# Lecture 16: Large Language Models

## Administrative: Assignment 5

Out! Due 3/13 11:59pm

- LSTMs
- Transformers
- Self-Supervised

## Exam 2

1 week from today in class, 3/5 11:59pm

- interpretability to the end of what we cover today
    - will definitely include attention and transformers
- cheat sheet allowed
- calculator allowed (but unnecessary)

# Administrative: Fridays

This Friday

**Robotics**

## Self-Supervised

## Supervised Learning

Train **directly** on labeled data **(x,y)** for **downstream task**

Ex: ImageNet has (image, class) pairs

Gets ***extremely*** expensive

Upper-bound on how much of such data exists in the world

## Self-Supervised Learning

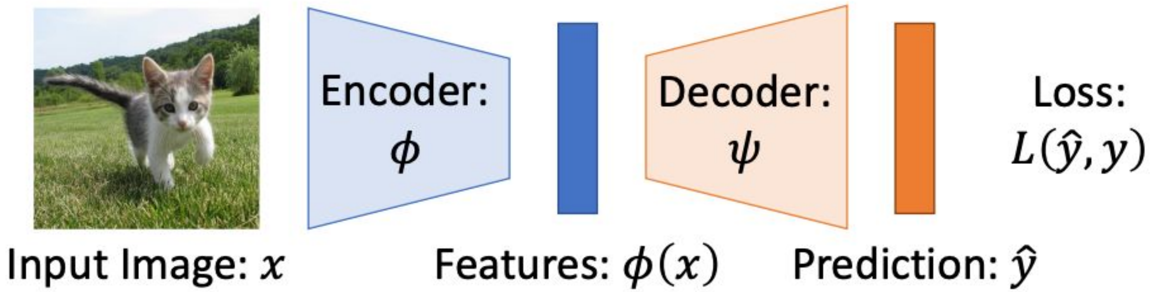
**Pre-train** on a **large, unlabeled** dataset (often generic)

**Post-train** on a **small, labeled** dataset (tailored to your downstream task)

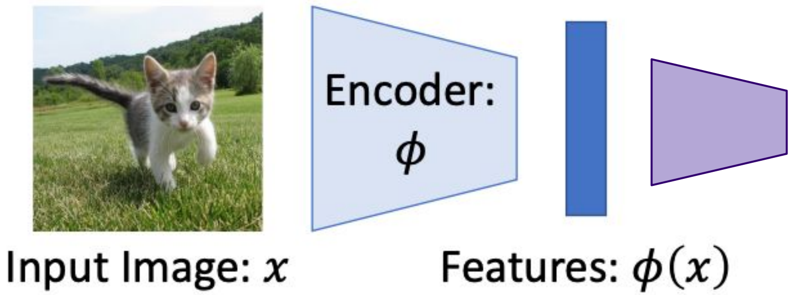
Intuition: pre-training helps model gain “common-sense” understanding

# Self-Supervised Learning: Pretext then Transfer

**Step 1:** Pretrain a network on a pretext task that doesn't require supervision

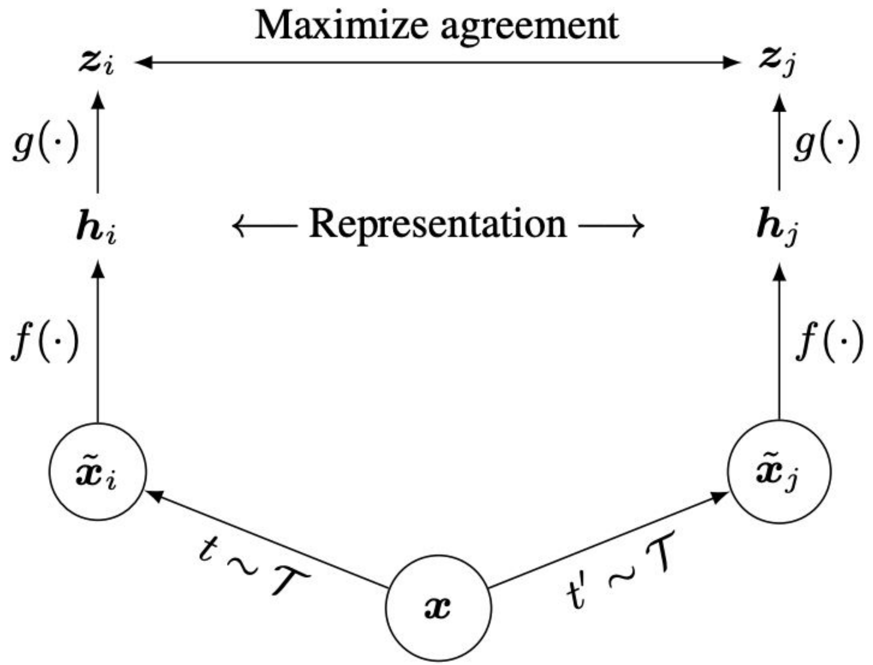


**Step 2:** Transfer encoder to downstream tasks via linear classifiers, KNN, finetuning



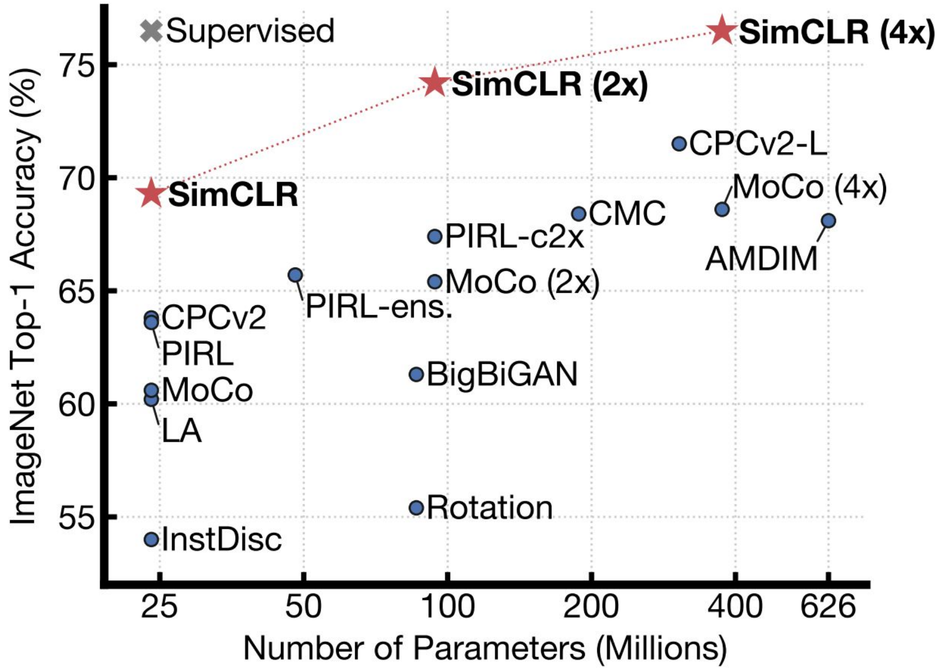
**Downstream tasks:** Image classification, object detection, semantic segmentation

## SimCLR: a really big pair-wise comparison



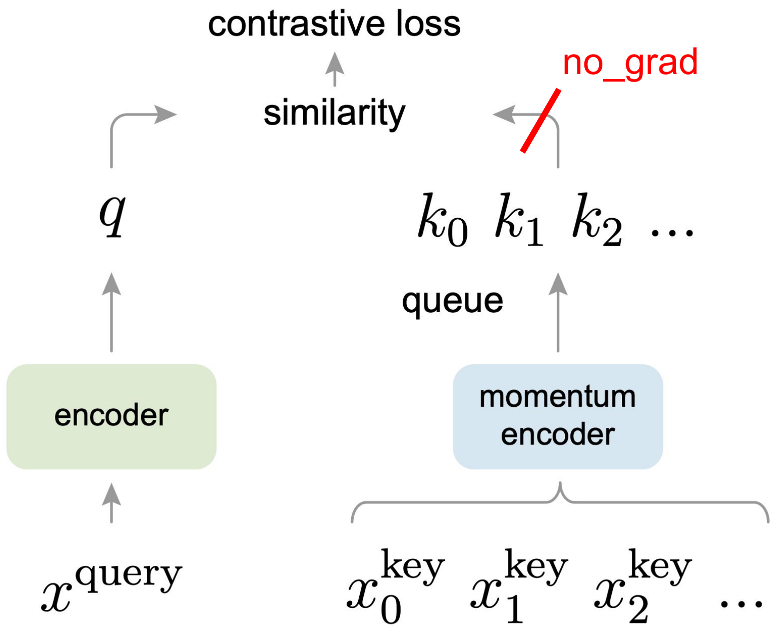
Source: Chen et al., 2020

## SimCLR: the proving ground for self-supervised



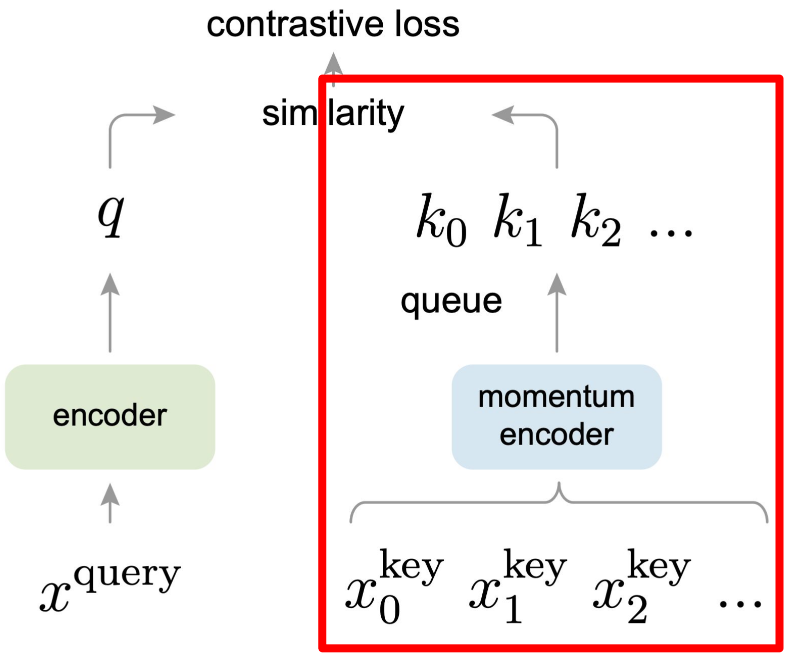
Source: Chen et al., 2020

# MoCo: efficient (ish) self-supervised learning



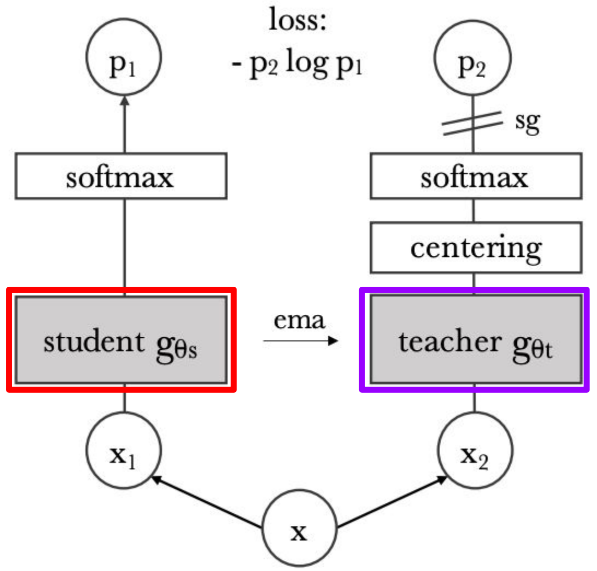
Source: He et al., 2020

# Problem with MoCoV2: Need to keep around a set of negatives



Do we need these negatives?

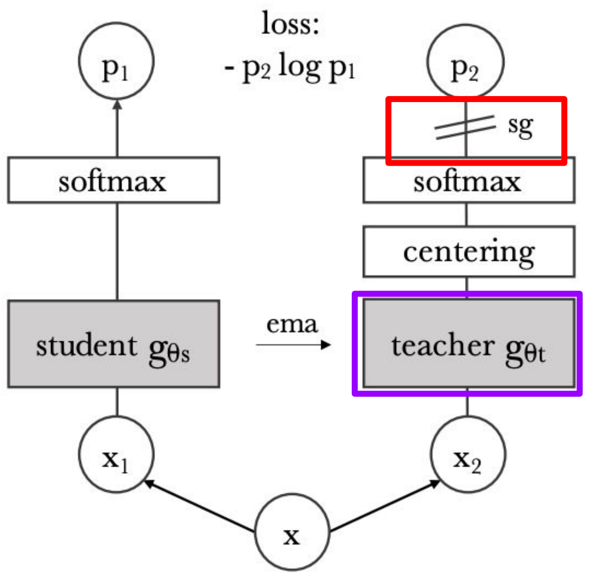
# Solution: DINO: self-distillation with no labels



- Similar to SimCLR and MOCO but with one big difference: no negatives
- Reformulates contrastive learning as knowledge distillation between a student and a teacher model.

Source: Caron et al. Emerging Properties in Self-Supervised Vision Transformers. 2021

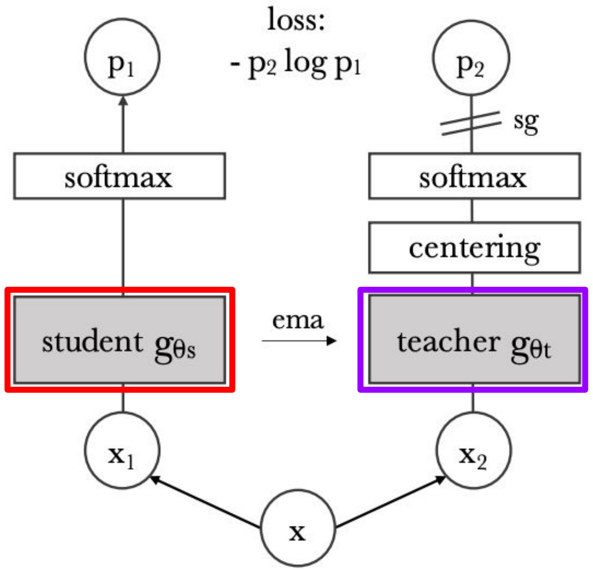
# Solution: DINO: self-distillation with no labels



- The teacher model is not trained: sg stands for stop-gradient: meaning that gradients are prevented from flowing back.

Source: Caron et al. Emerging Properties in Self-Supervised Vision Transformers. 2021

## Problem: But how do we choose the teacher model?



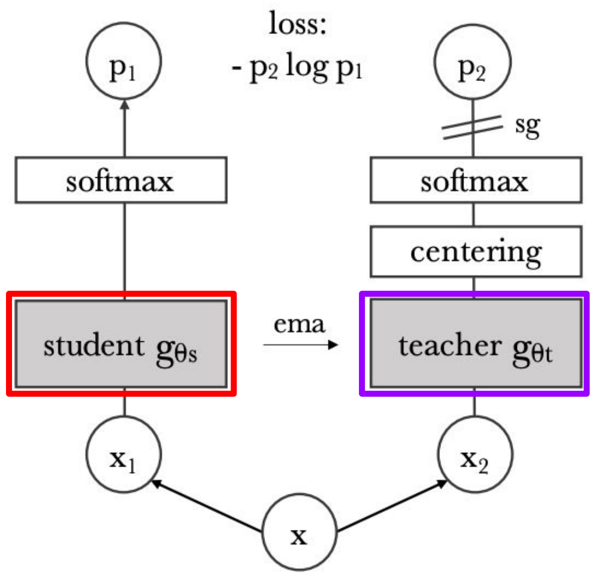
- The teacher model is like the momentum encoder. It is a running average of the student model

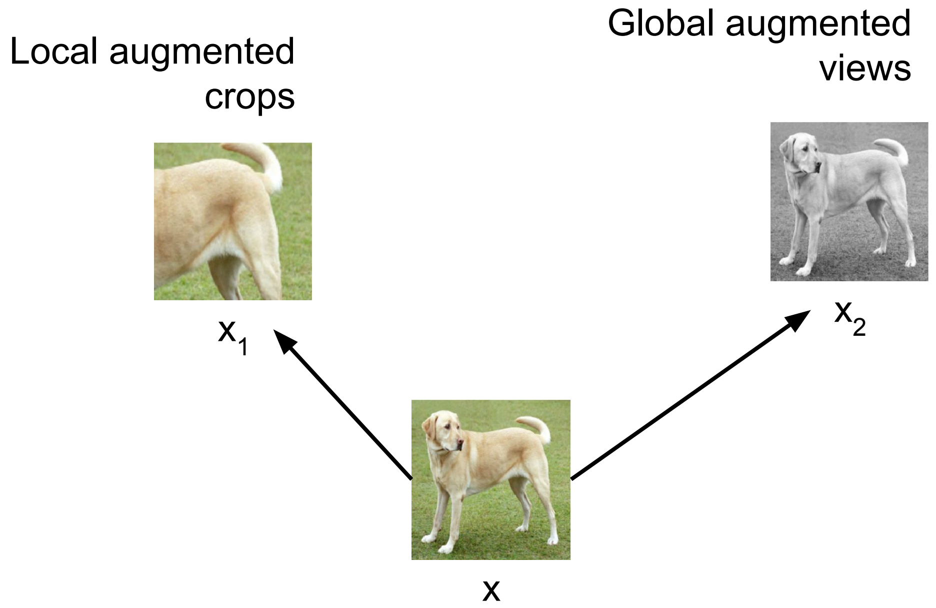
$$\theta_t \leftarrow \lambda \theta_t + (1 - \lambda)\theta_s$$

- The teacher sees a global view augmentation of the image
- Student only sees augmented local crops of the image

Source: Caron et al. Emerging Properties in Self-Supervised Vision Transformers. 2021

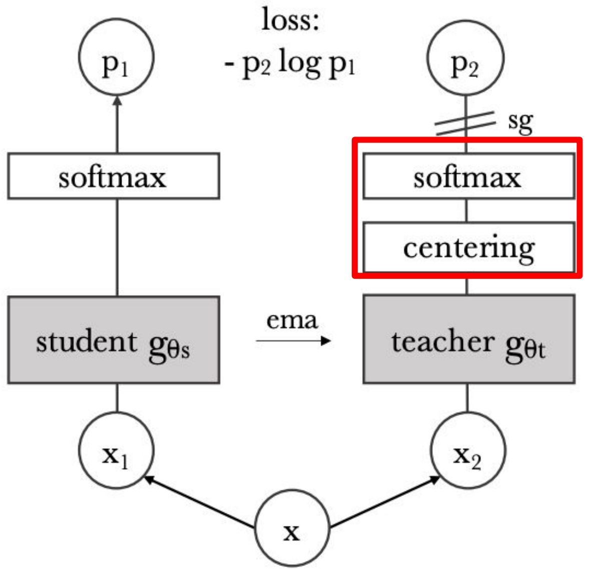
# Problem: But how do we choose the teacher model?





Source: Caron et al. Emerging Properties in Self-Supervised Vision Transformers. 2021

# Problem: But how do we choose the teacher model?



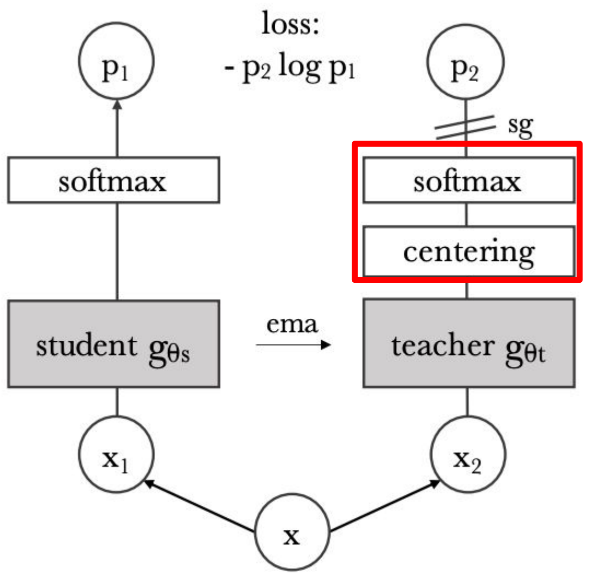
Training tricks:

- **Centering**: prevents one dimension from dominating.
  - A constant value c is added to all dimensions of the teacher's output.
  - c is a running average of outputs

$$g_t(x) \leftarrow g_t(x) + c,\ c \leftarrow mc + (1-m)\frac{1}{B}\sum_{i=1}^{B} g_{\theta_t}(x_i)$$

Source: Caron et al. Emerging Properties in Self-Supervised Vision Transformers. 2021

# Problem: But how do we choose the teacher model?



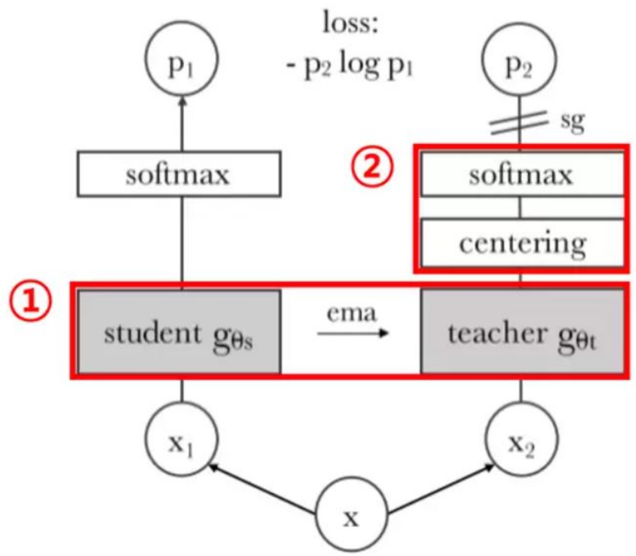
Training tricks:

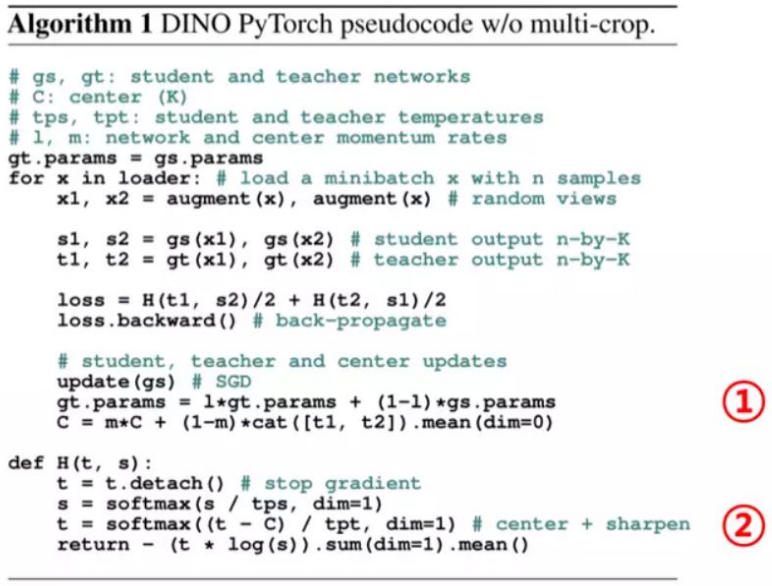
- **Sharpening**:
  - A temperature (Tau) hyperparameter is used to sharpen the distributions towards one dimension.

$$\frac{\exp(g_{\theta_s}(x)^{(i)}/\tau_s)}{\sum_{k=1}^{K}\exp(g_{\theta_s}(x)^{(k)}/\tau_s)}$$

Source: Caron et al. Emerging Properties in Self-Supervised Vision Transformers. 2021

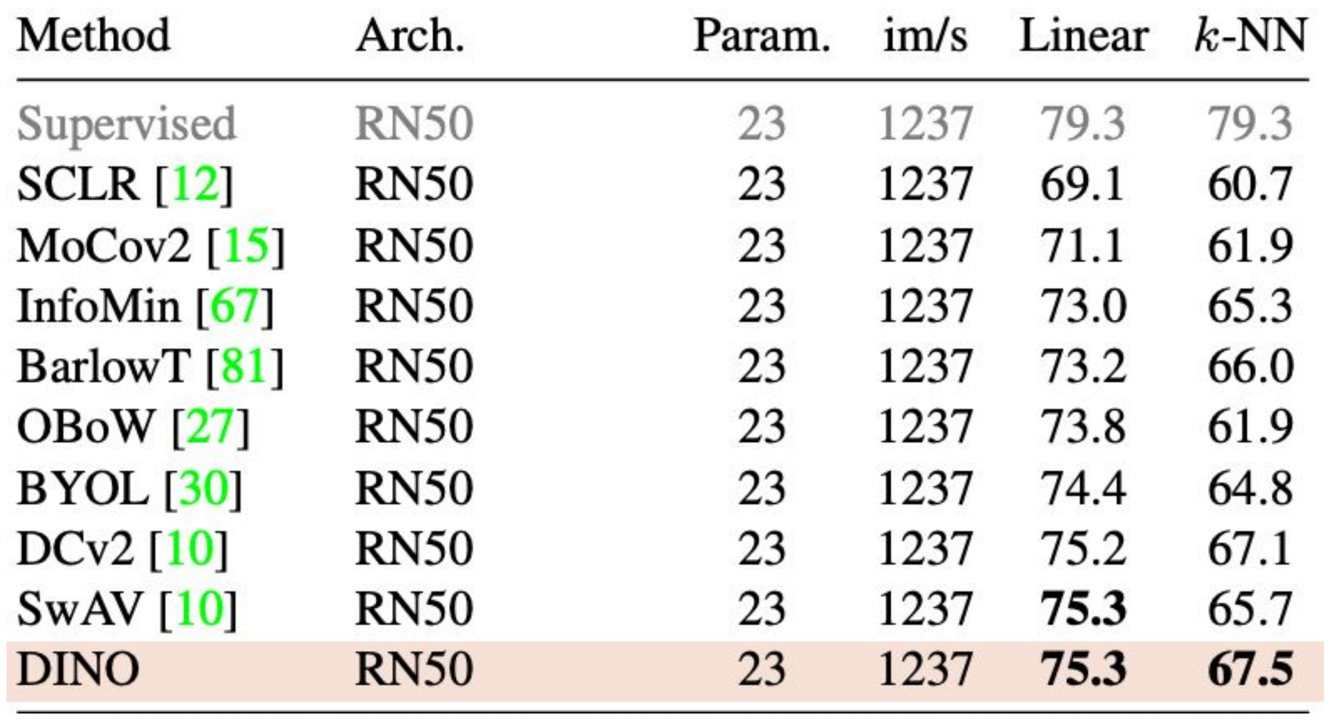
## DINO code




**Algorithm 1** DINO PyTorch pseudocode w/o multi-crop.

```
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# l, m: network and center momentum rates
gt.params = gs.params
for x in loader: # load a minibatch x with n samples
    x1, x2 = augment(x), augment(x) # random views

    s1, s2 = gs(x1), gs(x2) # student output n-by-K
    t1, t2 = gt(x1), gt(x2) # teacher output n-by-K

    loss = H(t1, s2)/2 + H(t2, s1)/2
    loss.backward() # back-propagate

    # student, teacher and center updates
    update(gs) # SGD
    gt.params = l*gt.params + (1-l)*gs.params                ①
    C = m*C + (1-m)*cat([t1, t2]).mean(dim=0)

def H(t, s):
    t = t.detach() # stop gradient
    s = softmax(s / tps, dim=1)
    t = softmax((t - C) / tpt, dim=1) # center + sharpen      ②
    return - (t * log(s)).sum(dim=1).mean()
```

## Results: DINO



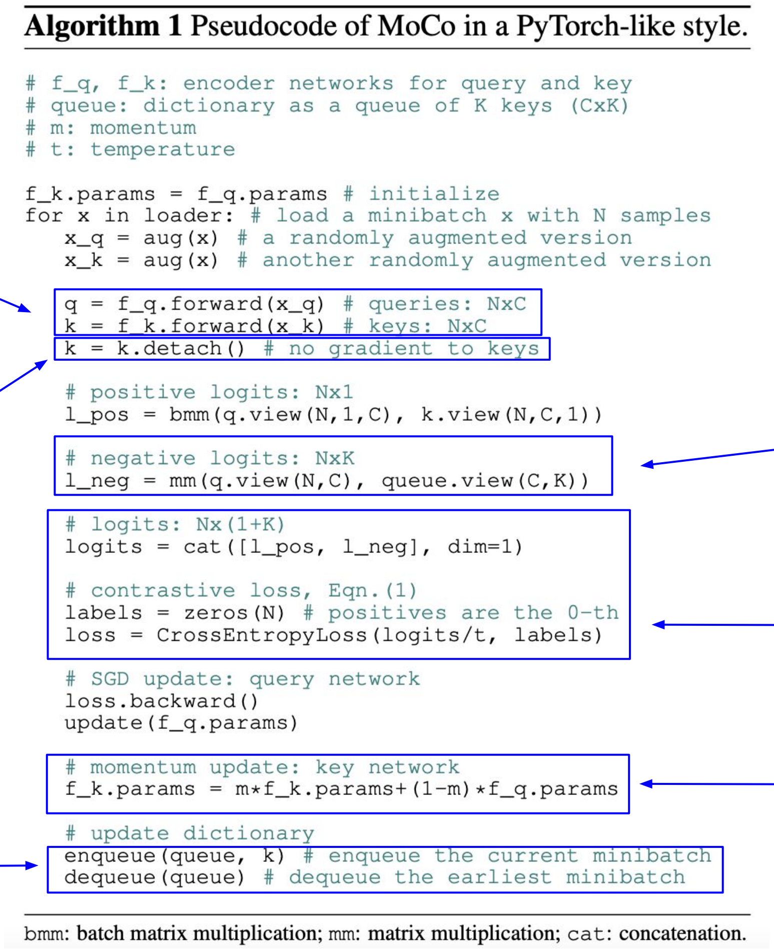
| Method | Arch. | Param. | im/s | Linear | $k$-NN |
|---|---|---|---|---|---|
| Supervised | RN50 | 23 | 1237 | 79.3 | 79.3 |
| SCLR [12] | RN50 | 23 | 1237 | 69.1 | 60.7 |
| MoCov2 [15] | RN50 | 23 | 1237 | 71.1 | 61.9 |
| InfoMin [67] | RN50 | 23 | 1237 | 73.0 | 65.3 |
| BarlowT [81] | RN50 | 23 | 1237 | 73.2 | 66.0 |
| OBoW [27] | RN50 | 23 | 1237 | 73.8 | 61.9 |
| BYOL [30] | RN50 | 23 | 1237 | 74.4 | 64.8 |
| DCv2 [10] | RN50 | 23 | 1237 | 75.2 | 67.1 |
| SwAV [10] | RN50 | 23 | 1237 | **75.3** | 65.7 |
| DINO | RN50 | 23 | 1237 | **75.3** | **67.5** |

# MoCo

Generate a positive pair by sampling data augmentation functions

No gradient through the negative samples

Update the FIFO negative sample queue



**Algorithm 1** Pseudocode of MoCo in a PyTorch-like style.

```
# f_q, f_k: encoder networks for query and key
# queue: dictionary as a queue of K keys (CxK)
# m: momentum
# t: temperature

f_k.params = f_q.params # initialize
for x in loader: # load a minibatch x with N samples
   x_q = aug(x) # a randomly augmented version
   x_k = aug(x) # another randomly augmented version

   q = f_q.forward(x_q) # queries: NxC
   k = f_k.forward(x_k) # keys: NxC
   k = k.detach() # no gradient to keys

   # positive logits: Nx1
   l_pos = bmm(q.view(N,1,C), k.view(N,C,1))

   # negative logits: NxK
   l_neg = mm(q.view(N,C), queue.view(C,K))

   # logits: Nx(1+K)
   logits = cat([l_pos, l_neg], dim=1)

   # contrastive loss, Eqn.(1)
   labels = zeros(N) # positives are the 0-th
   loss = CrossEntropyLoss(logits/t, labels)

   # SGD update: query network
   loss.backward()
   update(f_q.params)

   # momentum update: key network
   f_k.params = m*f_k.params+(1-m)*f_q.params

   # update dictionary
   enqueue(queue, k) # enqueue the current minibatch
   dequeue(queue) # dequeue the earliest minibatch
```

bmm: batch matrix multiplication; mm: matrix multiplication; cat: concatenation.

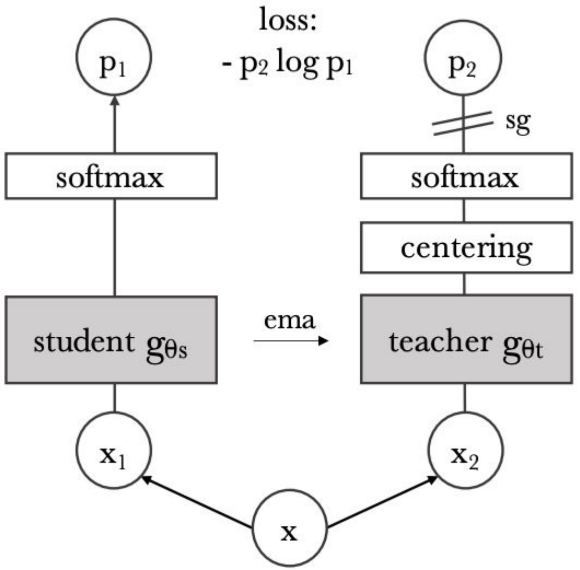
Use the running queue of keys as the negative samples

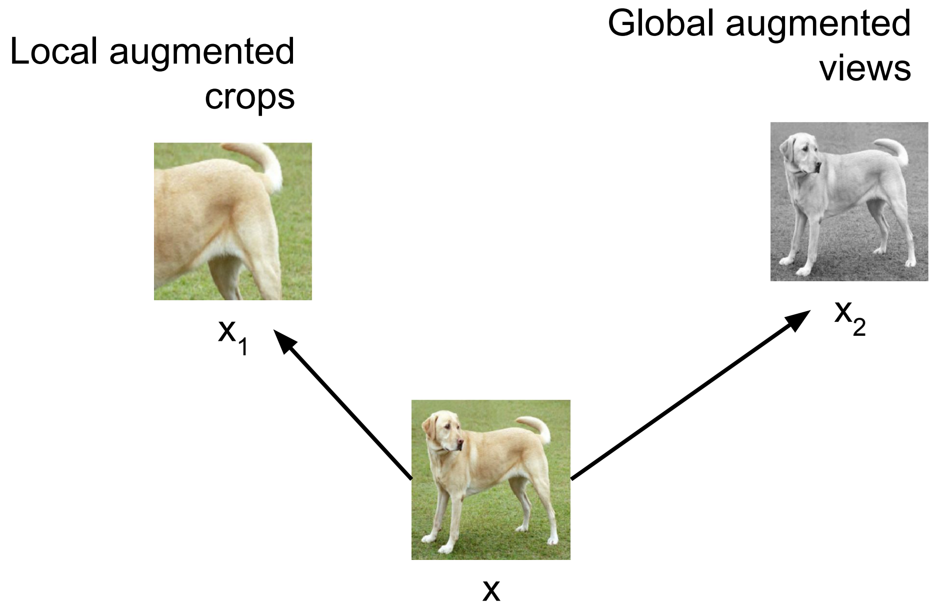
InfoNCE loss

Update f_k through momentum

Source: He et al., 2020

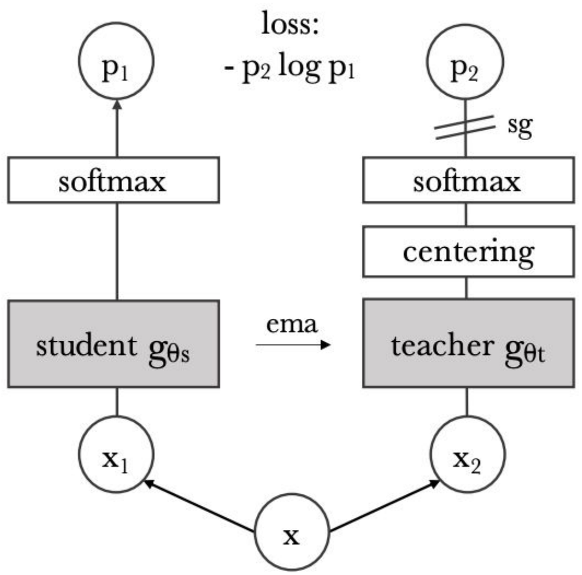
# DINO: get rid of the notion of negatives entirely





Source: Caron et al., 2021

## DINO: get rid of the notion of negatives entirely

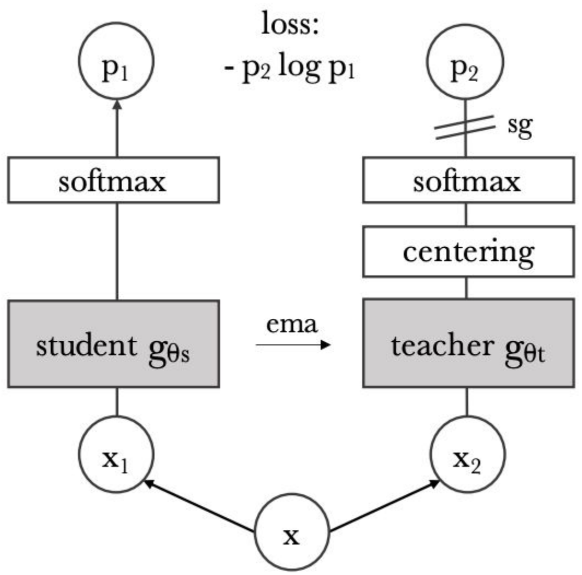


Simplifies loss – no need for InfoNCE bc we're comparing probability distributions

Allows use of Cross-Entropy loss!

Source: Caron et al., 2021

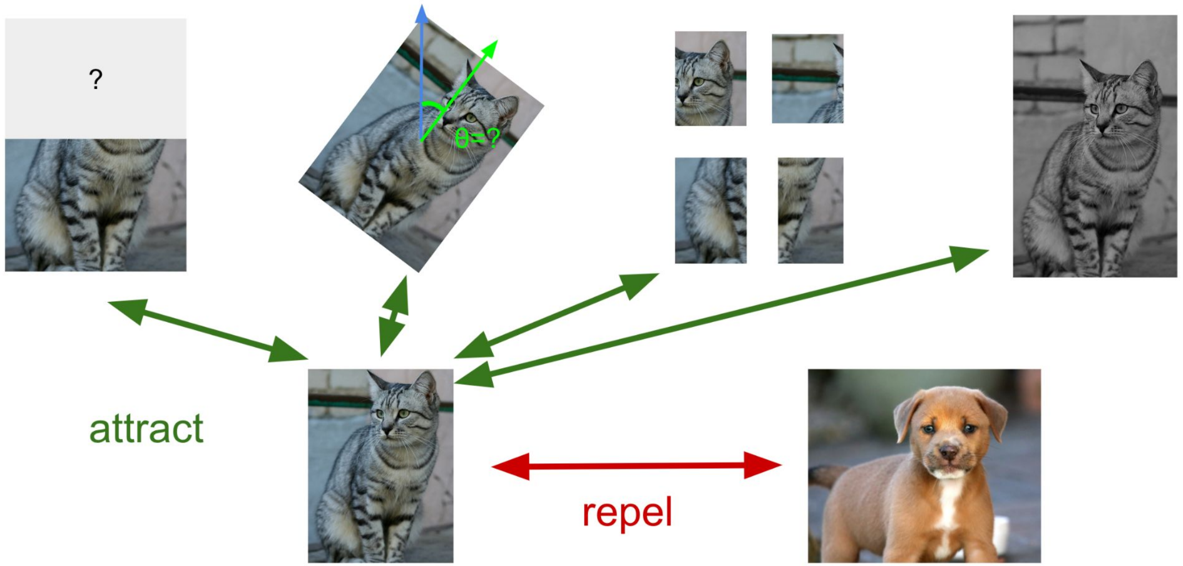
## DINO: get rid of the notion of negatives entirely



Efficient! Scalable! Hoorah

Source: Caron et al., 2021

## Aforementioned methods do instance-level C.L.

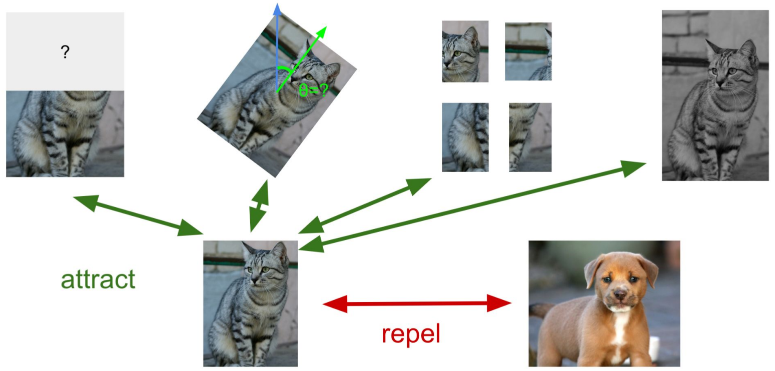


## What about sequence-level C.L.?

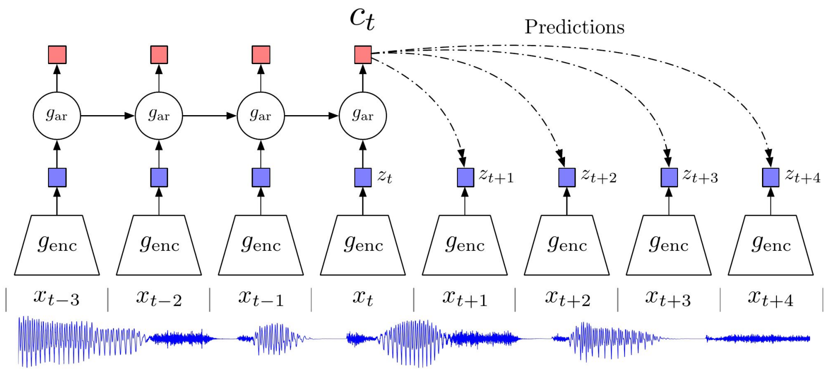
Instead of right and wrong instances (e.g., images)

What if we had right and wrong sequences (e.g., words in a sentence)

# Instance vs. Sequence Contrastive Learning



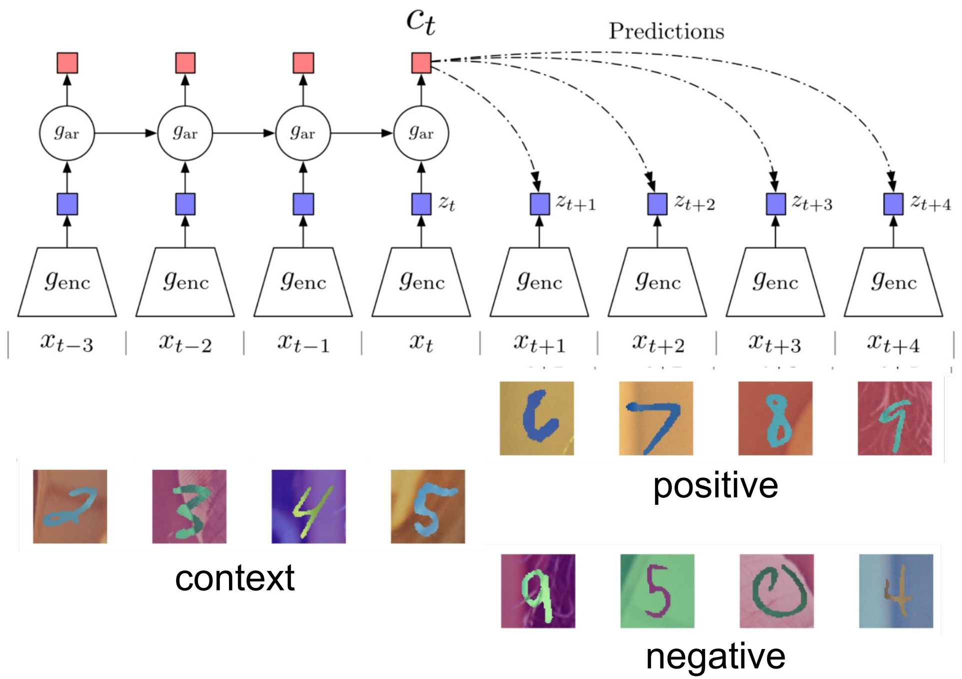
**Instance-level contrastive learning**:
contrastive learning based on
positive & negative instances.
Examples: SimCLR, MoCo



Source: van den Oord et al., 2018

**Sequence-level contrastive learning**:
contrastive learning based on
sequential / temporal orders.
Example: **Contrastive Predictive Coding (CPC)**

# Contrastive Predictive Coding (CPC)



**Contrastive**: contrast between "right" and "wrong" sequences using contrastive learning.

**Predictive**: the model has to predict future patterns given the current context.

**Coding**: the model learns useful feature vectors, or "code", for downstream tasks, similar to other self-supervised methods.

Figure source

Source: van den Oord et al., 2018,

# Contrastive Predictive Coding (CPC)

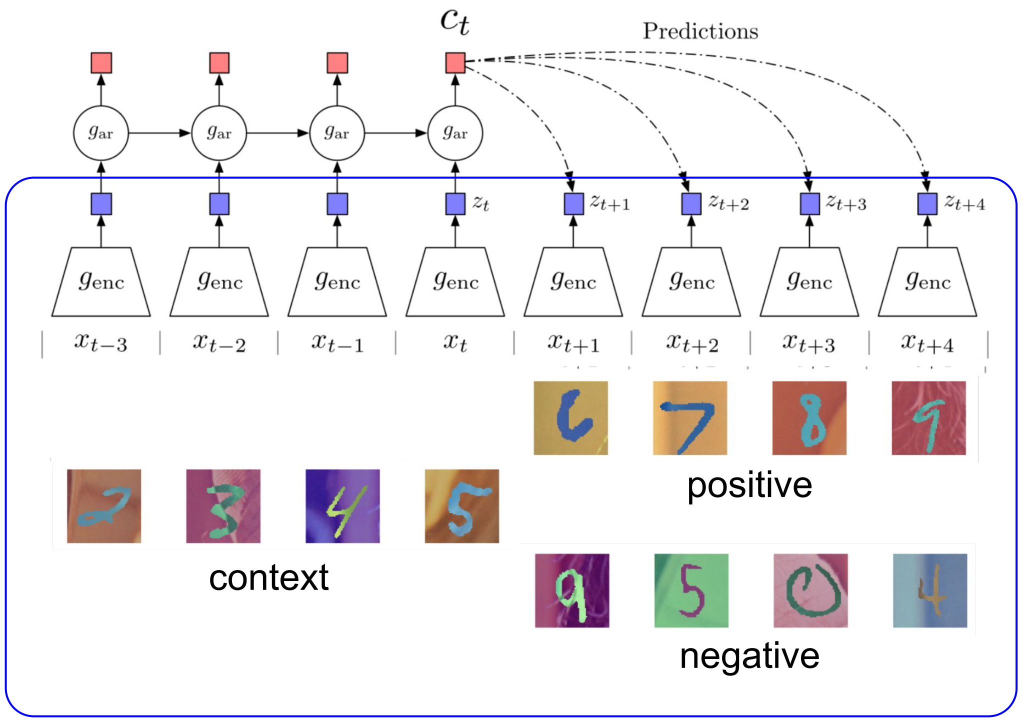
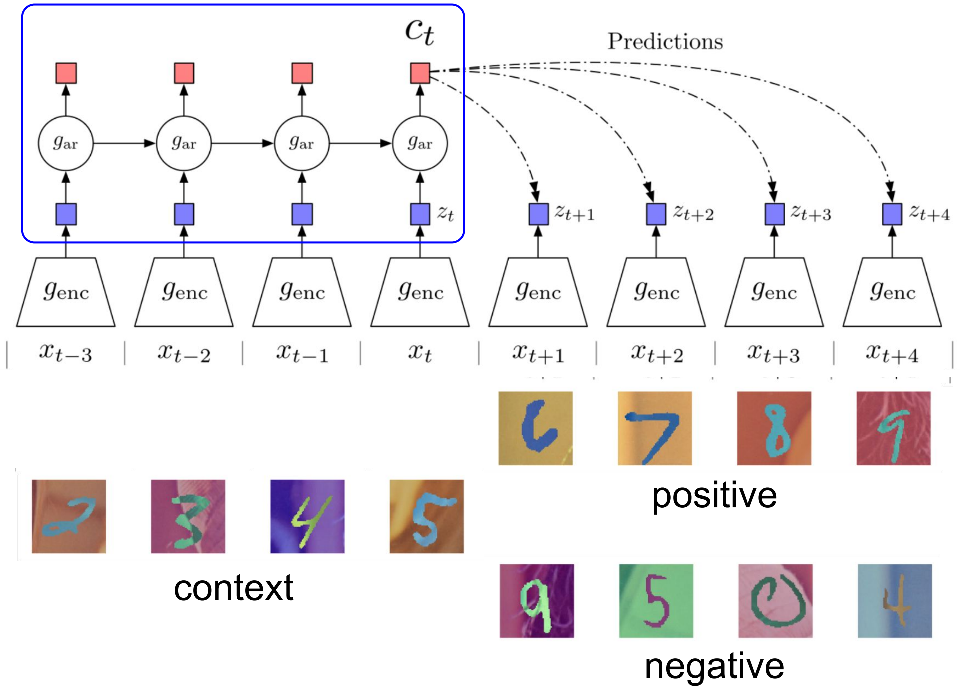


Figure source

1. Encode all samples in a sequence into vectors $z_t = g_{enc}(x_t)$

Source: van den Oord et al., 2018,

# Contrastive Predictive Coding (CPC)



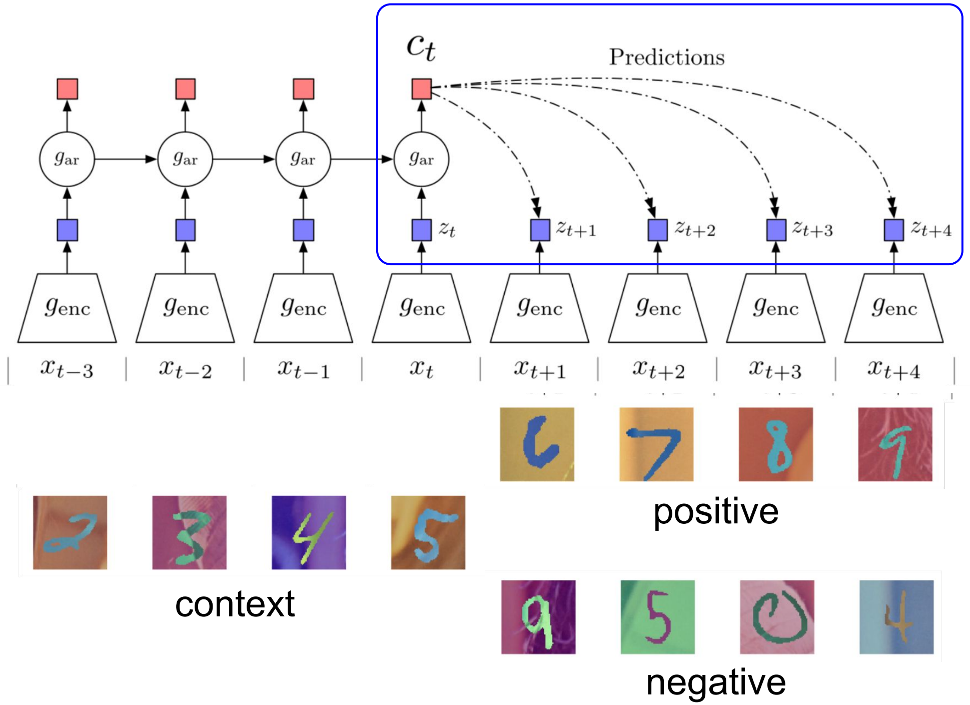
1. Encode all samples in a sequence into vectors $\boldsymbol{z_t} = \boldsymbol{g_{enc}(x_t)}$

2. Summarize context (e.g., half of a sequence) into a context code $\boldsymbol{c_t}$ using an auto-regressive model ($\boldsymbol{g_{ar}}$). The original paper uses GRU-RNN here.

Figure source

Source: van den Oord et al., 2018,

# Contrastive Predictive Coding (CPC)



1. Encode all samples in a sequence into vectors $\boldsymbol{z_t} = \boldsymbol{g_{enc}(x_t)}$

2. Summarize context (e.g., half of a sequence) into a context code $\boldsymbol{c_t}$ using an auto-regressive model ($\boldsymbol{g_{ar}}$)

3. Compute similarity score between the context $\boldsymbol{c_t}$ and future code $\boldsymbol{z_{t+k}}$ using the following time-dependent score function:

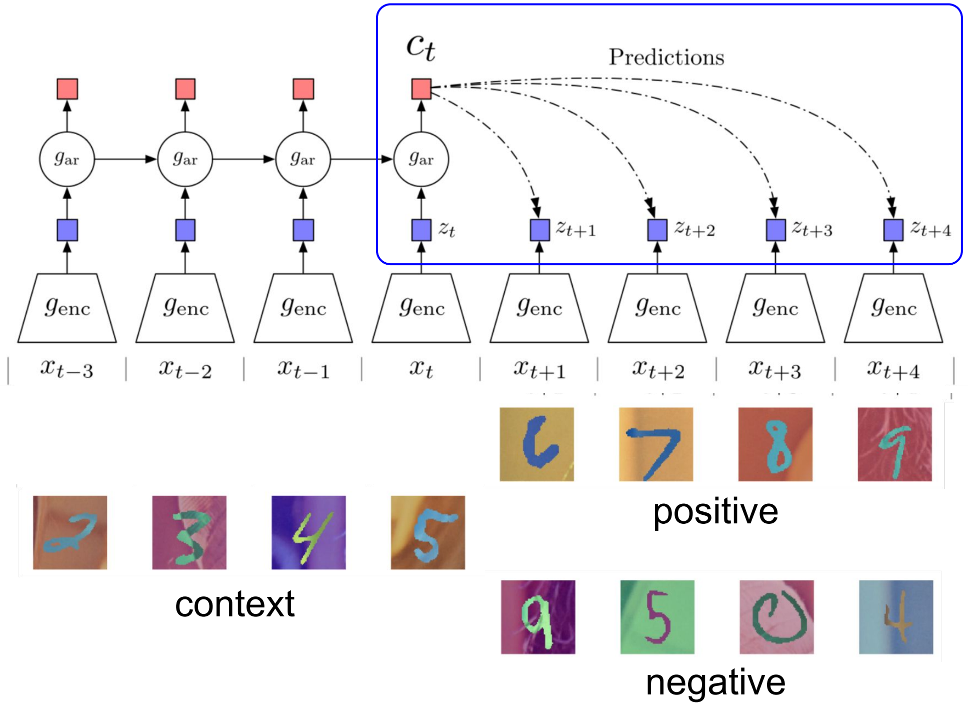
$$f_k(x_{t+k}, c_t) = \exp\left(z_{t+k}^T W_k c_t\right),$$

where $W_k$ is a trainable matrix.

Figure source

Source: van den Oord et al., 2018,

# Contrastive Predictive Coding (CPC)



1. Encode all samples in a sequence into vectors $\boldsymbol{z_t} = \boldsymbol{g_{enc}(x_t)}$

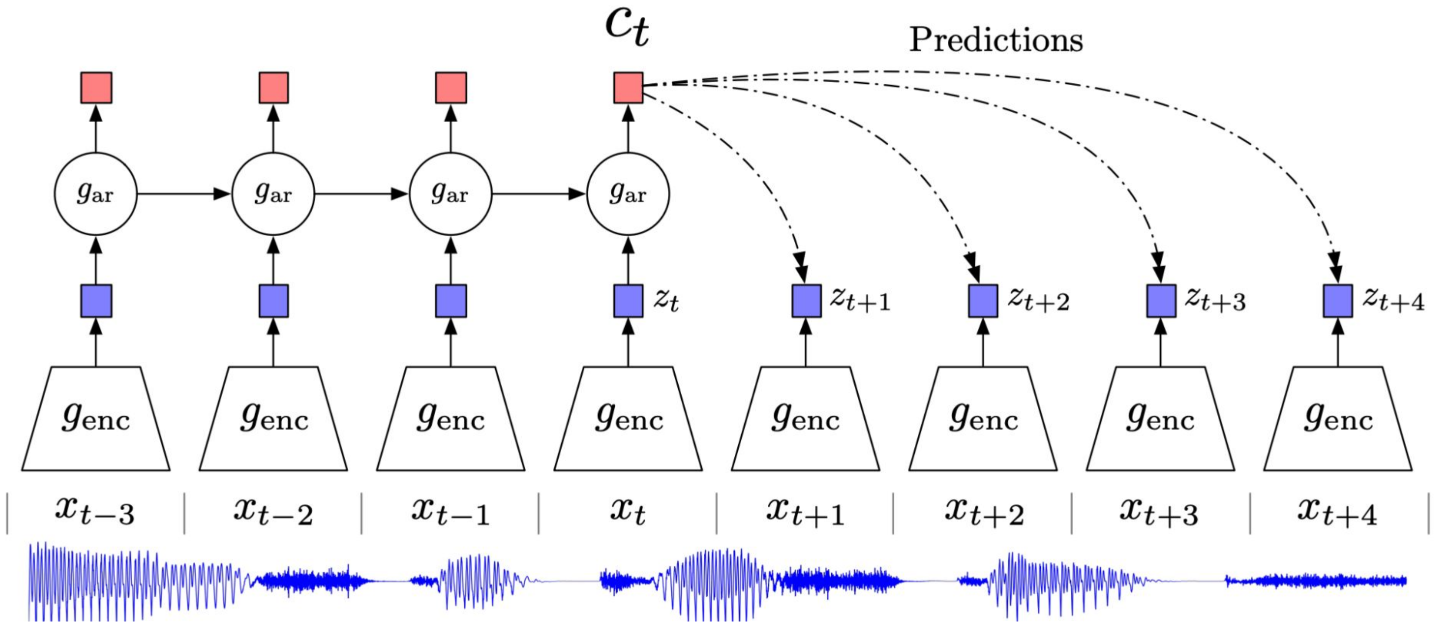
2. Summarize context (e.g., half of a sequence) into a context code $\boldsymbol{c_t}$ using an auto-regressive model ($\boldsymbol{g_{ar}}$)

3. Predict $\boldsymbol{z_{t+k}}$ using **c** and trainable weights. Loss is similarity to true $\boldsymbol{z_{t+k}}$ value over similarity to constrastng option

Figure source

Source: van den Oord et al., 2018,

## CPC example: modeling audio sequences



Source: van den Oord et al., 2018,

# CPC example: modeling audio sequences

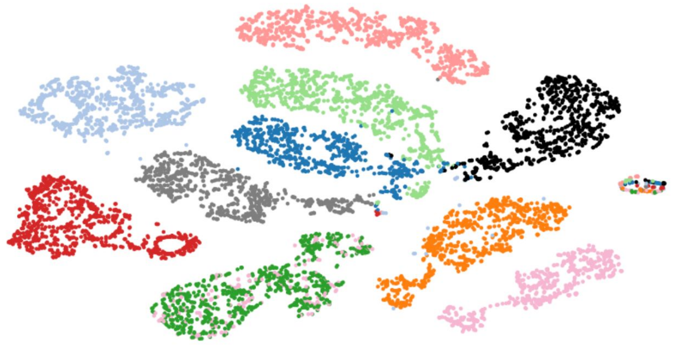


Figure 2: t-SNE visualization of audio (speech) representations for a subset of 10 speakers (out of 251). Every color represents a different speaker.

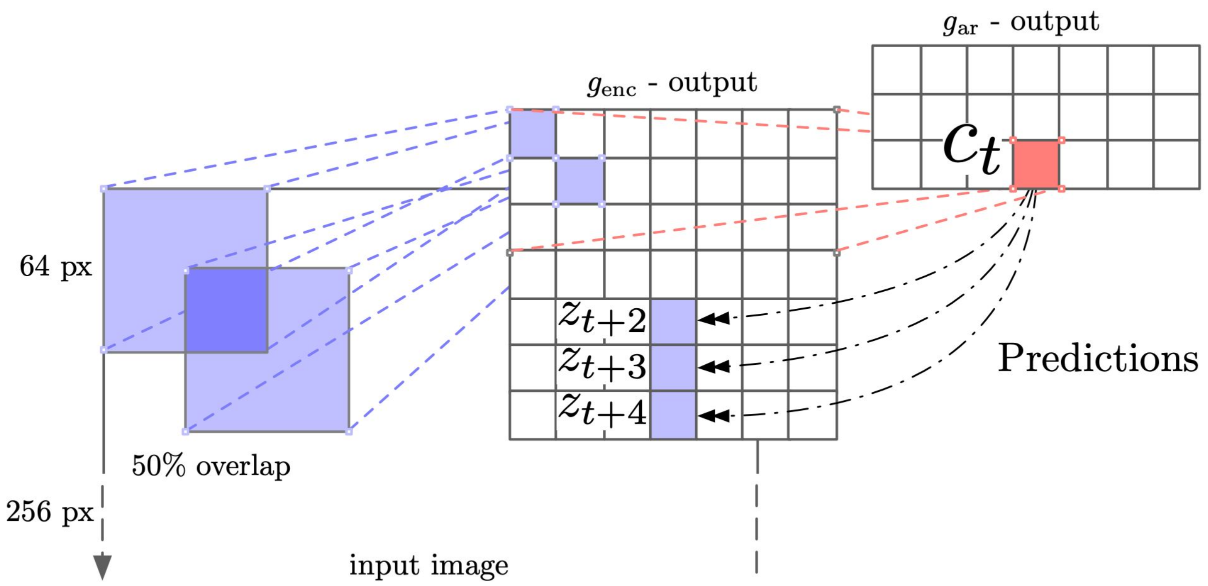
| Method | ACC |
|---|---|
| **Phone classification** | |
| Random initialization | 27.6 |
| MFCC features | 39.7 |
| CPC | 64.6 |
| Supervised | 74.6 |
| **Speaker classification** | |
| Random initialization | 1.87 |
| MFCC features | 17.6 |
| CPC | 97.4 |
| Supervised | 98.5 |

Linear classification on trained representations (LibriSpeech dataset)

Source: van den Oord et al., 2018,

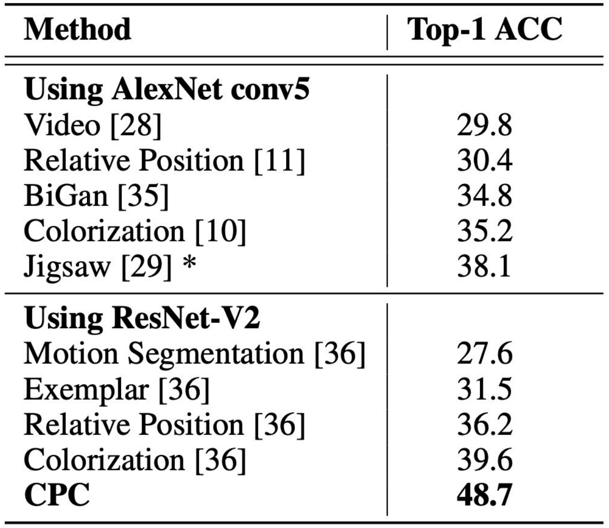
# CPC example: modeling visual context

**Idea**: split image into patches, model rows of patches from top to bottom as a sequence. I.e., use top rows as context to predict bottom rows.



Source: van den Oord et al., 2018,

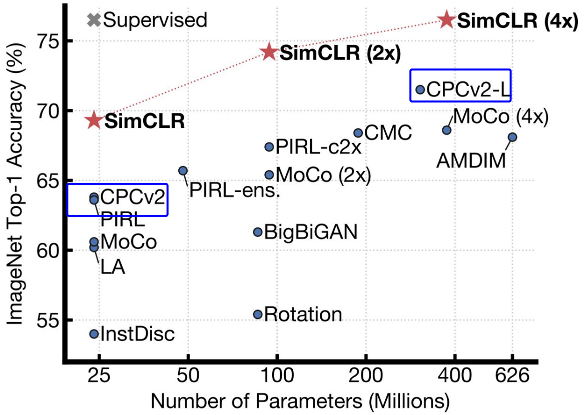
# CPC example: modeling visual context



| Method | Top-1 ACC |
|---|---|
| **Using AlexNet conv5** | |
| Video [28] | 29.8 |
| Relative Position [11] | 30.4 |
| BiGan [35] | 34.8 |
| Colorization [10] | 35.2 |
| Jigsaw [29] * | 38.1 |
| **Using ResNet-V2** | |
| Motion Segmentation [36] | 27.6 |
| Exemplar [36] | 31.5 |
| Relative Position [36] | 36.2 |
| Colorization [36] | 39.6 |
| **CPC** | **48.7** |

Table 3: ImageNet top-1 unsupervised classification results. *Jigsaw is not directly comparable to the other AlexNet results because of architectural differences.

- Compares favorably with other pretext task-based self-supervised learning method.
- Doesn't do as well compared to newer instance-based contrastive learning methods on image feature learning.



Source: van den Oord et al., 2018,

## Summary: Contrastive Representation Learning

A general formulation for contrastive learning:

$$\text{score}(f(x), f(x^+)) >> \text{score}(f(x), f(x^-))$$

InfoNCE loss: N-way classification among positive and negative samples

$$L = -\mathbb{E}_X \left[ \log \frac{\exp(s(f(x), f(x^+))}{\exp(s(f(x), f(x^+)) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-))} \right]$$

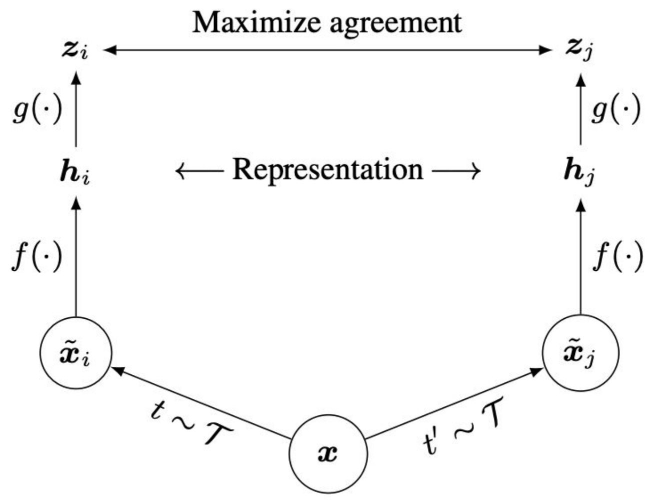
Commonly known as the InfoNCE loss (van den Oord et al., 2018)
A *lower bound* on the mutual information between *f(x)* and *f(x⁺)*

$$MI[f(x), f(x^+)] - \log(N) \geq -L$$

# Summary: Contrastive Representation Learning

**SimCLR**: a simple framework for contrastive representation learning

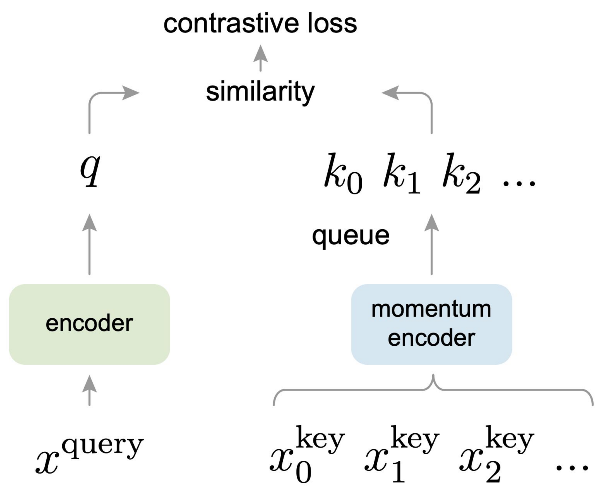
- **Key ideas**: non-linear projection head to allow flexible representation learning
- Simple to implement, effective in learning visual representation
- Requires large training batch size to be effective; large memory footprint



# Summary: Contrastive Representation Learning

**MoCo** (v1, v2): contrastive learning using momentum sample encoder

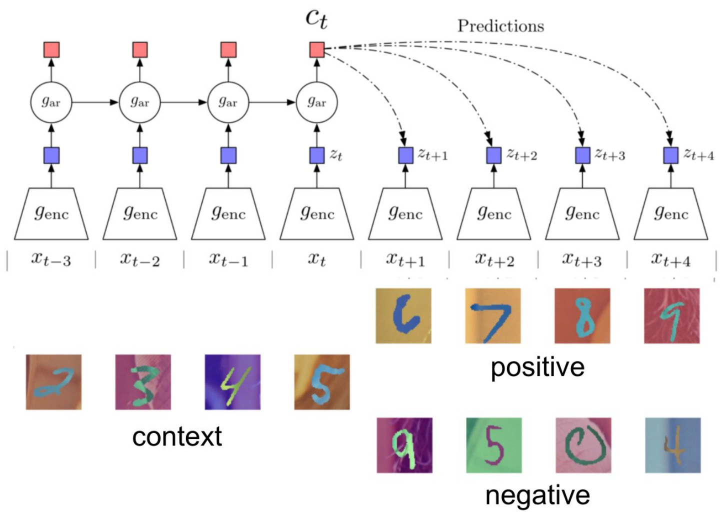
- Decouples negative sample size from minibatch size; allows large batch training without TPU
- MoCo-v2 combines the key ideas from SimCLR, i.e., nonlinear projection head, strong data augmentation, with momentum contrastive learning



# Summary: Contrastive Representation Learning

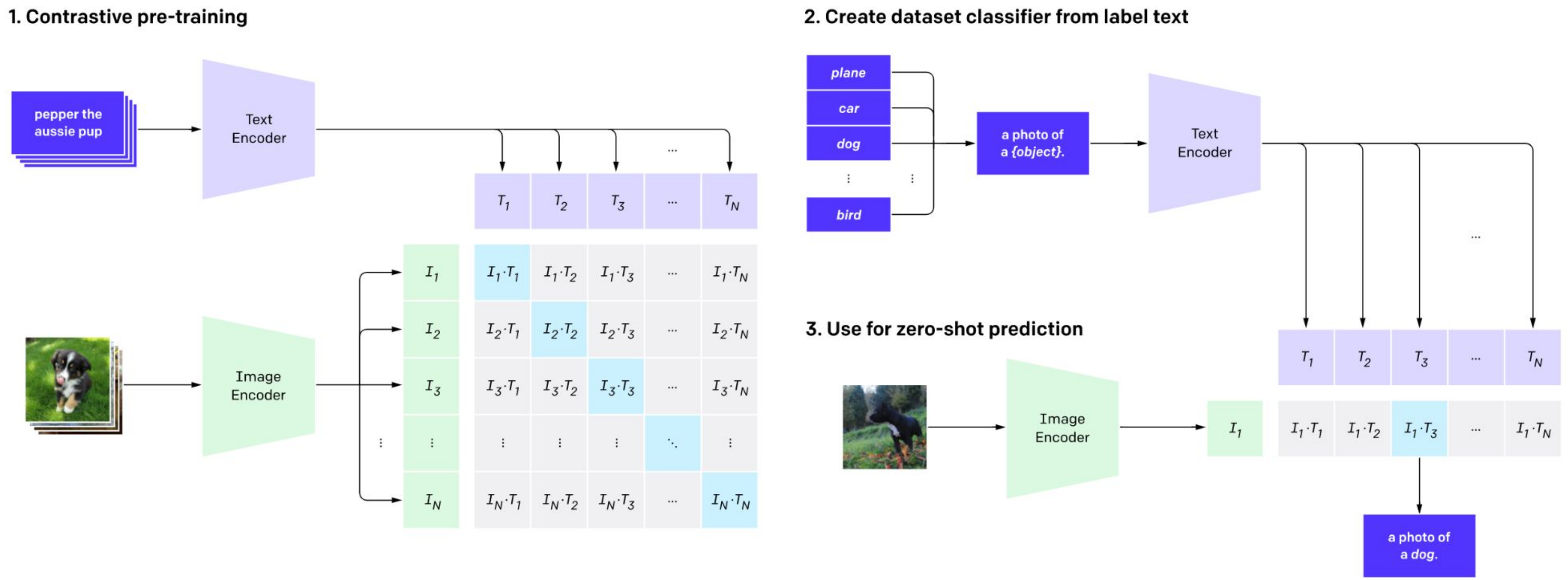
**CPC**: sequence-level contrastive learning

- Contrast “right” sequence with “wrong” sequence.
- InfoNCE loss with a time-dependent score function.
- Can be applied to a variety of learning problems, but not as effective in learning image representations compared to instance-level methods.



# Other examples: will be covered in next lecture

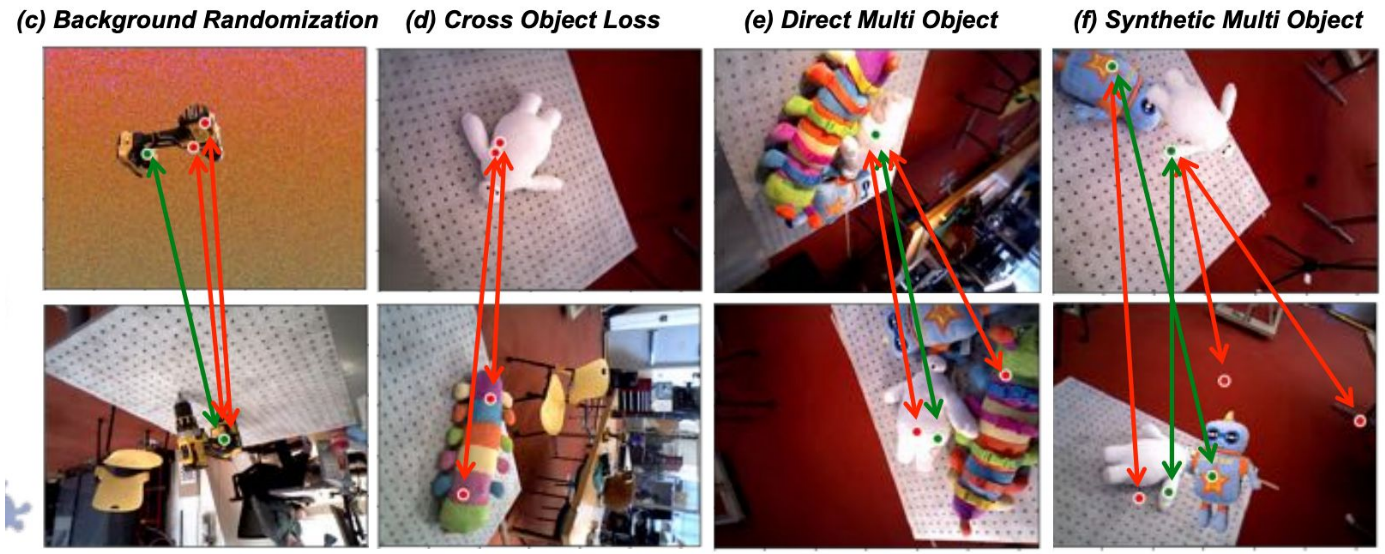
Contrastive learning between image and natural language sentences


1. Contrastive pre-training
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
$I_1$
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
a photo of a dog.

CLIP (*Contrastive Language–Image Pre-training*) Radford *et al.*, 2021

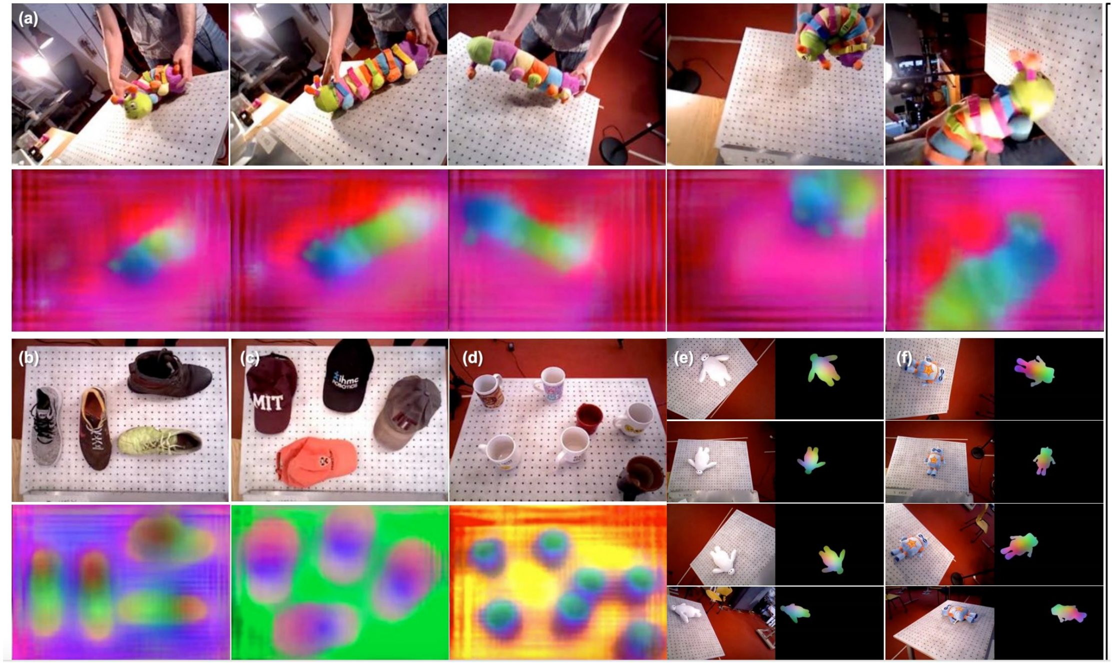
# Other examples

Contrastive learning on pixel-wise feature descriptors



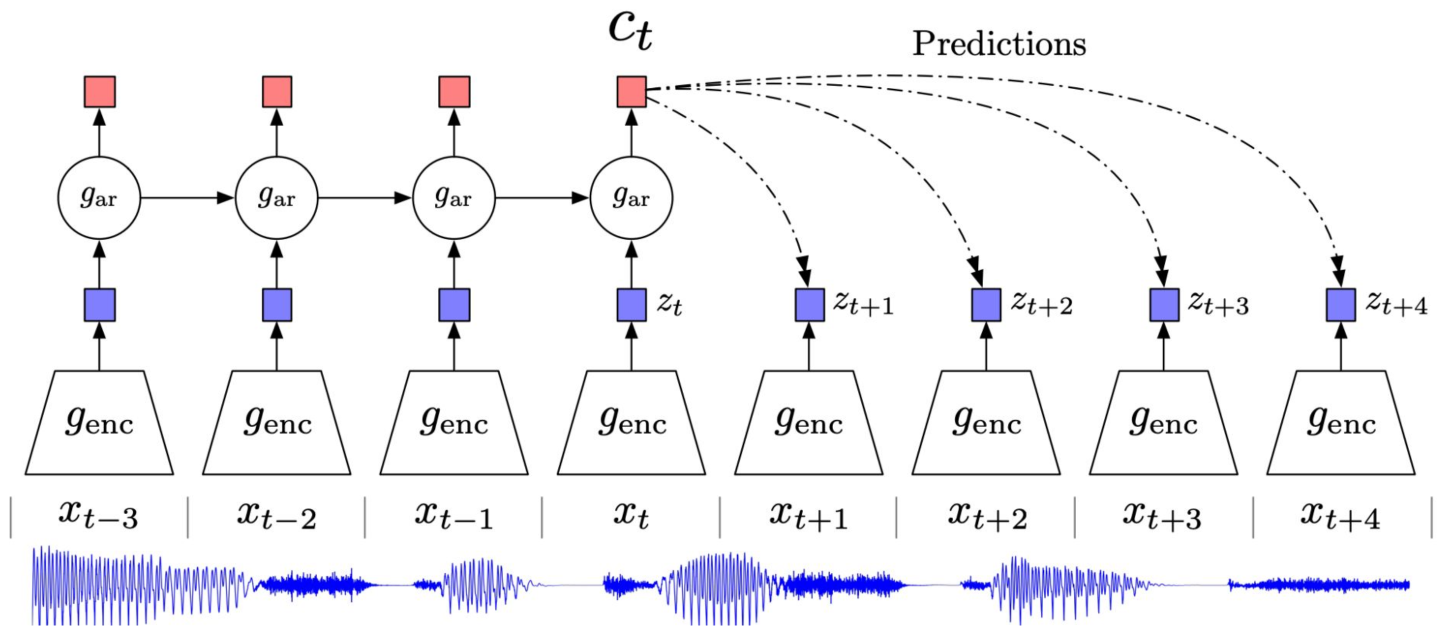
Dense Object Net, Florence et al., 2018

## Other examples



Dense Object Net, Florence et al., 2018

## CPC example: modeling audio sequences



Source: van den Oord et al., 2018,

# CPC example: modeling audio sequences

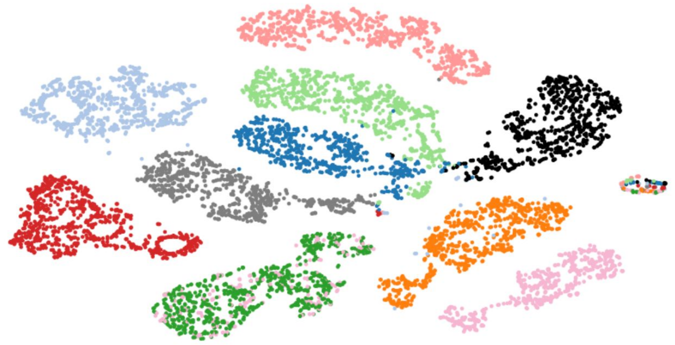


Figure 2: t-SNE visualization of audio (speech) representations for a subset of 10 speakers (out of 251). Every color represents a different speaker.

| Method | ACC |
|---|---|
| **Phone classification** | |
| Random initialization | 27.6 |
| MFCC features | 39.7 |
| CPC | 64.6 |
| Supervised | 74.6 |
| **Speaker classification** | |
| Random initialization | 1.87 |
| MFCC features | 17.6 |
| CPC | 97.4 |
| Supervised | 98.5 |

Linear classification on trained representations (LibriSpeech dataset)

Source: van den Oord et al., 2018,

## Other examples

Our robot then grasps the best match
for an instance-*specific* descriptor
12x

## LLMs

## Recap: Self-Supervised

Skills needed to classify dogs

Identify different textures
Identify different colors
Identify which pixels are parts of an object
Understand parts of objects which make up the whole
Understand the context of the object/animal in the image
Understand lighting conditions
Understand objects close up/far away
Have common-sense reasoning skills
Learn which features are associated with which dog

# Recap: Self-Supervised

| | |
|---|---|
| General visual modelling skills (Can be learned with Self Supervised) | Identify different textures<br>Identify different colors<br>Identify which pixels are parts of an object<br>Understand parts of objects which make up the whole<br>Understand the context of the object/animal in the image<br>Understand lighting conditions<br>Understand objects close up/far away<br>Have common-sense reasoning skills |
| Specific (Supervised) | Learn which features are associated with which dog |

# Self-Supervised for Language

Skills needed to classify book genres

Knowledge of words/letters
Knowledge of grammar
Meanings of words
Understanding context of words
Keeping track of entities over time
Understanding expressions/idioms
Understanding tone
Learning which features are associated with each genre

## Self-Supervised for Language

General language modelling skills (Can be learned with Self Supervised)

- Knowledge of words/letters
- Knowledge of grammar
- Meanings of words
- Understanding context of words
- Keeping track of entities over time
- Understanding expressions/idioms
- Understanding tone

Specific (Supervised)

- Learning which features are associated with each genre

## Self-Supervised for Language

| | |
|---|---|
| General language modelling skills (Can be learned with Self Supervised) | Knowledge of words/letters<br>Knowledge of grammar<br>Meanings of words<br>Understanding context of words<br>Keeping track of entities over time<br>Understanding expressions/idioms<br>Understanding tone |
| Specific (Supervised) | Learning which features are associated with each genre |

Want: model with general understanding of language (Language model!)

# LLMs

Building LLMs: Pre-training objectives + architectures

- Encoder only
- Decoder only
- Encoder Decoder

GPT

Gradient-Free Performance Improvement

## LLMs

**Building LLMs: Pre-training objectives + architectures**

- **Encoder only**
- **Decoder only**
- **Encoder Decoder**

GPT

Gradient-Free Performance Improvement

# Last time

## Pre-training tasks

rotation

Gidaris et al. 2018)

in-painting

Pathak et al., 2016z

colorization

Source: Google AI blog post

1. Solving the pretext tasks allow the model to learn good features.
2. We can automatically generate labels for the pretext tasks.

# Last time

## Pre-training tasks

Common recipe in vision:

1. Take original data
2. Remove/obscure information
3. Ask model to get back to the original instance

colorization

Source: Google AI blog post

rotation

Gidaris et al. 2018)

in-painting

Pathak et al., 2016z

# Last time

## Pre-training tasks

rotation

in-painting

colorization

What to use as pre-training task for language?

# Encoder only LLMs

It’s cold today! Don’t forget to wear a ______________.

The ______________ is a popular tourist attraction in Seattle.

I missed ___ bus.

I had 3 pencils and lost one so now I have ______ pencils.

# Encoder only LLMs

It’s cold today! Don’t forget to wear a jacket / coat / sweater.

The Space Needle is a popular tourist attraction in Seattle.

I missed the bus.

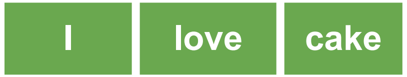
I had 3 pencils and lost one so now I have 2 / two pencils.

# Encoder only LLMs

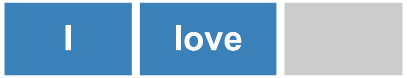
| | |
|---|---|
| It's cold today! Don't forget to wear a jacket / coat / sweater. | **Common Sense** |
| The Space Needle is a popular tourist attraction in Seattle. | **Factual knowledge** |
| I missed the bus. | **Grammar** |
| I had 3 pencils and lost one so now I have 2 / two pencils. | **Math/ Reasoning** |

## LLMs

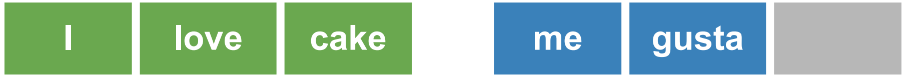
**Encoder Only:**



**Decoder Only:**

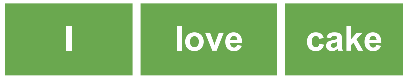


**Encoder-Decoder:**

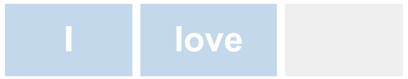


## LLMs

**Encoder Only:**



**Decoder Only:**



**Encoder-Decoder:**

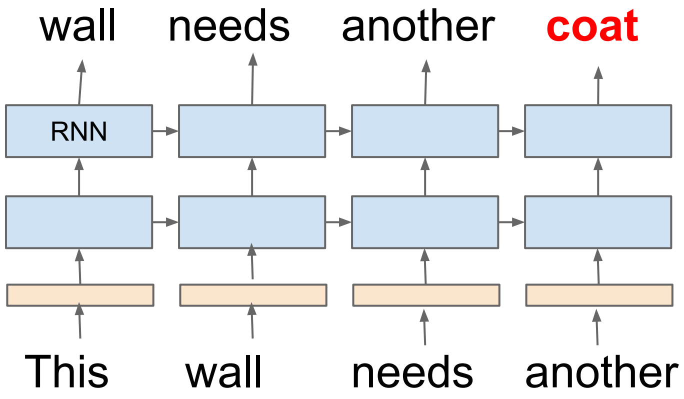
I
love
cake
me
gusta

# ELMo (Embeddings from Language Models)

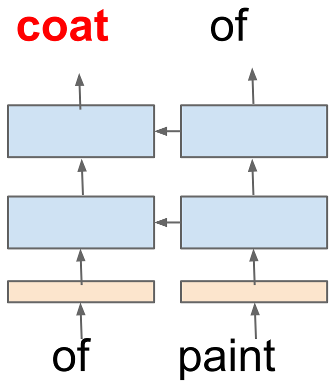
## Pre-training task

This wall needs another coat of paint

**Predict words based on previous words**



**Predict words based on following words**

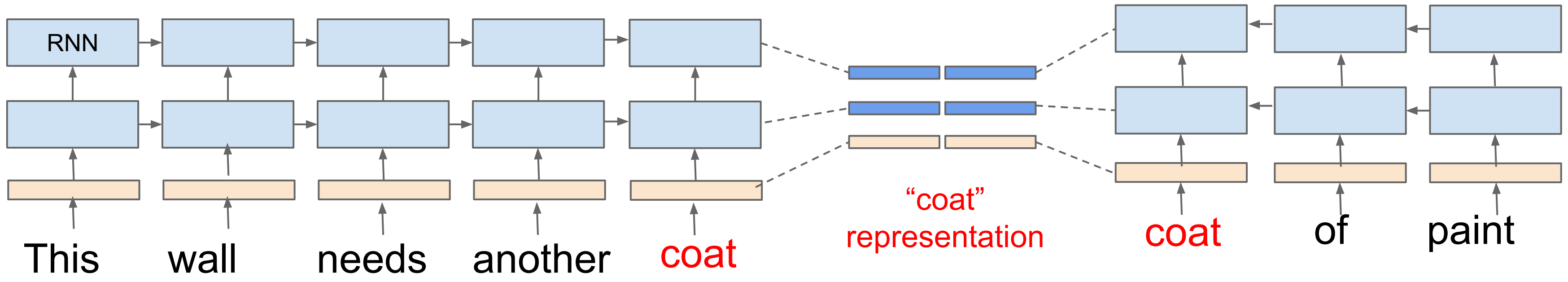


Peters et al. Deep contextualized word representations. 2018.

# ELMo (Embeddings from Language Models)

## Application to downstream tasks

This wall needs another coat of paint

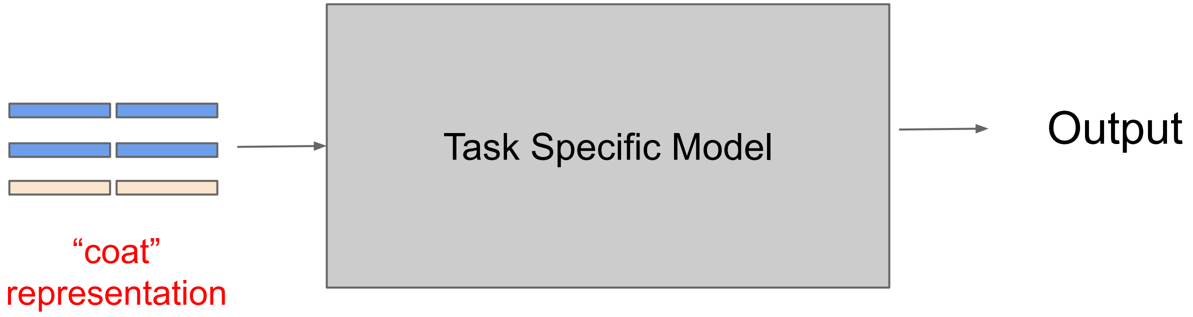


Peters et al. Deep contextualized word representations. 2018.

# ELMo (Embeddings from Language Models)

## Application to downstream tasks

This wall needs another coat of paint

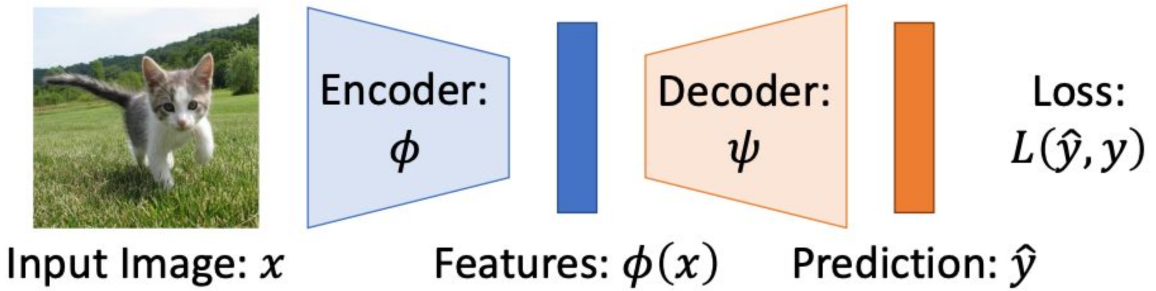


Fine-tune

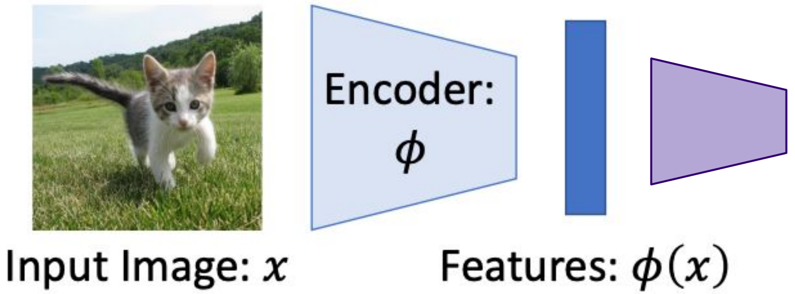
Peters et al. Deep contextualized word representations. 2018.

# Vision: Pre-train and then fine-tune

**Step 1:** Pretrain a network on a pretext task that doesn't require supervision



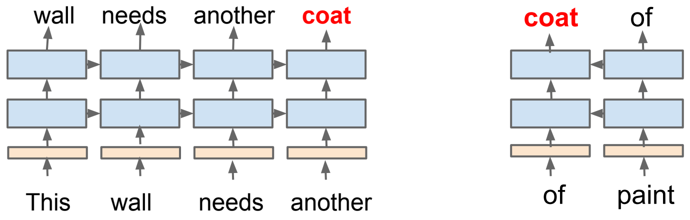
**Step 2:** Transfer encoder to downstream tasks via linear classifiers, KNN, finetuning



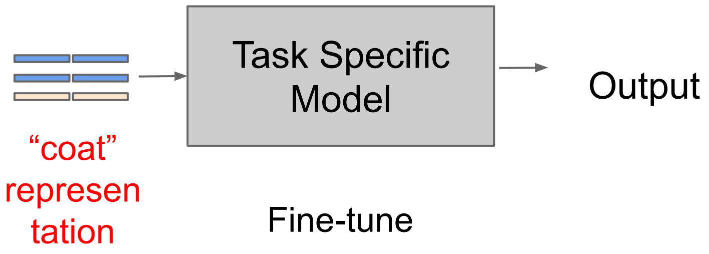
**Downstream tasks:**
Image classification, object detection, semantic segmentation

# Language: Pre-train and then fine-tune

**Step 1:** Pretrain a network on a pretext task that doesn't require supervision



**Step 2:** Transfer encoder to downstream tasks via linear classifiers, KNN, finetuning



**Downstream tasks:**
Sentiment classification, NLI, …

# Encoder only LLMs

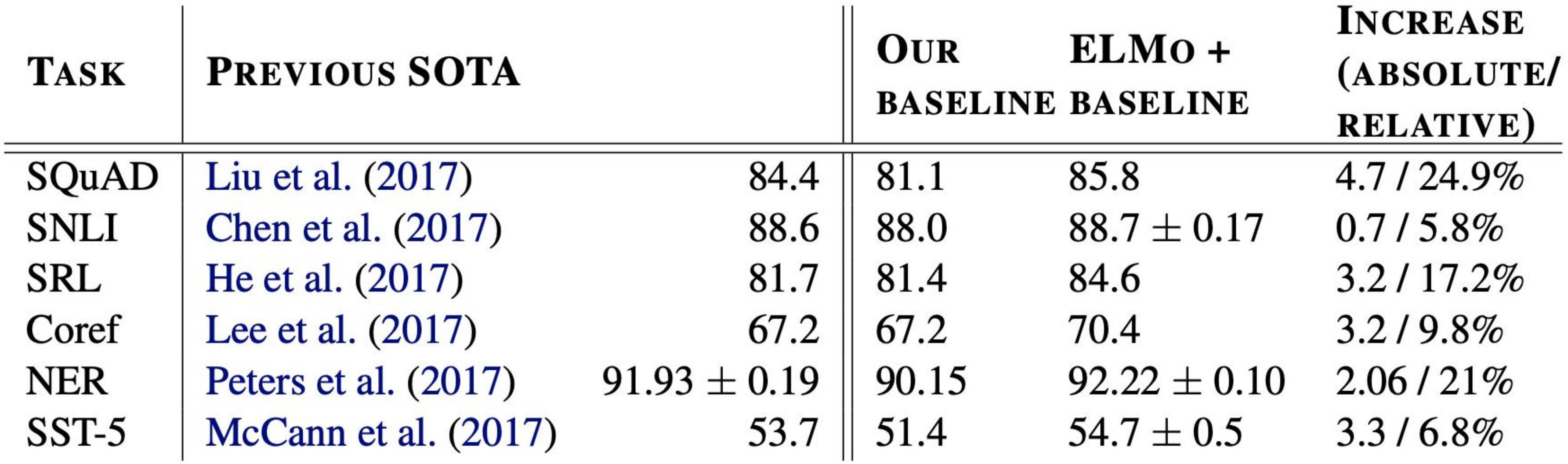
It’s cold today! Don’t forget to wear a coat.

This wall needs another coat of paint

$$\text{coat} = \begin{bmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{bmatrix}$$

Glove embeddings use the same vector for every instance of a word, no matter the context!

# ELMo Results



| TASK | PREVIOUS SOTA | | OUR BASELINE | ELMO + BASELINE | INCREASE (ABSOLUTE/ RELATIVE) |
|---|---|---|---|---|---|
| SQuAD | Liu et al. (2017) | 84.4 | 81.1 | 85.8 | 4.7 / 24.9% |
| SNLI | Chen et al. (2017) | 88.6 | 88.0 | 88.7 ± 0.17 | 0.7 / 5.8% |
| SRL | He et al. (2017) | 81.7 | 81.4 | 84.6 | 3.2 / 17.2% |
| Coref | Lee et al. (2017) | 67.2 | 67.2 | 70.4 | 3.2 / 9.8% |
| NER | Peters et al. (2017) | 91.93 ± 0.19 | 90.15 | 92.22 ± 0.10 | 2.06 / 21% |
| SST-5 | McCann et al. (2017) | 53.7 | 51.4 | 54.7 ± 0.5 | 3.3 / 6.8% |

Source: Peters et al. Deep contextualized word representations. 2018.

## ELMo

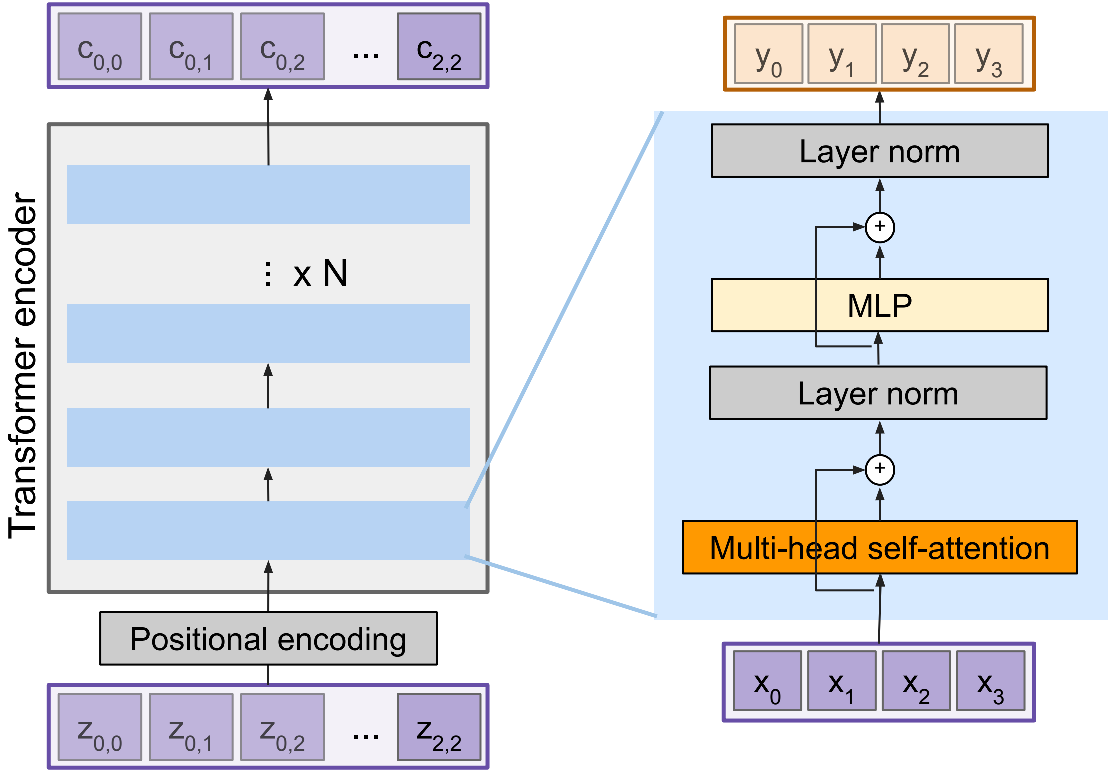
Used RNNs… what about transformers?

Let’s combine:

(1) ELMo approach of gathering general knowledge about world

(2) Powerful multi-purpose architecture that is the transformer

# Recall: Transformer encoder block



**Transformer Encoder Block:**

**Inputs**: Set of vectors **x**
**Outputs**: Set of vectors **y**

Self-attention is the only interaction between vectors.

Layer norm and MLP operate independently per vector.

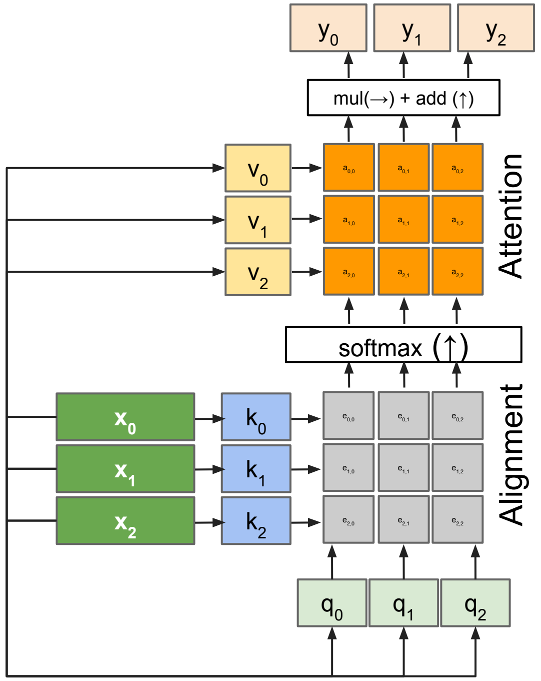
Highly scalable, highly parallelizable, but high memory usage.

Vaswani et al, "Attention is all you need", NeurIPS 2017

## Encoder Only: Bert (Bidirectional Encoder Representations from Transformers)

**Input:** Text sequence

**Output**: Feature Vector



$y_0$
$y_1$
$y_2$
mul(→) + add (↑)
$v_0$
$v_1$
$v_2$
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$
Attention
softmax (↑)
$x_0$
$x_1$
$x_2$
$k_0$
$k_1$
$k_2$
$e_{0,0}$ $e_{0,1}$ $e_{0,2}$
$e_{1,0}$ $e_{1,1}$ $e_{1,2}$
$e_{2,0}$ $e_{2,1}$ $e_{2,2}$
Alignment
$q_0$
$q_1$
$q_2$

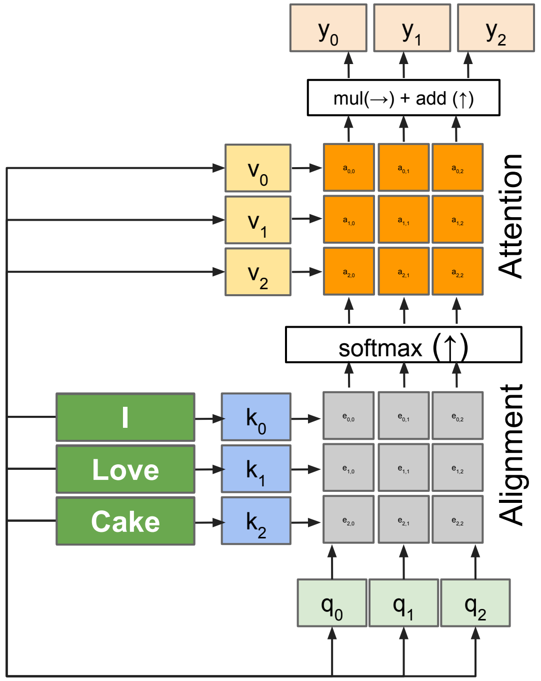
**Outputs:**
context vectors: **y** (shape: $D_v$)

Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.

## Encoder Only: Bert

**Input:** Text sequence

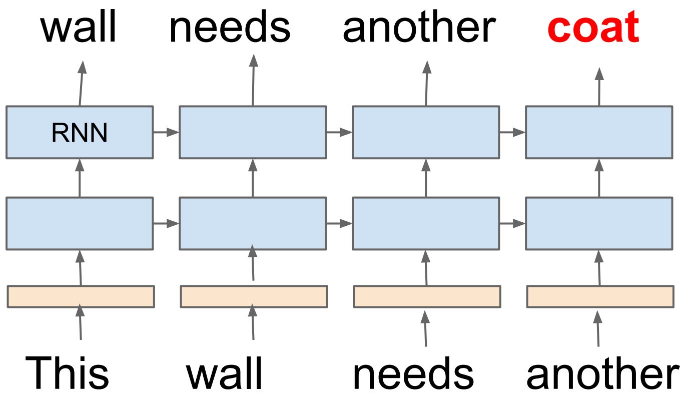
**Output**: Feature Vector

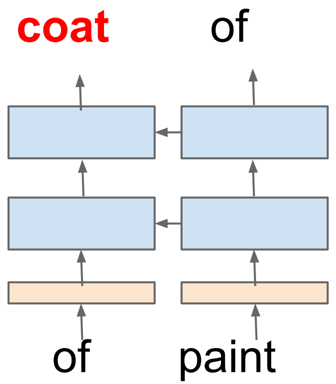


**Outputs:**
context vectors: **y** (shape: $D_v$)

Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.

# ELMo (Embeddings from Language Models)



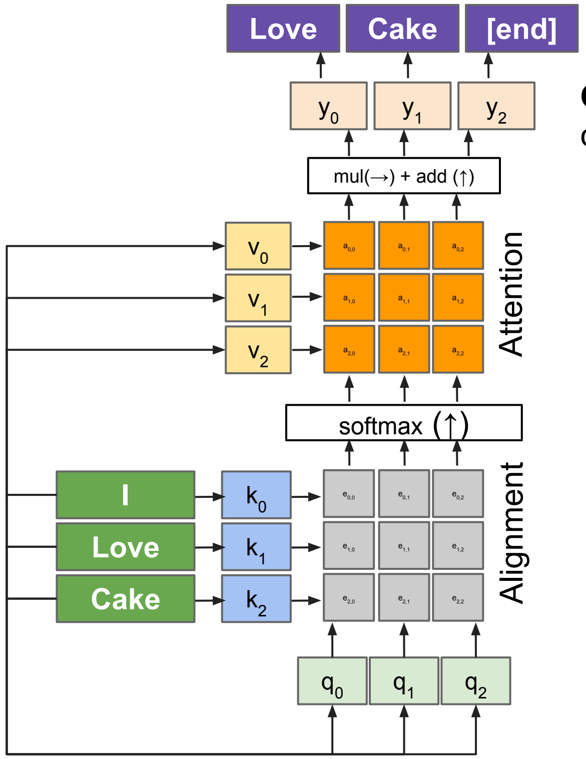


## Encoder Only: Bert

**Input:** Text sequence

**Output**: Feature Vector

**What information do the y vectors contain?**



**Outputs:**
context vectors: **y** (shape: $D_v$)

Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.

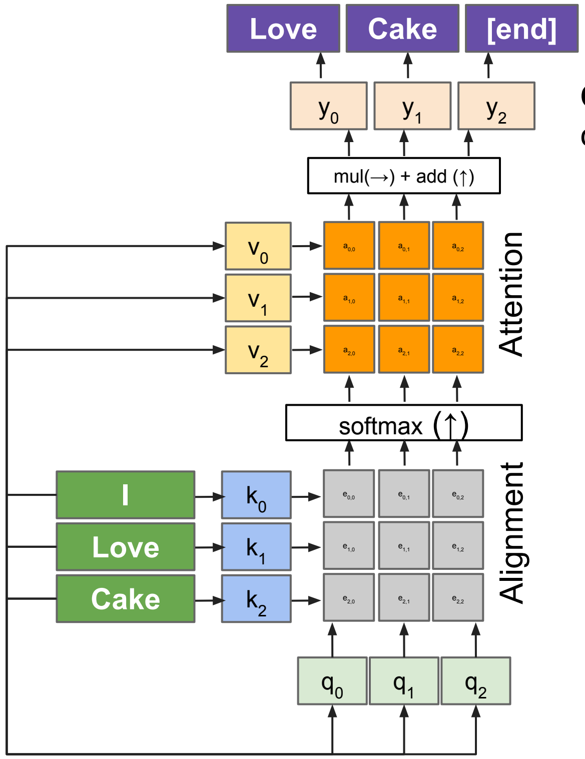
## Encoder Only: Bert

**Input:** Text sequence

**Output**: Feature Vector

**What information do the y vectors contain?**

**Just copying input**



**Outputs:**
context vectors: **y** (shape: $D_v$)

Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.

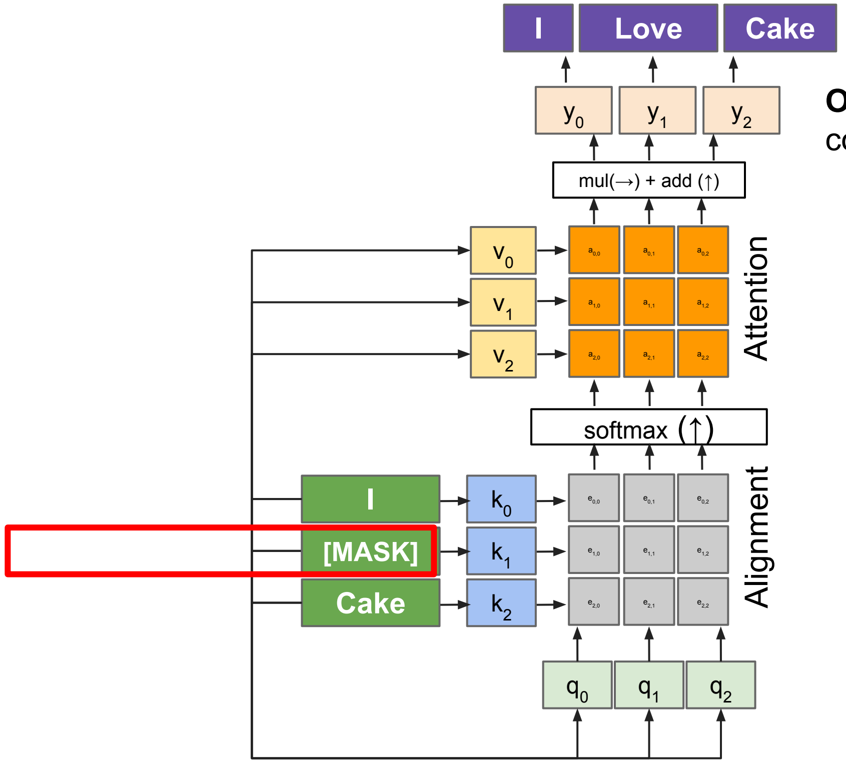
# Encoder Only: Bert

**Input:** Text sequence

**Output**: Feature Vector

**Randomly select 15% of tokens.**

- **80% - [MASK]**
- **10% - random token**
- **10% - keep same**



**Outputs:**
context vectors: **y** (shape: $D_v$)

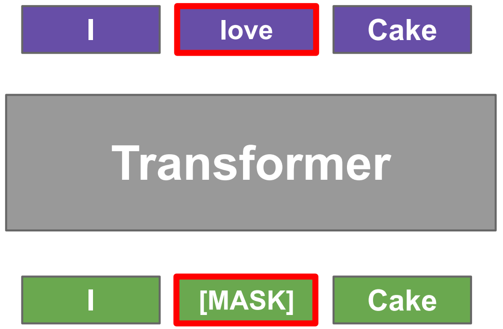
Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.

## Encoder Only: Bert

**Input:** Text sequence

**Output**: Feature Vector

1 Masked Language Model



**Randomly select 15% of tokens.**
**80% - [MASK]**
**10% - random token**
**10% - keep same**

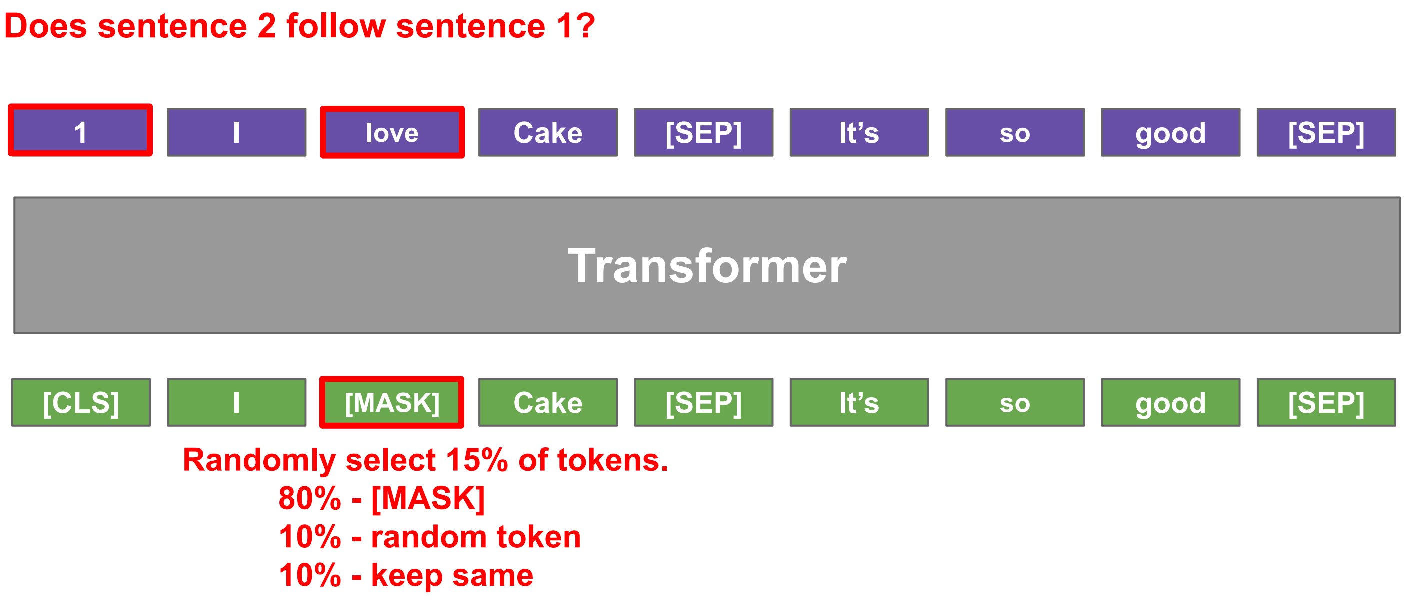
Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.

## Encoder Only: Bert

**Input:** Text sequence

**Output**: Feature Vector

1 Masked Language Model
2. Next Sentence Prediction



Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.

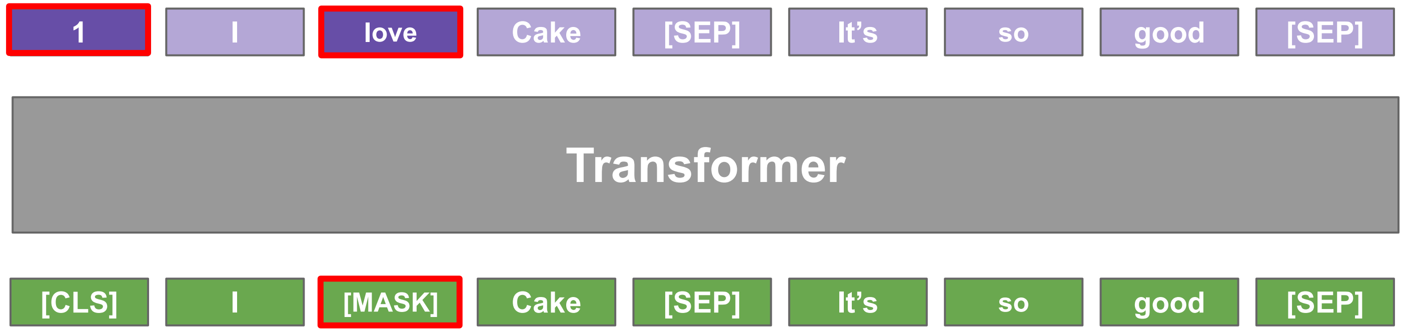
## Encoder Only: Bert

**Input:** Text sequence

**Output**: Feature Vector

1 Masked Language Model
2. Next Sentence Prediction

Does sentence 2 follow sentence 1?

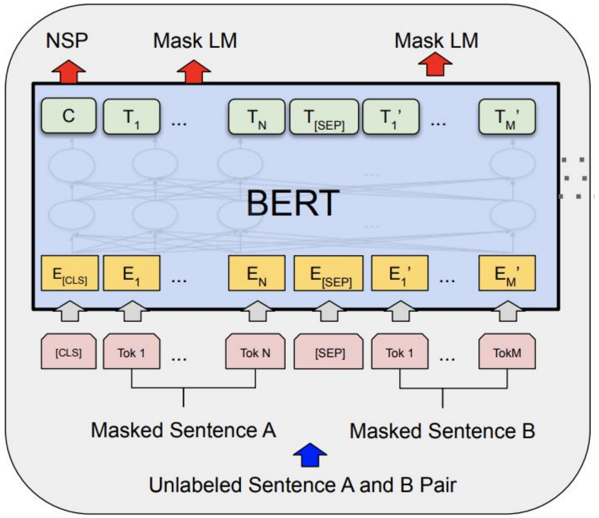


Randomly select 15% of tokens.
- 80% - [MASK]
- 10% - random token
- 10% - keep same

Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.

## Encoder Only: Bert

Application to downstream tasks



Pre-training

Image Source: Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.

## Encoder Only: Bert

### Application to downstream tasks

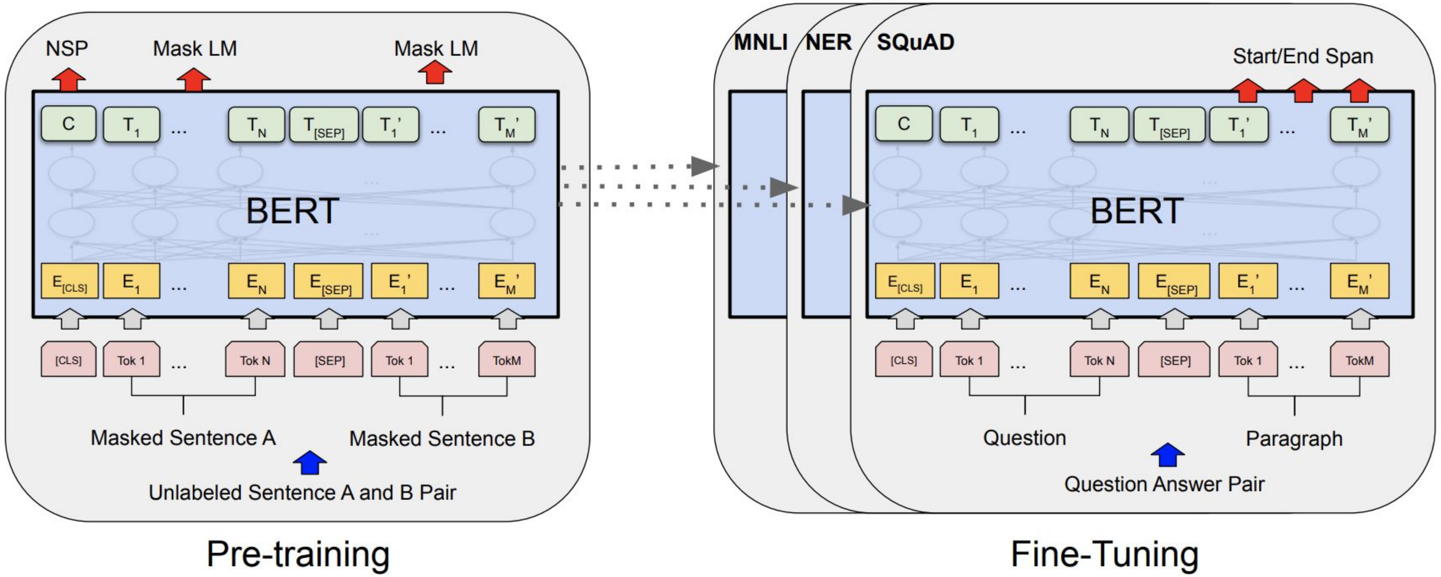
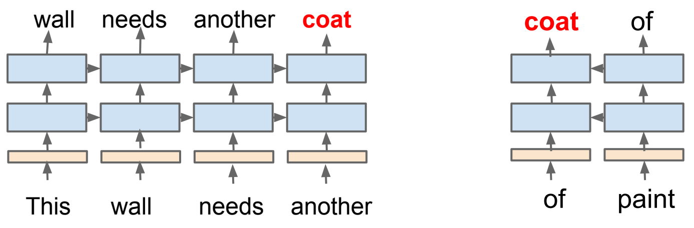


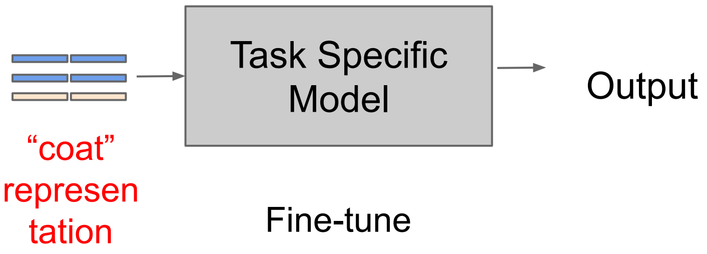
Image Source: Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.

# ELMO: Two step process

**Step 1:** Pretrain a network on a pretext task that doesn't require supervision



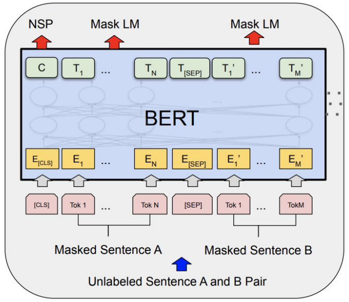
**Step 2:** Transfer encoder to downstream tasks via linear classifiers, KNN, finetuning



**Downstream tasks:** Sentiment classification, NLI, …

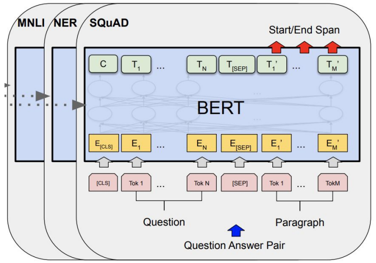
# BERT: Two step process

**Step 1:** Pretrain a network on a pretext task that doesn't require supervision



Pre-training

**Step 2:** Transfer encoder to downstream tasks via linear classifiers, KNN, finetuning

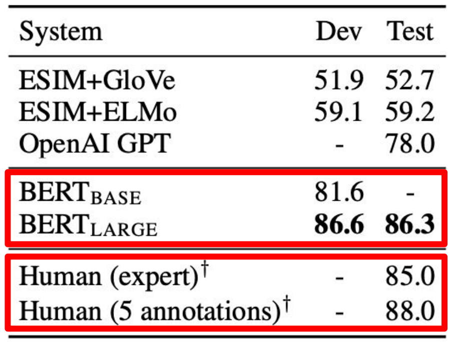


Fine-Tuning

**Downstream tasks:** Sentiment classification, NLI, …

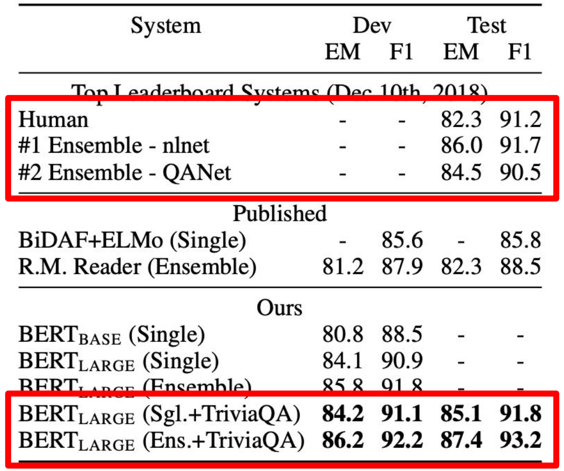
Image Source: Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.

# BERT Results



| System | Dev | Test |
|---|---|---|
| ESIM+GloVe | 51.9 | 52.7 |
| ESIM+ELMo | 59.1 | 59.2 |
| OpenAI GPT | - | 78.0 |
| BERT$_{\text{BASE}}$ | 81.6 | - |
| BERT$_{\text{LARGE}}$ | **86.6** | **86.3** |
| Human (expert)[†] | - | 85.0 |
| Human (5 annotations)[†] | - | 88.0 |

Table 4: SWAG Dev and Test accuracies. [†]Human performance is measured with 100 samples, as reported in the SWAG paper.



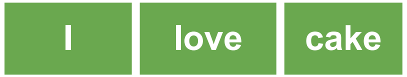
| System | Dev | | Test | |
|---|---|---|---|---|
| | EM | F1 | EM | F1 |
| **Top Leaderboard Systems (Dec 10th, 2018)** | | | | |
| Human | - | - | 82.3 | 91.2 |
| #1 Ensemble - nlnet | - | - | 86.0 | 91.7 |
| #2 Ensemble - QANet | - | - | 84.5 | 90.5 |
| **Published** | | | | |
| BiDAF+ELMo (Single) | - | 85.6 | - | 85.8 |
| R.M. Reader (Ensemble) | 81.2 | 87.9 | 82.3 | 88.5 |
| **Ours** | | | | |
| BERT$_{\text{BASE}}$ (Single) | 80.8 | 88.5 | - | - |
| BERT$_{\text{LARGE}}$ (Single) | 84.1 | 90.9 | - | - |
| BERT$_{\text{LARGE}}$ (Ensemble) | 85.8 | 91.8 | - | - |
| BERT$_{\text{LARGE}}$ (Sgl.+TriviaQA) | **84.2** | **91.1** | **85.1** | **91.8** |
| BERT$_{\text{LARGE}}$ (Ens.+TriviaQA) | **86.2** | **92.2** | **87.4** | **93.2** |

Table 2: SQuAD 1.1 results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.

Image Source: [Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.]

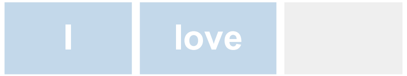
# LLMs

## Encoder Only:



**ELMO:** Bi-directional next word prediction,
**BERT:** Masked language objective, Next Sentence Prediction

## Decoder Only:



## Encoder-Decoder:

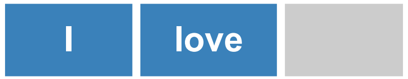
I
love
cake
me
gusta

# LLMs

## Encoder Only:

**ELMO:** Bi-directional next word prediction,
**BERT:** Masked language objective, Next Sentence Prediction

## Decoder Only:

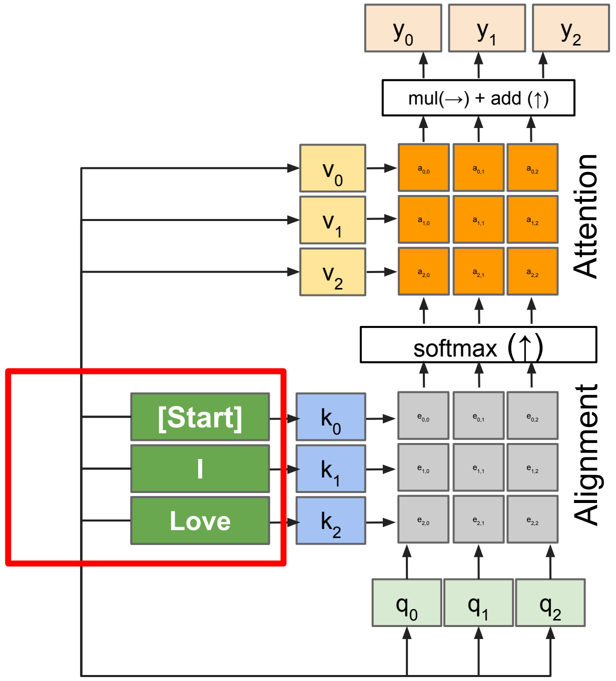


## Encoder-Decoder:

I
love
cake
me
gusta

## Decoder Only: GPT

**Input:** Text sequence

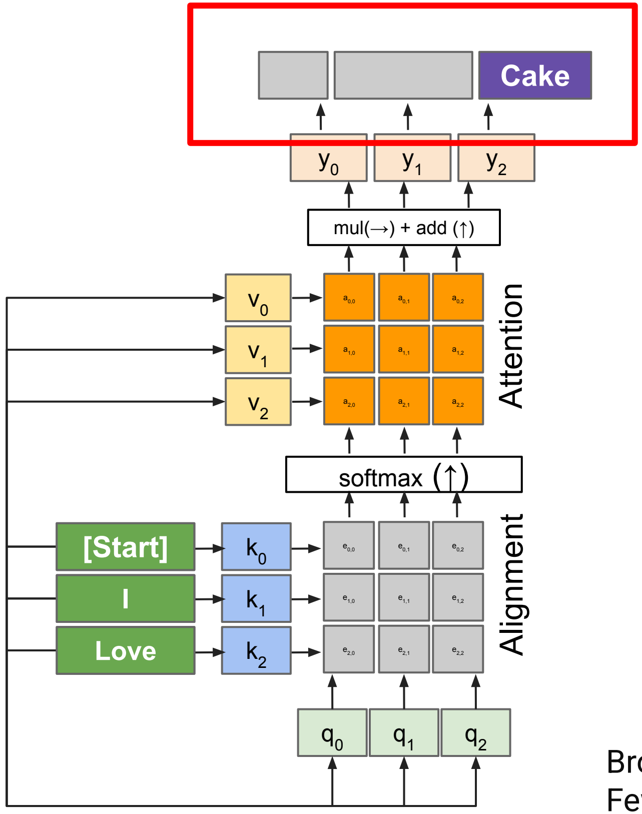


Brown et al. Language Models are Few-Shot Learners. 2020.

## Decoder Only: GPT

**Input:** Text sequence

**Output**: Completed text sequence



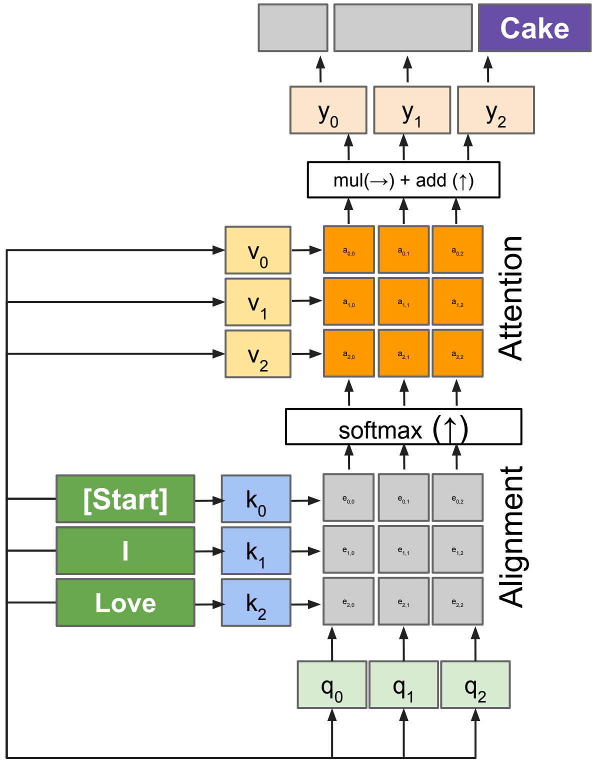
Brown et al. Language Models are Few-Shot Learners. 2020.

## Decoder Only: GPT

**Input:** Text sequence

**Output**: Completed text sequence

**Cons: Need to process entire sentence in order to get loss from one word - not very much signal for the amount of processing**



Brown et al. Language Models are Few-Shot Learners. 2020.

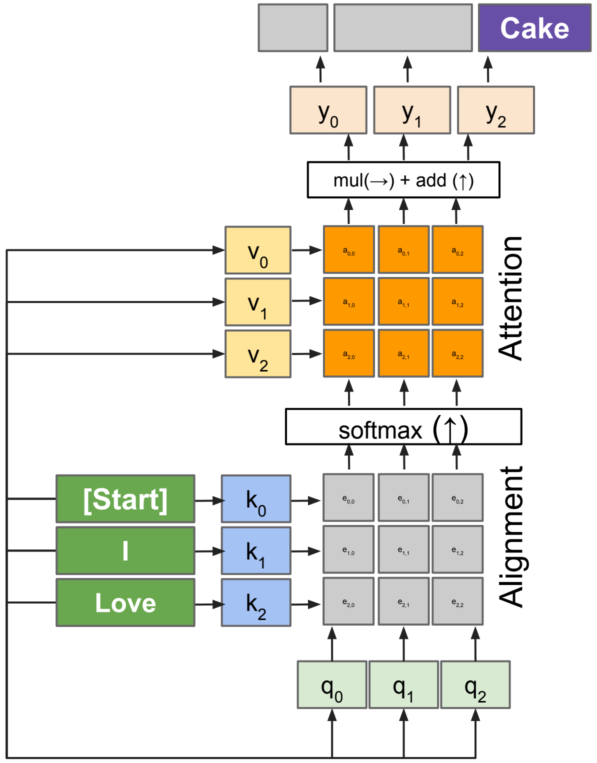
## Decoder Only: GPT

**Input:** Text sequence

**Output**: Completed text sequence

**Cons: Need to process entire sentence in order to get loss from one word - not very much signal for the amount of processing**

**Solution: predict each word given previous words so far**



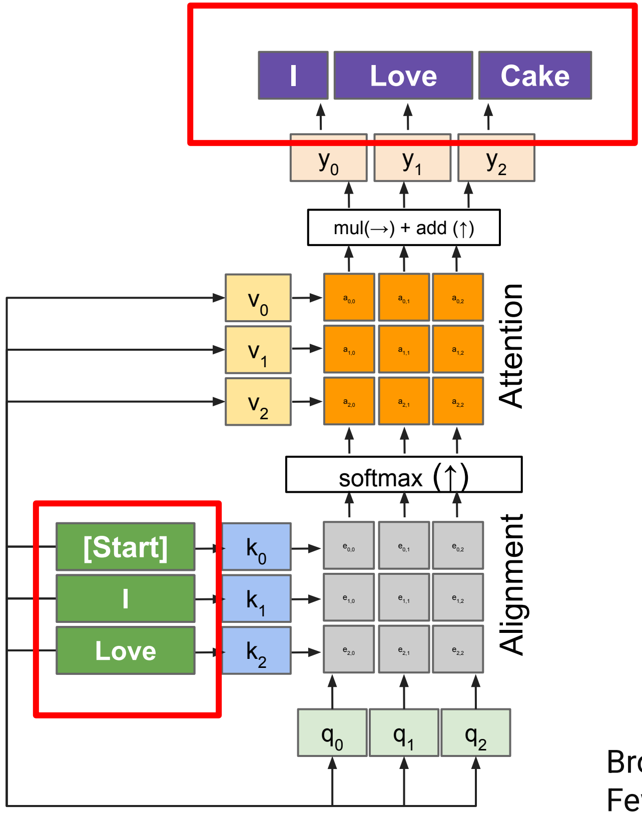
Brown et al. Language Models are Few-Shot Learners. 2020.

## Decoder Only: GPT

**Input:** Text sequence
**Output**: Completed text sequence

**Cons: Need to process entire sentence in order to get loss from one word - not very much signal for the amount of processing**

**Solution: predict each word given previous words so far**



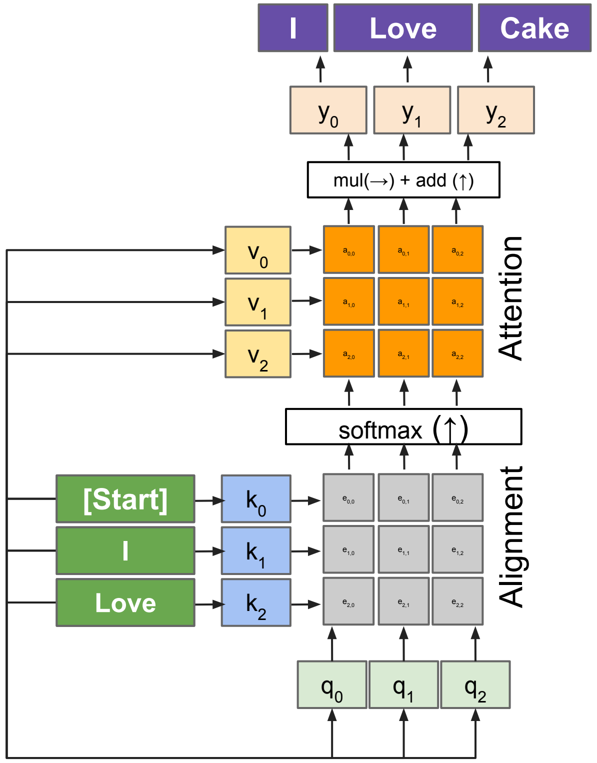
Brown et al. Language Models are Few-Shot Learners. 2020.

## Decoder Only: GPT

**Input:** Text sequence

**Output**: Completed text sequence

**What's wrong with this?**



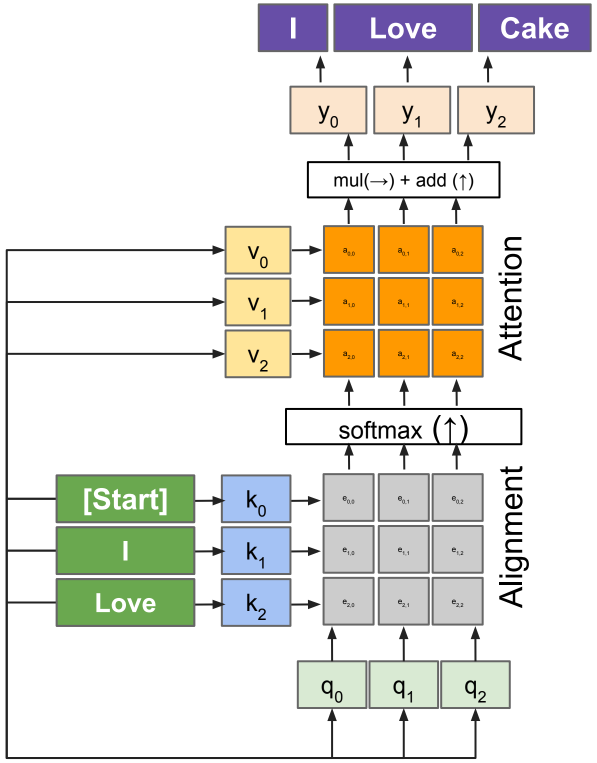
Brown et al. Language Models are Few-Shot Learners. 2020.

## Decoder Only: GPT

**Input:** Text sequence
**Output**: Completed text sequence

**What's wrong with this?**

**It can see the answer!**



Brown et al. Language Models are Few-Shot Learners. 2020.

## Decoder Only: GPT

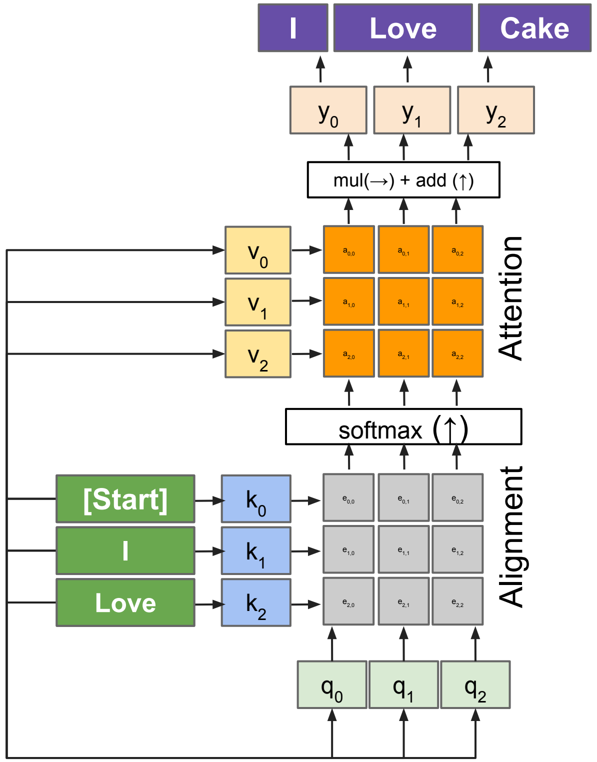
**Input:** Text sequence

**Output**: Completed text sequence

**What's wrong with this?**

**It can see the answer!**

**Solution: zero out values from future words**



Brown et al. Language Models are Few-Shot Learners. 2020.

# Decoder Only: GPT

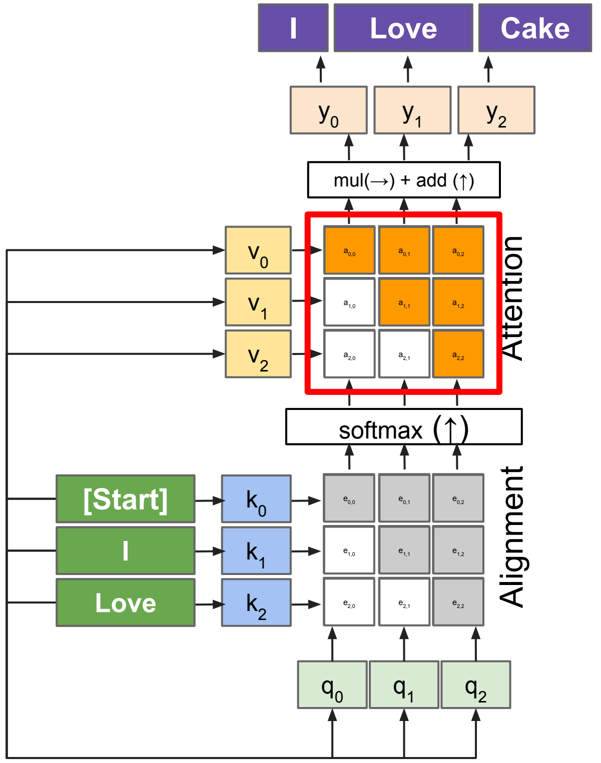
**Input:** Text sequence

**Output**: Completed text sequence

**What's wrong with this?**

**It can see the answer!**

**Solution: zero out values from future words**



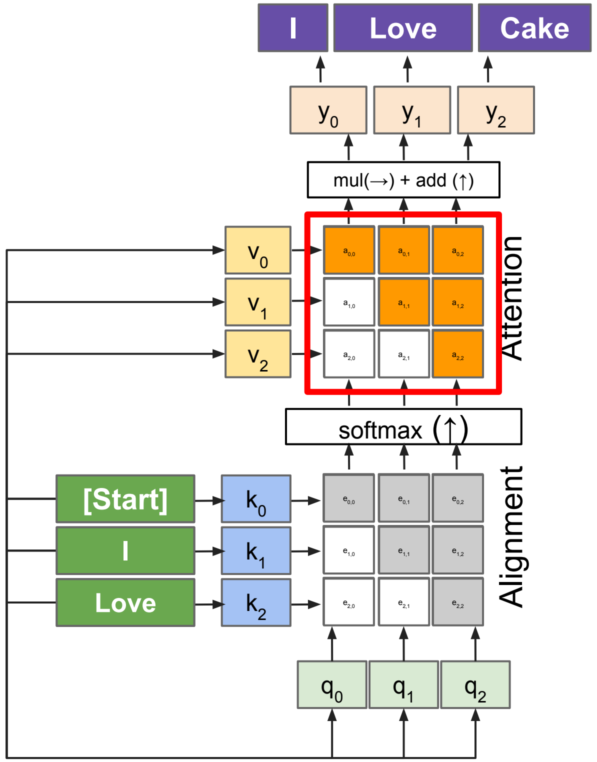
Brown et al. Language Models are Few-Shot Learners. 2020.

## Decoder Only: GPT

**Input:** Text sequence

**Output**: Completed text sequence

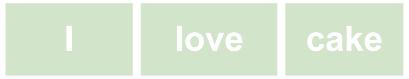
**To pre-train: predict next words from previous words for large text corpus**



Brown et al. Language Models are Few-Shot Learners. 2020.

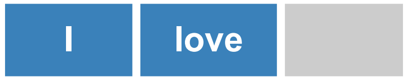
## LLMs

**Encoder Only:**



**ELMO:** Bi-directional next word prediction,
**BERT:** Masked language objective, Next Sentence Prediction

**Decoder Only:**



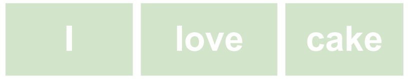
**GPT:** next token prediction (autoregressive)

**Encoder-Decoder:**

I
love
cake
me
gusta

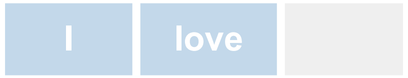
# LLMs

## Encoder Only:



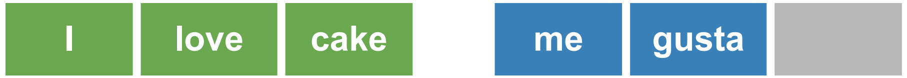
**ELMO:** Bi-directional next word prediction,
**BERT:** Masked language objective, Next Sentence Prediction

## Decoder Only:

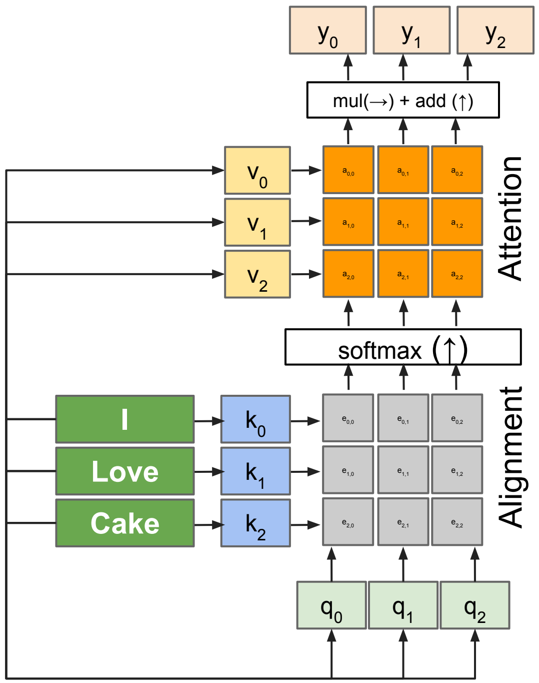


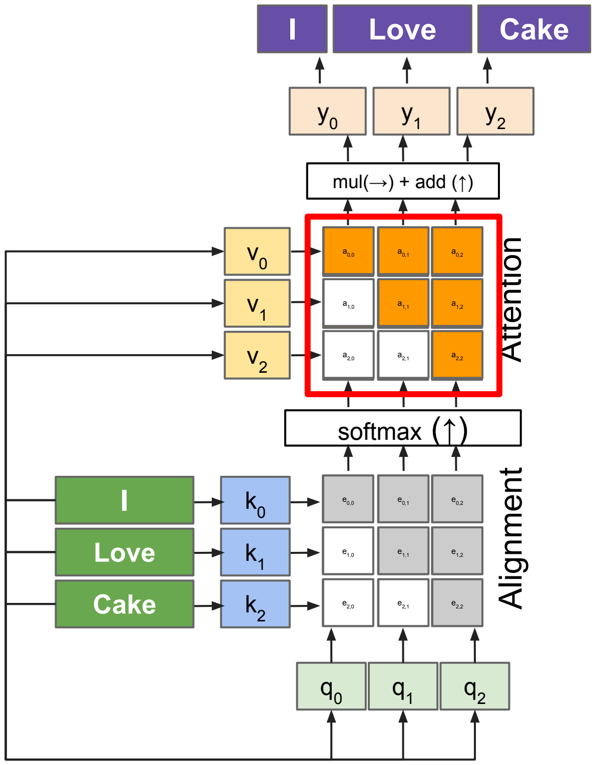
**GPT:** next token prediction (autoregressive)

## Encoder-Decoder:

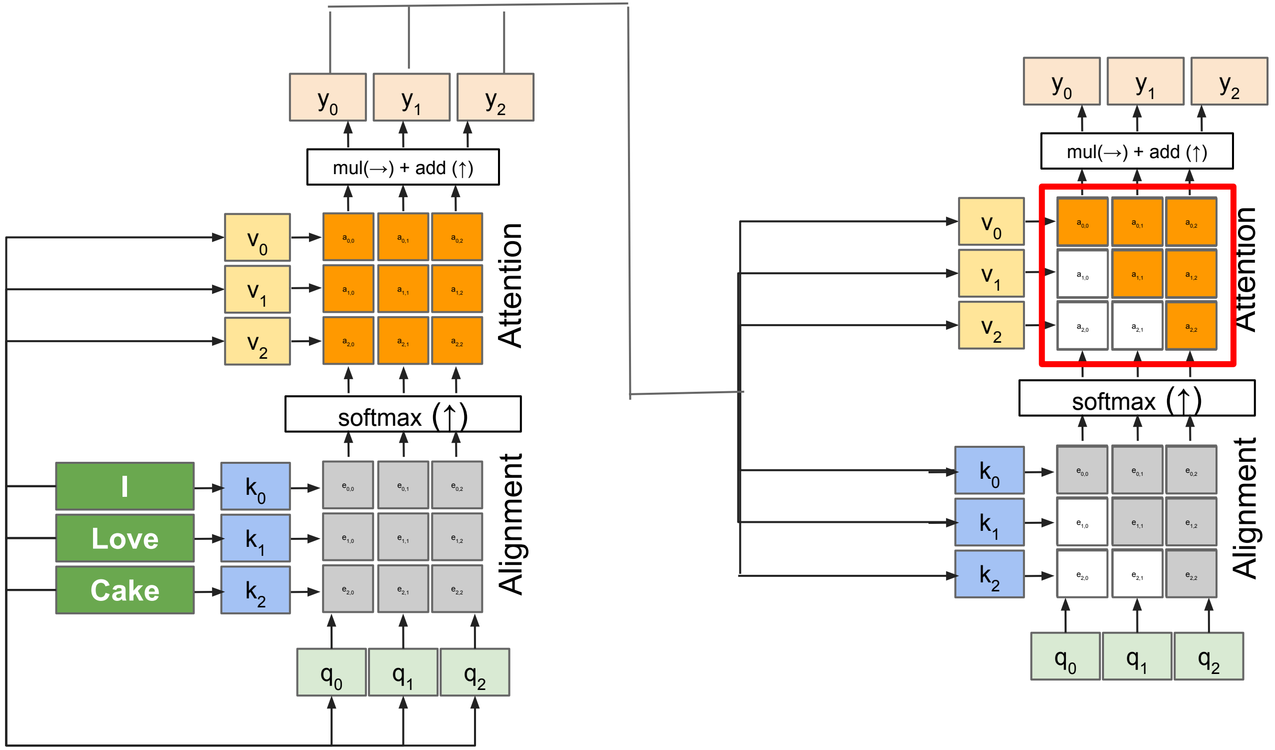


## Encoder-Decoder: Generate text based on previously generated text and the meaning of a separate sequence

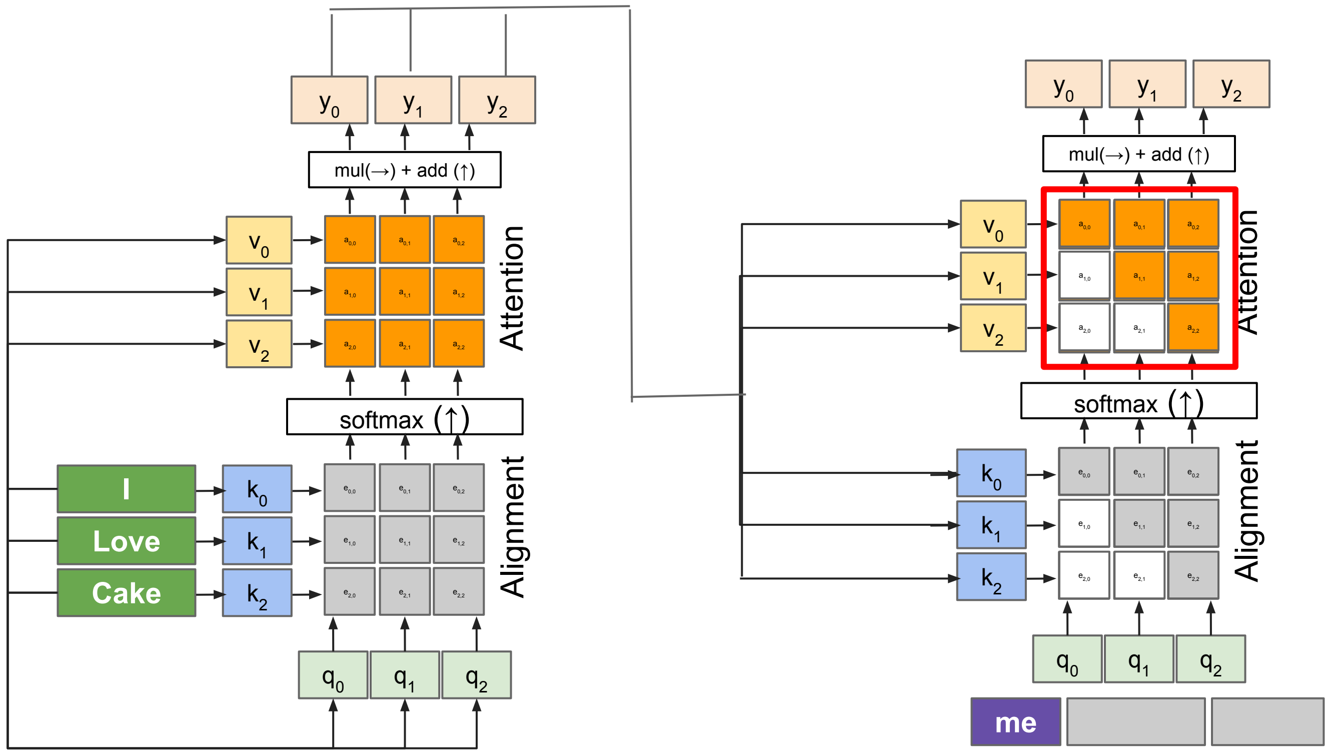




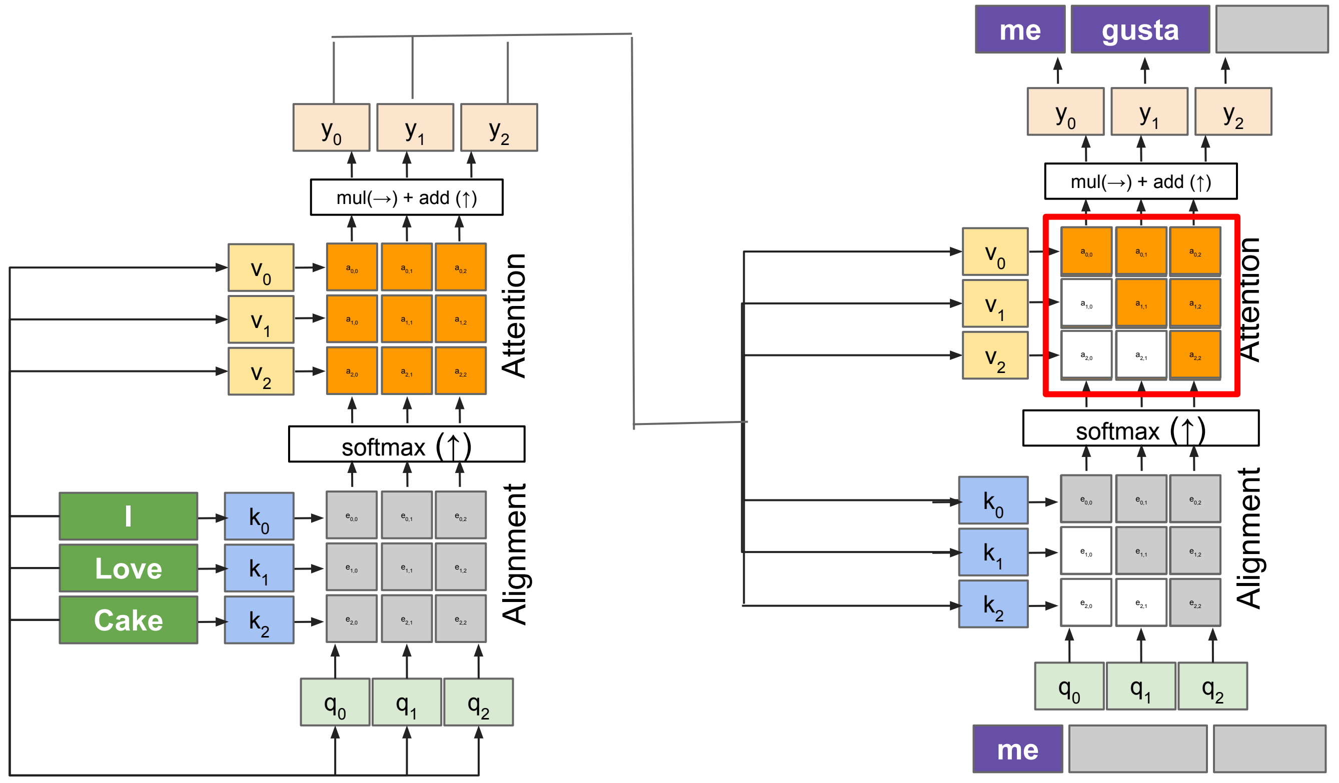
## Encoder-Decoder: Generate text based on previously generated text and the meaning of a separate sequence



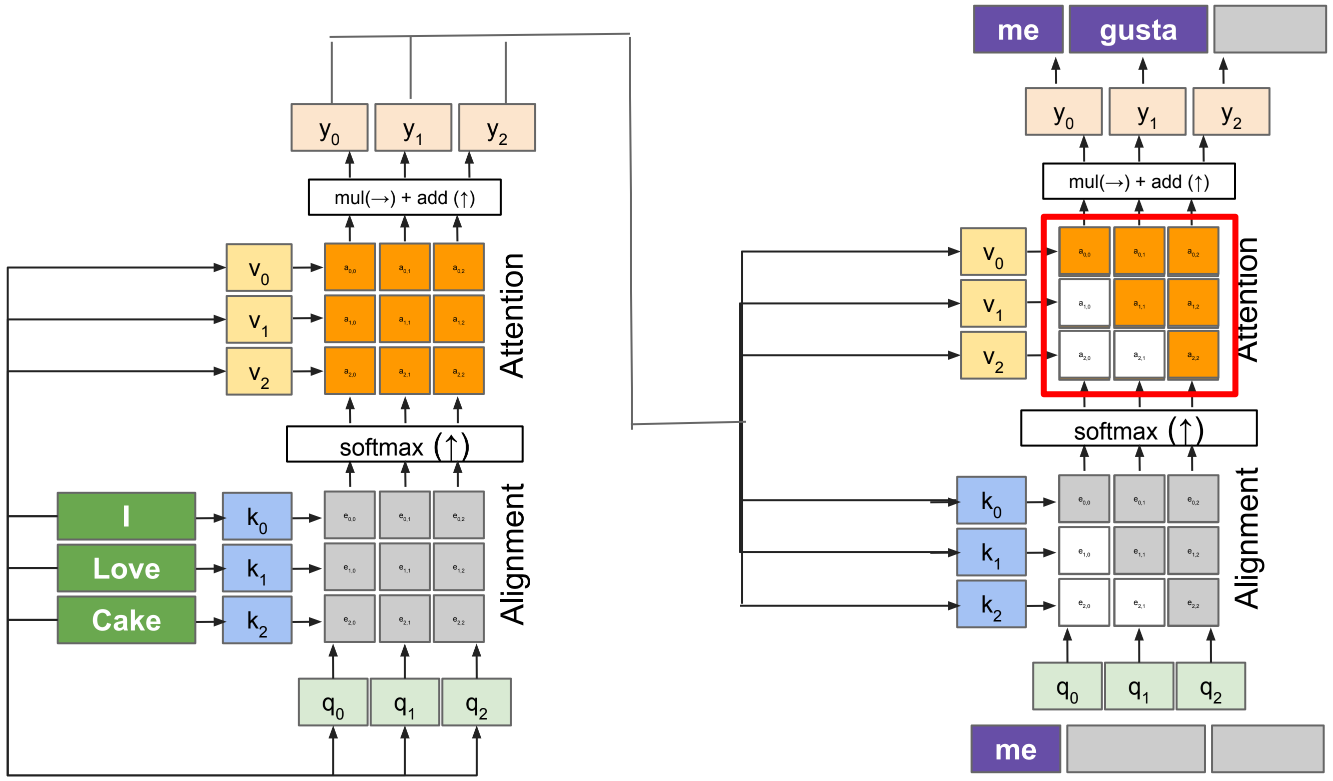
## Encoder-Decoder: Generate text based on previously generated text and the meaning of a separate sequence



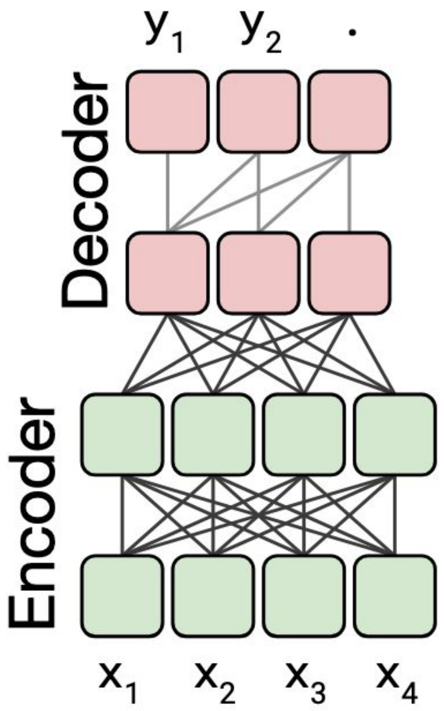
## Encoder-Decoder: Generate text based on previously generated text and the meaning of a separate sequence



# Encoder-Decoder: Generate text based on previously generated text and the meaning of a separate sequence

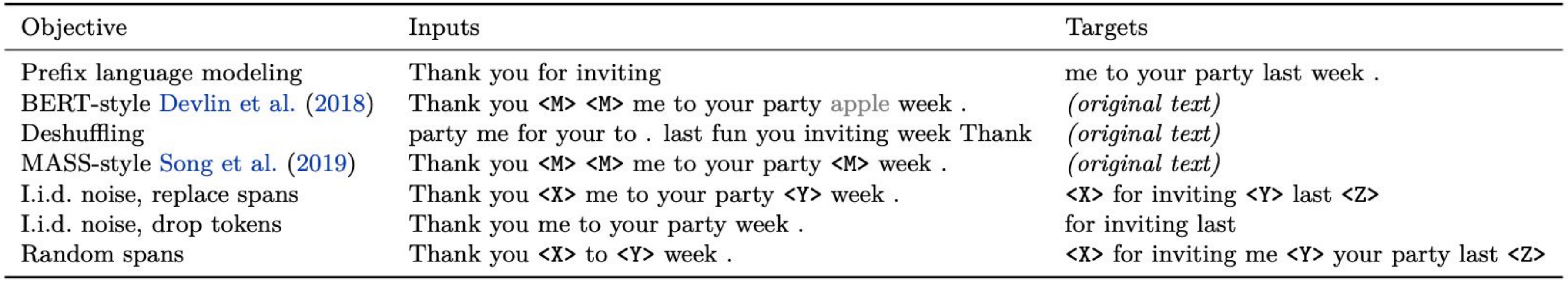


## Encoder-Decoder: Generate text based on previously generated text and the meaning of a separate sequence



Raffel et al., 2019

## Encoder-Decoder: Generate text based on previously generated text and the meaning of a separate sequence



| Objective | Inputs | Targets |
|---|---|---|
| Prefix language modeling | Thank you for inviting | me to your party last week . |
| BERT-style Devlin et al. (2018) | Thank you `<M>` `<M>` me to your party apple week . | *(original text)* |
| Deshuffling | party me for your to . last fun you inviting week Thank | *(original text)* |
| MASS-style Song et al. (2019) | Thank you `<M>` `<M>` me to your party `<M>` week . | *(original text)* |
| I.i.d. noise, replace spans | Thank you `<X>` me to your party `<Y>` week . | `<X>` for inviting `<Y>` last `<Z>` |
| I.i.d. noise, drop tokens | Thank you me to your party week . | for inviting last |
| Random spans | Thank you `<X>` to `<Y>` week . | `<X>` for inviting me `<Y>` your party last `<Z>` |

Image Source: Raffel et al. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. 2019

## Encoder-Decoder: Generate text based on previously generated text and the meaning of a separate sequence

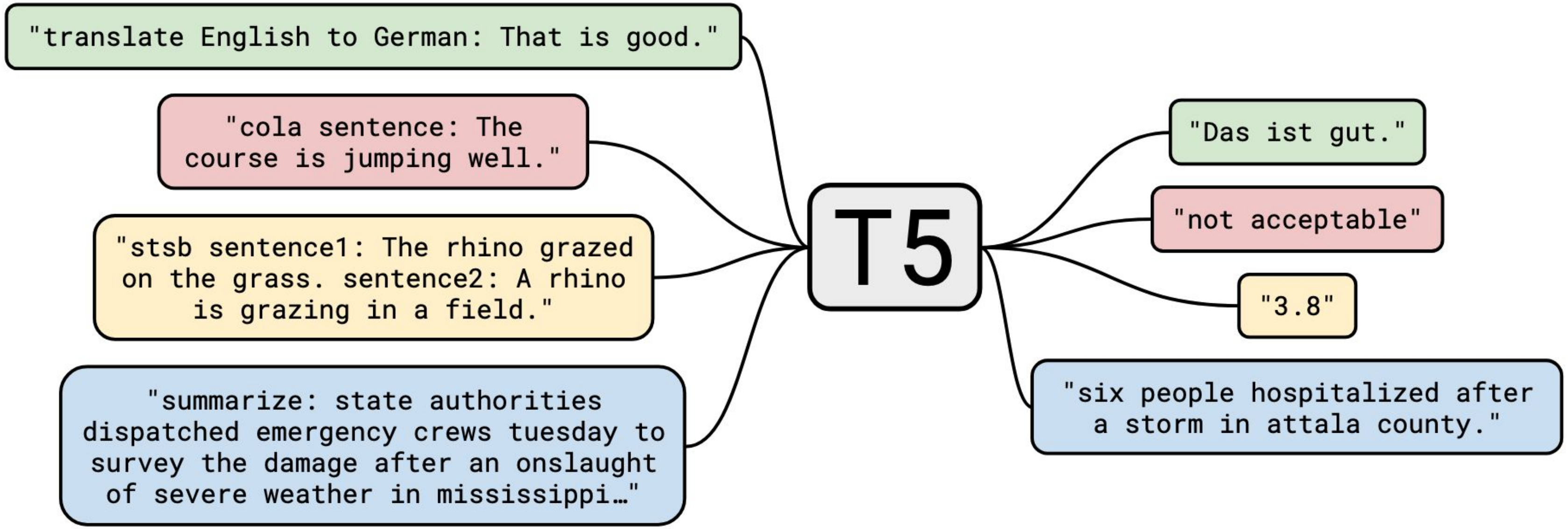
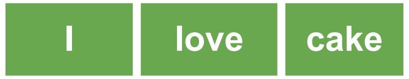

"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

Image Source: Raffel et al. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. 2019

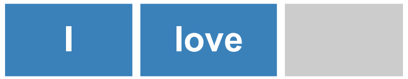
# LLMs

## Encoder Only:



**ELMO:** Bi-directional next word prediction,
**BERT:** Masked language objective, Next Sentence Prediction

## Decoder Only:



**GPT:** text token prediction

## Encoder-Decoder:

**T5:** Masked language objective

I
love
cake
me
gusta

## LLMs

Building LLMs: Pre-training objectives + architectures

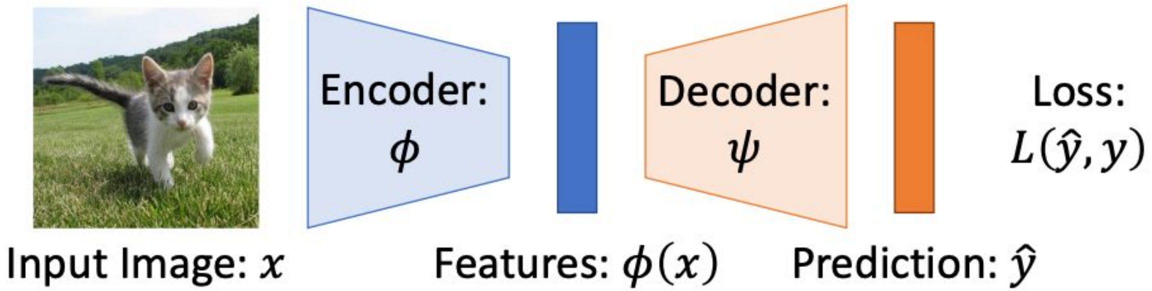
- Encoder only
- Decoder only
- Encoder Decoder

**GPT**

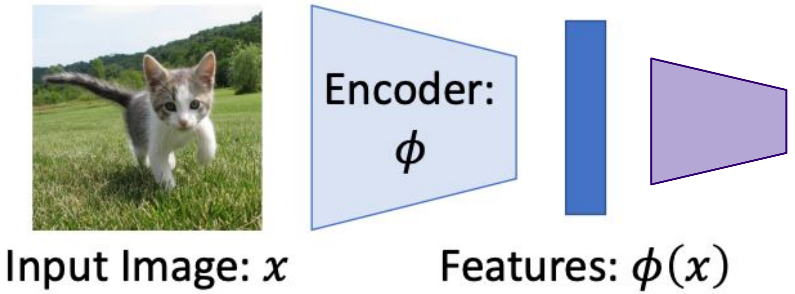
Gradient-Free Performance Improvement

# Using pre-trained models out of the box

**Step 1:** Pretrain a network on a pretext task that doesn't require supervision



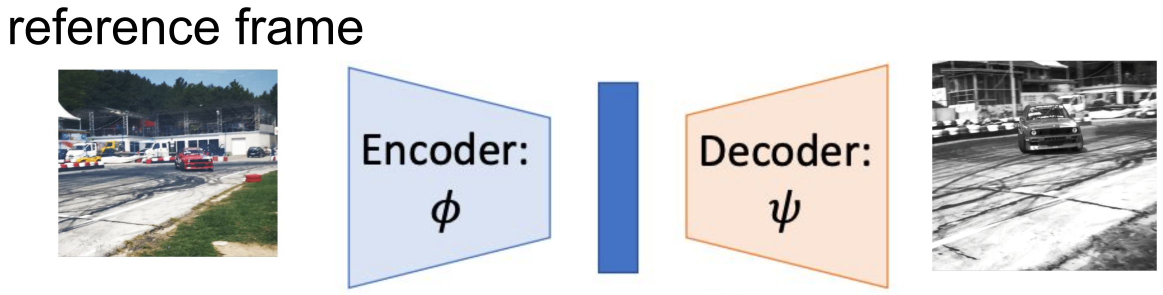
**Step 2:** Transfer encoder to downstream tasks via linear classifiers, KNN, finetuning



**Downstream tasks:**
Image classification, object detection, semantic segmentation

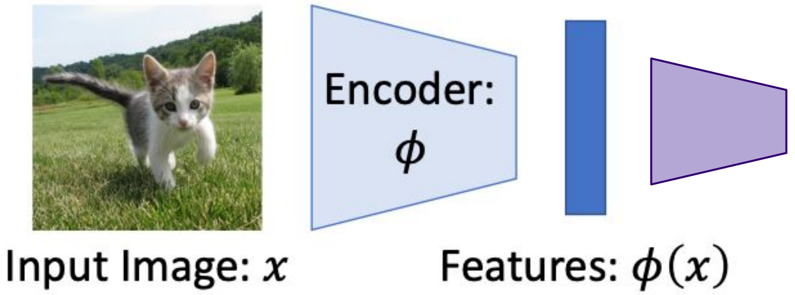
# Using pre-trained models out of the box

**Step 1:** Pretrain a network on a pretext task that doesn't require supervision



Source: Vondrick et al., 2018

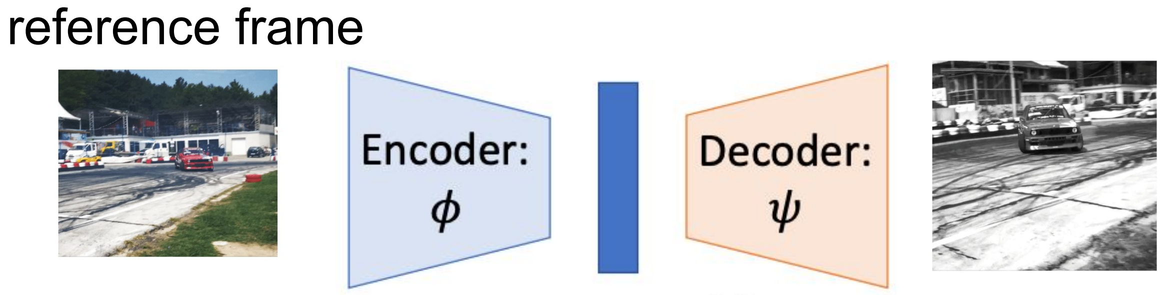
**Step 2:** Transfer encoder to downstream tasks via linear classifiers, KNN, finetuning



**Downstream tasks:** Image classification, object detection, semantic segmentation

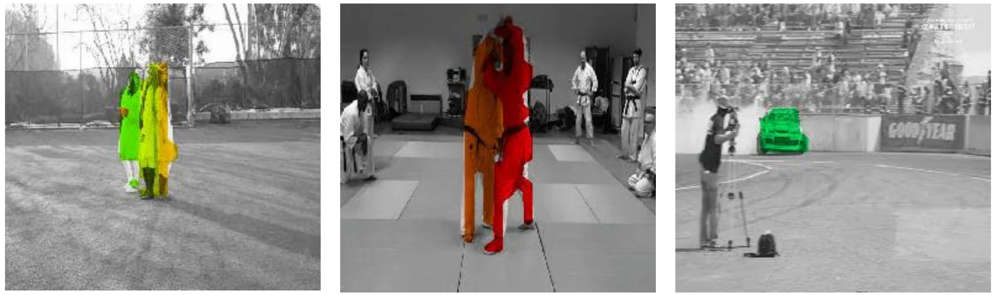
## Using pre-trained models out of the box

**Step 1:** Pretrain a network on a pretext task that doesn't require supervision



Source: Vondrick et al., 2018

**Step 2:** Use the model out of the box in a creative way!



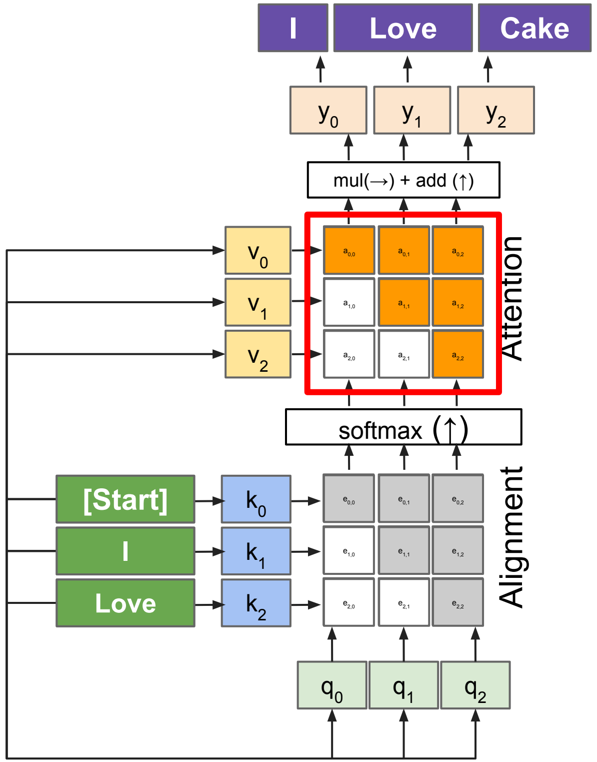
Source: Google AI blog post

## Decoder Only: GPT

**Input:** Text sequence

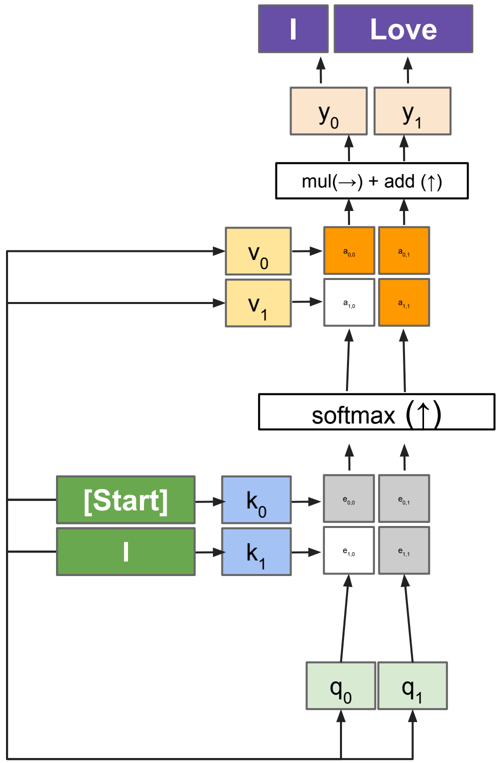
**Output**: Completed text sequence

**To pre-train: predict next words from previous words for large text corpus**

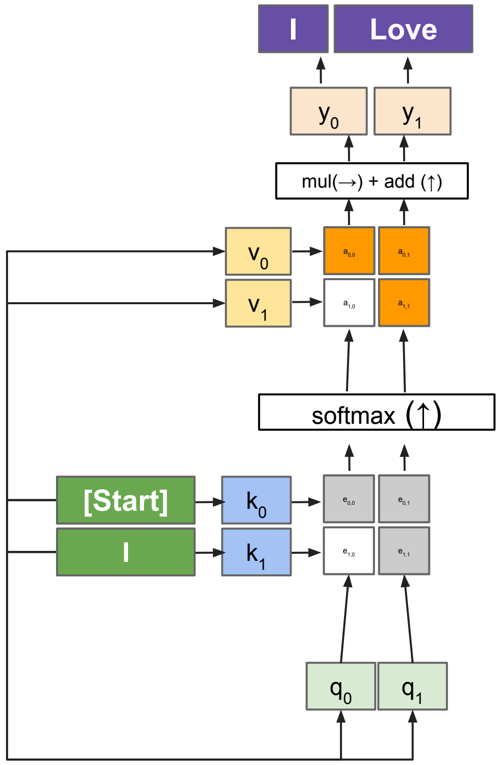


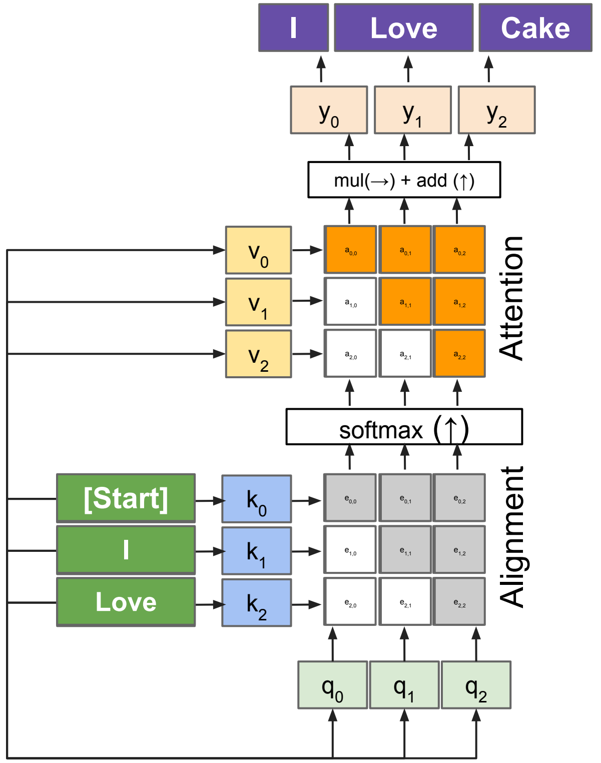
## Decoder Only: Inference

# Decoder Only: Inference



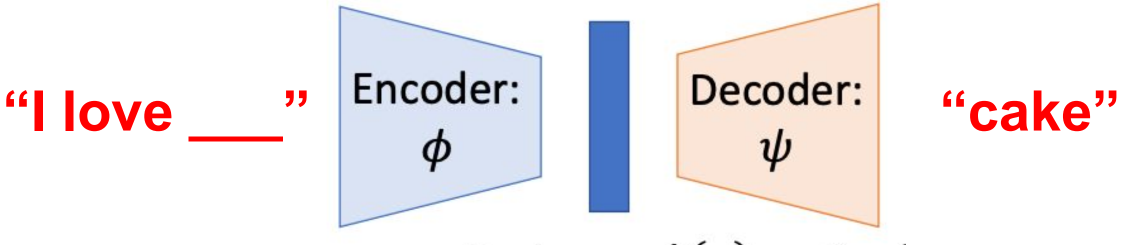
# Decoder Only: Inference



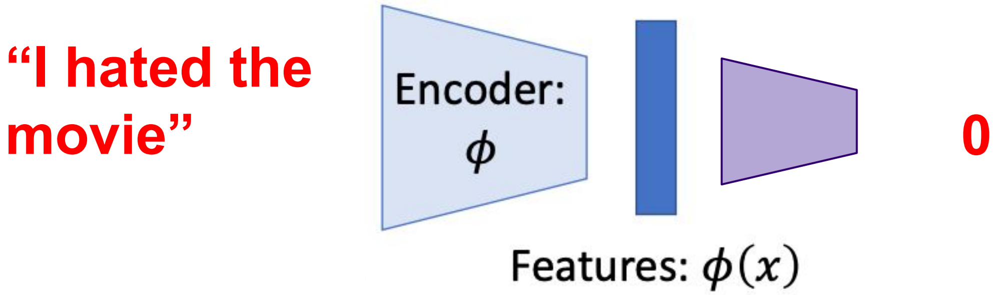


# Using pre-trained models out of the box

**Step 1:** Pretrain a network on a pretext task that doesn't require supervision



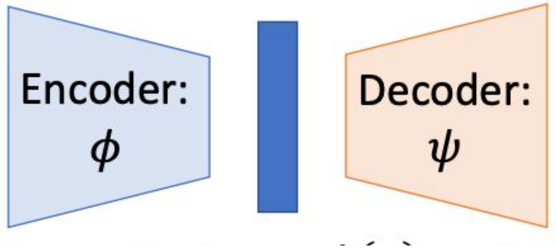
**Step 2:** Transfer encoder to downstream tasks via linear classifiers, KNN, finetuning



## Using pre-trained models out of the box

**Step 1:** Pretrain a network on a pretext task that doesn't require supervision

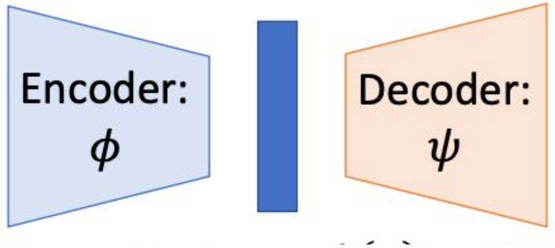
“I love ___”



“cake”

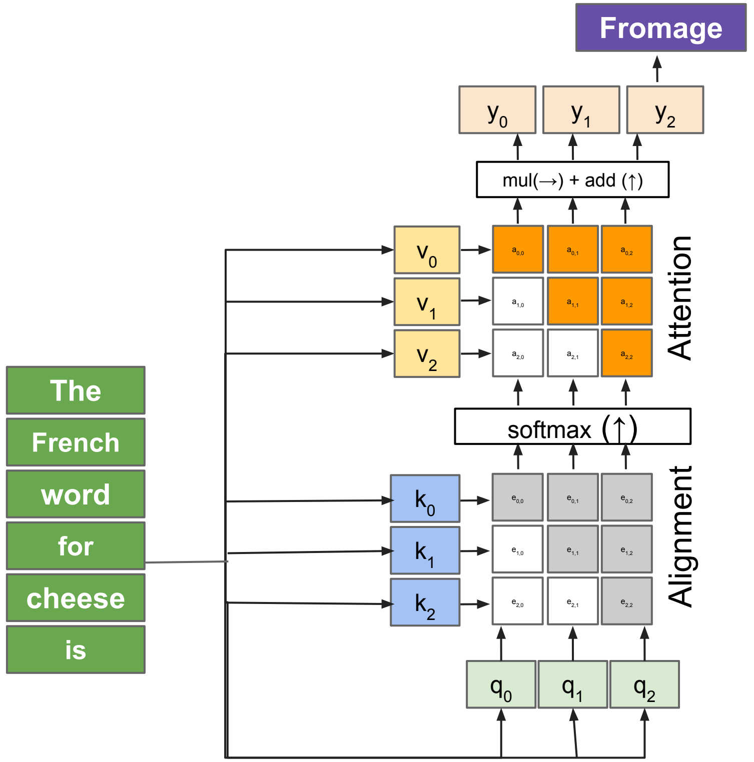
**Step 2:** Use the model out of the box in a creative way!

“The movie review ‘I hated the movie’ is ____”

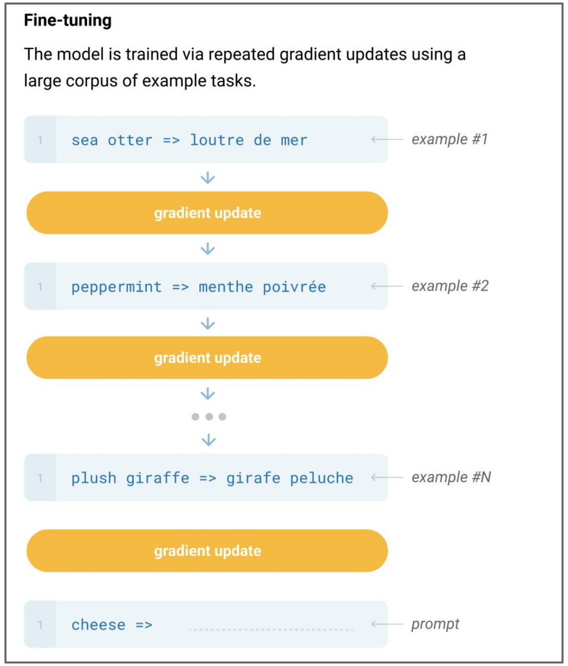


“negative”

## Zero-shot


Fromage
$y_0$
$y_1$
$y_2$
mul(→) + add (↑)
$v_0$
$v_1$
$v_2$
$a_{0,0}$
$a_{0,1}$
$a_{0,2}$
$a_{1,0}$
$a_{1,1}$
$a_{1,2}$
$a_{2,0}$
$a_{2,1}$
$a_{2,2}$
Attention
softmax (↑)
$k_0$
$k_1$
$k_2$
$e_{0,0}$
$e_{0,1}$
$e_{0,2}$
$e_{1,0}$
$e_{1,1}$
$e_{1,2}$
$e_{2,0}$
$e_{2,1}$
$e_{2,2}$
Alignment
$q_0$
$q_1$
$q_2$
The
French
word
for
cheese
is

# Fine-tuning is effective (but annoying)


**Fine-tuning**

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

```
sea otter => loutre de mer
```

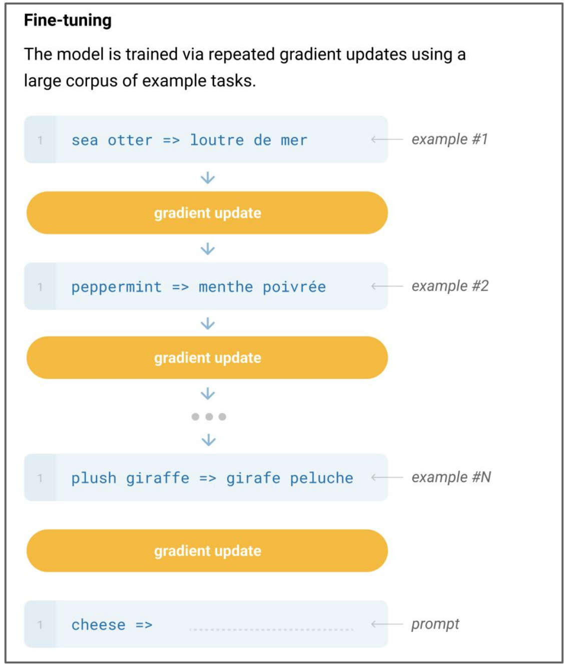
```
peppermint => menthe poivrée
```

```
plush giraffe => girafe peluche
```

```
cheese =>
```

Image Source: Language Models are Few-Shot Learners, Brown et al

# GPT-3 Results


**Fine-tuning**

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

**Language Models are Few-Shot Learners**

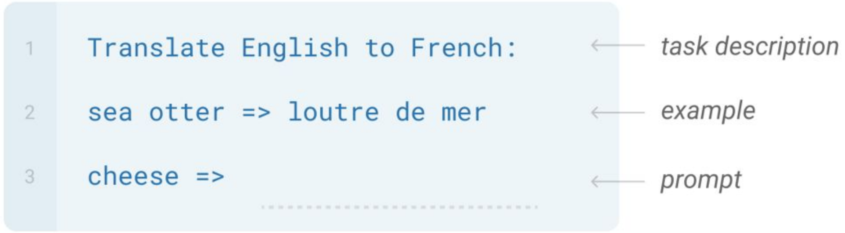


Image Source: [Language Models are Few-Shot Learners, Brown et al]

# In-Context Learning

context

```
Translate English to French:

sea otter => loutre de mer

cheese =>
```

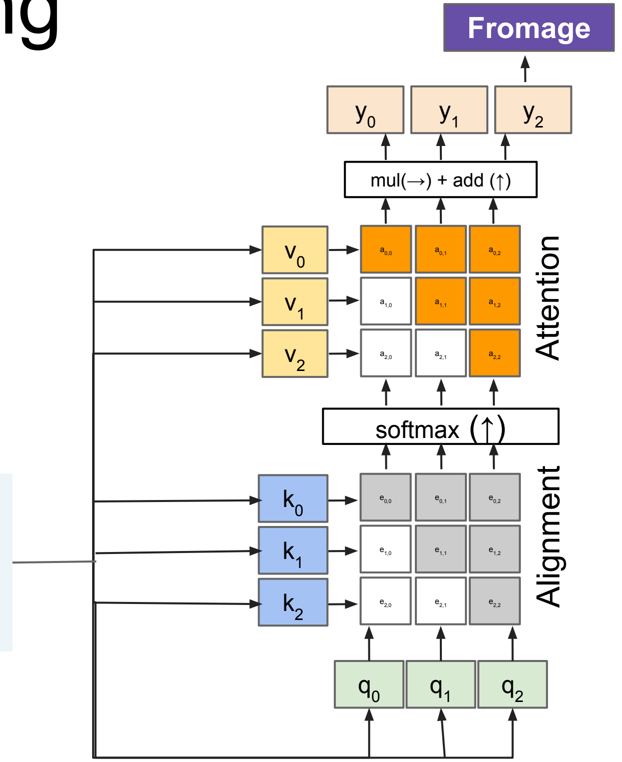
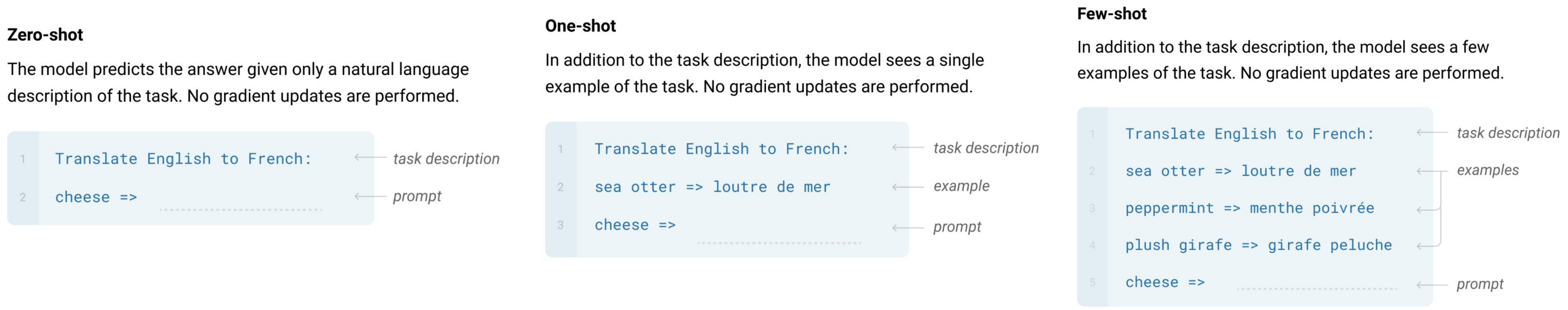


Image Source: Language Models are Few-Shot Learners, Brown et al

# In-Context Learning



**Zero-shot**

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

```
Translate English to French:
cheese =>
```

**One-shot**

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

```
Translate English to French:
sea otter => loutre de mer
cheese =>
```

**Few-shot**

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

```
Translate English to French:
sea otter => loutre de mer
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>
```

No training. No gradients.

Image Source: Language Models are Few-Shot Learners, Brown et al

# Effect of In-Context Learning

s.u!c/c!e.s s i/o/n = succession

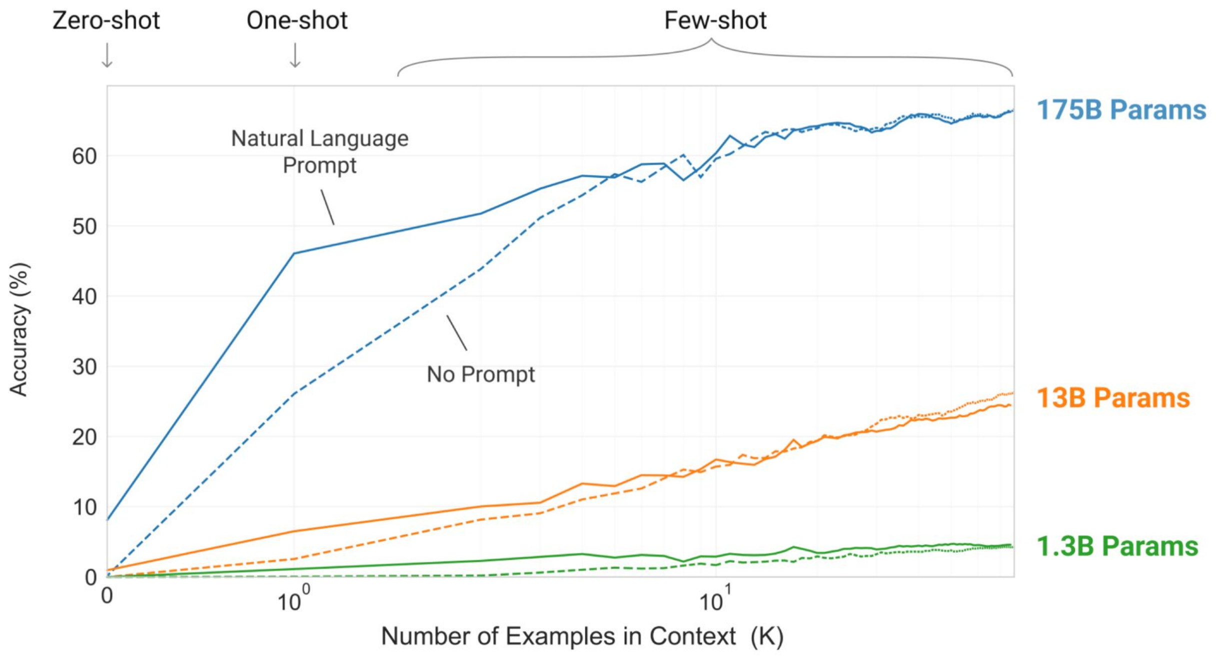
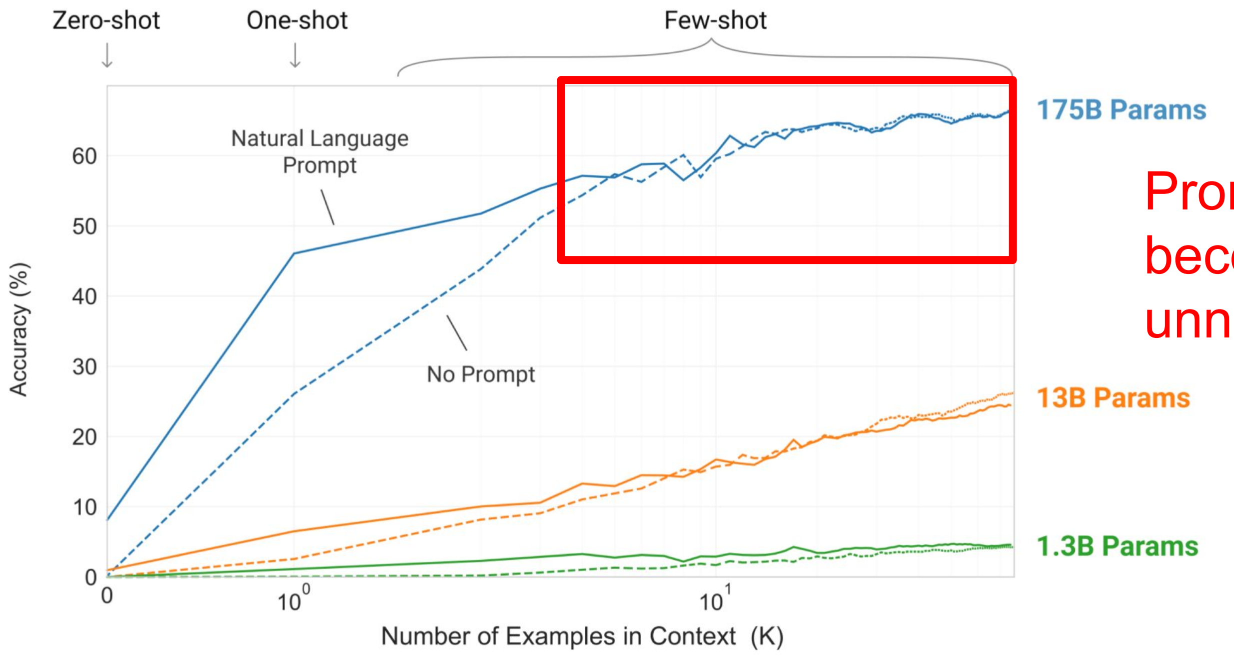


Image Source: Language Models are Few-Shot Learners, Brown et al

## Effect of In-Context Learning

s.u!c/c!e.s s i/o/n = succession



Prompt becomes unnecessary

Image Source: Language Models are Few-Shot Learners, Brown et al

# Effect of In-Context Learning

Example Question:

audacious is to boldness as
- (a) sanctimonious is to hypocrisy,
- (b) anonymous is to identity,
- (c) remorseful is to misdeed,
- (d) deleterious is to result,
- (e) impressionable is to temptation

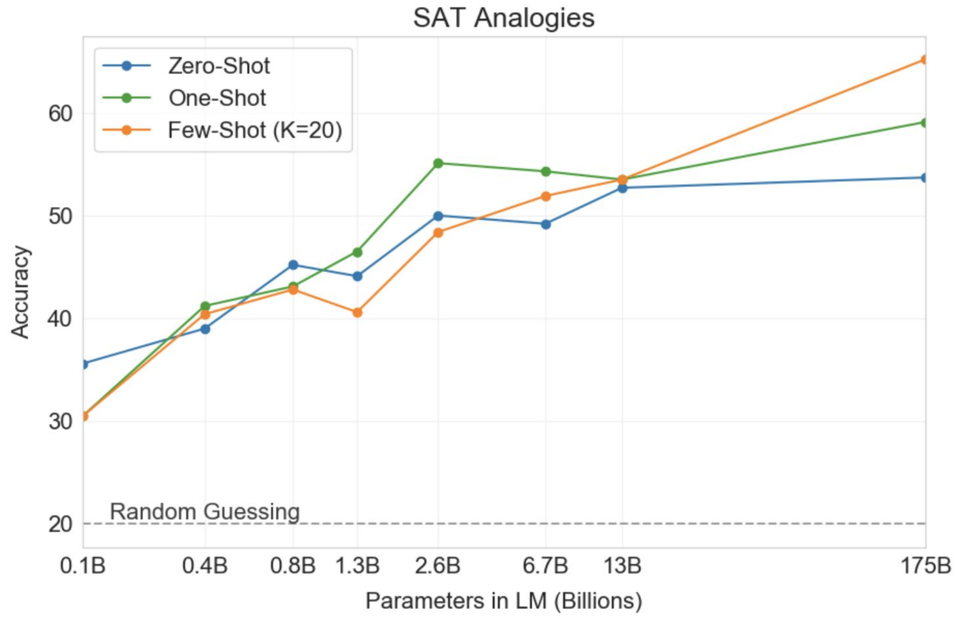


Image Source: [Language Models are Few-Shot Learners, Brown et al](#)

# GPT Results

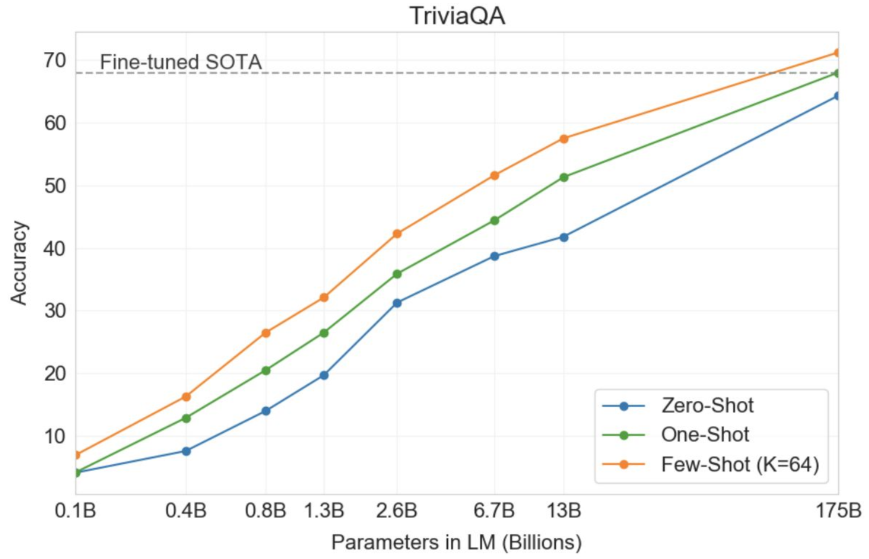


**Figure 3.3:** On TriviaQA GPT3's performance grows smoothly with model size, suggesting that language models continue to absorb knowledge as their capacity increases. One-shot and few-shot performance make significant gains over zero-shot behavior, matching and exceeding the performance of the SOTA fine-tuned open-domain model, RAG [LPP+20]

Image Source: Language Models are Few-Shot Learners, Brown et al

# GPT Results

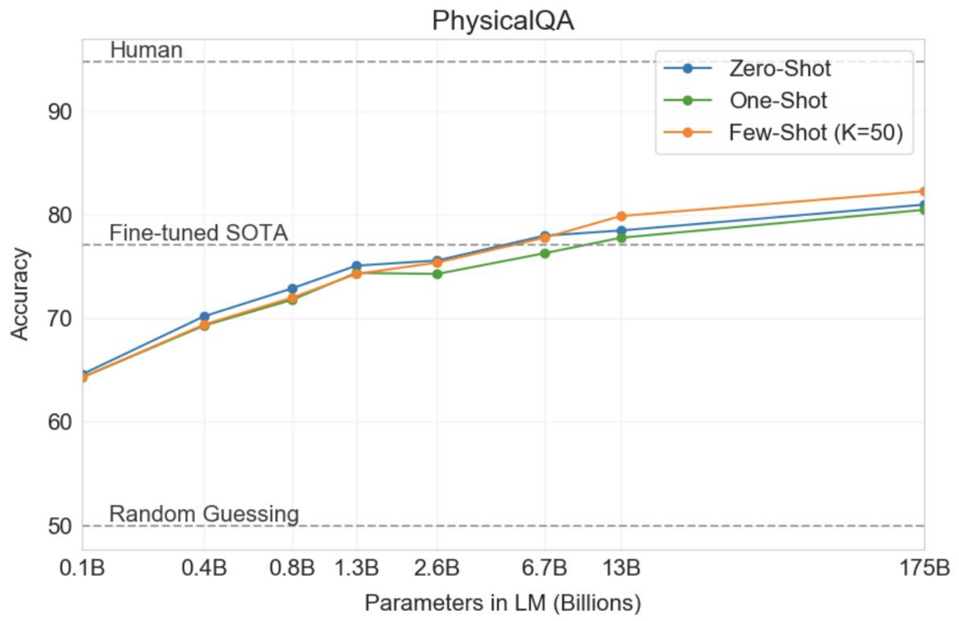
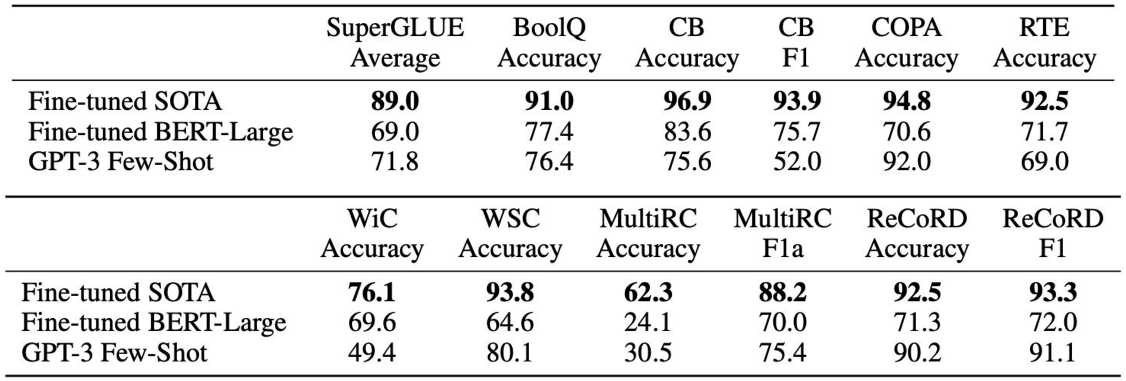


**Figure 3.6:** GPT-3 results on PIQA in the zero-shot, one-shot, and few-shot settings. The largest model achieves a score on the development set in all three conditions that exceeds the best recorded score on the task.

Image Source: [Language Models are Few-Shot Learners, Brown et al]()

# GPT Results



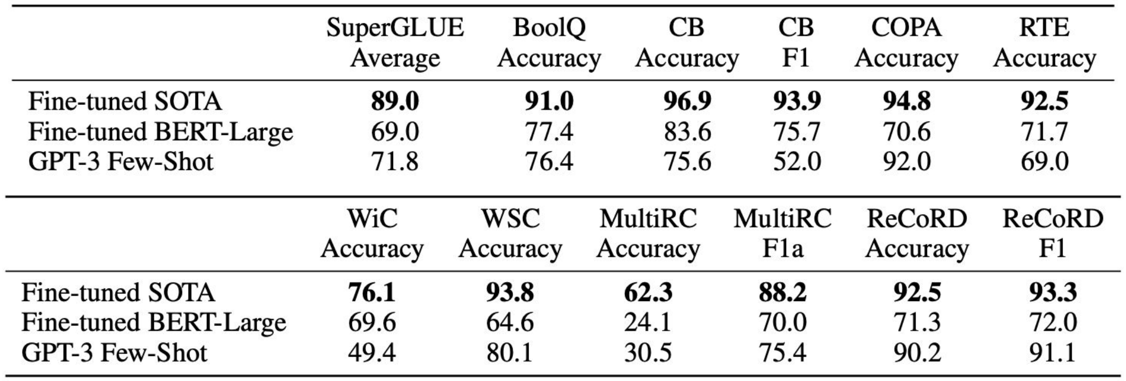
| | SuperGLUE Average | BoolQ Accuracy | CB Accuracy | CB F1 | COPA Accuracy | RTE Accuracy |
|---|---|---|---|---|---|---|
| Fine-tuned SOTA | **89.0** | **91.0** | **96.9** | **93.9** | **94.8** | **92.5** |
| Fine-tuned BERT-Large | 69.0 | 77.4 | 83.6 | 75.7 | 70.6 | 71.7 |
| GPT-3 Few-Shot | 71.8 | 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 | **76.1** | **93.8** | **62.3** | **88.2** | **92.5** | **93.3** |
| 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 |

**Table 3.8:** Performance of GPT-3 on SuperGLUE compared to fine-tuned baselines and SOTA. All results are reported on the test set. GPT-3 few-shot is given a total of 32 examples within the context of each task and performs no gradient updates.

Image Source: [Language Models are Few-Shot Learners, Brown et al]

# GPT Results



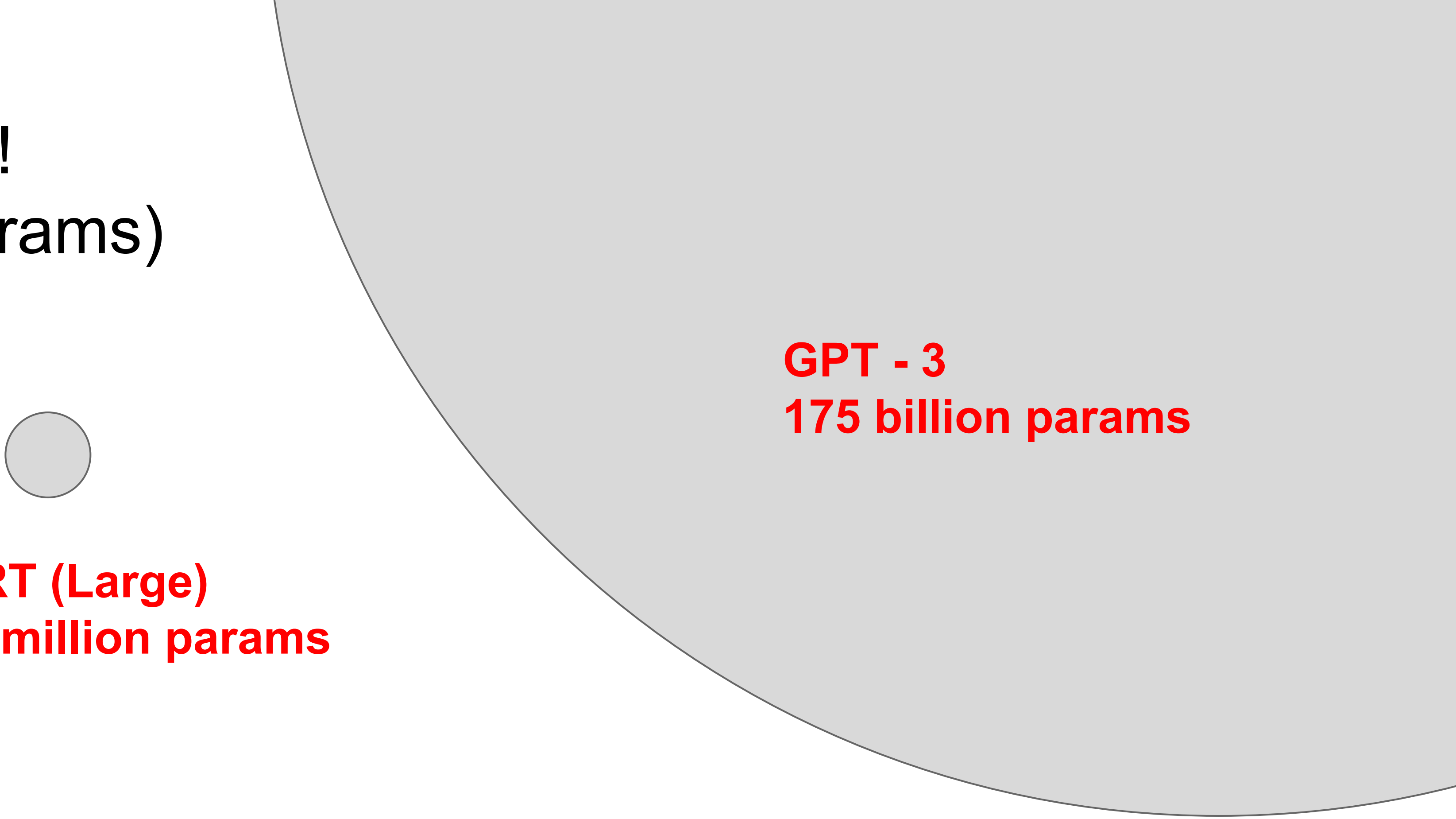
| | SuperGLUE Average | BoolQ Accuracy | CB Accuracy | CB F1 | COPA Accuracy | RTE Accuracy |
|---|---|---|---|---|---|---|
| Fine-tuned SOTA | **89.0** | **91.0** | **96.9** | **93.9** | **94.8** | **92.5** |
| Fine-tuned BERT-Large | 69.0 | 77.4 | 83.6 | 75.7 | 70.6 | 71.7 |
| GPT-3 Few-Shot | 71.8 | 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 | **76.1** | **93.8** | **62.3** | **88.2** | **92.5** | **93.3** |
| 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 |

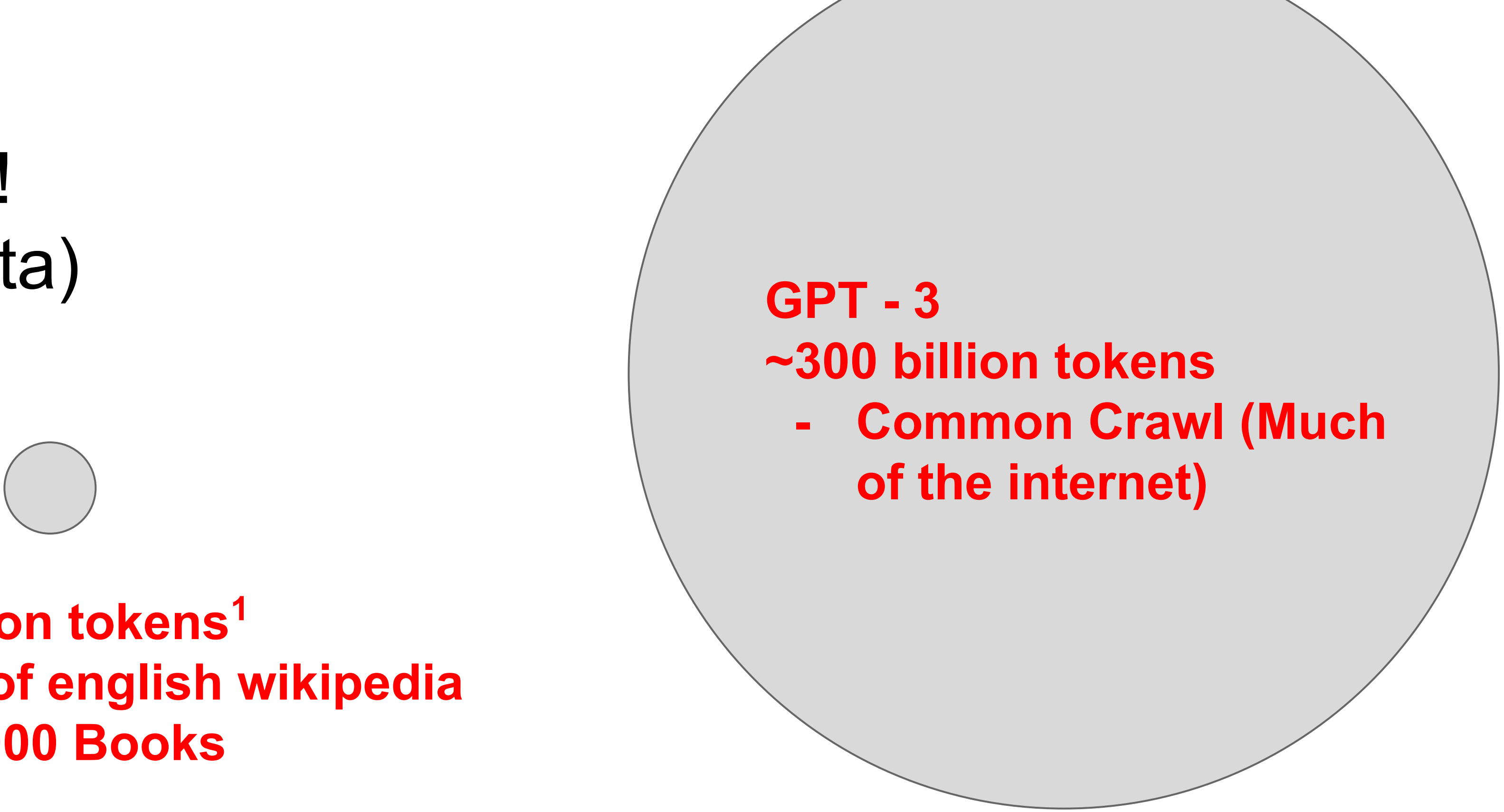
**Table 3.8:** Performance of GPT-3 on SuperGLUE compared to fine-tuned baselines and SOTA. All results are reported on the test set. GPT-3 few-shot is given a total of 32 examples within the context of each task and performs no gradient updates.

**All GPT was trained to do was predict the next token, how is it so good???**

## Scale!
## (in params)



## Scale!
## (In data)



**GPT - 3**
**~300 billion tokens**
- **Common Crawl (Much of the internet)**

**BERT**
**3.3 Billion tokens[1]**
- **All of english wikipedia**
- **11,000 Books**

[1]https://aclanthology.org/W19-4828.pdf

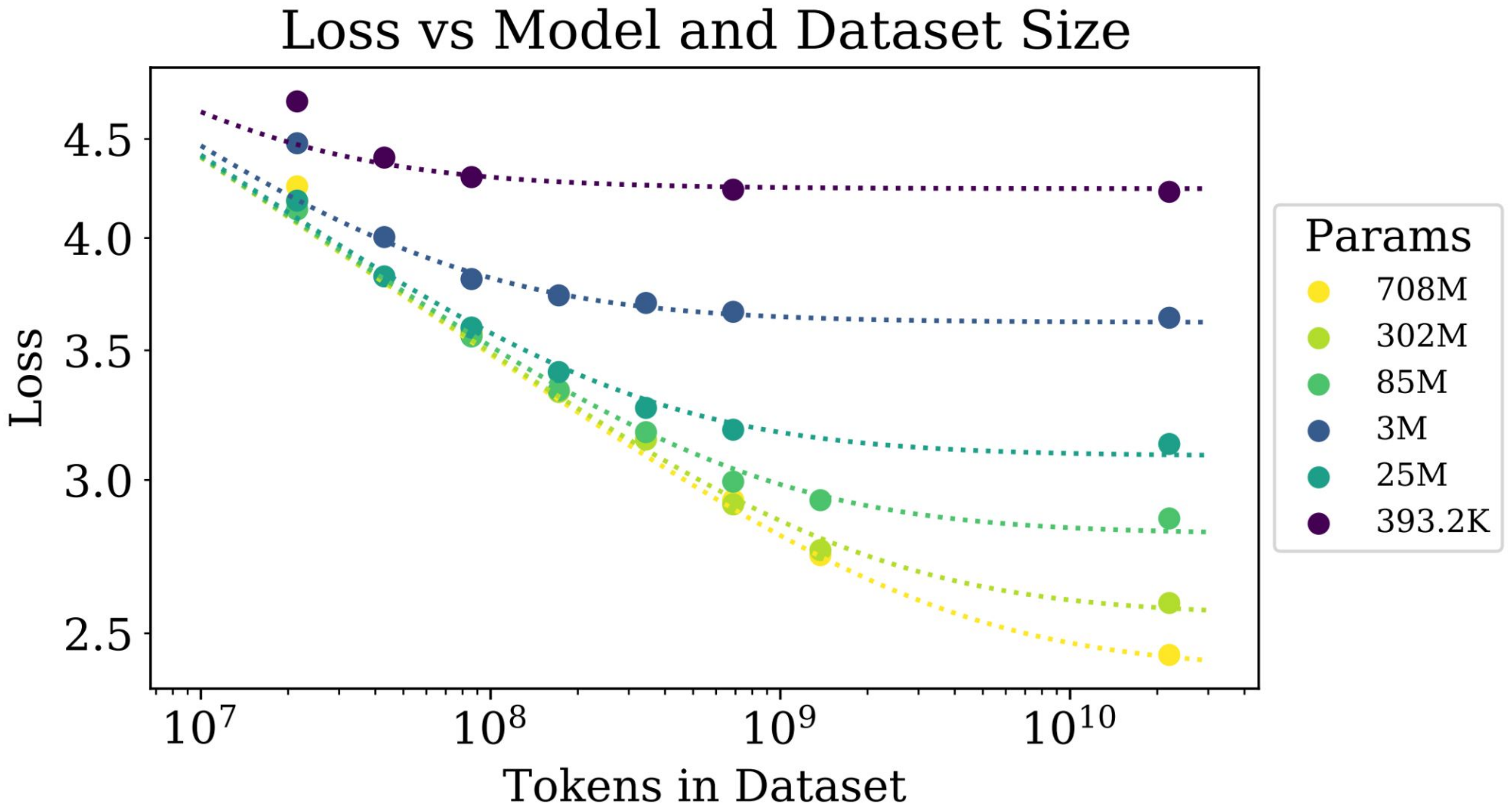


Image source: Kaplan et al. Scaling Laws for Neural Language Models. 2020.

# The Bitter Lesson

## Rich Sutton

**March 13, 2019**

The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin. The ultimate reason for this is Moore's law, or rather its generalization of continued exponentially falling cost per unit of computation. Most AI research has been conducted as if the computation available to the agent were constant (in which case leveraging human knowledge would be one of the only ways to improve performance) but, over a slightly longer time than a typical research project, massively more computation inevitably becomes available. Seeking an improvement that makes a difference in the shorter term, researchers seek to leverage their human knowledge of the domain, but the only thing that matters in the long run is the leveraging of computation. These two need not run counter to each other, but in practice they tend to. Time spent on one is time not spent on the other. There are psychological commitments to investment in one approach or the other. And the human-knowledge approach tends to complicate methods in ways that make them less suited to taking advantage of general methods leveraging computation. There were many examples of AI researchers' belated learning of this bitter lesson, and it is instructive to review some of the most prominent.

http://www.incompleteideas.net/IncIdeas/BitterLesson.html

# LLMs

Building LLMs: Pre-training objectives + architectures

- Encoder only
- Decoder only
- Encoder Decoder

GPT

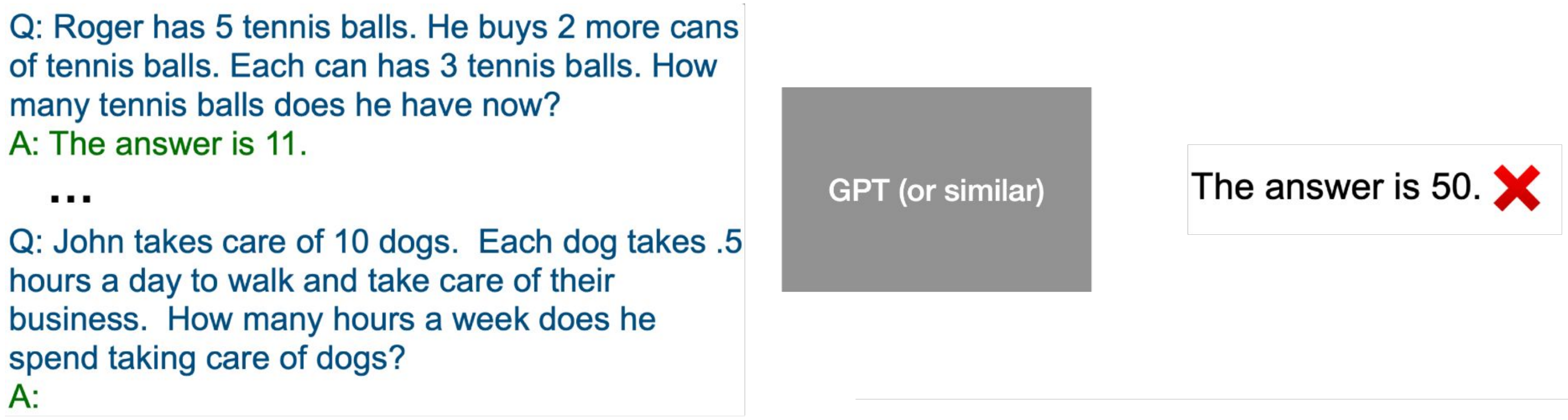
**Gradient-Free Performance Improvement**

## Improving accuracy without fine-tuning

In-context learning is better than zero-shot with no additional training

Can we do better than in context learning with no additional training?

# Chain of thought prompting

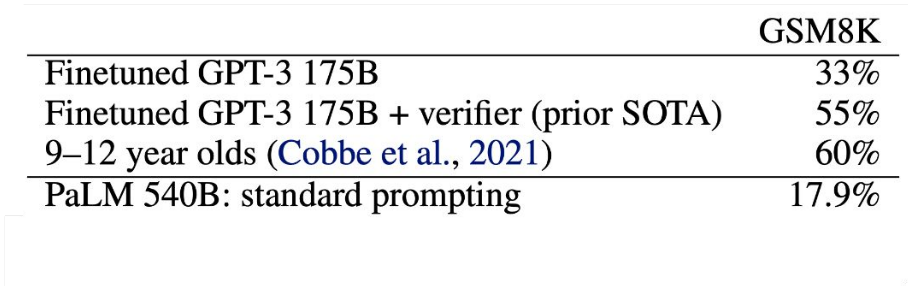


Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: The answer is 11.

...

Q: John takes care of 10 dogs. Each dog takes .5 hours a day to walk and take care of their business. How many hours a week does he spend taking care of dogs?
A:

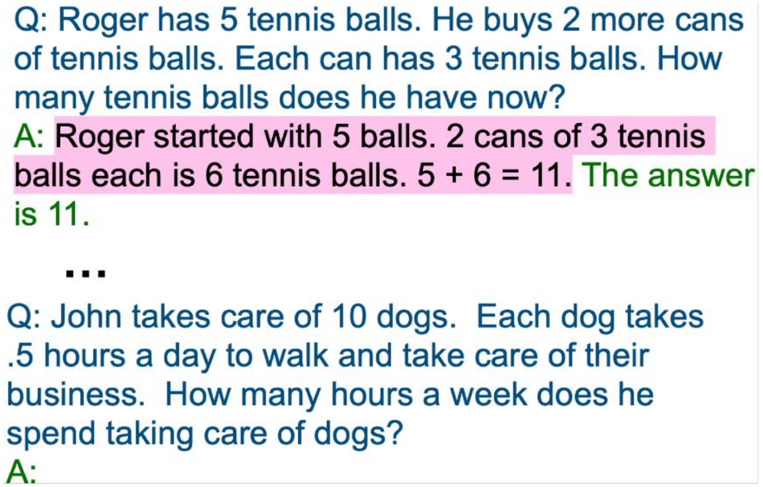
The answer is 50. ❌



| | GSM8K |
|---|---|
| Finetuned GPT-3 175B | 33% |
| Finetuned GPT-3 175B + verifier (prior SOTA) | 55% |
| 9–12 year olds (Cobbe et al., 2021) | 60% |
| PaLM 540B: standard prompting | 17.9% |

Image Source: Wei et al. Chain of Thought Prompting Elicits Reasoning in Large Language Models. 2022.

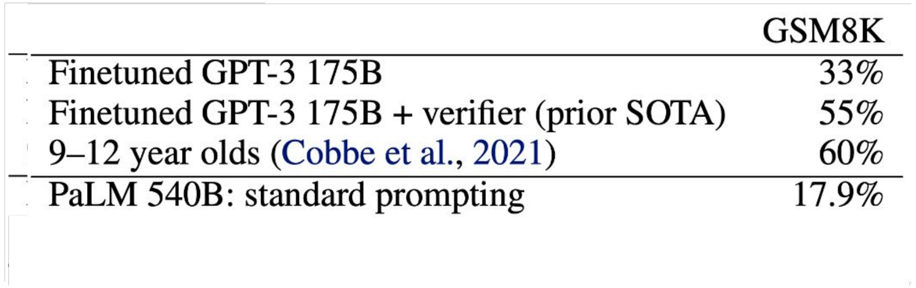
# Chain of thought prompting



Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

...

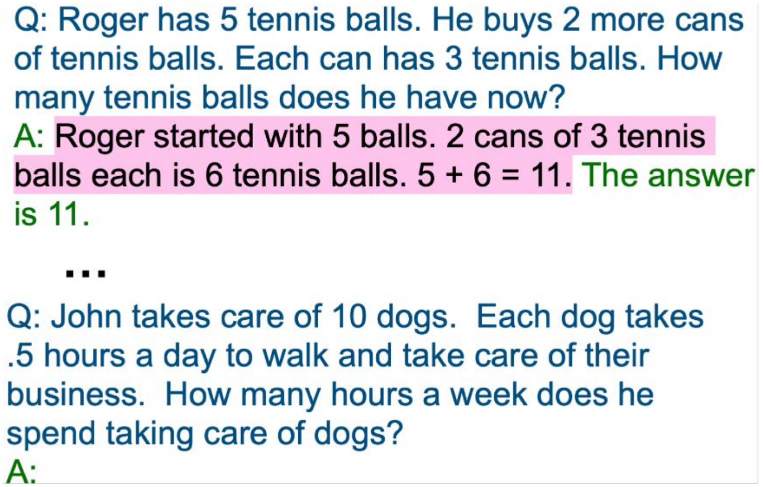
Q: John takes care of 10 dogs. Each dog takes .5 hours a day to walk and take care of their business. How many hours a week does he spend taking care of dogs?
A:



| | GSM8K |
|---|---|
| Finetuned GPT-3 175B | 33% |
| Finetuned GPT-3 175B + verifier (prior SOTA) | 55% |
| 9–12 year olds (Cobbe et al., 2021) | 60% |
| PaLM 540B: standard prompting | 17.9% |

Image Source: Wei et al. Chain of Thought Prompting Elicits Reasoning in Large Language Models. 2022.

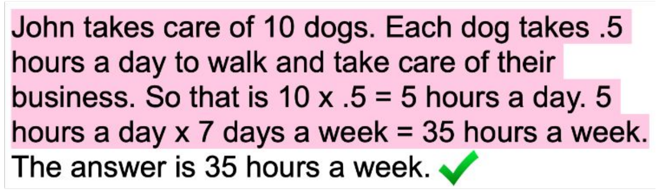
# Chain of thought prompting



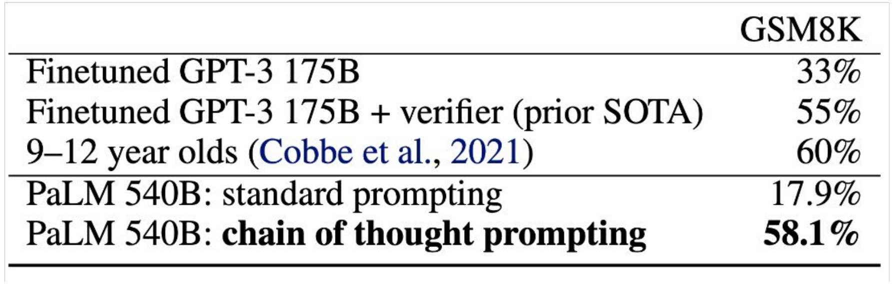
Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

...

Q: John takes care of 10 dogs. Each dog takes .5 hours a day to walk and take care of their business. How many hours a week does he spend taking care of dogs?
A:



John takes care of 10 dogs. Each dog takes .5 hours a day to walk and take care of their business. So that is 10 x .5 = 5 hours a day. 5 hours a day x 7 days a week = 35 hours a week. The answer is 35 hours a week. ✓



| | GSM8K |
|---|---|
| Finetuned GPT-3 175B | 33% |
| Finetuned GPT-3 175B + verifier (prior SOTA) | 55% |
| 9–12 year olds (Cobbe et al., 2021) | 60% |
| PaLM 540B: standard prompting | 17.9% |
| PaLM 540B: **chain of thought prompting** | **58.1%** |

Image Source: Wei et al. Chain of Thought Prompting Elicits Reasoning in Large Language Models. 2022.

# Chain of thought for Math Problems

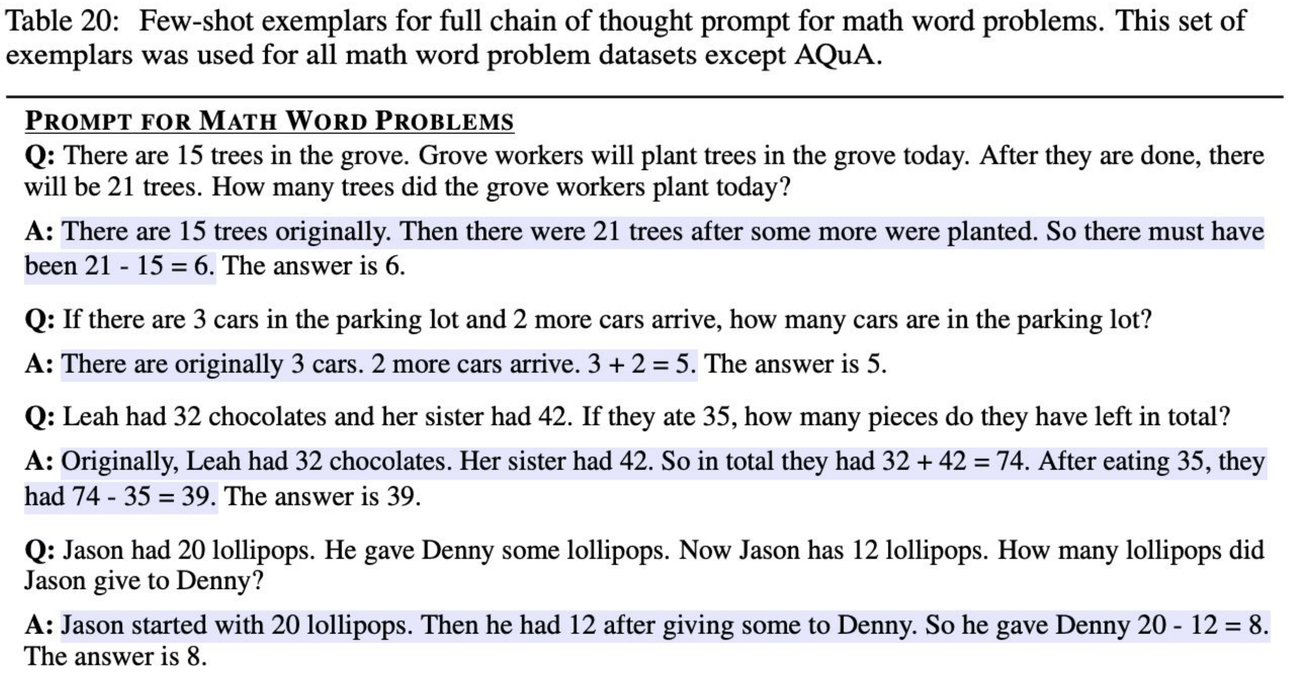


Table 20: Few-shot exemplars for full chain of thought prompt for math word problems. This set of exemplars was used for all math word problem datasets except AQuA.

**PROMPT FOR MATH WORD PROBLEMS**

**Q:** There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?

**A:** There are 15 trees originally. Then there were 21 trees after some more were planted. So there must have been 21 - 15 = 6. The answer is 6.

**Q:** If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?

**A:** There are originally 3 cars. 2 more cars arrive. 3 + 2 = 5. The answer is 5.

**Q:** Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total?

**A:** Originally, Leah had 32 chocolates. Her sister had 42. So in total they had 32 + 42 = 74. After eating 35, they had 74 - 35 = 39. The answer is 39.

**Q:** Jason had 20 lollipops. He gave Denny some lollipops. Now Jason has 12 lollipops. How many lollipops did Jason give to Denny?

**A:** Jason started with 20 lollipops. Then he had 12 after giving some to Denny. So he gave Denny 20 - 12 = 8. The answer is 8.

Image Source: [Wei et al. Chain of Thought Prompting Elicits Reasoning in Large Language Models. 2022.]

# Chain of thought for Math Problems

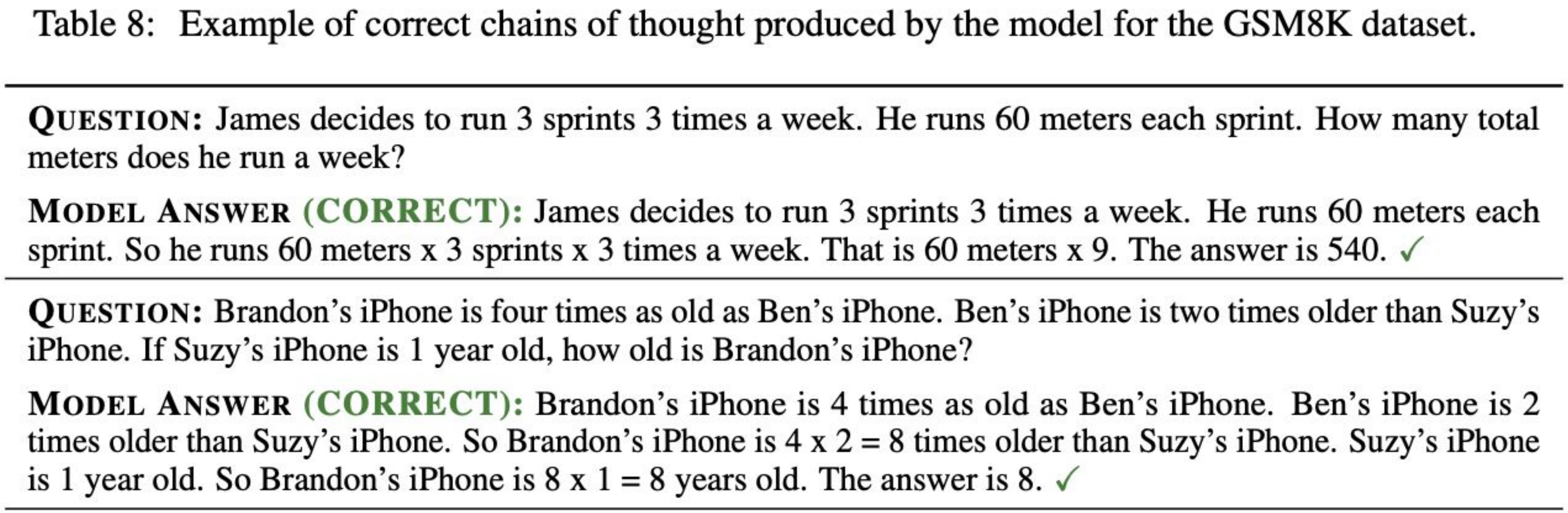


Table 8: Example of correct chains of thought produced by the model for the GSM8K dataset.

**QUESTION:** James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. How many total meters does he run a week?

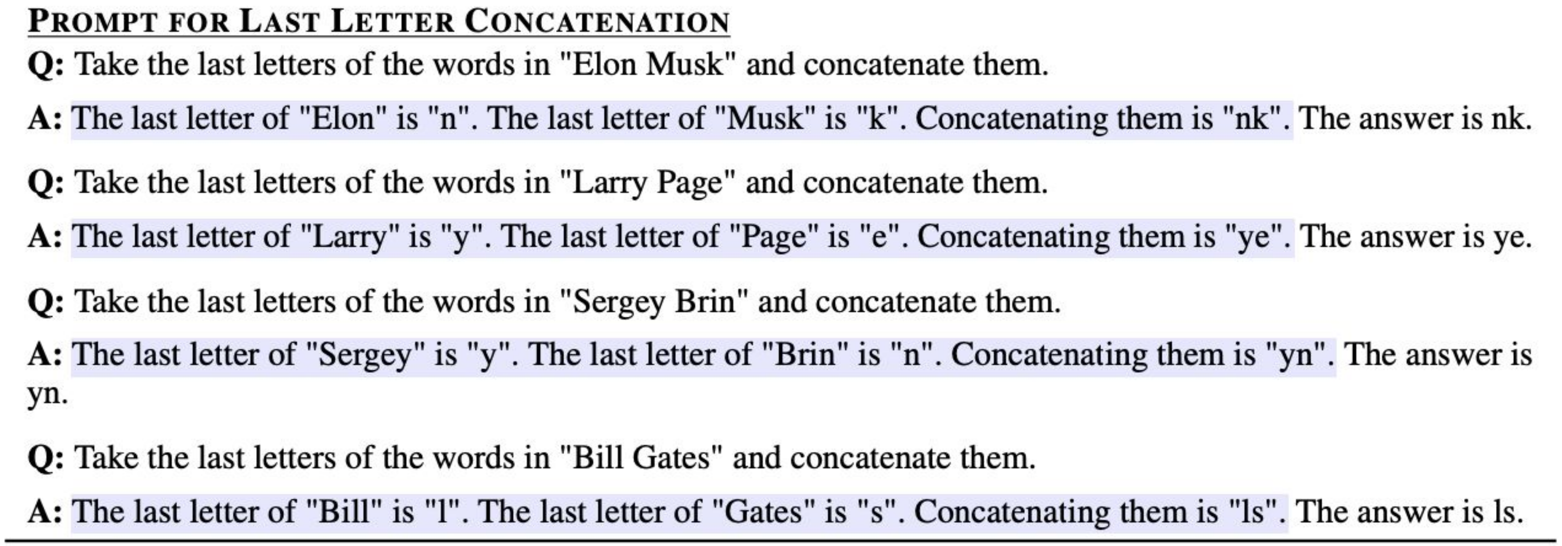
**MODEL ANSWER (CORRECT):** James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. So he runs 60 meters x 3 sprints x 3 times a week. That is 60 meters x 9. The answer is 540. ✓

**QUESTION:** Brandon's iPhone is four times as old as Ben's iPhone. Ben's iPhone is two times older than Suzy's iPhone. If Suzy's iPhone is 1 year old, how old is Brandon's iPhone?

**MODEL ANSWER (CORRECT):** Brandon's iPhone is 4 times as old as Ben's iPhone. Ben's iPhone is 2 times older than Suzy's iPhone. So Brandon's iPhone is 4 x 2 = 8 times older than Suzy's iPhone. Suzy's iPhone is 1 year old. So Brandon's iPhone is 8 x 1 = 8 years old. The answer is 8. ✓

Image Source: Wei et al. Chain of Thought Prompting Elicits Reasoning in Large Language Models. 2022.

# Chain of thought for Symbolic Reasoning



PROMPT FOR LAST LETTER CONCATENATION

**Q:** Take the last letters of the words in "Elon Musk" and concatenate them.

**A:** The last letter of "Elon" is "n". The last letter of "Musk" is "k". Concatenating them is "nk". The answer is nk.

**Q:** Take the last letters of the words in "Larry Page" and concatenate them.

**A:** The last letter of "Larry" is "y". The last letter of "Page" is "e". Concatenating them is "ye". The answer is ye.

**Q:** Take the last letters of the words in "Sergey Brin" and concatenate them.

**A:** The last letter of "Sergey" is "y". The last letter of "Brin" is "n". Concatenating them is "yn". The answer is yn.

**Q:** Take the last letters of the words in "Bill Gates" and concatenate them.

**A:** The last letter of "Bill" is "l". The last letter of "Gates" is "s". Concatenating them is "ls". The answer is ls.

Image Source: Wei et al. Chain of Thought Prompting Elicits Reasoning in Large Language Models. 2022.

# Chain of thought for Symbolic Reasoning

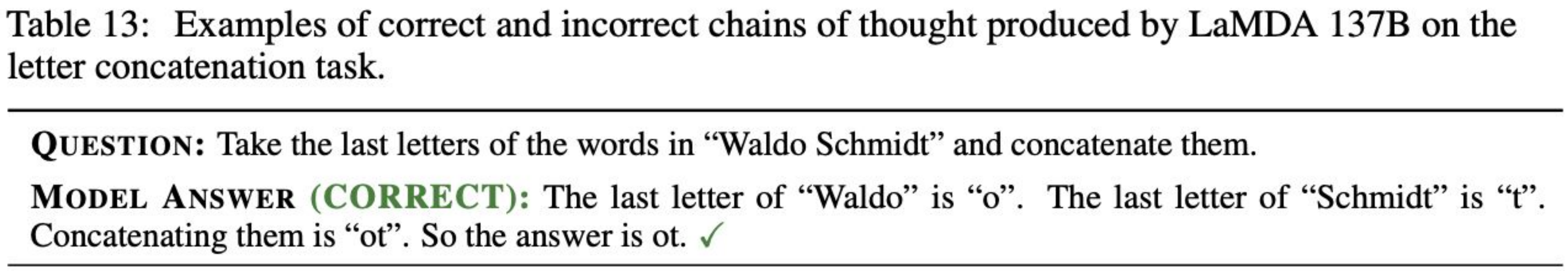


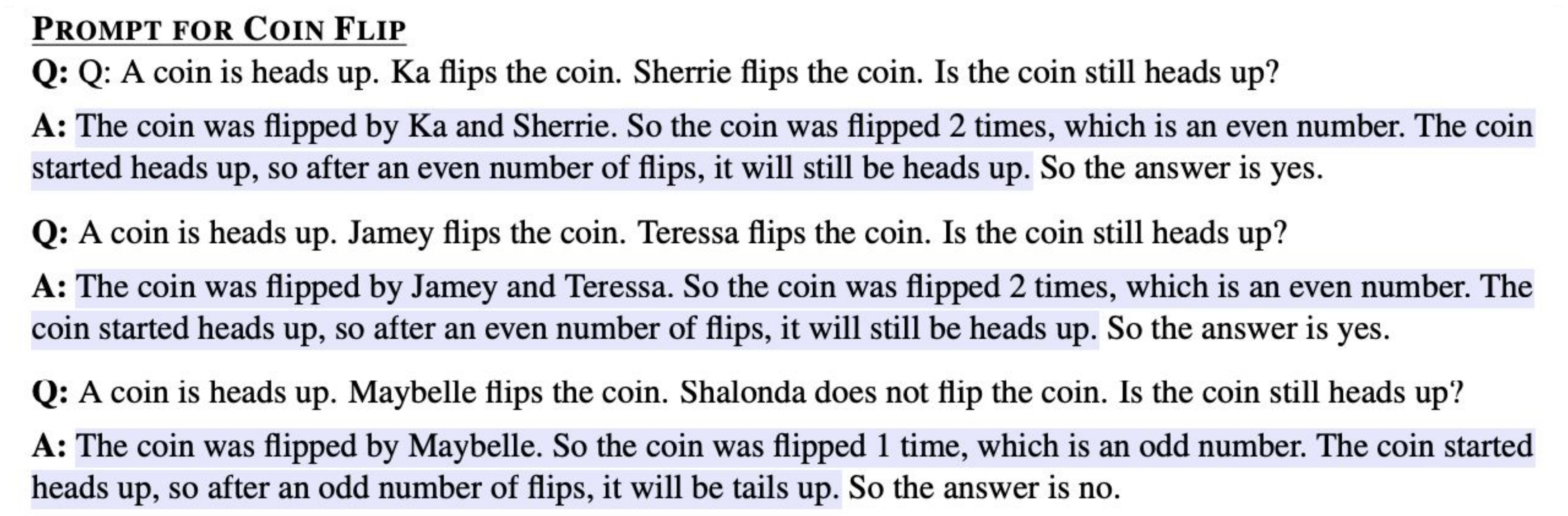
Table 13: Examples of correct and incorrect chains of thought produced by LaMDA 137B on the letter concatenation task.

**QUESTION:** Take the last letters of the words in "Waldo Schmidt" and concatenate them.

**MODEL ANSWER (CORRECT):** The last letter of "Waldo" is "o". The last letter of "Schmidt" is "t". Concatenating them is "ot". So the answer is ot. ✓

Image Source: Wei et al. Chain of Thought Prompting Elicits Reasoning in Large Language Models. 2022.

## Chain of thought for Physical Reasoning



**PROMPT FOR COIN FLIP**

**Q:** Q: A coin is heads up. Ka flips the coin. Sherrie flips the coin. Is the coin still heads up?

**A:** The coin was flipped by Ka and Sherrie. So the coin was flipped 2 times, which is an even number. The coin started heads up, so after an even number of flips, it will still be heads up. So the answer is yes.

**Q:** A coin is heads up. Jamey flips the coin. Teressa flips the coin. Is the coin still heads up?

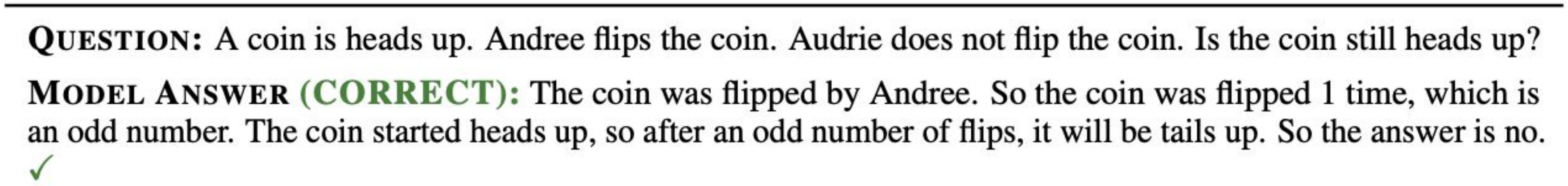
**A:** The coin was flipped by Jamey and Teressa. So the coin was flipped 2 times, which is an even number. The coin started heads up, so after an even number of flips, it will still be heads up. So the answer is yes.

**Q:** A coin is heads up. Maybelle flips the coin. Shalonda does not flip the coin. Is the coin still heads up?

**A:** The coin was flipped by Maybelle. So the coin was flipped 1 time, which is an odd number. The coin started heads up, so after an odd number of flips, it will be tails up. So the answer is no.

Image Source: Wei et al. Chain of Thought Prompting Elicits Reasoning in Large Language Models. 2022.

## Chain of thought for Physical Reasoning

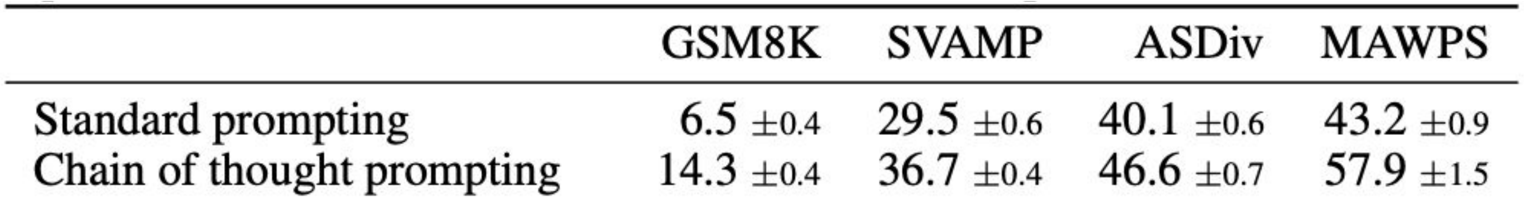


**QUESTION:** A coin is heads up. Andree flips the coin. Audrie does not flip the coin. Is the coin still heads up?

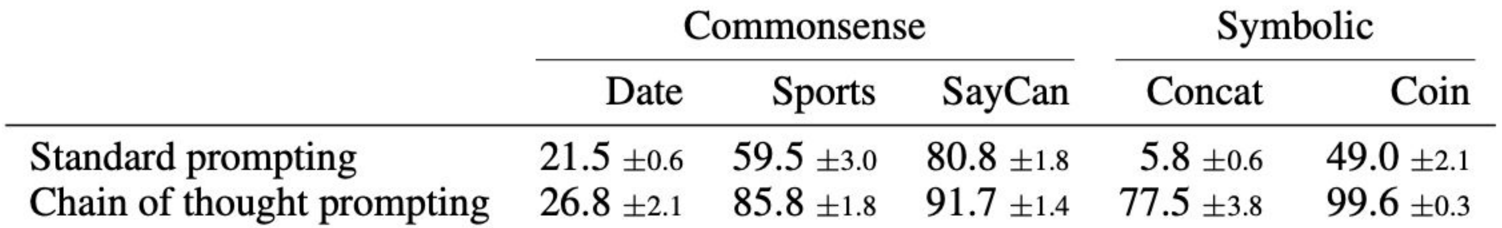
**MODEL ANSWER (CORRECT):** The coin was flipped by Andree. So the coin was flipped 1 time, which is an odd number. The coin started heads up, so after an odd number of flips, it will be tails up. So the answer is no. ✓

Image Source: [Wei et al. Chain of Thought Prompting Elicits Reasoning in Large Language Models. 2022.]

## Chain of thought results



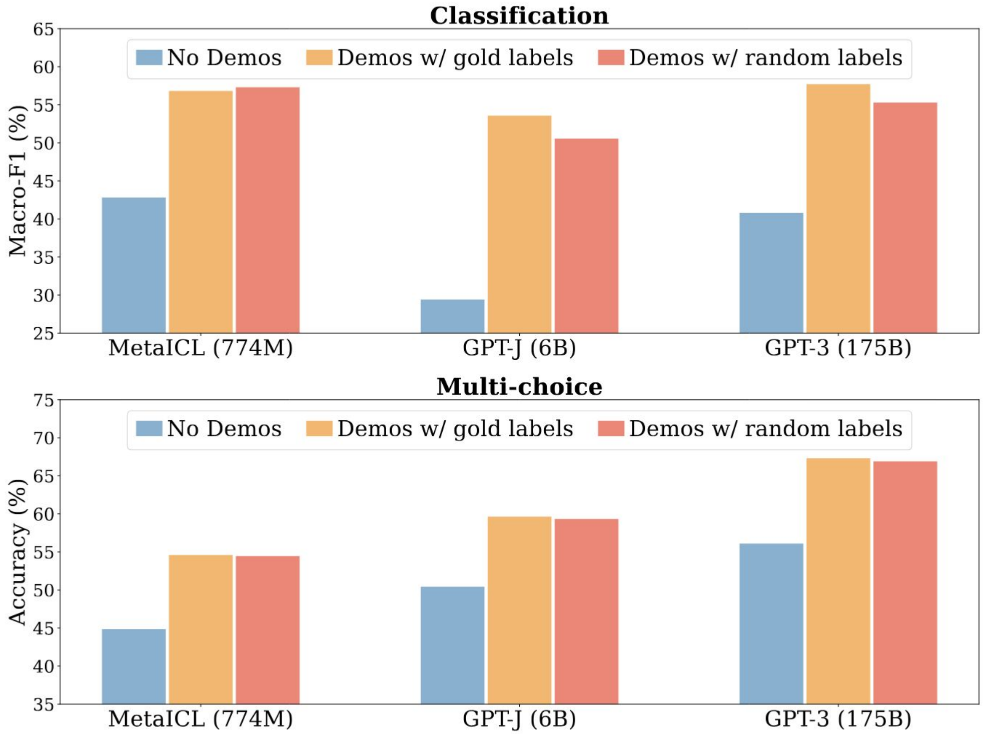
| | GSM8K | SVAMP | ASDiv | MAWPS |
|---|---|---|---|---|
| Standard prompting | 6.5 ±0.4 | 29.5 ±0.6 | 40.1 ±0.6 | 43.2 ±0.9 |
| Chain of thought prompting | 14.3 ±0.4 | 36.7 ±0.4 | 46.6 ±0.7 | 57.9 ±1.5 |



| | Commonsense | | | Symbolic | |
|---|---|---|---|---|---|
| | Date | Sports | SayCan | Concat | Coin |
| Standard prompting | 21.5 ±0.6 | 59.5 ±3.0 | 80.8 ±1.8 | 5.8 ±0.6 | 49.0 ±2.1 |
| Chain of thought prompting | 26.8 ±2.1 | 85.8 ±1.8 | 91.7 ±1.4 | 77.5 ±3.8 | 99.6 ±0.3 |

Image Source: Wei et al. Chain of Thought Prompting Elicits Reasoning in Large Language Models. 2022.

## In-context CoT traces need not be accurate (!)



Min et al., 2022

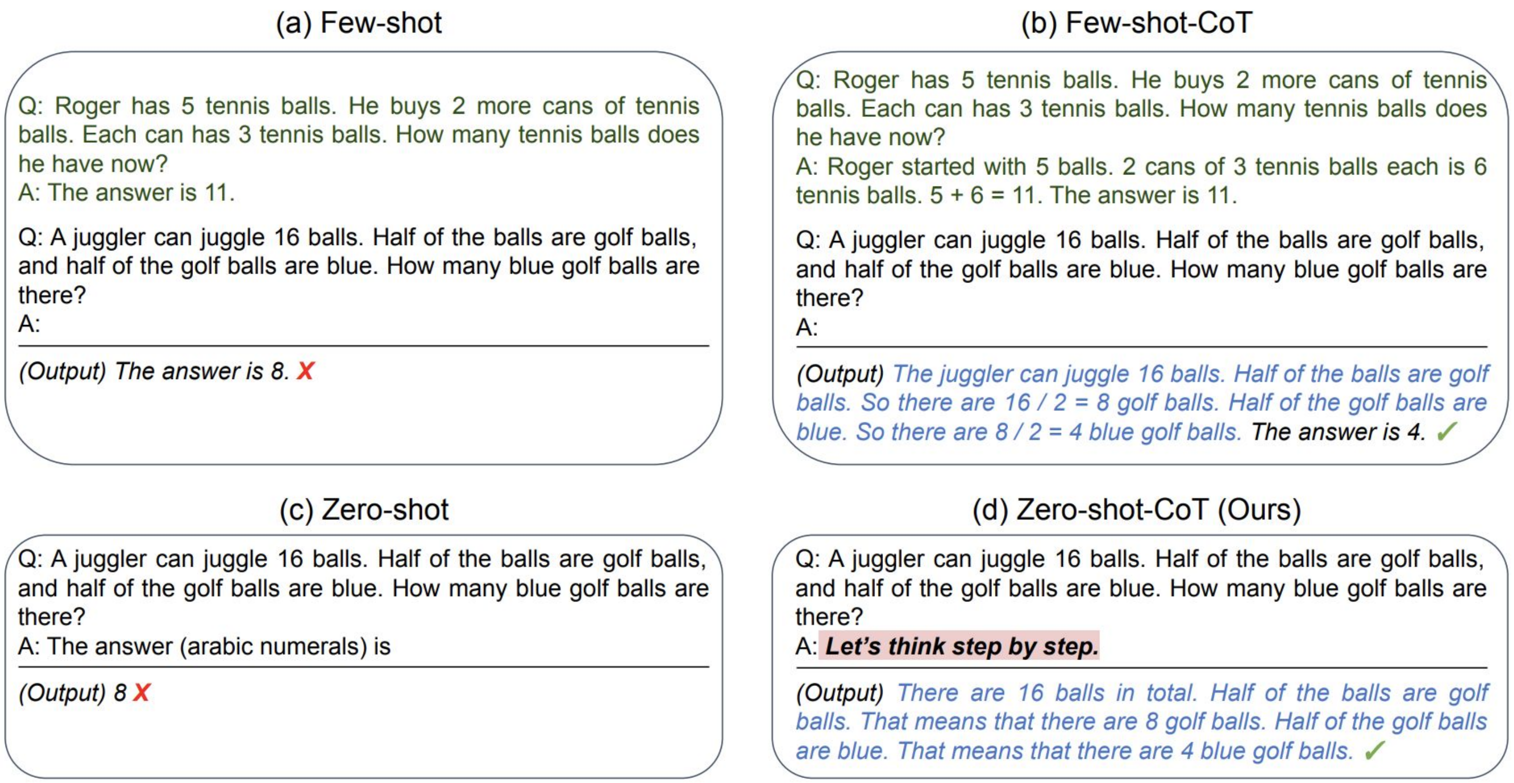
# Explicit CoT is usually not required today

Models are **fine-tuned** on CoT traces, so they often do CoT on their own

However, few-shot examples are still useful
    b/c CoT elicits a *behavior*, while few-shot provides useful information

Min et al., 2022

# Think step by step



(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

*(Output) The answer is 8.* ✗

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

*(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are 16 / 2 = 8 golf balls. Half of the golf balls are blue. So there are 8 / 2 = 4 blue golf balls. The answer is 4.* ✓

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: The answer (arabic numerals) is

*(Output) 8* ✗

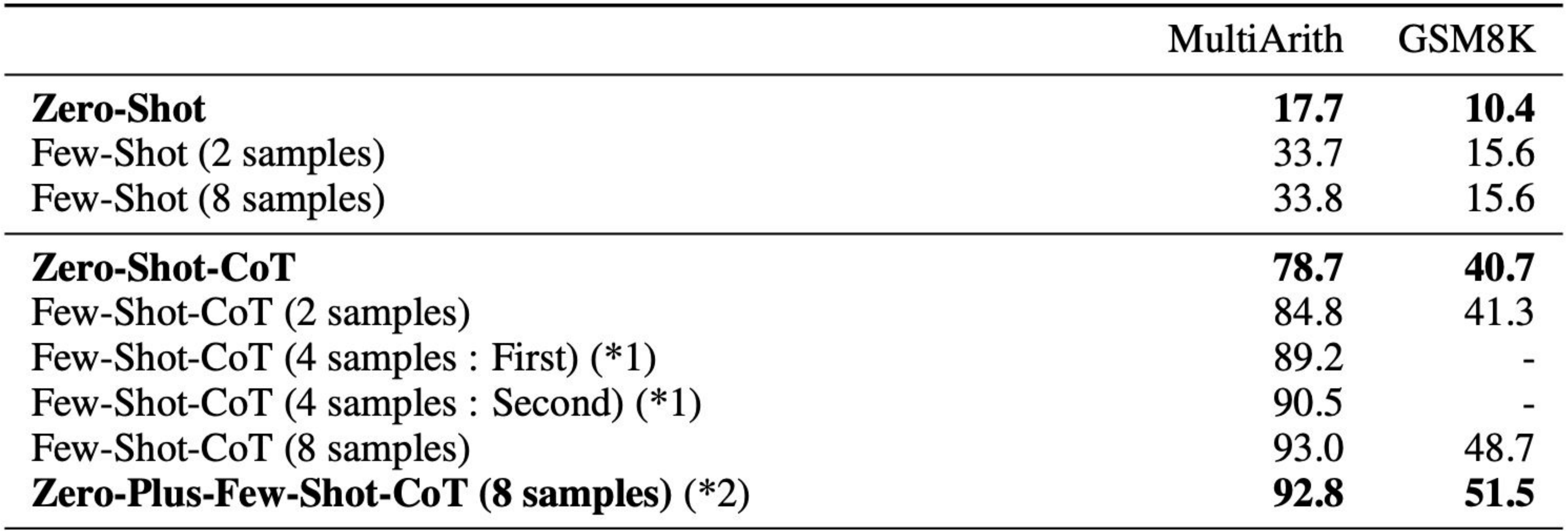
(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: ***Let's think step by step.***

*(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls.* ✓

Image Source: Chowdhery et al. Large Language Models are Zero-Shot Reasoners. 2022.

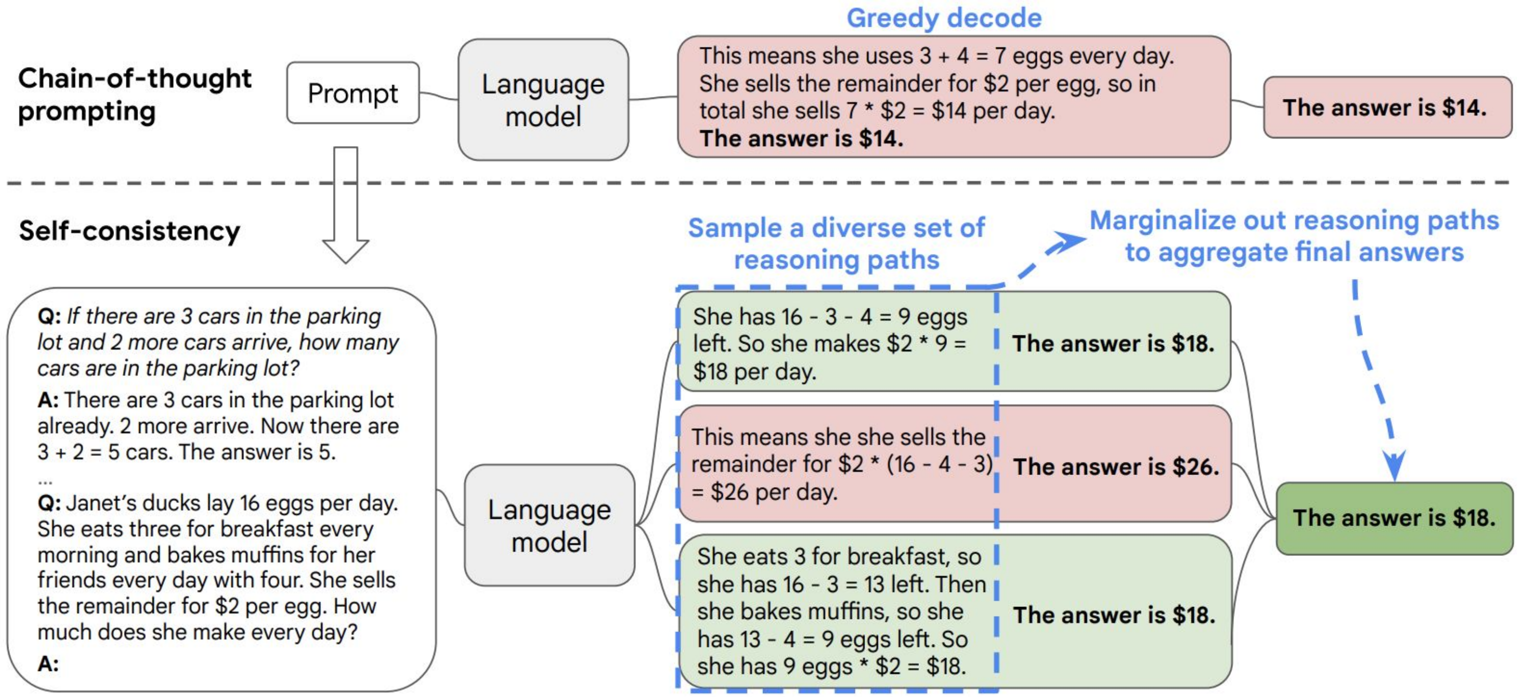
# Think step by step



| | MultiArith | GSM8K |
|---|---|---|
| **Zero-Shot** | **17.7** | **10.4** |
| Few-Shot (2 samples) | 33.7 | 15.6 |
| Few-Shot (8 samples) | 33.8 | 15.6 |
| **Zero-Shot-CoT** | **78.7** | **40.7** |
| Few-Shot-CoT (2 samples) | 84.8 | 41.3 |
| Few-Shot-CoT (4 samples : First) (*1) | 89.2 | - |
| Few-Shot-CoT (4 samples : Second) (*1) | 90.5 | - |
| Few-Shot-CoT (8 samples) | 93.0 | 48.7 |
| **Zero-Plus-Few-Shot-CoT (8 samples)** (*2) | **92.8** | **51.5** |

Image Source: Chowdhery et al. Large Language Models are Zero-Shot Reasoners. 2022.

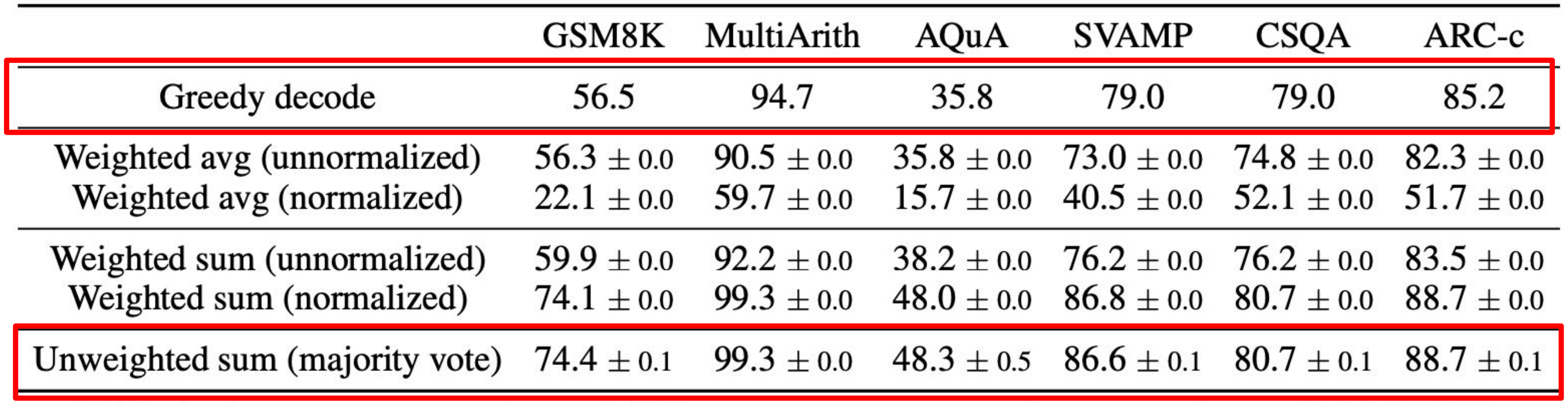
# Self Consistency



Chain-of-thought prompting
Prompt
Language model
Greedy decode
This means she uses 3 + 4 = 7 eggs every day. She sells the remainder for $2 per egg, so in total she sells 7 * $2 = $14 per day. The answer is $14.
The answer is $14.
Self-consistency
Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?
A: There are 3 cars in the parking lot already. 2 more arrive. Now there are 3 + 2 = 5 cars. The answer is 5.
...
Q: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder for $2 per egg. How much does she make every day?
A:
Language model
Sample a diverse set of reasoning paths
Marginalize out reasoning paths to aggregate final answers
She has 16 - 3 - 4 = 9 eggs left. So she makes $2 * 9 = $18 per day.
The answer is $18.
This means she she sells the remainder for $2 * (16 - 4 - 3) = $26 per day.
The answer is $26.
She eats 3 for breakfast, so she has 16 - 3 = 13 left. Then she bakes muffins, so she has 13 - 4 = 9 eggs left. So she has 9 eggs * $2 = $18.
The answer is $18.
The answer is $18.

Image Source: Xie et al. Self-Consistency Improves Chain of Thought Reasoning in Language Models. 2022.

## Self Consistency



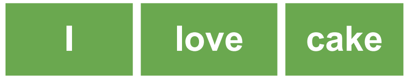
| | GSM8K | MultiArith | AQuA | SVAMP | CSQA | ARC-c |
|---|---|---|---|---|---|---|
| Greedy decode | 56.5 | 94.7 | 35.8 | 79.0 | 79.0 | 85.2 |
| Weighted avg (unnormalized) | 56.3 ± 0.0 | 90.5 ± 0.0 | 35.8 ± 0.0 | 73.0 ± 0.0 | 74.8 ± 0.0 | 82.3 ± 0.0 |
| Weighted avg (normalized) | 22.1 ± 0.0 | 59.7 ± 0.0 | 15.7 ± 0.0 | 40.5 ± 0.0 | 52.1 ± 0.0 | 51.7 ± 0.0 |
| Weighted sum (unnormalized) | 59.9 ± 0.0 | 92.2 ± 0.0 | 38.2 ± 0.0 | 76.2 ± 0.0 | 76.2 ± 0.0 | 83.5 ± 0.0 |
| Weighted sum (normalized) | 74.1 ± 0.0 | 99.3 ± 0.0 | 48.0 ± 0.0 | 86.8 ± 0.0 | 80.7 ± 0.0 | 88.7 ± 0.0 |
| Unweighted sum (majority vote) | 74.4 ± 0.1 | 99.3 ± 0.0 | 48.3 ± 0.5 | 86.6 ± 0.1 | 80.7 ± 0.1 | 88.7 ± 0.1 |

Table 1: Accuracy comparison of different answer aggregation strategies on PaLM-540B.

Image Source: Xie et al. Self-Consistency Improves Chain of Thought Reasoning in Language Models. 2022.

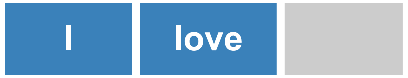
## Summary Slide

**Encoder Only:** Capture the meaning of an entire sequence



**ELMO:** Bi-directional next word prediction,
**BERT:** Masked language objective, Next Sentence Prediction

**Decoder Only:** Generate text based on previously generated text

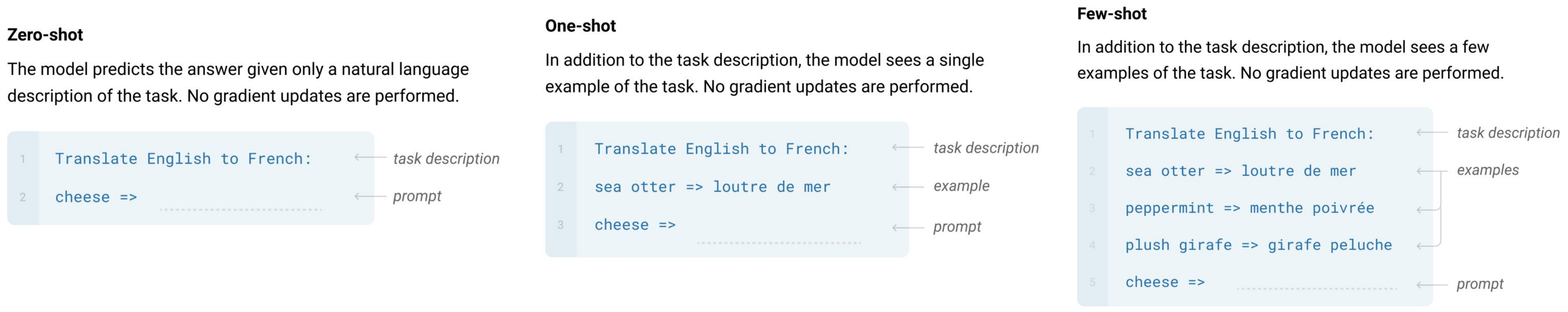


**GPT:** next token prediction (autoregressive)

**Encoder-Decoder:** Generate text based on previously generated text and the meaning of a separate sequence **T5:** Masked language objective

I
love
cake
me
gusta

# Summary



**Zero-shot**

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

```
Translate English to French:
cheese =>
```

**One-shot**

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

```
Translate English to French:
sea otter => loutre de mer
cheese =>
```

**Few-shot**

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

```
Translate English to French:
sea otter => loutre de mer
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>
```

Image Source: [Language Models are Few-Shot Learners, Brown et al]()

## Summary

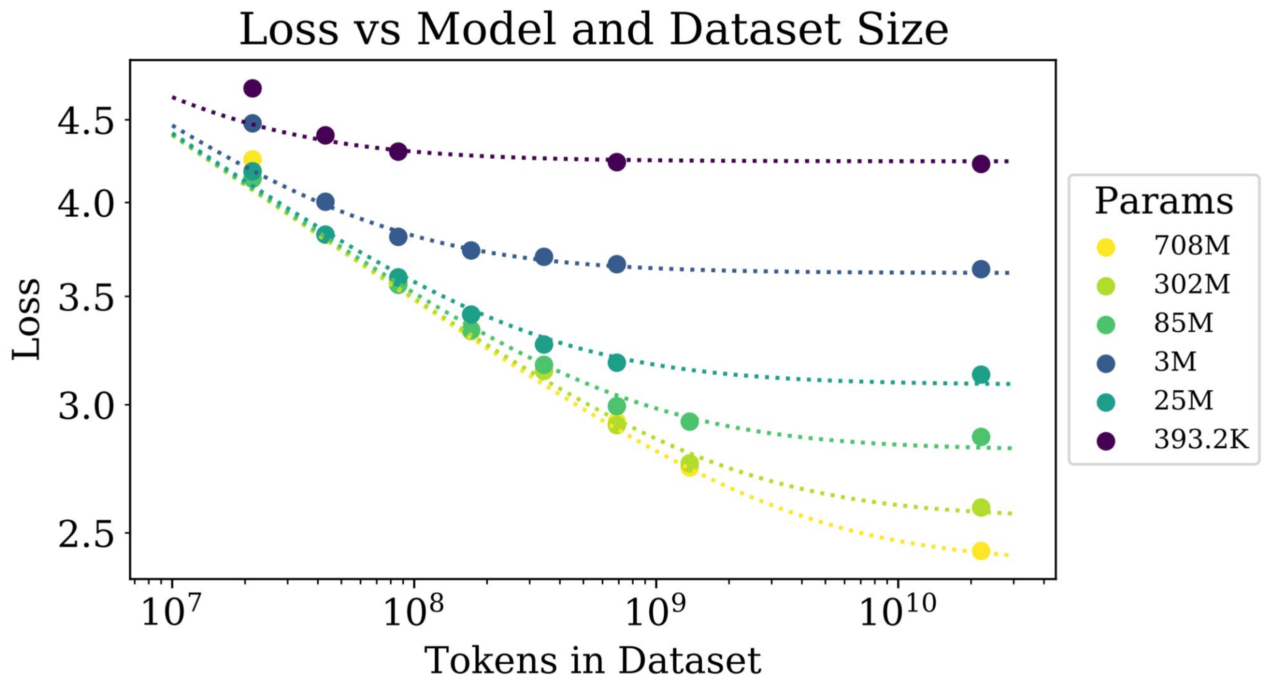


Image source: [Kaplan et al. Scaling Laws for Neural Language Models. 2020.]

## Summary

**Inference Only Performance Improvement**

- **Chain-of-thought**
- **Think step by step**
- **Self consistency**

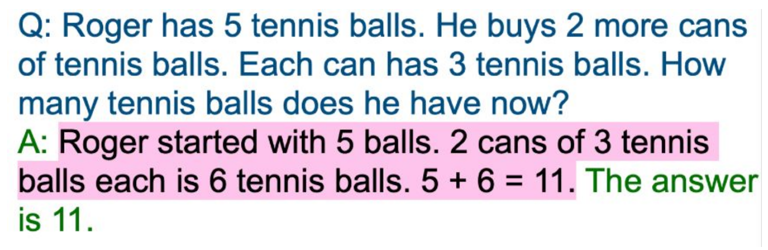

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Image Source: Wei et al. Chain of Thought Prompting Elicits Reasoning in Large Language Models. 2022.

Intelligence is just an emergent property of predicting the next token?
imgflip.com

https://twitter.com/alirahimi0