facebook AI	RoBERTa	84.6	87.1	90.5/95.2	90.6	84.4/52.5	90.6/90.0	88.2	69.9	89.0	91.0/78.1	57.9
+	11	Anuar Sharafudinov	AILabs Team, Transformers	82.6	88.1	91.6/94.8	86.8	85.1/54.7	82.8/79.8	88.9	74.1	78.8	100.0/100.0	100.0
	12	Rakesh Radhakrishnan Menon	ADAPET (ALBERT) - few-shot	76.0	80.0	82.3/92.0	85.4	76.2/35.7	86.1/85.5	75.0	53.5	85.6	100.0/50.0	-0.4
+	13	Timo Schick	iPET (ALBERT) - Few-Shot (32 Examples)	75.4	81.2	79.9/88.8	90.8	74.1/31.7	85.9/85.4	70.8	49.3	88.4	97.8/57.9	36.2
	14	Adrian de Wynter	Bort (Alexa AI)	74.1	83.7	81.9/86.4	89.6	83.7/54.1	49.8/49.0	81.2	70.1	65.8	96.1/61.5	48.0
	15	IBM Research AI	BERT-mtl	73.5	84.8	89.6/94.0	73.8	73.2/30.5	74.6/74.0	84.1	66.2	61.0	97.8/57.3	29.6
	16	Ben Mann	GPT-3 few-shot - OpenAI	71.8	76.4	52.0/75.6	92.0	75.4/30.5	91.1/90.2	69.0	49.4	80.1	90.4/55.3	21.1
	17	SuperGLUE Baselines	BERT++	71.5	79.0	84.8/90.4	73.8	70.0/24.1	72.0/71.3	79.0	69.6	64.4	99.4/51.4	38.0
			BERT	69.0	77.4	75.7/83.6	70.6	70.0/24.1	72.0/71.3	71.7	69.6	64.4	97.8/51.7	23.0
			Most Frequent Class	47.1	62.3	21.7/48.4	50.0	61.1/0.3	33.4/32.5	50.3	50.0	65.1	100.0/50.0	0.0
			CBoW	44.5	62.2	49.0/71.2	51.6	0.0/0.5	14.0/13.6	49.7	53.1	65.1	100.0/50.0	-0.4

# Language Model Pre-training

# Where to transfer *from*?

# Where to transfer *from*?

- Goal: find a linguistic task that will build general-purpose / *transferable* representations

# Where to transfer *from*?

- Goal: find a linguistic task that will build general-purpose / *transferable* representations
- Possibilities:

# Where to transfer *from*?

- Goal: find a linguistic task that will build general-purpose / *transferable* representations
- Possibilities:
  - Constituency or dependency parsing



# Where to transfer *from*?

- Goal: find a linguistic task that will build general-purpose / *transferable* representations
- Possibilities:
  - Constituency or dependency parsing
  - Semantic parsing

# Where to transfer *from*?

- Goal: find a linguistic task that will build general-purpose / *transferable* representations
- Possibilities:
  - Constituency or dependency parsing
  - Semantic parsing
  - Machine translation

# Where to transfer *from*?

- Goal: find a linguistic task that will build general-purpose / *transferable* representations
- Possibilities:
  - Constituency or dependency parsing
  - Semantic parsing
  - Machine translation
  - QA

# Where to transfer *from*?

- Goal: find a linguistic task that will build general-purpose / *transferable* representations
- Possibilities:
  - Constituency or dependency parsing
  - Semantic parsing
  - Machine translation
  - QA
  - ...

# Where to transfer *from*?

- Goal: find a linguistic task that will build general-purpose / *transferable* representations
- Possibilities:
  - Constituency or dependency parsing
  - Semantic parsing
  - Machine translation
  - QA
  - ...
- Scalability issue: all require expensive annotation

# Language Modeling

# Language Modeling

- A good language model should produce good *general-purpose* and *transferable* representations

# Language Modeling

- A good language model should produce good *general-purpose* and *transferable* representations
- Linguistic knowledge:
  - The bicycles, even though old, were in good shape because \_\_\_\_\_ ...
  - The bicycle, even though old, was in good shape because \_\_\_\_\_ ...

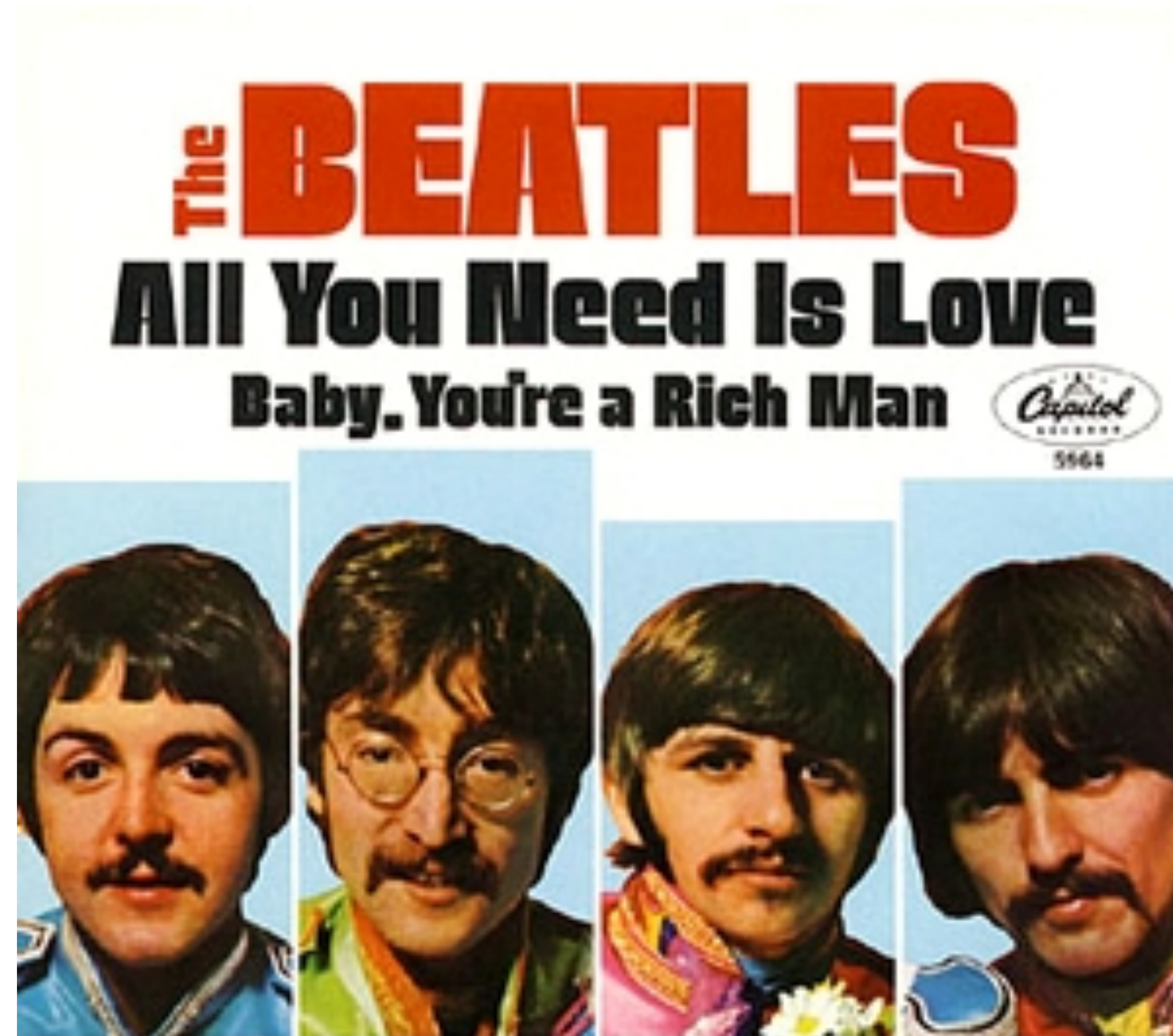


# Language Modeling

- A good language model should produce good *general-purpose* and *transferable* representations
- Linguistic knowledge:
  - The bicycles, even though old, were in good shape because \_\_\_\_\_ ...
  - The bicycle, even though old, was in good shape because \_\_\_\_\_ ...
- World knowledge:
  - The University of Washington was founded in \_\_\_\_\_
  - Seattle had a huge population boom as a launching point for expeditions to \_\_\_\_\_

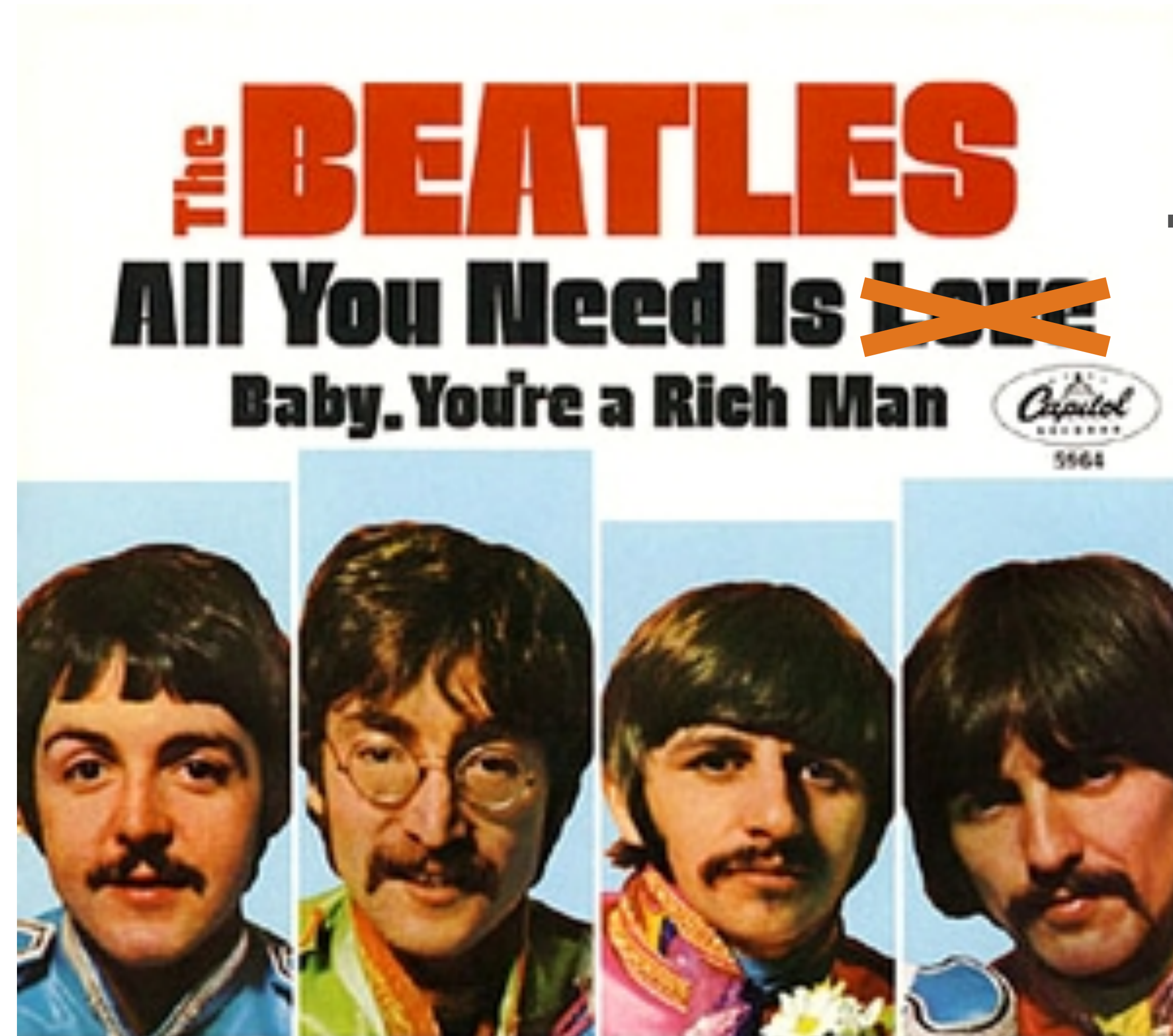
# Data for LM is cheap

# Data for LM is cheap





# Data for LM is cheap



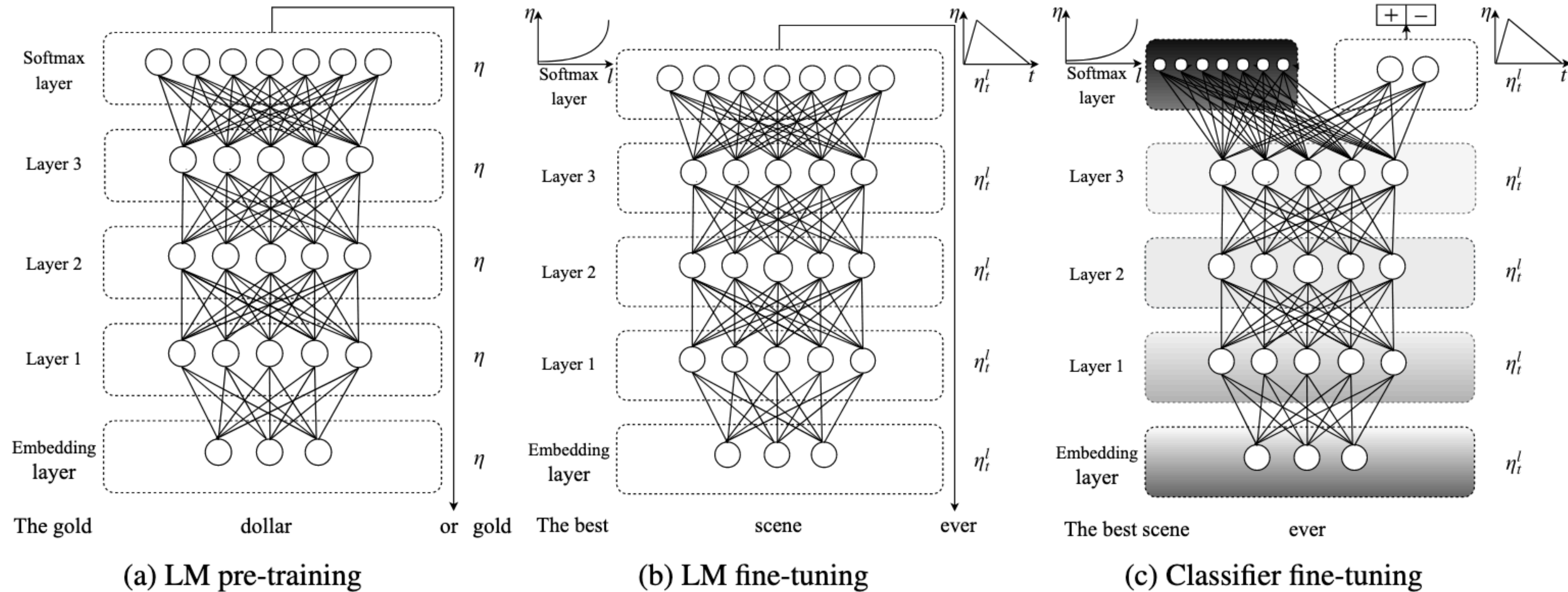
Text

# Language Model Pre-training

- A currently powerful paradigm for training models for NLP tasks:
  - *Pre-train* a large language model on a large amount of raw text
  - *Fine-tune* a small model on top of the LM for the task you care about
    - (or use the LM as a general feature extractor)



# ULMFiT

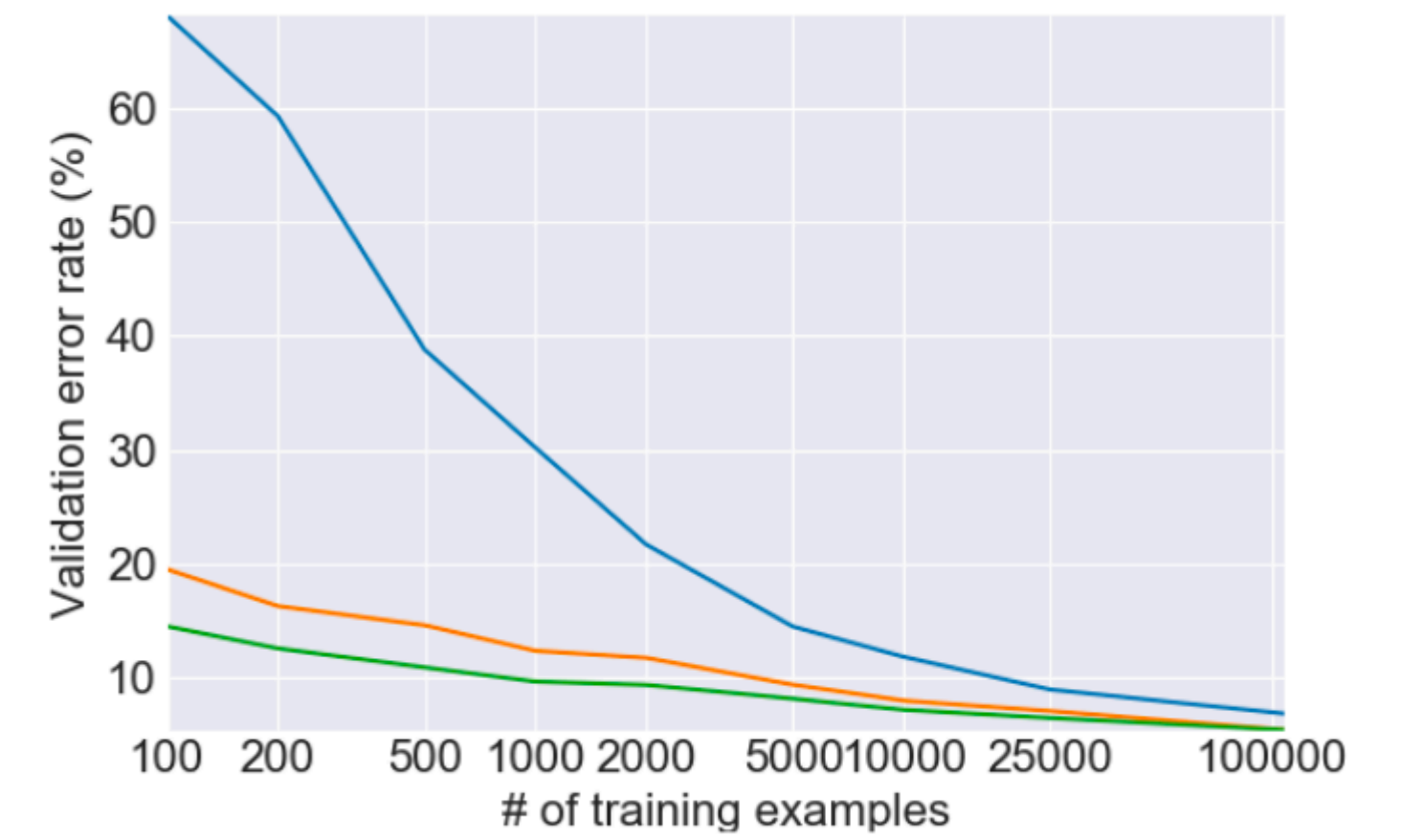
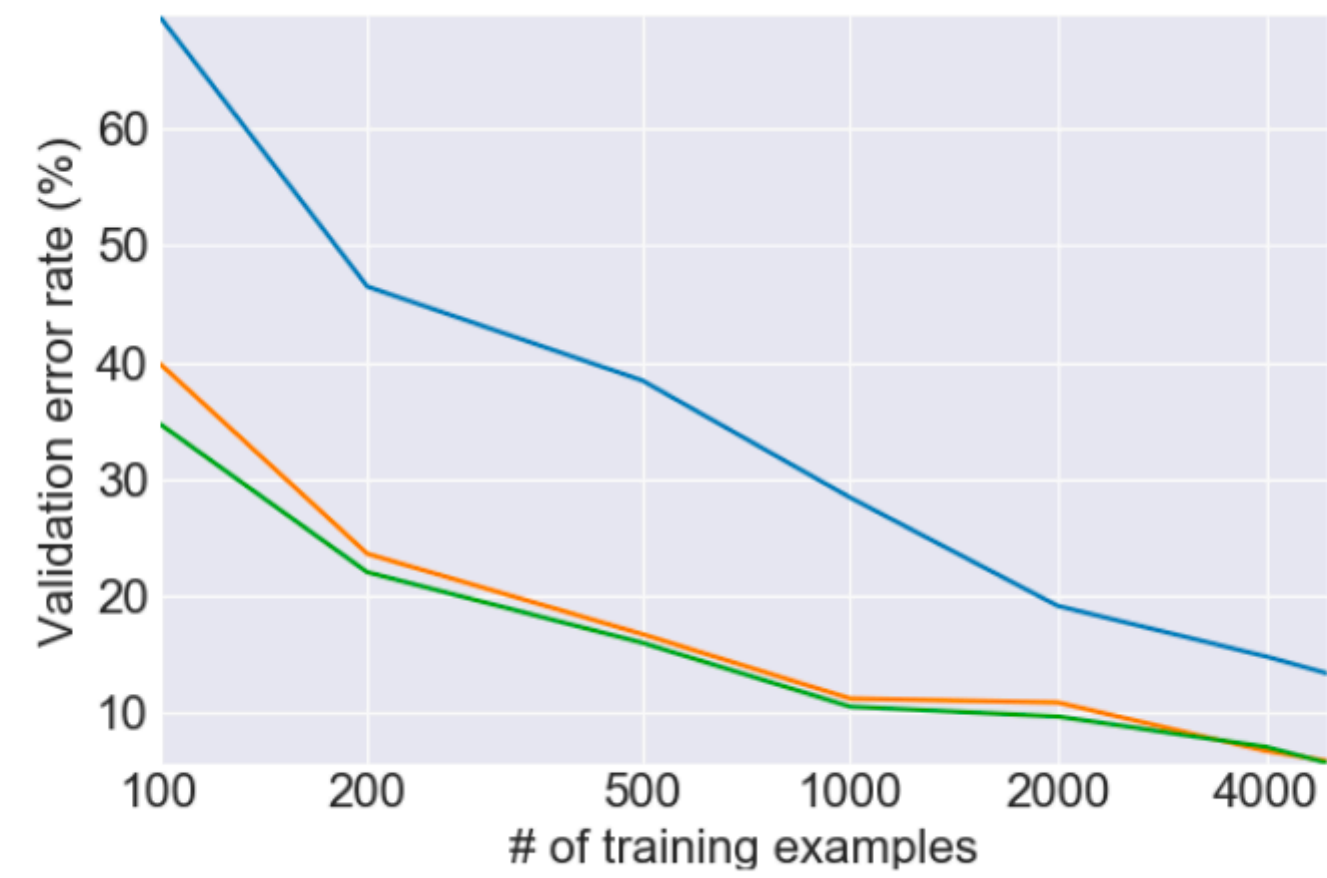
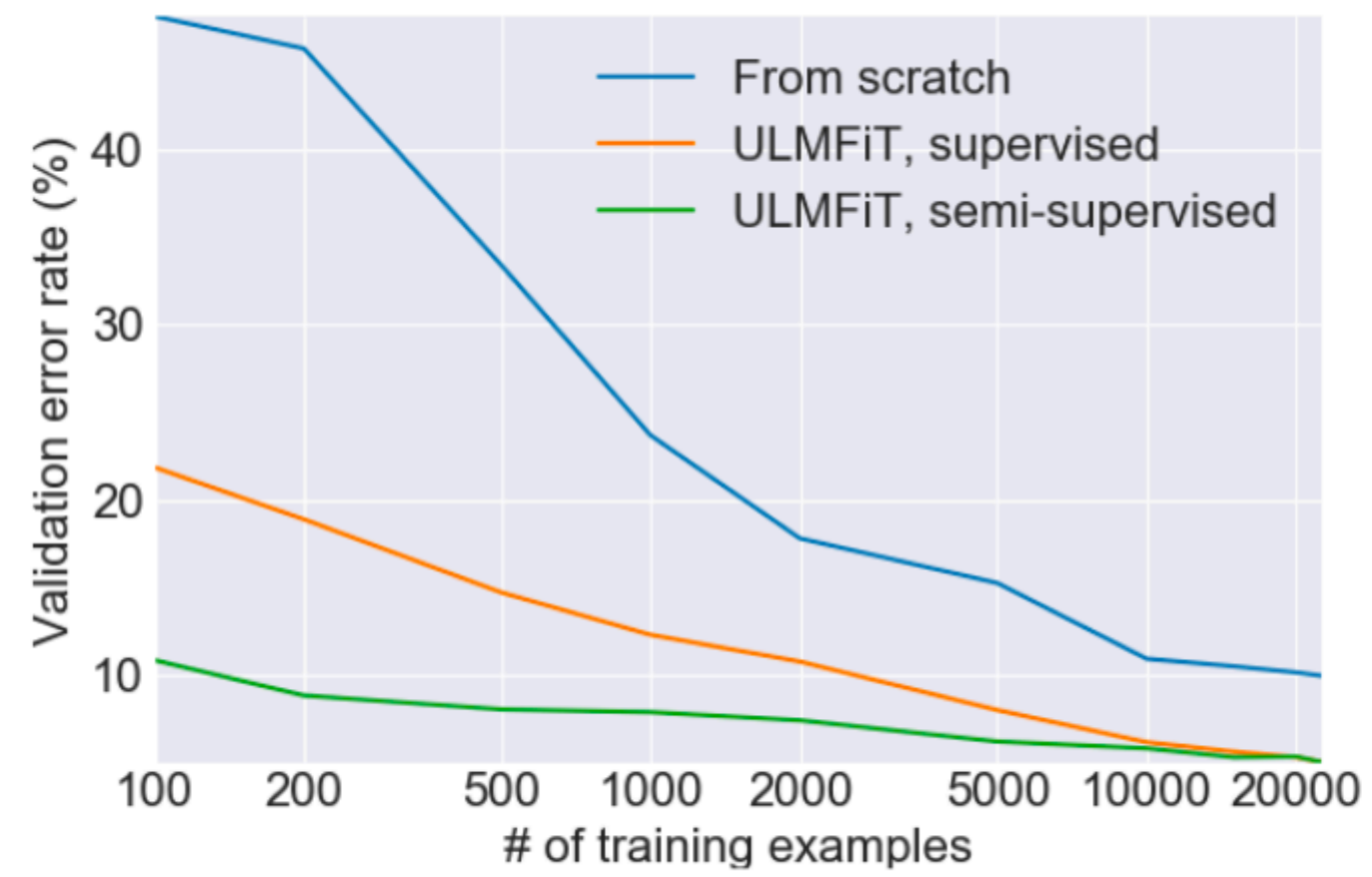


## Universal Language Model Fine-tuning for Text Classification (ACL '18)

# ULMFiT

	Model	Test		Model	Test
IMDb	CoVe (McCann et al., 2017)	8.2	TREC-6	CoVe (McCann et al., 2017)	4.2
	oh-LSTM (Johnson and Zhang, 2016)	5.9		TBCNN (Mou et al., 2015)	4.0
	Virtual (Miyato et al., 2016)	5.9		LSTM-CNN (Zhou et al., 2016)	3.9
	ULMFiT (ours)	<b>4.6</b>		ULMFiT (ours)	<b>3.6</b>

# ULMFiT





# Deep Contextualized Word Representations

Peters et. al (2018)

# Deep Contextualized Word Representations

Peters et. al (2018)

- NAACL 2018 Best Paper Award

# Deep Contextualized Word Representations

Peters et. al (2018)

- NAACL 2018 Best Paper Award
- **E**MBEDDINGS FROM **L**ANGUAGE **M**ODELS (ELMo)
  - aka the OG NLP Muppet



# ELMo

## Deep contextualized word representations

**Matthew E. Peters<sup>†</sup>, Mark Neumann<sup>†</sup>, Mohit Iyyer<sup>†</sup>, Matt Gardner<sup>†</sup>,**  
{matthewp, markn, mohiti, mattg}@allenai.org

**Christopher Clark\*, Kenton Lee\*, Luke Zettlemoyer<sup>†\*</sup>**  
{csquared, kentonl, lsz}@cs.washington.edu

<sup>†</sup>Allen Institute for Artificial Intelligence

\*Paul G. Allen School of Computer Science & Engineering, University of Washington

### Abstract

We introduce a new type of *deep contextualized* word representation that models both (1) complex characteristics of word use (e.g., syntax and semantics), and (2) how these uses vary across linguistic contexts (i.e., to model polysemy). Our word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pre-trained on a large text corpus. We show that these representations can be easily added to existing models and significantly improve the state of the art across six challenging NLP problems, including question answering, textual entailment and sentiment analysis. We also present an analysis showing that exposing the deep internals of the pre-trained network is crucial, allowing downstream models to mix different types of semi-supervision signals.

guage model (LM) objective on a large text corpus. For this reason, we call them ELMo (Embeddings from Language Models) representations. Unlike previous approaches for learning contextualized word vectors (Peters et al., 2017; McCann et al., 2017), ELMo representations are deep, in the sense that they are a function of all of the internal layers of the biLM. More specifically, we learn a linear combination of the vectors stacked above each input word for each end task, which markedly improves performance over just using the top LSTM layer.

Combining the internal states in this manner allows for very rich word representations. Using intrinsic evaluations, we show that the higher-level LSTM states capture context-dependent aspects of word meaning (e.g., they can be used without modification to perform well on supervised



# ELMo

## Deep contextualized word representations

**Matthew E. Peters<sup>†</sup>, Mark Neumann<sup>†</sup>, Mohit Iyyer<sup>†</sup>, Matt Gardner<sup>†</sup>,**  
{matthewp, markn, mohiti, mattg}@allenai.org

**Christopher Clark\*, Kenton Lee\*, Luke Zettlemoyer<sup>†\*</sup>**  
{csquared, kentonl, lsz}@cs.washington.edu

<sup>†</sup>Allen Institute for Artificial Intelligence

\*Paul G. Allen School of Computer Science & Engineering, University of Washington

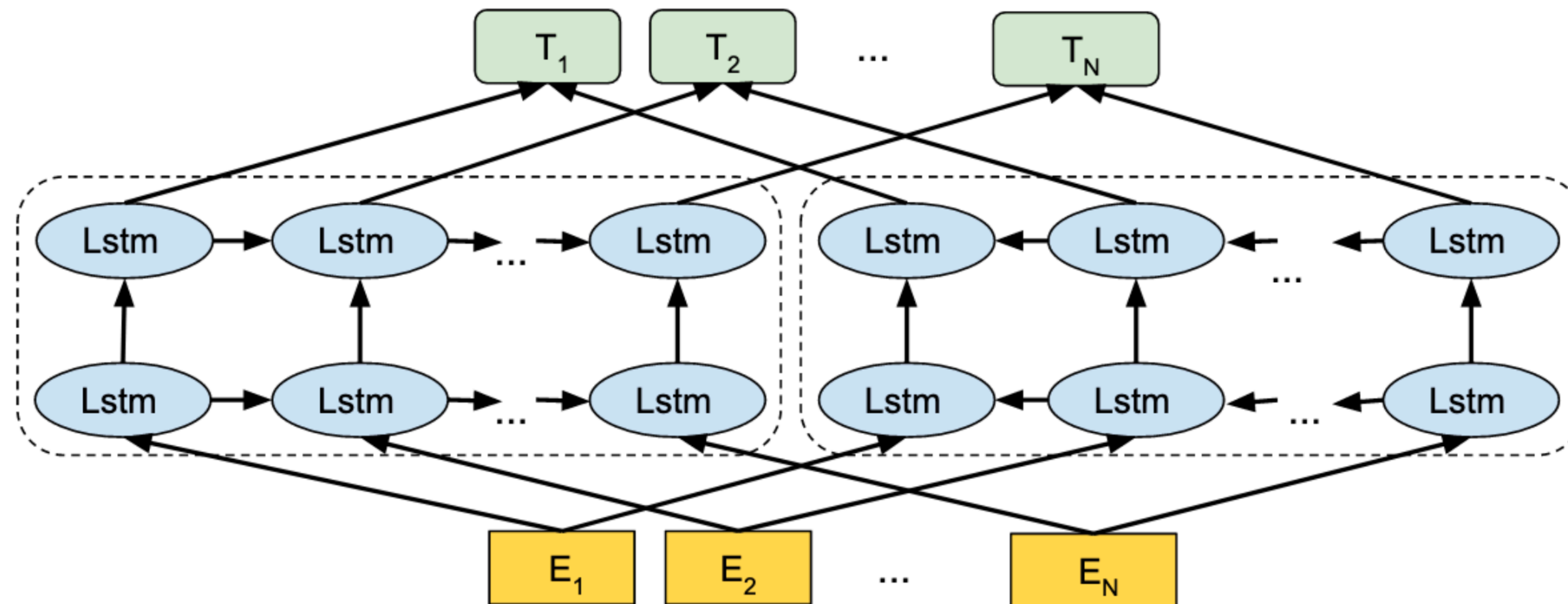
### Abstract

We introduce a new type of *deep contextualized* word representation that models both (1) complex characteristics of word use (e.g., syntax and semantics), and (2) how these uses vary across linguistic contexts (i.e., to model polysemy). Our word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pre-trained on a large text corpus. We show that these representations can be easily added to existing models and significantly improve the state of the art across six challenging NLP problems, including question answering, textual entailment and sentiment analysis. We also present an analysis showing that exposing the deep internals of the pre-trained network is crucial, allowing downstream models to mix different types of semi-supervision signals.

guage model (LM) objective on a large text corpus. For this reason, we call them ELMo (Embeddings from Language Models) representations. Unlike previous approaches for learning contextualized word vectors (Peters et al., 2017; McCann et al., 2017), ELMo representations are deep, in the sense that they are a function of all of the internal layers of the biLM. More specifically, we learn a linear combination of the vectors stacked above each input word for each end task, which markedly improves performance over just using the top LSTM layer.

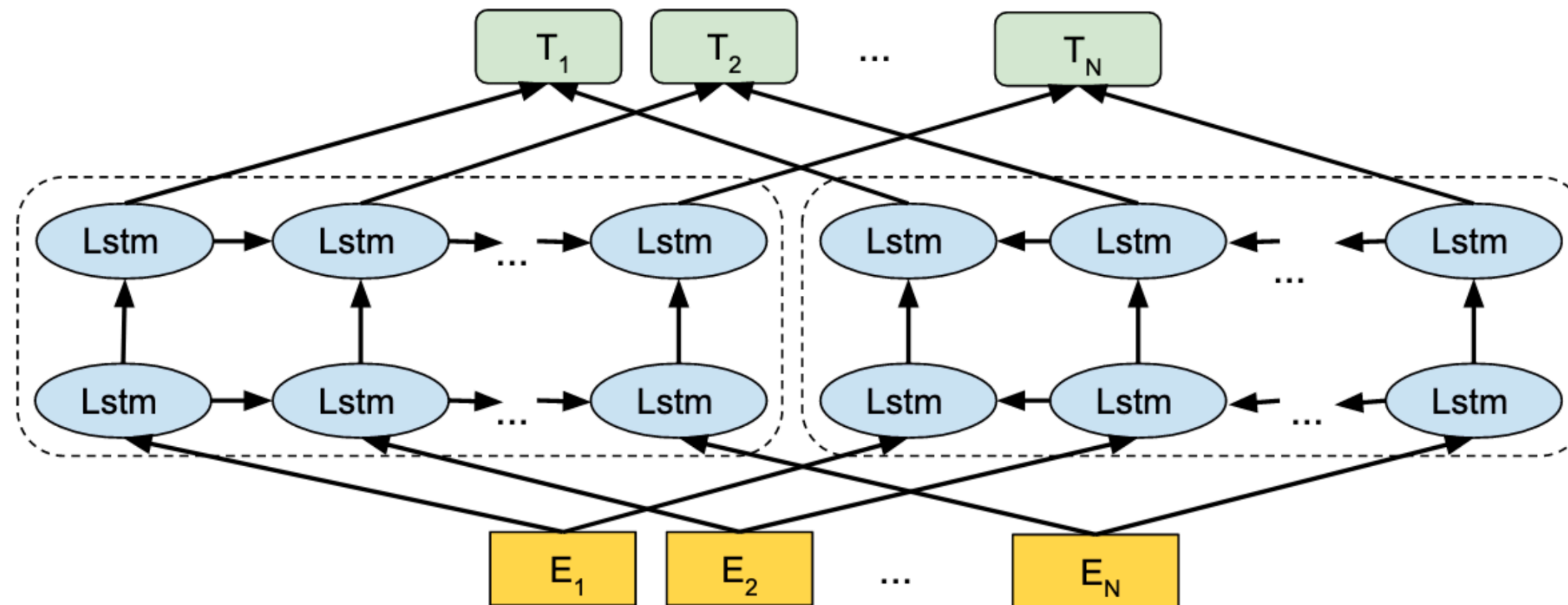
Combining the internal states in this manner allows for very rich word representations. Using intrinsic evaluations, we show that the higher-level LSTM states capture context-dependent aspects of word meaning (e.g., they can be used without modification to perform well on supervised

# ELMo Model



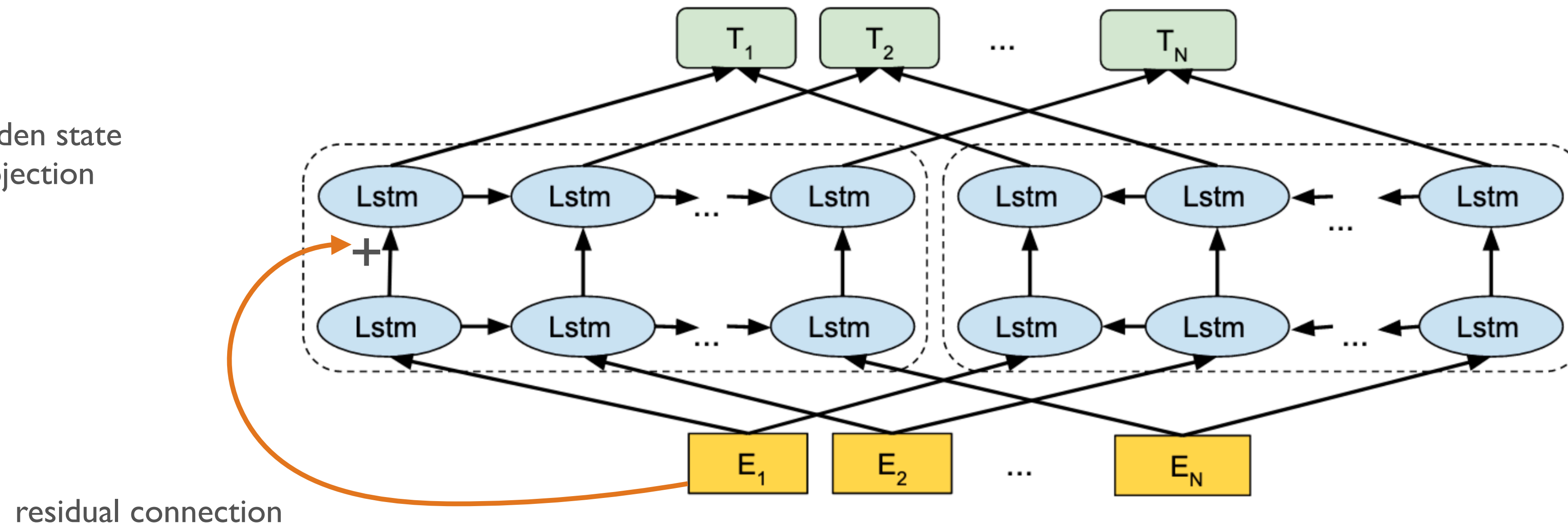
# ELMo Model

4096-d hidden state  
512d projection



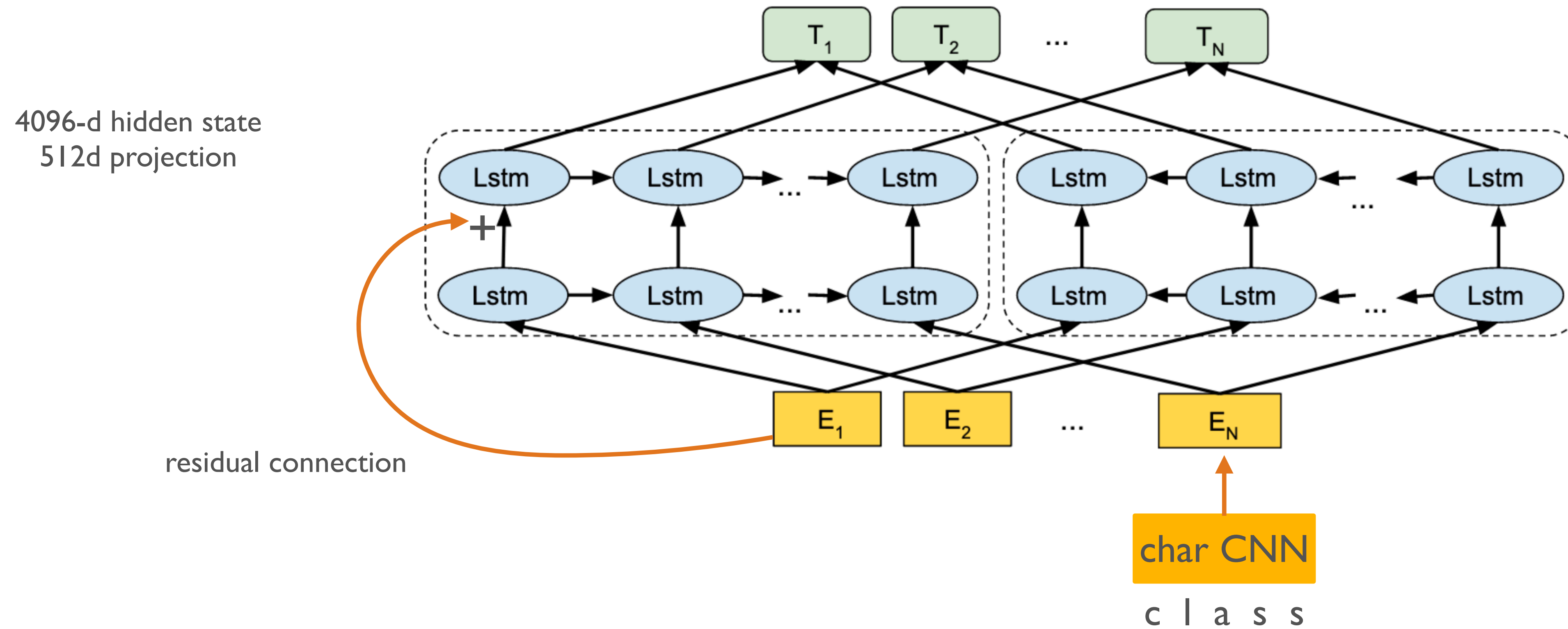
# ELMo Model

4096-d hidden state  
512d projection



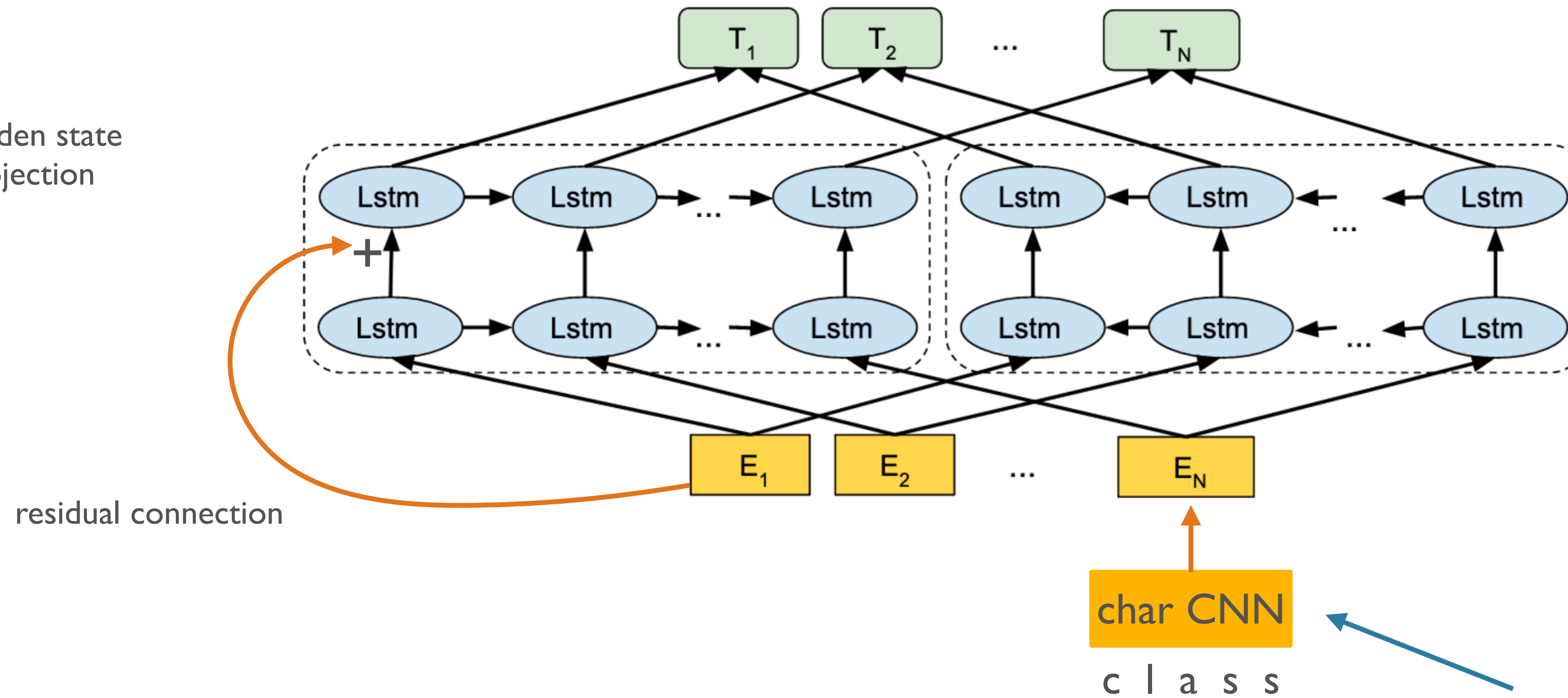


# ELMo Model



# ELMo Model

4096-d hidden state  
512d projection



residual connection

char CNN  
c l a s s

Helps with rare / new words (no OOV)

# ELMo Training

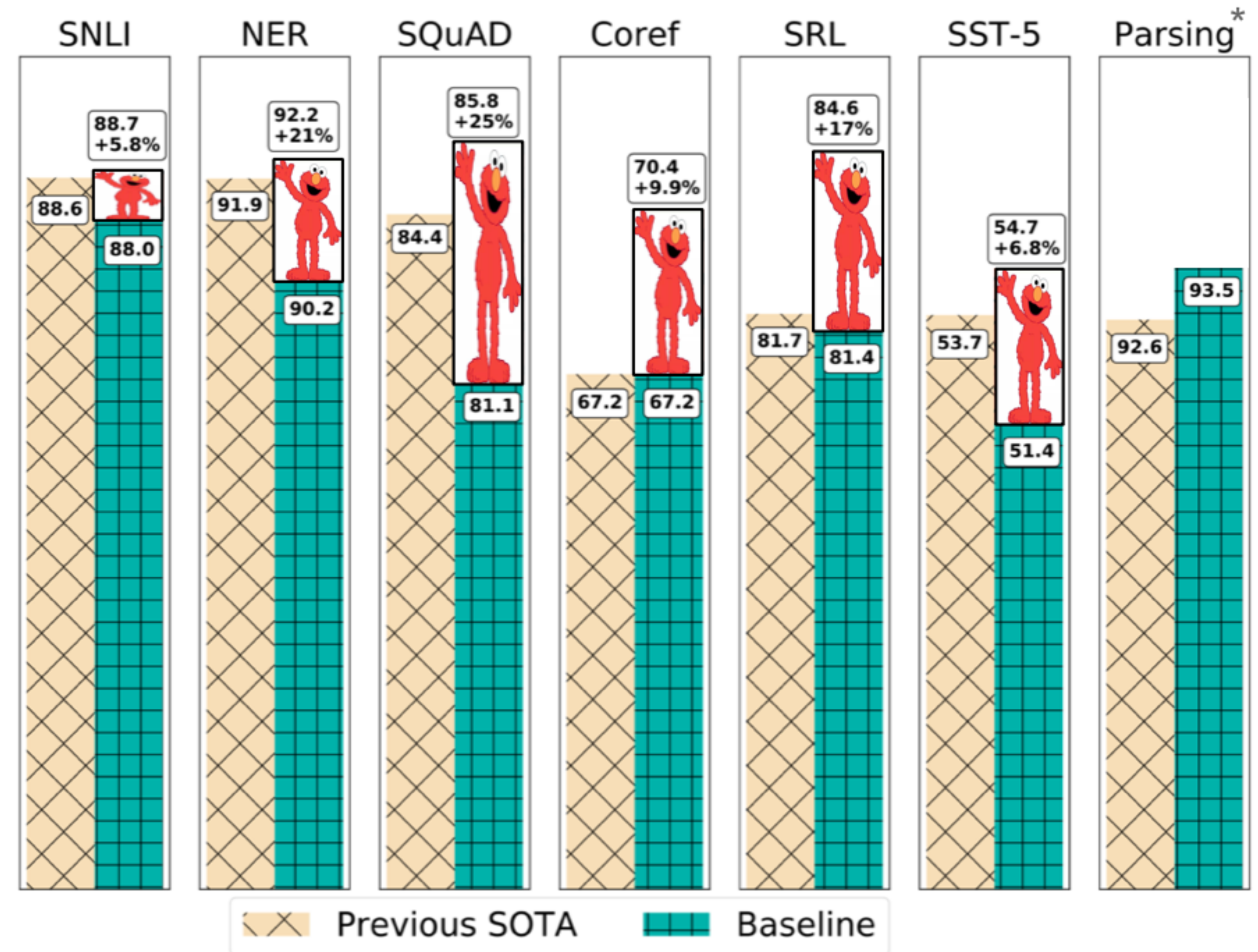
- 10 epochs on 1B Word Benchmark
- NB: not SOTA perplexity even at time of publishing
  - See “Exploring the Limits of Language Modeling” paper
- Regularization:
  - Dropout
  - L2 norm

# Deep Contextualized Word Representations

Peters et. al (2018)

- Used in place of other embeddings on multiple tasks:

SQuAD = [Stanford Question Answering Dataset](#)  
SNLI = [Stanford Natural Language Inference Corpus](#)  
SST-5 = [Stanford Sentiment Treebank](#)



\*Kitaev and Klein, ACL 2018 (see also Joshi et al., ACL 2018)



# Global vs. Contextual Word Vectors

- Global vectors: one vector per word-type
  - E.g. word2vec, GloVe
  - No difference between e.g. “play” as a verb, noun, or its different senses
- Contextual vectors: one vector per word-occurrence
  - “We saw a really great **play** last week.”
  - “Do you want to **play** basketball tomorrow?”
  - Each *occurrence* gets its own vector representation.

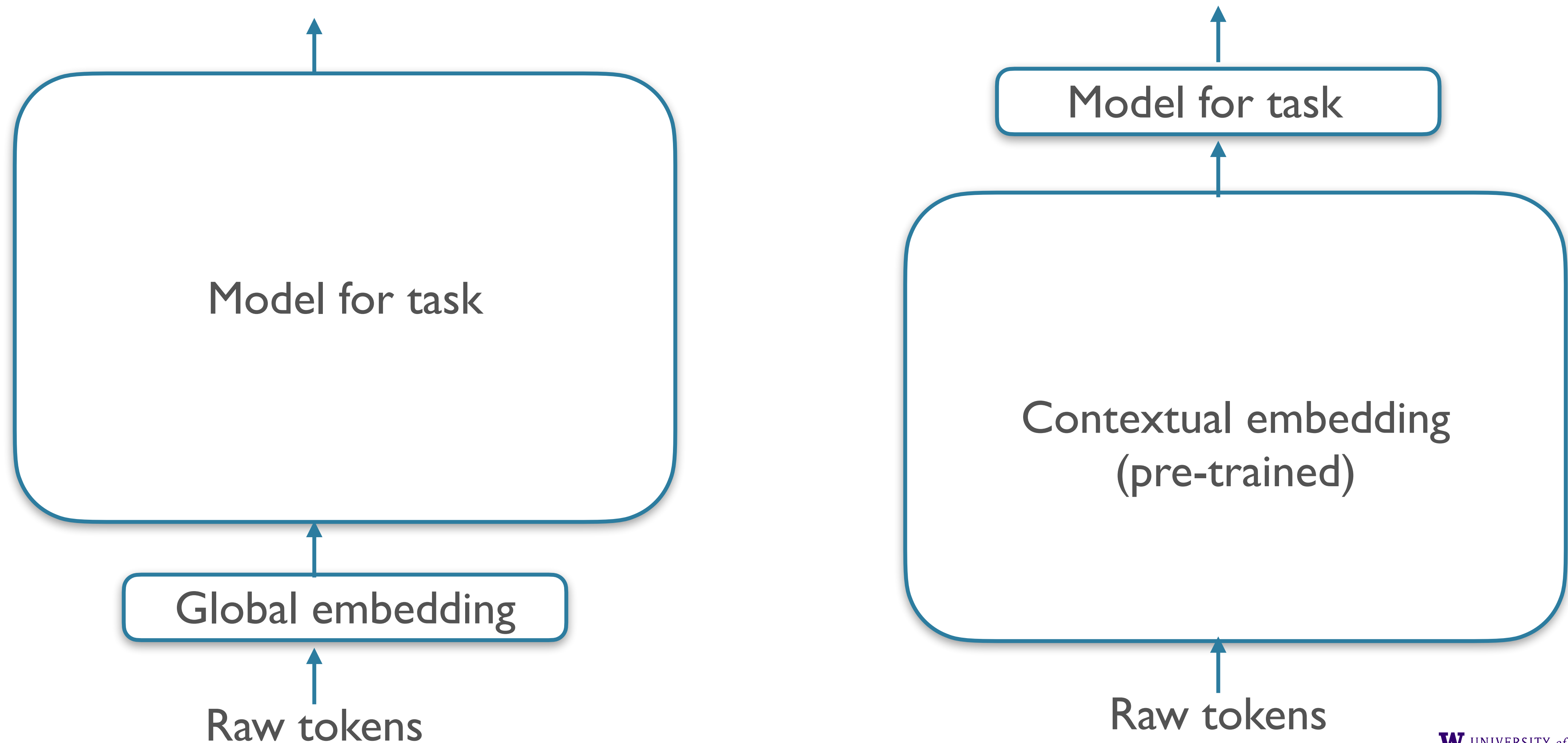
# Deep Contextualized Word Representations

Peters et. al (2018)

- Comparison to GloVe:

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular <b>play</b> on Alusik's grounder...	Kieffer, the only junior in the group, was commended for his ability to hit in the clutch, as well as his all-round excellent <b>play</b> .
	Olivia De Havilland signed to do a Broadway <b>play</b> for Garson...	...they were actors who had been handed fat roles in a successful <b>play</b> , and had talent enough to fill the roles competently, with nice understatement.

# Shallow vs Deep Pre-training





# Pre-trained Transformers

# Paralellizability + Scale

- ULMFiT + ELMo:
  - Demonstrate the value of LM pre-training + transfer learning
    - Noted that there are “virtually unlimited” quantities of data for LM
  - Used bi-LSTMs for the LM
- Concurrently: Transformer paper introduced
- Triggered an explosion in the pretraining approach
  - Lack of recurrence → paralellizability → scaling up both model size and dataset size

# Pre-trained Transformers: Encoder-only

# BERT: Bidirectional Encoder Representations from Transformers

[Devlin et al NAACL 2019](#)



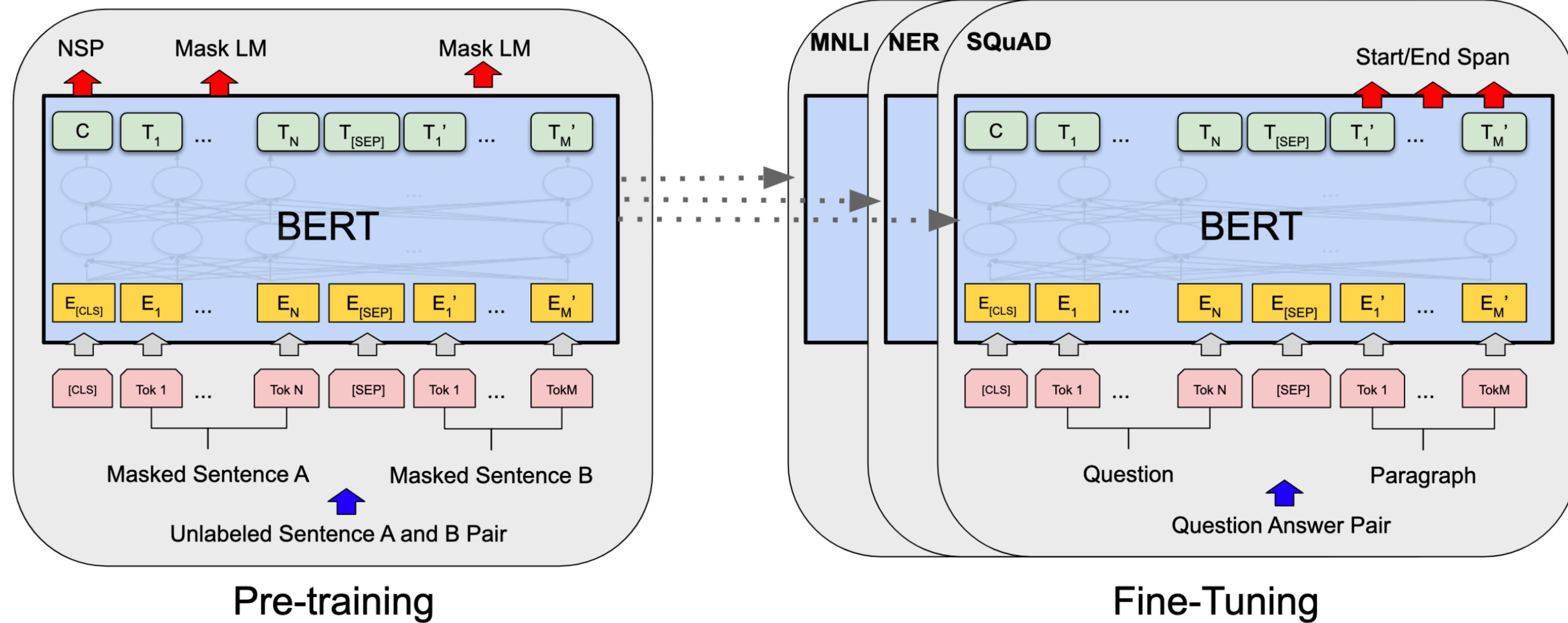
# Overview

- Encoder Representations from Transformers: ✓
- Bidirectional: .....?
  - BiLSTM (ELMo): left-to-right and right-to-left
  - Self-attention: every token can see every other
  - *adirectional* probably a better term
- How do you treat the encoder as an LM computing  $P(w_t | w_{t-1}, w_{t-2}, \dots, w_1)$ ?
  - You don't: modify the task instead

# Masked Language Modeling

- Language modeling: next word prediction
- *Masked* Language Modeling (a.k.a. cloze task): fill-in-the-blank
  - Nancy Pelosi sent the articles of \_\_\_\_\_ to the Senate.
  - Seattle \_\_\_\_\_ some snow, so UW was delayed due to \_\_\_\_\_ roads.
- i.e.  $P(w_t | w_{t+k}, w_{t+(k-1)}, \dots, w_{t+1}, w_{t-1}, \dots, w_{t-(m+1)}, w_{t-m})$ 
  - (very similar to CBOW: continuous bag of words from word2vec)
- Auxiliary training task: next sentence prediction.
  - Given sentences A and B, binary classification: did B follow A in the corpus or not?

# Schematically





# Some details

# Some details

- BASE model:
  - 12 Transformer Blocks
  - Hidden vector size: 768
  - Attention heads / layer: 12
  - Total parameters: 110M

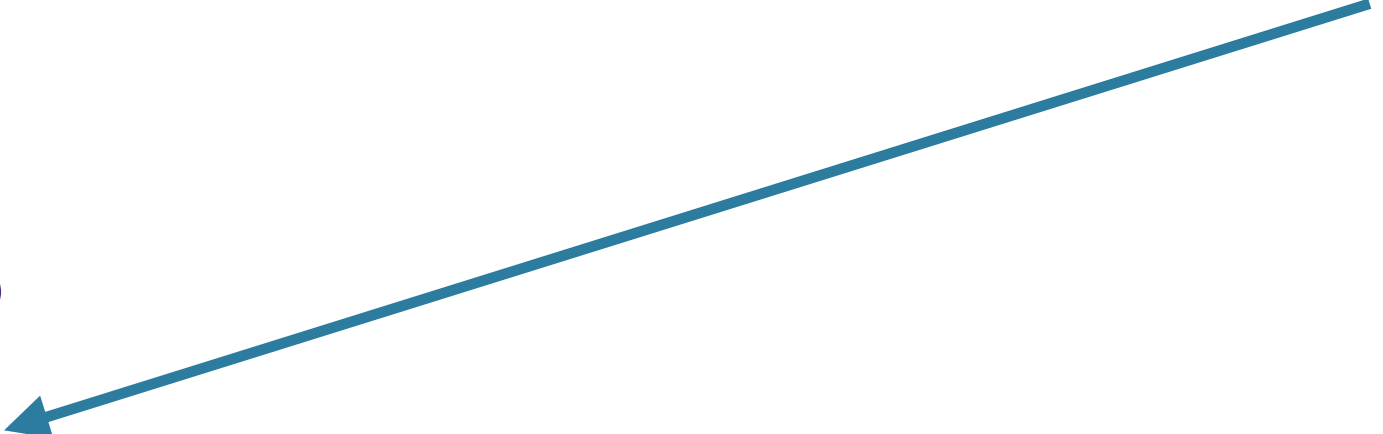
# Some details

- BASE model:
  - 12 Transformer Blocks
  - Hidden vector size: 768
  - Attention heads / layer: 12
  - Total parameters: 110M
- LARGE model:
  - 24 Transformer Blocks
  - Hidden vector size: 1024
  - Attention heads / layer: 16
  - Total parameters: 340M

# Some details

- BASE model:
  - 12 Transformer Blocks
  - Hidden vector size: 768
  - Attention heads / layer: 12
  - Total parameters: 110M
- LARGE model:
  - 24 Transformer Blocks
  - Hidden vector size: 1024
  - Attention heads / layer: 16
  - Total parameters: 340M

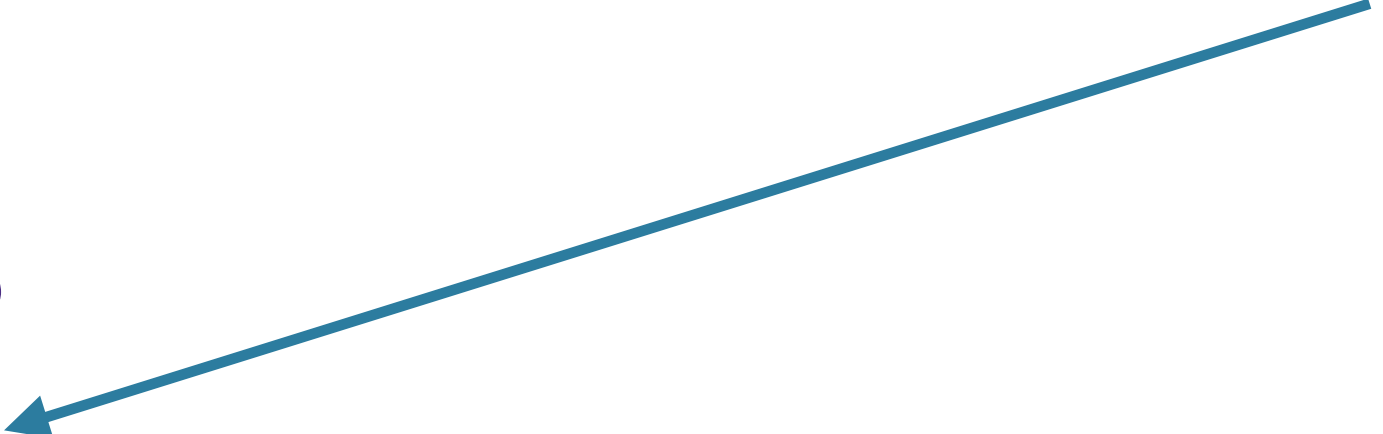
this is the first work to demonstrate convincingly that scaling to extreme model sizes also leads to large improvements on very small scale tasks, provided that the model has been sufficiently pre-trained. [Peters et al. \(2018b\)](#) presented



# Some details

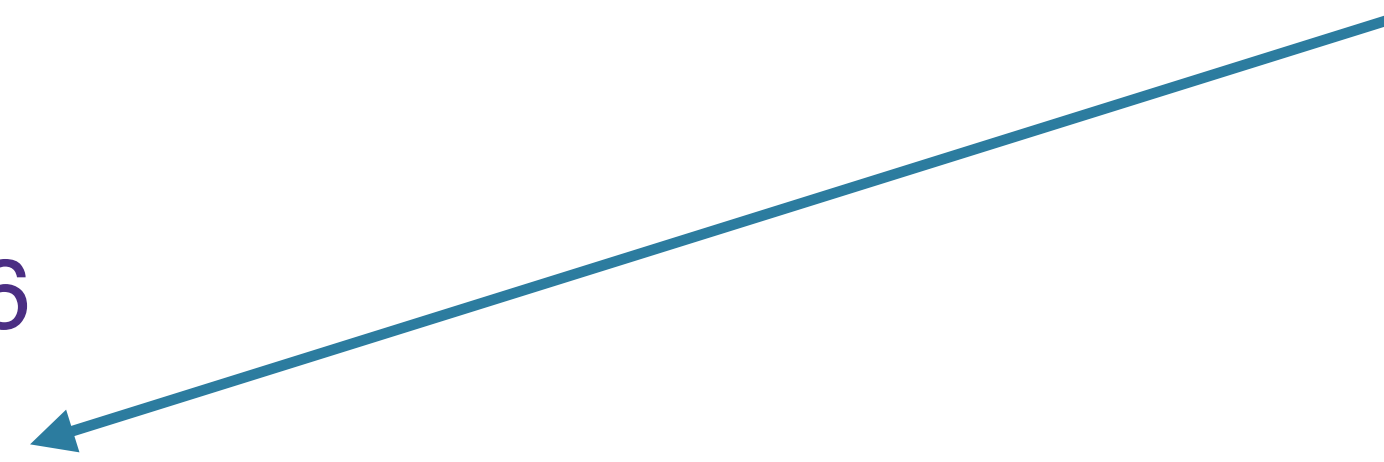
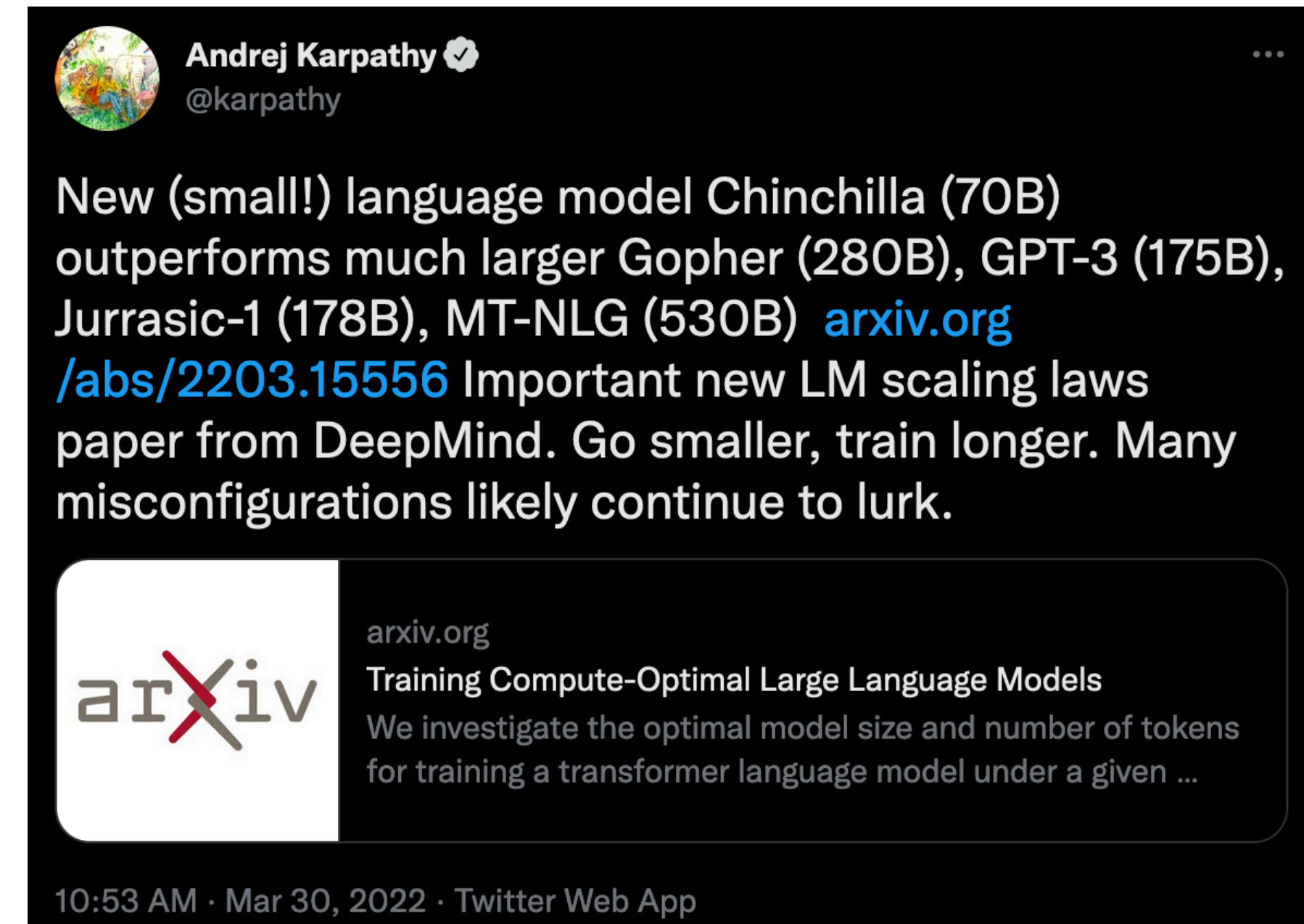
- BASE model:
  - 12 Transformer Blocks
  - Hidden vector size: 768
  - Attention heads / layer: 12
  - Total parameters: 110M
- LARGE model:
  - 24 Transformer Blocks
  - Hidden vector size: 1024
  - Attention heads / layer: 16
  - Total parameters: 340M

this is the first work to demonstrate convincingly that scaling to **extreme model sizes** also leads to large improvements on very small scale tasks, provided that the model has been sufficiently pre-trained. [Peters et al. \(2018b\)](#) presented



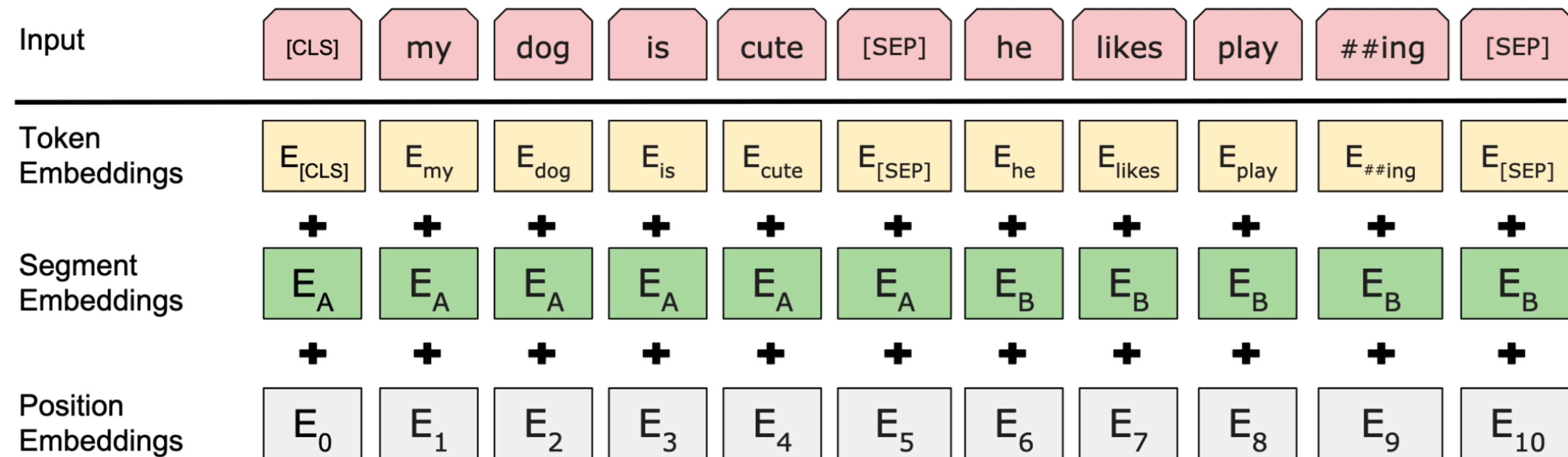
# Some details

- BASE model:
  - 12 Transformer Blocks
  - Hidden vector size: 768
  - Attention heads / layer: 12
  - Total parameters: 110M
- LARGE model:
  - 24 Transformer Blocks
  - Hidden vector size: 1024
  - Attention heads / layer: 16
  - Total parameters: 340M



convinc-  
es) also  
leads to large improvements on very small scale  
tasks, provided that the model has been suffi-  
ciently pre-trained. Peters et al. (2018b) presented

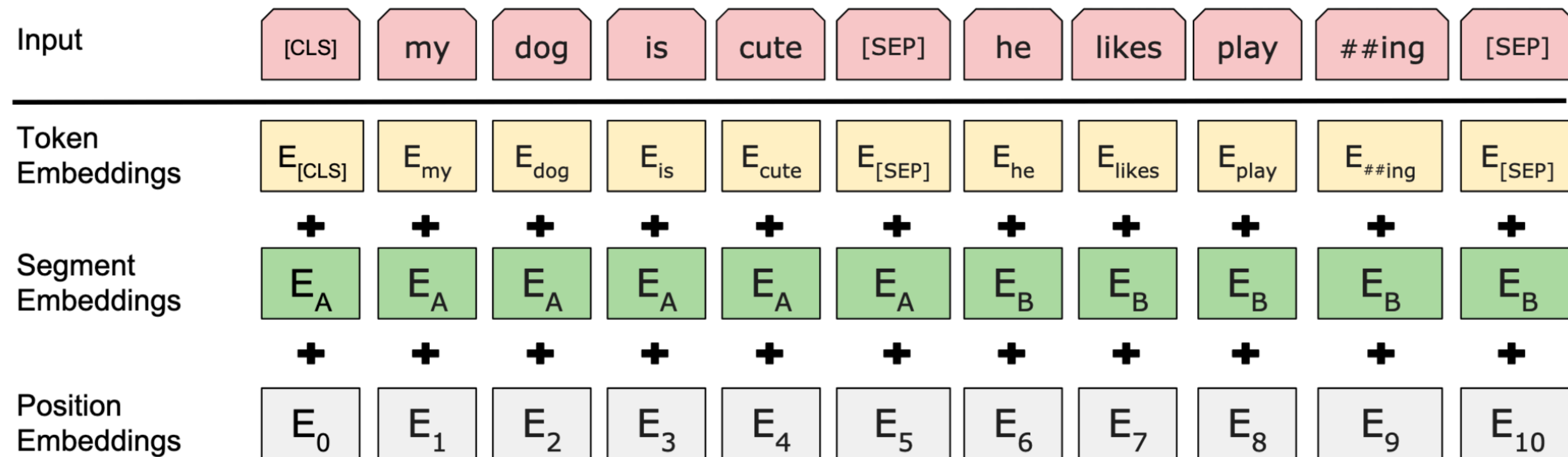
# Input Representation





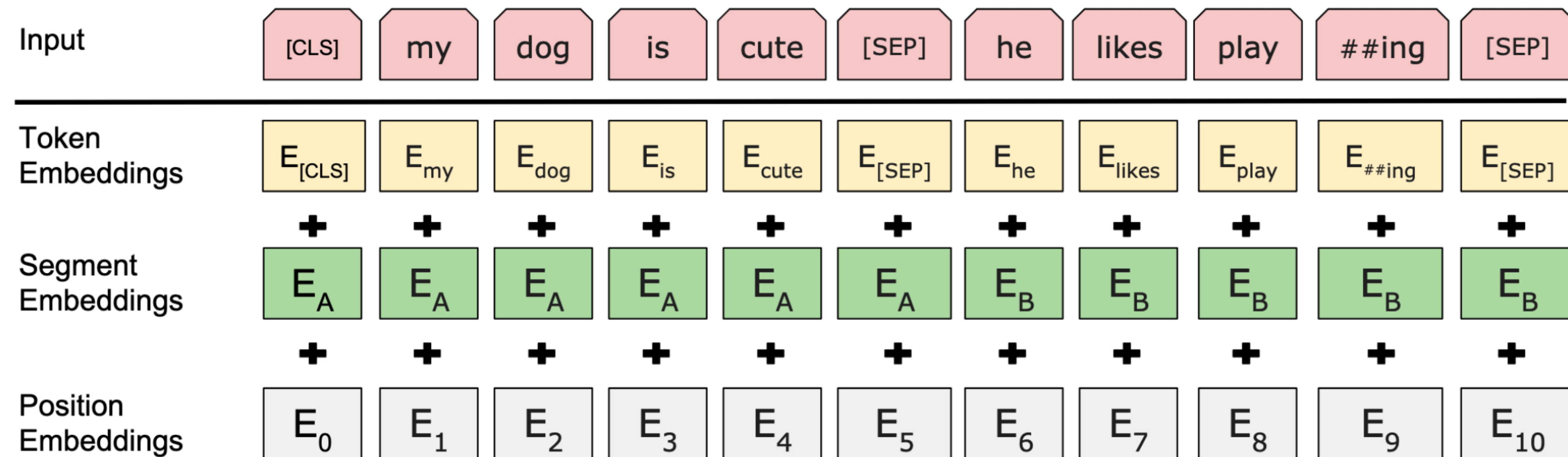
# Input Representation

- [CLS], [SEP]: special tokens



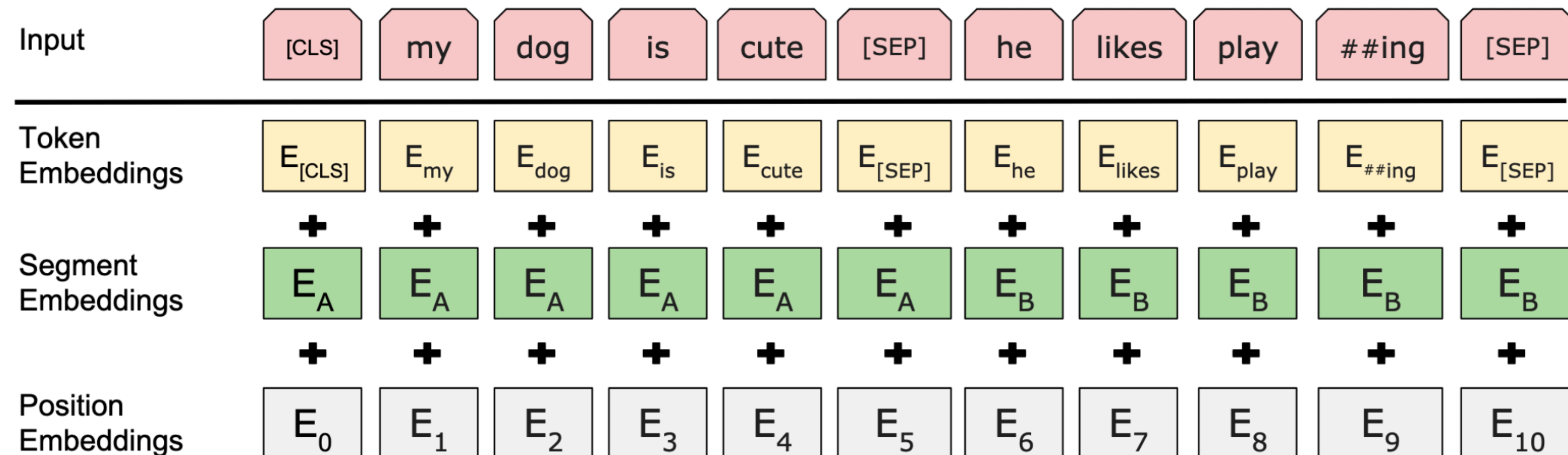
# Input Representation

- [CLS], [SEP]: special tokens
- Segment: is this a token from sentence A or B?



# Input Representation

- [CLS], [SEP]: special tokens
- Segment: is this a token from sentence A or B?
- Position embeddings: provide position in sequence (*learned* in this case, not fixed)



# Training Details

# Training Details

- BooksCorpus (800M words) + Wikipedia (2.5B)

# Training Details

- BooksCorpus (800M words) + Wikipedia (2.5B)
- Masking the input text. 15% of all tokens are chosen. Then:
  - 80% of the time: replaced by designated '[MASK]' token
  - 10% of the time: replaced by random token
  - 10% of the time: unchanged

# Training Details

- BooksCorpus (800M words) + Wikipedia (2.5B)
- Masking the input text. 15% of all tokens are chosen. Then:
  - 80% of the time: replaced by designated '[MASK]' token
  - 10% of the time: replaced by random token
  - 10% of the time: unchanged
- Loss is cross-entropy of the prediction at the masked positions.



# Training Details

- BooksCorpus (800M words) + Wikipedia (2.5B)
- Masking the input text. 15% of all tokens are chosen. Then:
  - 80% of the time: replaced by designated '[MASK]' token
  - 10% of the time: replaced by random token
  - 10% of the time: unchanged
- Loss is cross-entropy of the prediction at the masked positions.
- Max seq length: 128 tokens for first 90%, 512 tokens for final 10%

# Training Details

- BooksCorpus (800M words) + Wikipedia (2.5B)
- Masking the input text. 15% of all tokens are chosen. Then:
  - 80% of the time: replaced by designated '[MASK]' token
  - 10% of the time: replaced by random token
  - 10% of the time: unchanged
- Loss is cross-entropy of the prediction at the masked positions.
- Max seq length: 128 tokens for first 90%, 512 tokens for final 10%
- 1M training steps, batch size 256 = 4 days on 4 or 16 TPUs

# Initial Results

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT <sub>BASE</sub>	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT <sub>LARGE</sub>	<b>86.7/85.9</b>	<b>72.1</b>	<b>92.7</b>	<b>94.9</b>	<b>60.5</b>	<b>86.5</b>	<b>89.3</b>	<b>70.1</b>	<b>82.1</b>

# Ablations

- Helpfulness of depth not a given (depth doesn't help ELMo)
  - Possibly a difference between fine-tuning vs. feature extraction
  - Many more variations to explore

Hyperparams				Dev Set Accuracy		
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

Tasks	Dev Set				
	MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
BERT <sub>BASE</sub>	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

# Other Prominent Encoders

- RoBERTa: robustly optimized BERT approach
  - BERT was very *under-trained*: give it more data, train it longer
  - (keep model the same otherwise)
  - Good default encoder
- ELECTRA: replace Masked Language Modeling with “replaced token detection”, trains just as well with much less data
- SpanBERT: mask out entire *spans* instead of single tokens

# Limitation of Encoders

- No left-to-right modeling assumption
- Good for NLU (understanding/comprehension) tasks
- Does not straightforwardly *generate* text

# Pre-training + Fine-tuning Paradigm

Ling 575j: Deep Learning for NLP

C.M. Downey

Spring 2023



# Today's Plan

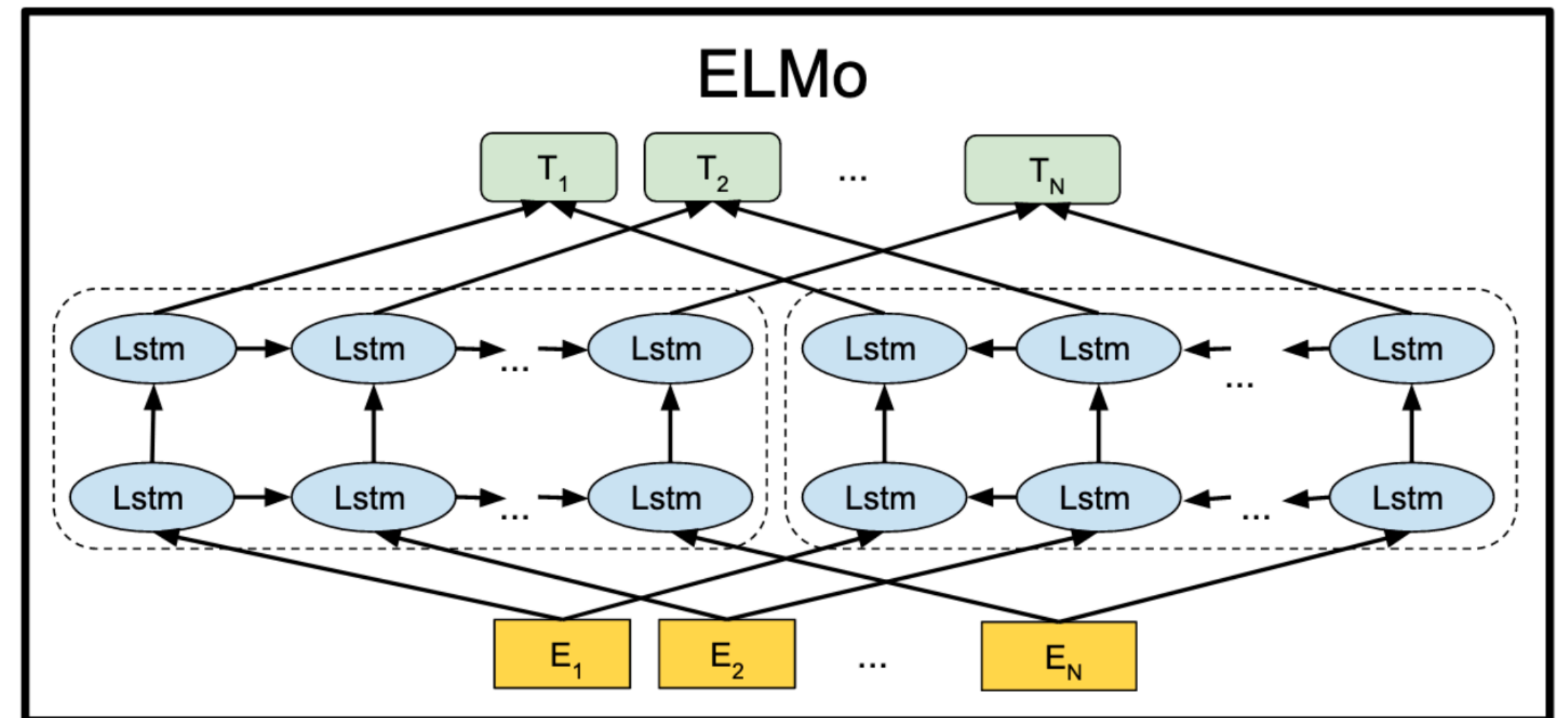
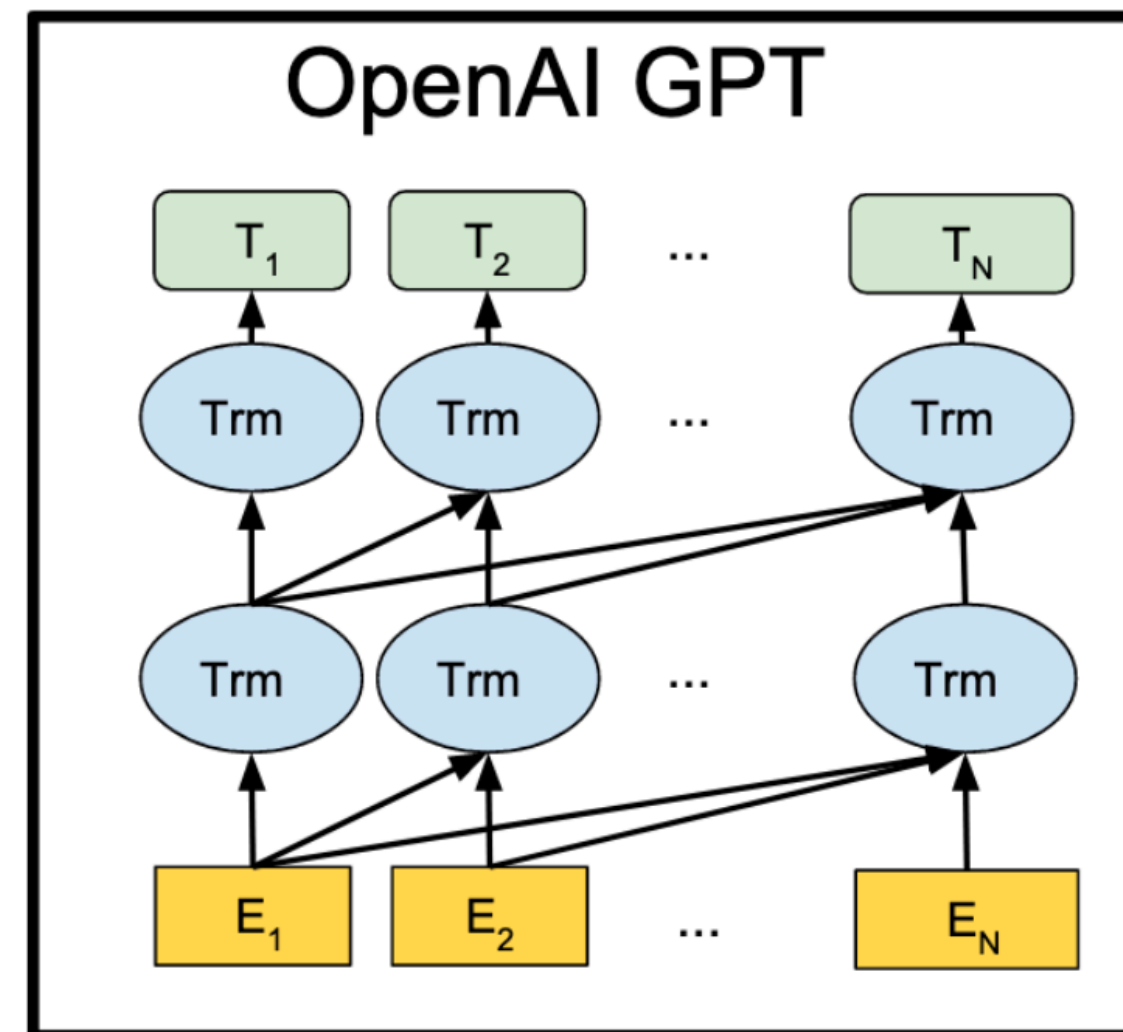
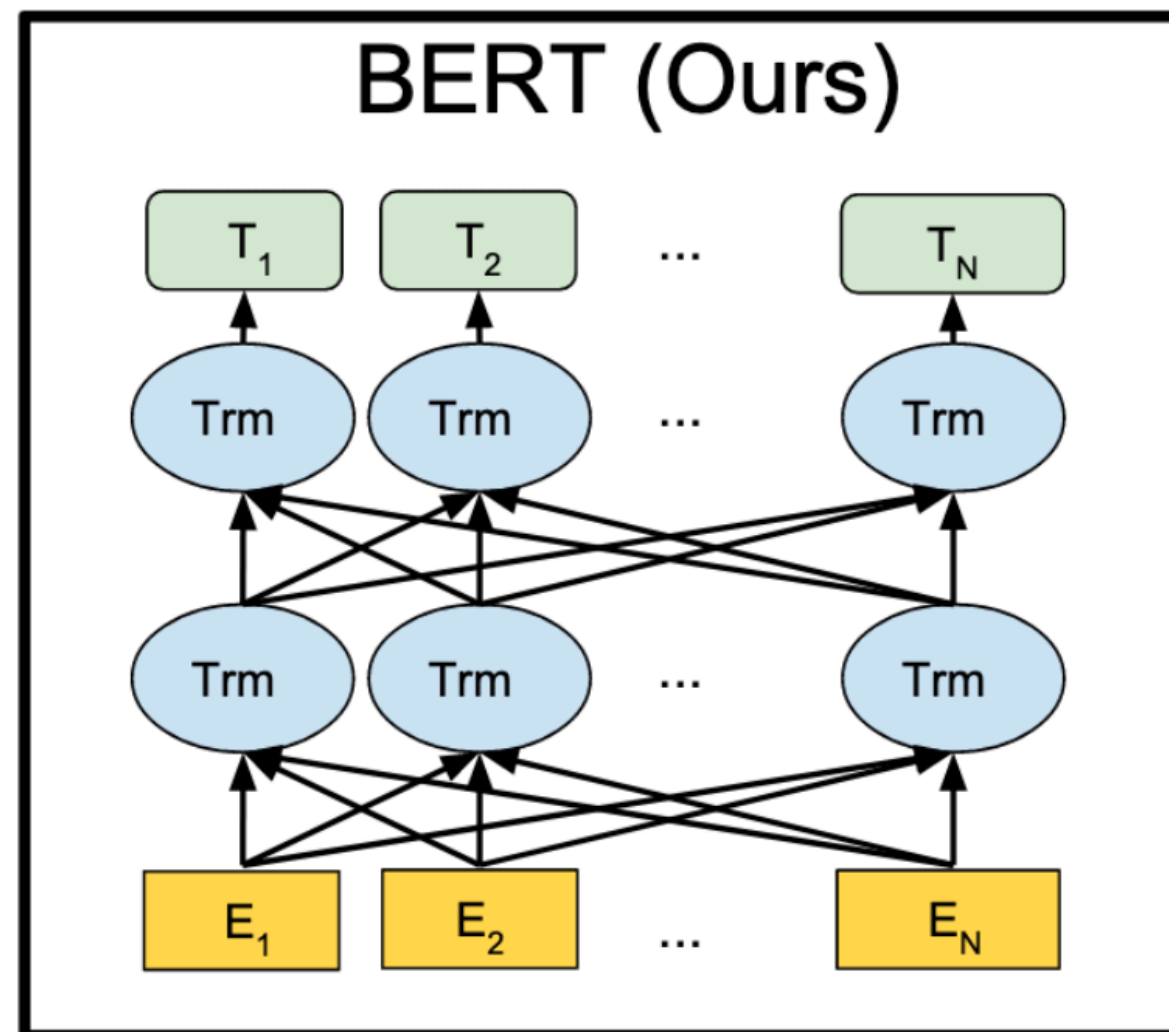
- Pre-training + fine-tuning, cont.
  - Last time:
    - Recurrent (ULMFIT, ELMo)
    - Encoder-only (BERT)
  - Decoder-only
  - Encoder-decoder
  - Risks
  - Accessing / using pre-trained LMs
- Next time: interpreting and analyzing pre-trained language models. What kind of linguistic and non-linguistic information do they acquire?

# Pre-trained Transformers: Decoder-only

# GPT(2,3,4,ChatGPT)

- Generative Pre-training
  - Radford et al [2018](#); [2019](#) (GPT2); Brown et al [2020](#) (GPT3)
- Uses Transformer *decoder* instead of *encoder*
  - “Self”-attention: masked so that only can attend to *previous* tokens
  - Pure LM training objective (GPT2, GPT3 only)
  - Can be used for text generation
- GPT: same params as BERT-BASE; GPT2 much bigger; GPT3 muuuuuch bigger (175B params)
- Training data: crawled from outbound Reddit links w/ >3 karma, not public

# Comparison



[Source: BERT paper](#)

# Samples from GPT2 (Cherry-picked)

SYSTEM PROMPT (HUMAN-WRITTEN)	<i>A train carriage containing controlled nuclear materials was stolen in Cincinnati today. Its whereabouts are unknown.</i>
MODEL COMPLETION (MACHINE-WRITTEN, FIRST TRY)	<p>The incident occurred on the downtown train line, which runs from Covington and Ashland stations.</p> <p>In an email to Ohio news outlets, the U.S. Department of Energy said it is working with the Federal Railroad Administration to find the thief.</p> <p>“The theft of this nuclear material will have significant negative consequences on public and environmental health, our workforce and the economy of our nation,” said Tom Hicks, the U.S. Energy Secretary, in a statement. “Our top priority is to secure the theft and ensure it doesn’t happen again.”</p> <p>The stolen material was taken from the University of Cincinnati’s Research Triangle Park nuclear research site, according to a news release from Department officials.</p> <p>The Nuclear Regulatory Commission did not immediately release any information.</p>



# Samples from GPT2 (Cherry-picked)

SYSTEM PROMPT  
(HUMAN-WRITTEN)

*Recycling is good for the world.*

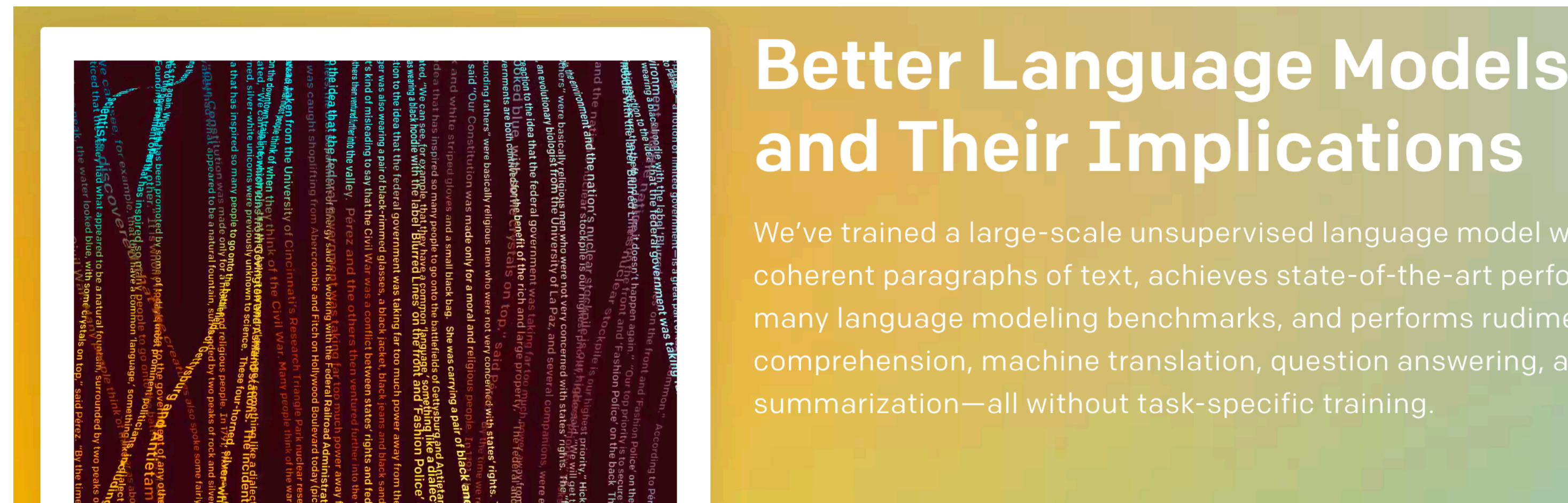
*NO! YOU COULD NOT BE MORE WRONG!!*

MODEL  
COMPLETION  
(MACHINE-  
WRITTEN, 25  
TRIES)

Recycling is NOT good for the world. It is bad for the environment, it is bad for our health, and it is bad for our economy. I'm not kidding. Recycling is not good for the environment. It is destructive to the earth and it is a major contributor to global warming. Recycling is not good for our health. It contributes to obesity and diseases like heart disease and cancer. Recycling is bad for our economy. It increases the cost of a product, and in turn, the price of everything that is made with that product. Recycling is not good for our nation. We pay a tremendous price for the privilege of having the world's most advanced and efficient recycling system. Recycling is a huge, colossal waste of time, energy, money, and resources. And THAT is why we need to get back to basics and get back to basics in our recycling efforts. One of the best ways to start is to look at the process of creating a paper product. When you make a paper product, it is basically a long chain of materials. Everything from the raw materials (wood, cardboard, paper, etc.), to the reagents (dyes, solvents, etc.) to the printing equipment (chemicals, glue, paper, ink, etc.), to the packaging, to the packaging materials (mercury, chemicals, etc.) to the processing equipment (heating, cooling, etc.), to the packaging materials, to the packaging materials that are shipped overseas and to the packaging materials that are used in the United States. Each step along the way creates tons of waste that we constantly have to clean up. The process

# GPT2 Release Controversy

- Worry: it's so good, it could be used by malicious actors to make deep-fakes for text (incl. fake news, character assassination, ...)
- So the model was not released until a partially independent report on possible misuses (but has been now)





# Write With Transformer

- Prompt: **Studying computational linguistics is a good idea. There are many interesting problems.**
- However, there are many bad problems. The research into computational linguistics has not helped very much in the development of better solutions. One problem with computer science and machine learning is that, until recently, the problems they were supposed to solve were insufficiently well studied. Most of the people who are doing research in computer science (at least in my profession) have no understanding of how to design a particularly natural problem, for example. In the case of the reasoning problem above, most of the people I know who work on computer science or machine learning have no understanding of how to make computer programs work in ways which do not have unavoidable adverse consequences. In the best cases, computational linguistics was little help.

# Write With Transformer

- Prompt: **I didn't ride my bike to campus today because of the snowy weather.**
- There is a lot of construction and I wanted to have a few hours of skiing. All that snow is a real pain in the ass. Good thing there was an LRT planned for that area. See you guys on campus in a few days! Photo credit: Tapto@archivosos.gr – August 26, 2016: Windy Weather on the Mountain (photo credit: Diana Prat)  
From Diana Prat, via [tapto@archivosos.gr](mailto:tapto@archivosos.gr):  
Ever since the camp that I organized to protest the modernisation of the Barranco, my beautiful alma mater, you know, going in for a press conference, I was asking why the Italian government

# ...fastforward to ChatGPT



I didn't ride my bike to campus today because of the snowy weather.



It's understandable that you didn't ride your bike to campus today due to the snowy weather. Riding a bike in such conditions can be dangerous and difficult. It's important to prioritize your safety and well-being when making decisions like this. Is there anything else you would like to know or talk about?





# GPT3

- Same approach: pure Transformer decoder trained on LM
  - Scale: 175B params
  - Data size: ~500billion tokens, majority from filtered Common Crawl
- Few-shot “fine-tuning” paradigm:
  - Prompt with a few examples, ask to continue
  - *No parameter updates*
  - See “in-context learning” in a few slides

The three settings we explore for in-context learning

### Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 cheese => ..... ← prompt
```

### One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← example
3 cheese => ..... ← prompt
```

### Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← examples
3 peppermint => menthe poivrée ←
4 plush girafe => girafe peluche ←
5 cheese => ..... ← prompt
```

Traditional fine-tuning (not used for GPT-3)

### Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.



# GPT3 Few-Shot Results

	SuperGLUE Average	BoolQ Accuracy	CB Accuracy	CB F1	COPA Accuracy	RTE Accuracy
Fine-tuned SOTA	<b>89.0</b>	<b>91.0</b>	<b>96.9</b>	<b>93.9</b>	<b>94.8</b>	<b>92.5</b>
Fine-tuned BERT-Large	69.0	77.4	83.6	75.7	70.6	71.7
GPT-3 Few-Shot	71.8	76.4	75.6	52.0	92.0	69.0

	WiC Accuracy	WSC Accuracy	MultiRC Accuracy	MultiRC F1a	ReCoRD Accuracy	ReCoRD F1
Fine-tuned SOTA	<b>76.1</b>	<b>93.8</b>	<b>62.3</b>	<b>88.2</b>	<b>92.5</b>	<b>93.3</b>
Fine-tuned BERT-Large	69.6	64.6	24.1	70.0	71.3	72.0
GPT-3 Few-Shot	49.4	80.1	30.5	75.4	90.2	91.1

k=32

# Some follow-ups on GPT3

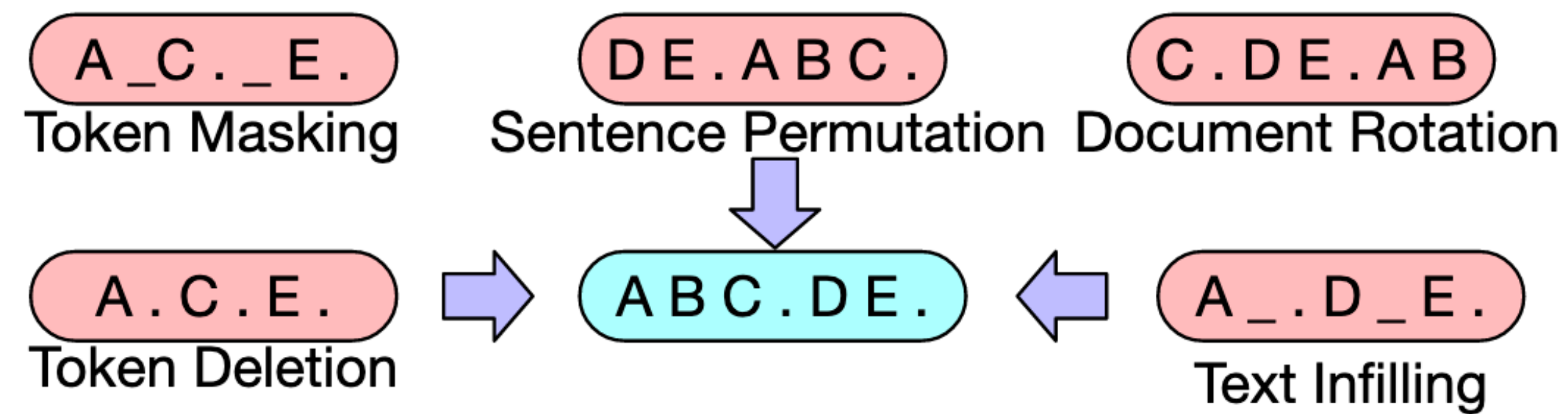
- Has ushered in a lot of work on “prompt tuning”: how to best engineer the prompts to produce the behavior that you want
  - Very useful survey paper/website on that front: <http://pretrain.nlpedia.ai/>
- Putting the “open” back in:
  - [EleutherAI](#): “A grassroots collective of researchers working to open source AI research.”
    - Reproduce GPT-like models + datasets in entirely open way
  - [OPT-175B](#): Meta’s recent open (incl logbook, etc) non-commercial replication

# Pretrained Transformers: Encoder-Decoder



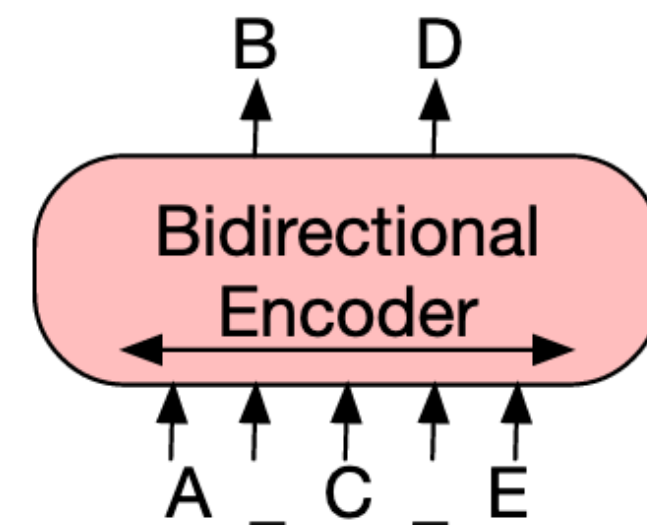
# BART

- Full Transformer, i.e. encoder-decoder transducer
  - Many composable transformations of raw text, presented to encoder
  - Goal of decoder is to reconstruct the original text

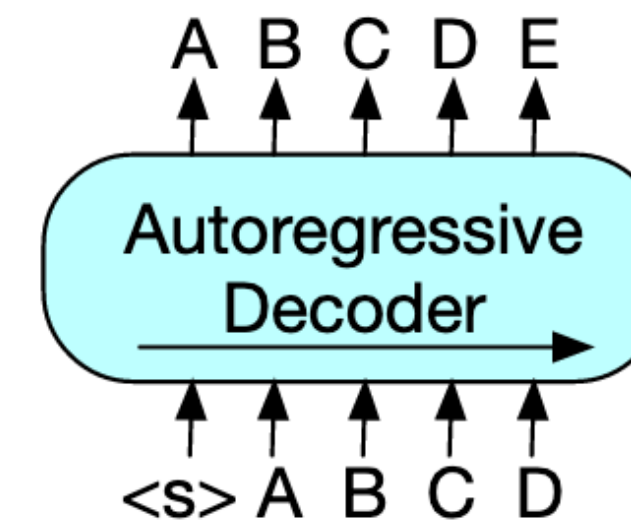


- Good for both discrimination and generation

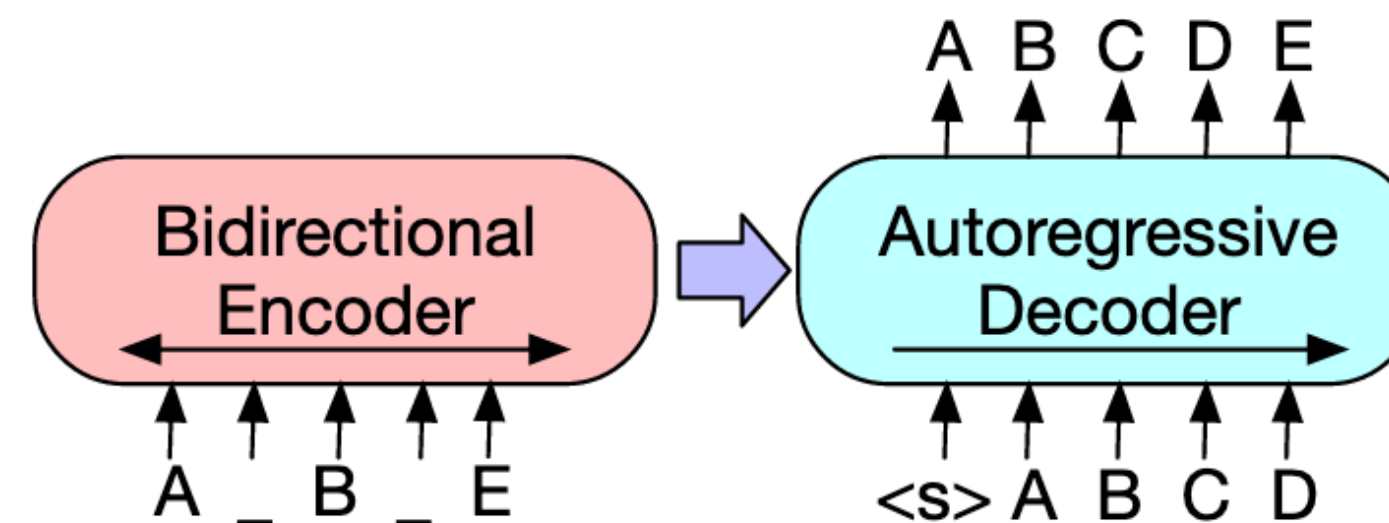
# High-level Overview



(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.



(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.



(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitrary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

# Comparison of Pre-training Objectives

<b>Model</b>	<b>SQuAD 1.1</b>	<b>MNLI</b>	<b>ELI5</b>	<b>XSum</b>	<b>ConvAI2</b>	<b>CNN/DM</b>
	F1	Acc	PPL	PPL	PPL	PPL
BERT Base (Devlin et al., 2019)	88.5	<b>84.3</b>	-	-	-	-
Masked Language Model	90.0	83.5	24.77	7.87	12.59	7.06
Masked Seq2seq Language Model	87.0	82.1	23.40	6.80	11.43	6.19
Permutated Language Model	76.7	80.1	<b>21.40</b>	7.00	11.51	6.56
Multitask Masked Language Model	89.1	83.7	24.03	7.69	12.23	6.96
	89.2	82.4	23.73	7.50	12.39	6.74
BART Base						
w/ Token Masking	90.4	84.1	25.05	7.08	11.73	6.10
w/ Token Deletion	90.4	84.1	24.61	6.90	11.46	5.87
w/ Text Infilling	<b>90.8</b>	84.0	24.26	<b>6.61</b>	<b>11.05</b>	5.83
w/ Document Rotation	77.2	75.3	53.69	17.14	19.87	10.59
w/ Sentence Shuffling	85.4	81.5	41.87	10.93	16.67	7.89
w/ Text Infilling + Sentence Shuffling	<b>90.8</b>	83.8	24.17	6.62	11.12	<b>5.41</b>



# Advantages of Encoder-Decoder Models

- “Best of both worlds”
  - On a par with RoBERTa on NLU / discrimination tasks
  - State-of-the-art on many generation tasks (e.g. summarization)
- Others:
  - [MASS](#)
  - [T5](#)
    - uses labeled data
    - “Unified” text-to-text format

Source Document (abbreviated)	BART Summary
The researchers examined three types of coral in reefs off the coast of Fiji ... The researchers found when fish were plentiful, they would eat algae and seaweed off the corals, which appeared to leave them more resistant to the bacterium <i>Vibrio coralliilyticus</i> , a bacterium associated with bleaching. The researchers suggested the algae, like warming temperatures, might render the corals’ chemical defenses less effective, and the fish were protecting the coral by removing the algae.	Fisheries off the coast of Fiji are protecting coral reefs from the effects of global warming, according to a study in the journal Science.
Sacoolas, who has immunity as a diplomat’s wife, was involved in a traffic collision ... Prime Minister Johnson was questioned about the case while speaking to the press at a hospital in Watford. He said, “I hope that Anne Sacoolas will come back ... if we can’t resolve it then of course I will be raising it myself personally with the White House.”	Boris Johnson has said he will raise the issue of US diplomat Anne Sacoolas’ diplomatic immunity with the White House.

# Multilingual Pre-training

- One other main dimension: *mono- vs multi-lingual* pre-training
  - Roughly: concatenate (in fancy way) corpora from many languages, then do the same kind of pre-training
  - Much more info in upcoming lecture on multilingual NLP

	Encoder-only	Decoder-only	Encoder-decoder
English-only *	BERT, RoBERTa, XLNet, ALBERT, ...	GPT-n	BART
Multilingual	mBERT, XLM(-R), ...	<u>BLOOM</u> , <u>XGLM</u>	mBART, MASS, mT5

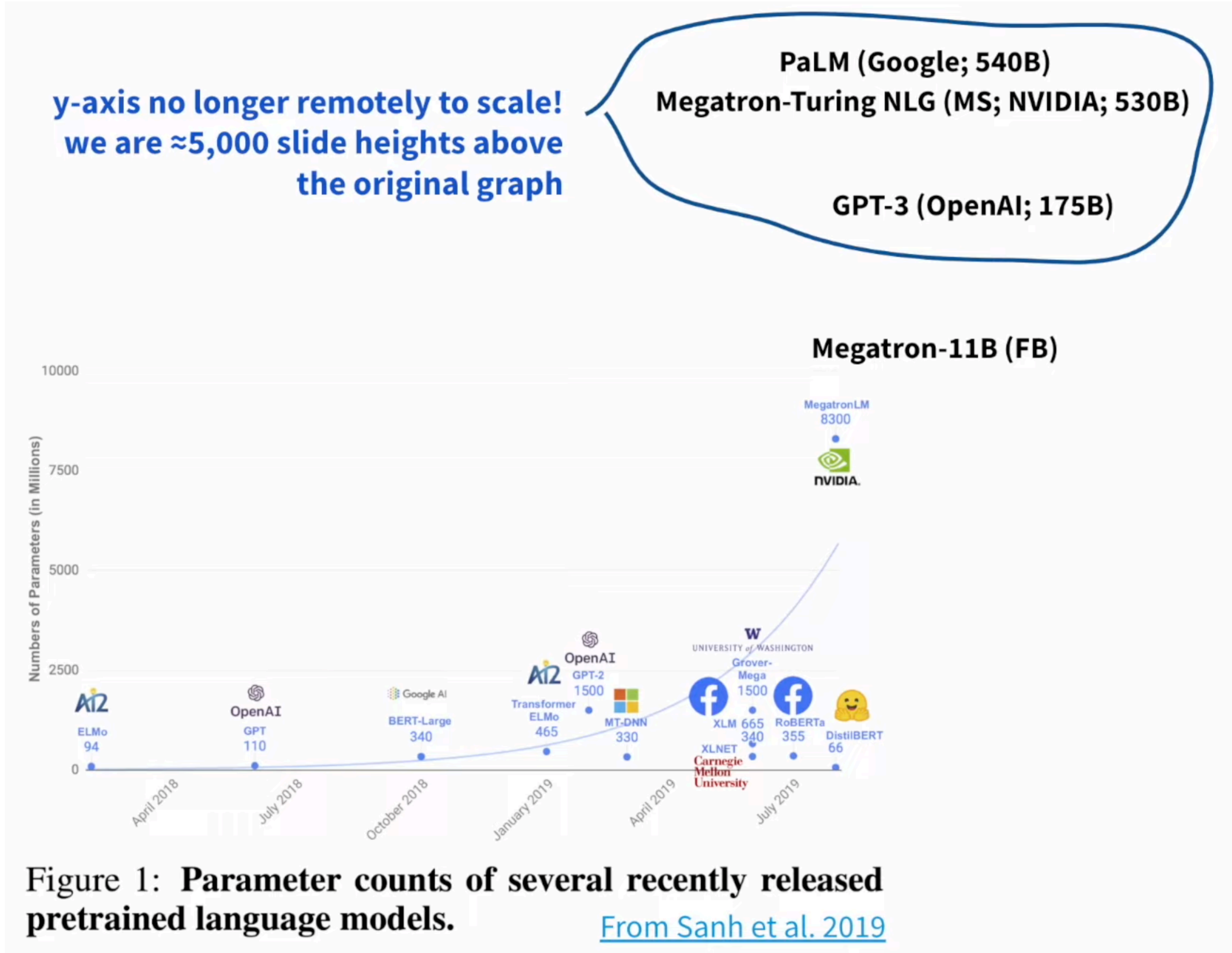
# Limitations of Pre-training + Fine-tuning

# State of the Field

- Manning 2017: “The BiLSTM Hegemony”
- ~2020-2022: “The Pre-trained Transformer Hegemony”
  - By default: fine-tune a large pre-trained Transformer on the task you care about
  - Will often yield the best results
  - Beware: often not significantly better than *very simple* baselines (SVM, etc)
- Now: “The LLM-prompting Hegemony??”
  - Actually fine-tuning models is less common
  - Some exceptions, as with all these “hegemonies”



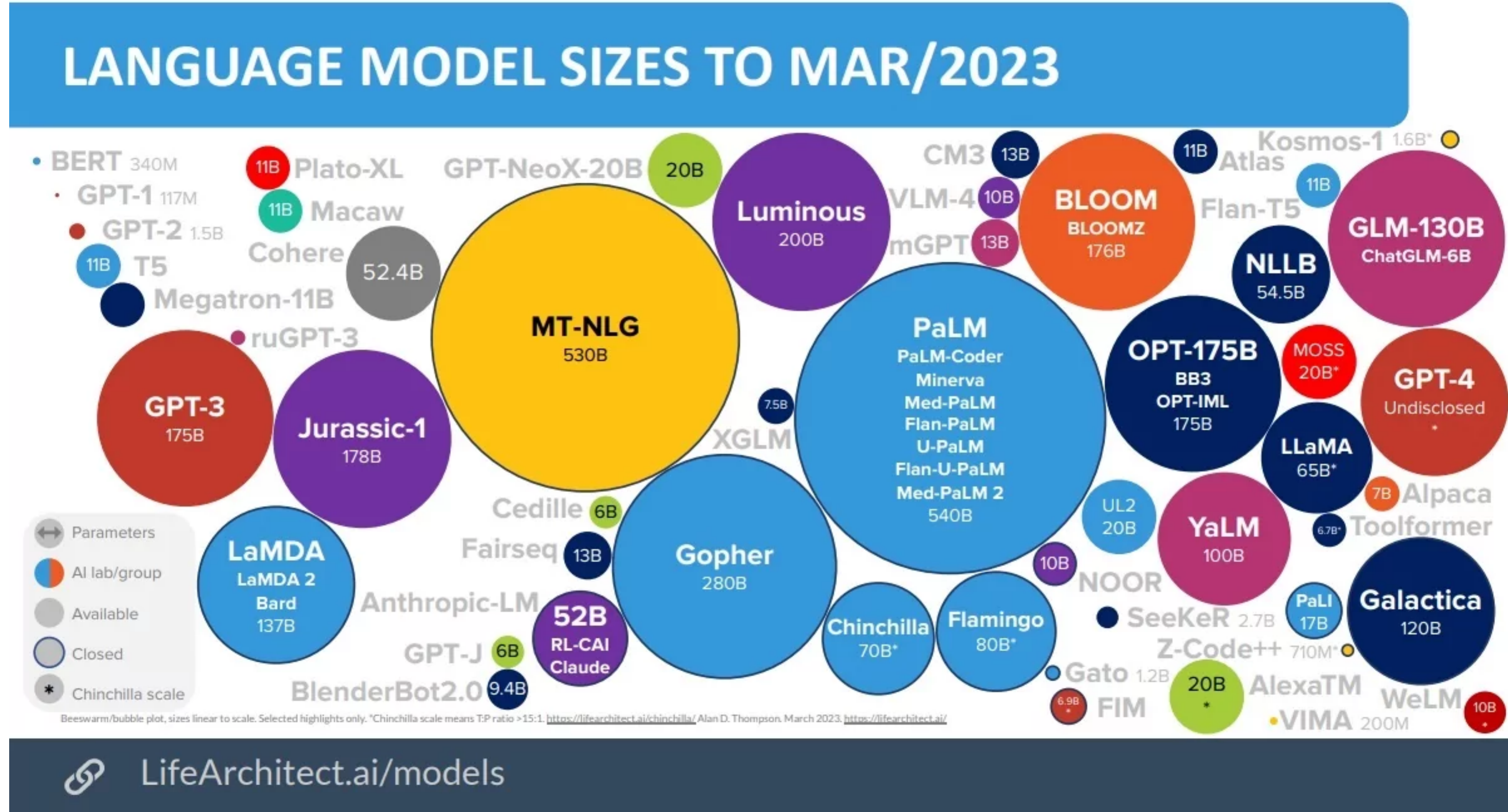
# Scale scale scale



source



# More Comprehensive

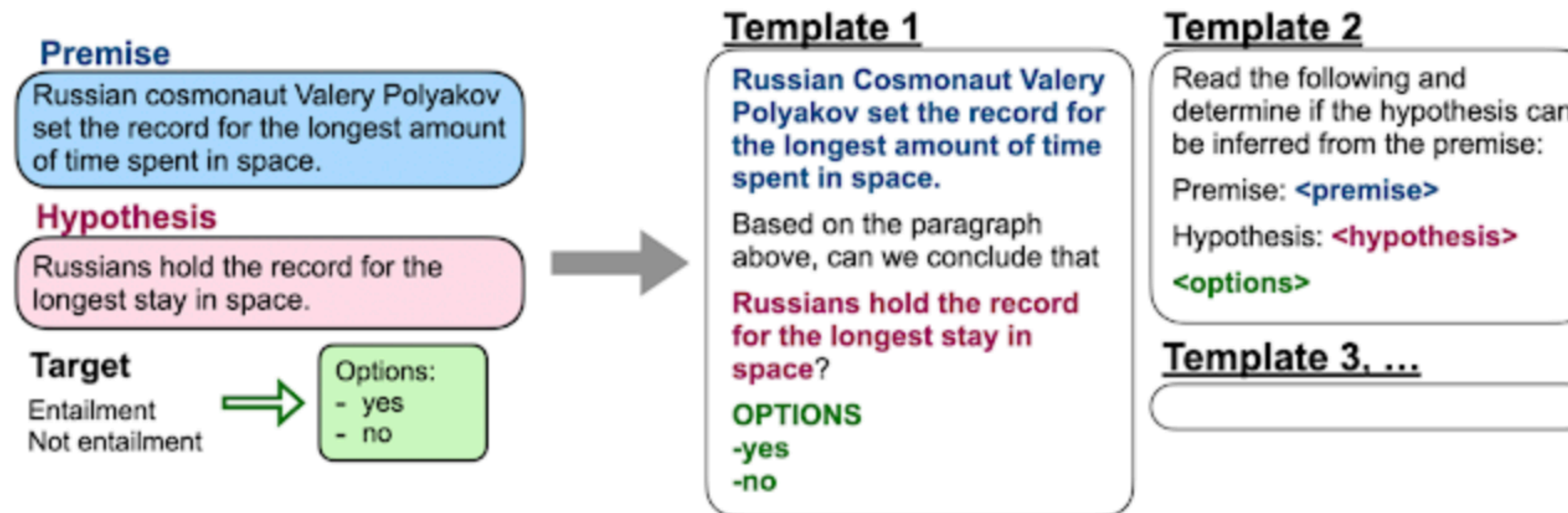




# Newest Trends (Sp 2023)

- **Instruction Fine-tuning**

- Large models are fine-tuned with various natural-language **cast as language modeling**; i.e. have the model answer **using language**
- examples: Google's [FLAN](#) (137B), OpenAI's davinci, Stanford [Alpaca](#) (7B)



Example templates for a natural language inference dataset.

# Newest Trends (Sp 2023)

- **“In-Context Learning”**
  - Models so large that it’s impractical to fine-tune
  - Instead, design ways to **“prompt” the model to get the results you want**
    - Model takes in task prompt as input to be continued as language-modeling
    - So-called “prompt engineering”
  - Many models now **only have API access**
    - i.e. user can give a prompt, get outputs and sometimes final probabilities
    - No access to model weights
    - **ChatGPT** has sent this paradigm into hyperdrive

# Newest Trends (Sp 2023)

- **RLHF**
  - (Reinforcement Learning from Human Feedback)
  - Key idea: have annotators rate the relative “quality” or “acceptability” of generations
  - Hard to use human feedback at large scale
    - Train a small, auxillary model to **replicate human judgements**
    - Have the auxillary model **provide acceptability judgements as loss**
  - One of the key ingredients of ChatGPT

# Note on the costs of LMs



# Note on the costs of LMs

- Currently something of an ‘arms race’ between e.g. Google, Facebook, OpenAI, MS, Baidu, ...

# Note on the costs of LMs

- Currently something of an ‘arms race’ between e.g. Google, Facebook, OpenAI, MS, Baidu, ...
- Hugely expensive
  - Carbon emissions
  - Monetarily
    - Inequitable access
  - Dataset debt/documentation

# Note on the costs of LMs

- Currently something of an ‘arms race’ between e.g. Google, Facebook, OpenAI, MS, Baidu, ...

- Hugely expensive

- Carbon emissions

- Monetarily

- Inequitable access

- Dataset debt/documentation

## Energy and Policy Considerations for Deep Learning in NLP

Emma Strubell    Ananya Ganesh    Andrew McCallum  
College of Information and Computer Sciences  
University of Massachusetts Amherst  
{strubell, aganesh, mccallum}@cs.umass.edu

### Abstract

Recent progress in hardware and methodology for training neural networks has ushered in a new generation of large networks trained on abundant data. These models have obtained notable gains in accuracy across many NLP tasks. However, these accuracy improvements depend on the availability of exceptionally large computational resources that necessitate similarly substantial energy consumption. As a result these models are costly to train and develop, both financially, due to the cost of hardware and electricity or cloud compute time, and environmentally, due to the carbon footprint required to fuel modern tensor

Consumption	CO <sub>2</sub> e (lbs)
Air travel, 1 person, NY↔SF	1984
Human life, avg, 1 year	11,023
American life, avg, 1 year	36,156
Car, avg incl. fuel, 1 lifetime	126,000
<b>Training one model (GPU)</b>	
NLP pipeline (parsing, SRL)	39
w/ tuning & experiments	78,468
Transformer (big)	192
w/ neural arch. search	626,155

Table 1: Estimated CO<sub>2</sub> emissions from training common NLP models, compared to familiar consumption.<sup>1</sup>

# Note on the costs of LMs

- Currently something of an ‘arms race’ between e.g. Google, Facebook, OpenAI, MS, Baidu, ...
- Hugely expensive
  - Carbon emissions
  - Monetarily
    - Inequitable access
  - Dataset debt/documentation

# Note on the costs of LMs

- Currently something of an ‘arms race’ between e.g. Google, Facebook, OpenAI, MS, Baidu, ...

## Green AI

- Hugely expensive

- Carbon emissions

- Monetarily

- Inequitable access

- Dataset debt/documentation

Roy Schwartz\*<sup>◇</sup> Jesse Dodge\*<sup>◇♣</sup> Noah A. Smith<sup>◇♡</sup> Oren Etzioni<sup>◇</sup>

<sup>◇</sup>Allen Institute for AI, Seattle, Washington, USA

<sup>♣</sup>Carnegie Mellon University, Pittsburgh, Pennsylvania, USA

<sup>♡</sup>University of Washington, Seattle, Washington, USA

July 2019

### Abstract

The computations required for deep learning research have been doubling every few months, resulting in an estimated 300,000x increase from 2012 to 2018 [2]. These computations have a surprisingly large carbon footprint [40]. Ironically, deep learning was inspired by the human brain, which is remarkably energy efficient. Moreover, the financial cost of the computations can make it difficult for academics, students, and researchers, in particular those from emerging economies, to engage in deep learning research.

This position paper advocates a practical solution by making **efficiency** an evaluation criterion for research alongside accuracy and related measures. In addition, we propose reporting the financial cost or “price tag” of developing, training, and running models to provide baselines for the investigation of increasingly efficient methods. Our goal is to make AI both greener and more inclusive—enabling any inspired undergraduate with a laptop to write high-quality research papers. **Green AI** is an emerging focus at the Allen Institute for AI.



# More on the Costs of LMs

- For more on the reactions to this paper: <https://faculty.washington.edu/ebender/stochasticparrots.html>

## On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?

Emily M. Bender\*

ebender@uw.edu

University of Washington  
Seattle, WA, USA

Angelina McMillan-Major

aymm@uw.edu

University of Washington  
Seattle, WA, USA

Timnit Gebru\*

timnit@blackinai.org

Black in AI  
Palo Alto, CA, USA

Shmargaret Shmitchell

shmargaret.shmitchell@gmail.com

The Aether

### ABSTRACT

The past 3 years of work in NLP have been characterized by the development and deployment of ever larger language models, especially for English. BERT, its variants, GPT-2/3, and others, most recently Switch-C, have pushed the boundaries of the possible both through architectural innovations and through sheer size. Using these pretrained models and the methodology of fine-tuning them for specific tasks, researchers have extended the state of the art

alone, we have seen the emergence of BERT and its variants [39, 70, 74, 113, 146], GPT-2 [106], T-NLG [112], GPT-3 [25], and most recently Switch-C [43], with institutions seemingly competing to produce ever larger LMs. While investigating properties of LMs and how they change with size holds scientific interest, and large LMs have shown improvements on various tasks (§2), we ask whether enough thought has been put into the potential risks associated with developing them and strategies to mitigate these risks.



# Some Reasons to Pause

- Leaderboard chasing (via larger models and more data) funnels research activity into one specific and limited goal
- Amplifies harmful biases
- Equity costs
- Climate costs
- Data documentation debt
- Does not promote human-like linguistic generalization ([Linzen 2020](#) summary)

# Outsourcing Harmful Content

- OpenAI outsourced labeling of harmful/disturbing content to workers making \$2/hr in Kenya
  - Content warning!
- Essentially: have annotators label content as harmful/inappropriate; use to train model to **not** output such things
  - Employees described having to label large amounts of disturbing content, some of which was illegal
- The takeaway: the success of models like ChatGPT is **still largely ensured by annotator labor**

# Transformers

<https://huggingface.co/transformers>

# Overview of the Library

- Access to many variants of many very large LMs (BERT, RoBERTa, XLNET, ALBERT, T5, language-specific models, ...) with fairly consistent API
  - Build tokenizer + model from string for name or config
  - Then use just like any PyTorch nn.Module
- Emphasis on ease-of-use
  - E.g. low barrier-to-entry to *using* the models, including for analysis
- Interoperable with PyTorch or TensorFlow 2.0

# Example: Tokenization

```
>>> from transformers import AutoTokenizer
```

```
>>> tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
```

```
>>> encoded_input = tokenizer("Do not meddle in the affairs of wizards, for they are subtle and quick to  
>>> print(encoded_input)  
{'input_ids': [101, 2079, 2025, 19960, 10362, 1999, 1996, 3821, 1997, 16657, 1010, 2005, 2027, 2024, 1125],  
'token_type_ids': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}
```

# Example: Tokenization

```
>>> from transformers import AutoTokenizer
```

```
>>> tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
```

```
>>> encoded_input = tokenizer("Do not meddle in the affairs of wizards, for they are subtle and quick to  
>>> print(encoded_input)  
{'input_ids': [101, 2079, 2025, 19960, 10362, 1999, 1996, 3821, 1997, 16657, 1010, 2005, 2027, 2024, 1125],  
  'token_type_ids': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
  'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}
```

```
>>> tokenizer.decode(encoded_input["input_ids"])  
'[CLS] Do not meddle in the affairs of wizards, for they are subtle and quick to anger. [SEP]'
```



# Example: Tokenizing a Batch

```
>>> batch_sentences = [  
...     "But what about second breakfast?",  
...     "Don't think he knows about second breakfast, Pip.",  
...     "What about elevensies?",  
... ]  
>>> encoded_input = tokenizer(batch_sentences, padding=True)  
>>> print(encoded_input)  
{'input_ids': [[101, 1252, 1184, 1164, 1248, 6462, 136, 102, 0, 0, 0, 0, 0, 0, 0],  
               [101, 1790, 112, 189, 1341, 1119, 3520, 1164, 1248, 6462, 117, 21902, 1643, 119, 102],  
               [101, 1327, 1164, 5450, 23434, 136, 102, 0, 0, 0, 0, 0, 0, 0, 0]],  
'token_type_ids': [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
                    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
                    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]],  
'attention_mask': [[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0],  
                   [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],  
                   [1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]]}
```

# Example: Tokenizing a Batch

```
>>> batch_sentences = [  
...     "But what about second breakfast?",  
...     "Don't think he knows about second breakfast, Pip.",  
...     "What about elevensies?",  
... ]  
>>> encoded_input = tokenizer(batch_sentences, padding=True)  
>>> print(encoded_input)  
{'input_ids': [[101, 1252, 1184, 1164, 1248, 6462, 136, 102, 0, 0, 0, 0, 0, 0, 0],  
               [101, 1790, 112, 189, 1341, 1119, 3520, 1164, 1248, 6462, 117, 21902, 1643, 119, 102],  
               [101, 1327, 1164, 5450, 23434, 136, 102, 0, 0, 0, 0, 0, 0, 0, 0]],  
'token_type_ids': [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
                   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
                   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]],  
'attention_mask': [[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0],  
                   [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],  
                   [1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]]}
```

Add `return\_tensors="pt"` to get these outputs as PyTorch Tensors

# Example: Forward Pass

```
>>> from transformers import BertTokenizer, BertModel
>>> import torch

>>> tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
>>> model = BertModel.from_pretrained("bert-base-uncased")

>>> inputs = tokenizer("Hello, my dog is cute", return_tensors="pt")
>>> outputs = model(**inputs)

>>> last_hidden_states = outputs.last_hidden_state
```

# More on HuggingFace

- Main library: <https://huggingface.co/transformers>
- Model repository (w/ search, tags, etc): <https://huggingface.co/models>
- Datasets: <https://huggingface.co/datasets>