



Large Pretrained Models

Vision Transformers, Vision-Language Pretraining, Zero-shot Learning, Scaling Laws

Antonio Carta

antonio.carta@unipi.it

Motivation



Foundation models open the doors to novel large-scale CL applications

- They work in a very dynamic environment
 - They must be aware of factual knowledge («what is the last album released by Spiritbox?»)
 - They need to be personalized
 - Advanced models are proper agents that use tools (browser, command line) and interact with their environment
- They require rethinking what «learning» means
 - Weight-based learning is unfeasible
 - In-context-learning provides cheap and effective adaptation

Outline



- **Data:** quality, evaluation, scaling laws
- **Models:** Vision and Vision-Language Transformers
- **Efficient Finetuning:** Distillation, LoRA, Prompt Tuning
- **Research:** Model patching, model editing, explaining ICL

Large-Scale Pretraining

Training on a Web-Scale



- We have a very bad intuition about how much diversity there is in web-scale data
 - In general, much more diversity than we expect
 - There are examples of general tasks such as translation
- language provides a flexible way to specify tasks, inputs, and outputs all as a sequence of symbols
 - How do we use language to achieve few-shot and zero-shot learning?

"I'm not the cleverest man in the world, but like they say in French: **Je ne suis pas un imbecile** [I'm not a fool]."

In a now-deleted post from Aug. 16, Soheil Eid, Tory candidate in the riding of Joliette, wrote in French: "**Mentez mentez, il en restera toujours quelque chose**," which translates as, "Lie lie and something will always remain."

"I hate the word 'perfume,'" Burr says. 'It's somewhat better in French: 'parfum.'

If listened carefully at 29:55, a conversation can be heard between two guys in French: "**-Comment on fait pour aller de l'autre coté? -Quel autre coté?**", which means "- How do you get to the other side? - What side?"

If this sounds like a bit of a stretch, consider this question in French: **As-tu aller au cinéma?**, or **Did you go to the movies?**, which literally translates as Have-you to go to movies/theater?

"Brevet Sans Garantie Du Gouvernement", translated to English: **"Patented without government warranty"**.

Table 1. Examples of naturally occurring demonstrations of English to French and French to English translation found throughout the WebText training set.

Few-Shot and Zero-Shot Learning



Given a large pretrained language model

- **In-Context-Learning/Prompting:** at inference time we give a description of the task, (optionally) followed by a sequence of examples
 - No weights are updated
- **Few-Shot:** the model is given a few samples of the task as conditioning
- **One-Shot:** a single example
- **Zero-Shot:** only the task description is given

Examples



Context → Article:
Informal conversation is an important part of any business relationship. Before you start a discussion, however, make sure you understand which topics are suitable and which are considered taboo in a particular culture. Latin Americans enjoy sharing information about their local history, art and customs. You may expect questions about your family, and be sure to show pictures of your children. You may feel free to ask similar questions of your Latin American friends. The French think of conversation as an art form, and they enjoy the value of lively discussions as well as disagreements. For them, arguments can be interesting and they can cover pretty much or any topic ---- as long as they occur in a respectful and intelligent manner.
In the United States, business people like to discuss a wide range of topics, including opinions about work, family, hobbies, and politics. In Japan, China, and Korea, however, people are much more private. They do not share much about their thoughts, feelings, or emotions because they feel that doing so might take away from the harmonious business relationship they're trying to build. Middle Easterners are also private about their personal lives and family matters. It is considered rude, for example, to ask a businessman from Saudi Arabia about his wife or children. As a general rule, it's best not to talk about politics or religion with your business friends. This can get you into trouble, even in the United States, where people hold different religious views. In addition, discussing one's salary is usually considered unsuitable. Sports is typically a friendly subject in most parts of the world, although be careful not to criticize national sport. Instead, be friendly and praise your host's team.

Q: What shouldn't you do when talking about sports with colleagues from another country?

A: Criticizing the sports of your colleagues' country.

Q: Which is typically a friendly topic in most places according to the author?

A: Sports.

Q: Why are people from Asia more private in their conversation with others?

A: They don't want to have their good relationship with others harmed by informal conversation.

Q: The author considers politics and religion . .

A:

Correct Answer → taboo
Incorrect Answer → cheerful topics
Incorrect Answer → rude topics
Incorrect Answer → topics that can never be talked about

Figure G.1: Formatted dataset example for RACE-h. When predicting, we normalize by the unconditional probability of each answer.

Examples of Prompts



Context →	How to apply sealant to wood.
Correct Answer →	Using a brush, brush on sealant onto wood until it is fully saturated with the sealant.
Incorrect Answer →	Using a brush, drip on sealant onto wood until it is fully saturated with the sealant.

Figure G.4: Formatted dataset example for PIQA

LLMs are Few-Shot Learners



Setting	NaturalQS	WebQS	TriviaQA
RAG (Fine-tuned, Open-Domain) [LPP ⁺ 20]	44.5	45.5	68.0
T5-11B+SSM (Fine-tuned, Closed-Book) [RRS20]	36.6	44.7	60.5
T5-11B (Fine-tuned, Closed-Book)	34.5	37.4	50.1
GPT-3 Zero-Shot	14.6	14.4	64.3
GPT-3 One-Shot	23.0	25.3	68.0
GPT-3 Few-Shot	29.9	41.5	71.2

Table 3.2: Results on three Open-Domain QA tasks. GPT-3 is shown in the few-, one-, and zero-shot settings, as compared to prior SOTA results for closed book and open domain settings. TriviaQA few-shot result is evaluated on the wiki split test server.

Setting	ARC (Easy)	ARC (Challenge)	CoQA	DROP
Fine-tuned SOTA	92.0^a	78.5^b	90.7^c	89.1^d
GPT-3 Zero-Shot	68.8	51.4	81.5	23.6
GPT-3 One-Shot	71.2	53.2	84.0	34.3
GPT-3 Few-Shot	70.1	51.5	85.0	36.5

Table 3.3: GPT-3 results on a selection of QA / RC tasks. CoQA and DROP are F1 while ARC reports accuracy. See the appendix for additional experiments. ^a[KKS⁺20] ^b[KKS⁺20] ^c[JZC⁺19] ^d[JN20]

Challenges



- Prompting requires rethinking the adaptation and finetuning of models
- It's hard to evaluate LLMs zero-shot ability
 - How can we ensure that the task is really unseen?
- Still hard to control the model or estimate uncertainty

Challenges in the Evaluation



- Github CoPilot memorized the solution of many programming challenges
- Example: Project Euler #14
 - A simple problem about the convergence of Collatz sequences

How to get your solution

- Add a comment with the link to project euler
- `#`
`https://projecteuler.net/problem=14`
- Press tab or enter until the code is complete. Copilot will write the full solution

```
# https://projecteuler.net/problem=14

# imports
import time

# Measure runtime
start = time.time()

# Variables
longest_chain = 0
longest_chain_start = 0

# Functions
def collatz(n):
    chain = 1
    while n != 1:
        if n % 2 == 0:
            n = n/2
            chain += 1
        else:
            n = 3*n + 1
            chain += 1
    return chain

# Main
for i in range(1, 1000000):
    chain = collatz(i)
    if chain > longest_chain:
        longest_chain = chain
        longest_chain_start = i

print(longest_chain_start)

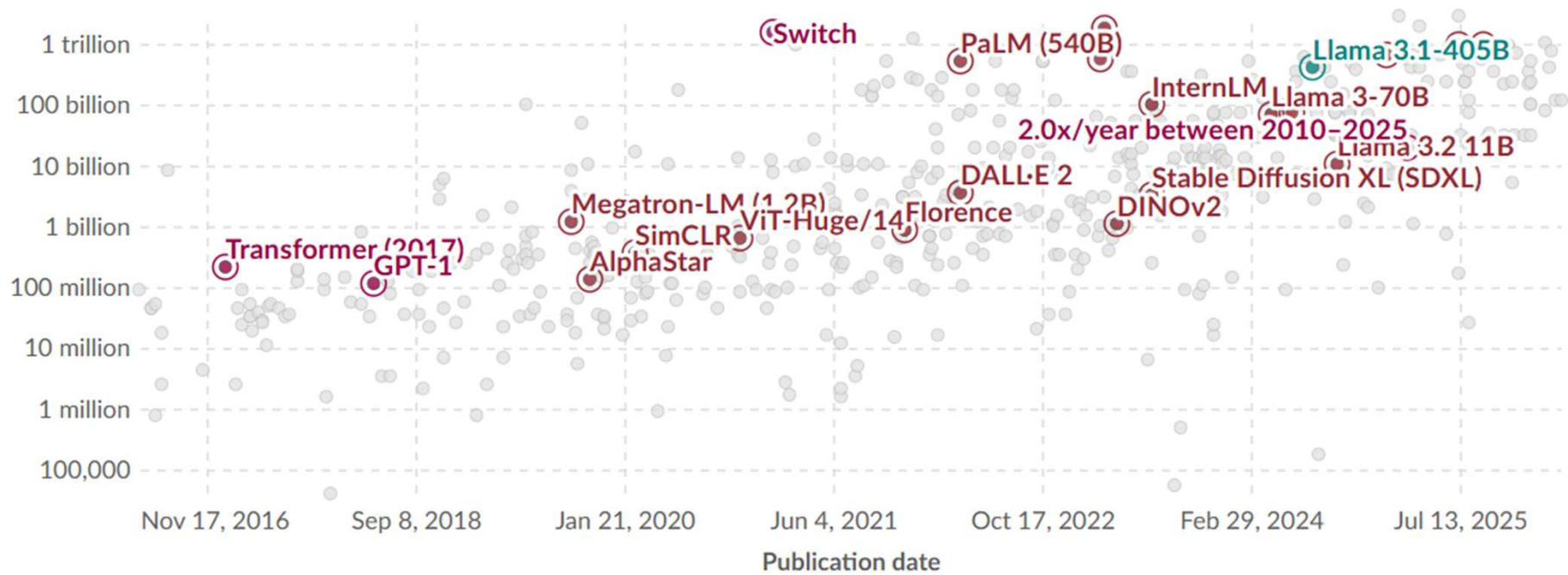
# Print runtime
print("--- %.2f seconds ---" %(time.time() - start))
```

Data and Scaling Laws

Model Scale



Number of parameters (plotted on a logarithmic axis)



Source: ourworldindata

Training Large (Huge) Models



Training large models is very expensive

- You can train a single run (it cost M\$)
- You would like to know the cost in advance, whether the performance improvements are worth the cost

Problems:

- How do we do model selection?
- How do we choose the model size?
- How much data do we need?
- What error will we get?

Model Selection



- You can't do it at large scale. A training is a single run, guided by experience gained from multiple runs at smaller scales
 - We need the ability to infer how performance will scale
- Lots of infrastructure issues happen at scale
 - If models become unstable recover from previous checkpoints
 - Needs to account for machine failures (the model is trained on a cluster)
- Helpful guide: [GitHub - stas00/ml-engineering: Machine Learning Engineering Open Book](#)

Chinchilla Scaling Law



- **Chinchilla Scaling Law:** an empirical law that determines the loss given
 - **N:** number of parameters of the model
 - **D:** number of tokens in the dataset
 - Three components: model error, data error, irreducible loss

$$L(N, D) = \underbrace{\frac{406.4}{N^{0.34}}}_{\text{finite model}} + \underbrace{\frac{410.7}{D^{0.28}}}_{\text{finite data}} + \underbrace{1.69}_{\text{irreducible}}$$

Compute-Optimal Models

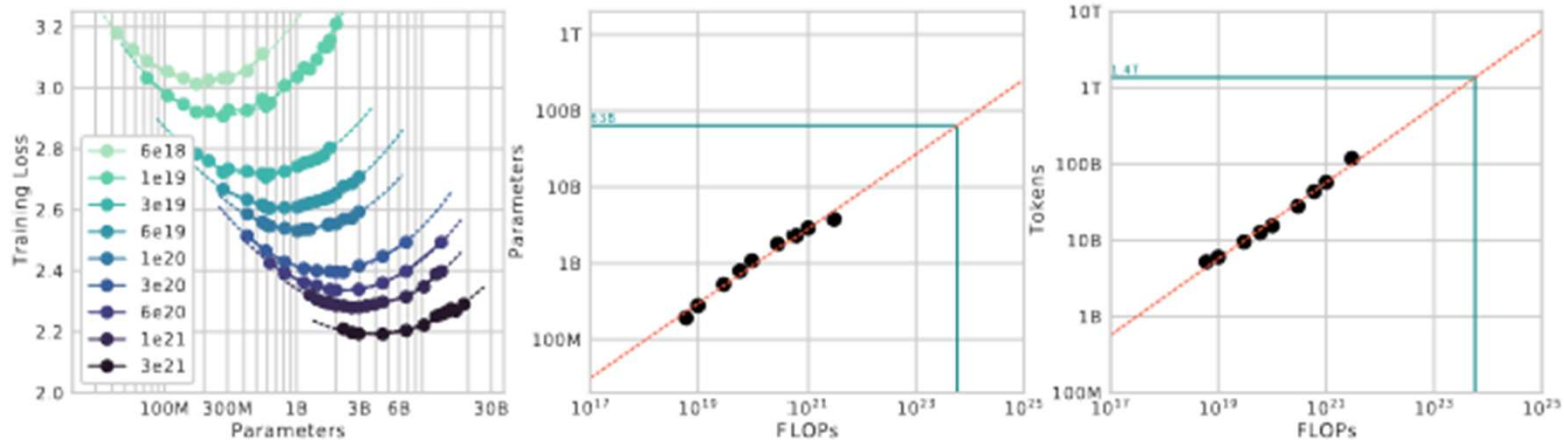


Figure 3 | IsoFLOP curves. For various model sizes, we choose the number of training tokens such that the final FLOPs is a constant. The cosine cycle length is set to match the target FLOP count. We find a clear valley in loss, meaning that for a given FLOP budget there is an optimal model to train (left). Using the location of these valleys, we project optimal model size and number of tokens for larger models (center and right). In green, we show the estimated number of parameters and tokens for an *optimal* model trained with the compute budget of *Gopher*.

Compute-Optimal Models



- Larger models are not always better, you also need larger data
- Given a fixed computational budget and a dataset, you can find the optimal model size
- The optimality is wrt to the loss and training cost.
 - Inference cost is ignored and it can be high due to the large model size suggested by the Eq.
- Unclear if the scaling law will hold for larger models

How much data do we have?



If the trend continues

- **high-quality language data** will be exhausted soon; likely before 2026.
- **low-quality** language data and image data will be exhausted only much later; between 2030 and 2050 (for low-quality language) and between 2030 and 2060 (for images).

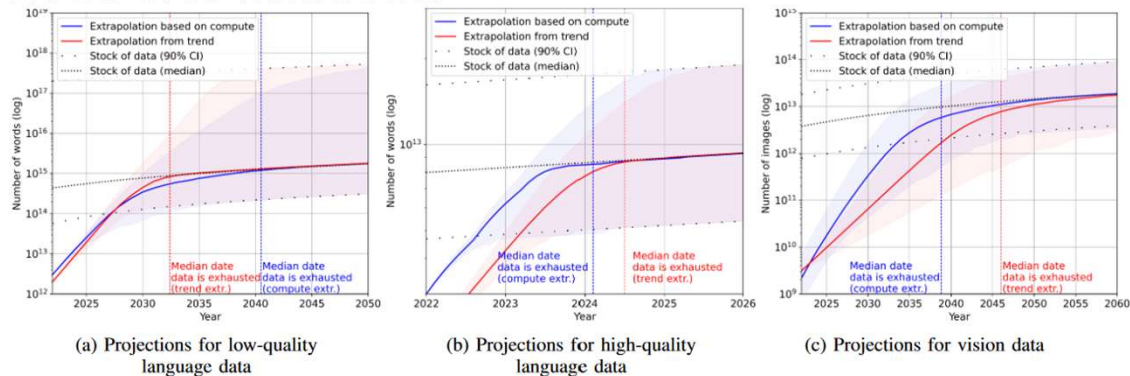


Fig. 1: Projections of data usage. Each graph shows two extrapolations of data usage, one from past trends and one from compute availability estimations plus scaling laws. Both projections are constrained to be lower than the estimated data stock. In all three cases, this constraint causes a slowdown in data usage growth.

Model	Stock of data (#words)	Growth rate
Recorded speech	1.46e17	5.2%
	[3.41e16; 4.28e17]	[4.95%; 5.2%]
Internet users	2.01e15	8.14%
	[6.47e14; 6.28e15]	[7.89%; 8.14%]
Popular platforms	4.41e14	8.14%
	[1.21e14; 1.46e15]	[7.89%; 8.14%]
CommonCrawl	9.62e13	16.68%
	[4.45e13; 2.84e14]	[16.41%; 16.68%]
Indexed websites	2.21e14	NA
	[5.16e13; 6.53e15]	
Aggregated model	7.41e14	7.15%
	[6.85e13; 7.13e16]	[6.41%; 17.49%]

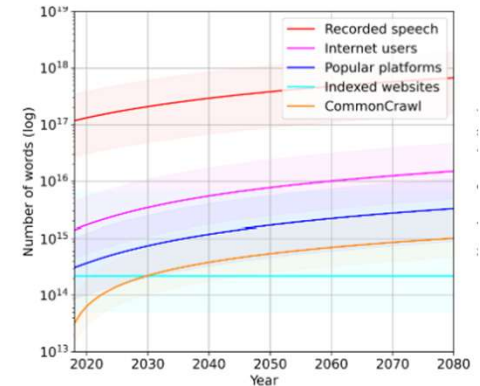
Domain	Doubling time median and CI (months)	Largest training dataset (datapoints)
Language	15.8 [11.2; 20.9]	2e12
Vision	41.5 [30.4; 48.3]	3e9

TABLE I: Trends in training dataset size for vision and language models.

Component	Raw Size	Data source
Pile-CC	227.12 GiB	Social media conversations
PubMed Central	90.27 GiB	Filtered webpages
Books3 [†]	100.96 GiB	Books (English)
OpenWebText2	62.77 GiB	GitHub (code)
ArXiv	56.21 GiB	Wikipedia (multilingual)
GitHub	95.16 GiB	News (English)
FreeLaw	51.15 GiB	
Stack Exchange	32.20 GiB	
USPTO Backgrounds	22.90 GiB	
PubMed Abstracts	19.26 GiB	
Gutenberg (PG-19) [‡]	10.88 GiB	
OpenSubtitles [†]	12.98 GiB	

	Disk Size	Documents
MassiveWeb	1.9 TB	604M
Books	2.1 TB	4M
C4	0.75 TB	361M
News	2.7 TB	1.1B
GitHub	3.1 TB	142M
Wikipedia	0.001 TB	6M

(a) Composition of high-quality datasets: The Pile (left), PaLM (top-right), MassiveText (bottom-right)



(a) Comparison of the different data stock models.

Conclusions



- Training large models is both hard and simple
- **Hard:** we are working with a large distributed system
 - Hardware is going to break
 - Need careful optimization
 - We can afford only one training run
- **Simple:** the results are relatively predictable
 - We can estimate improvements due to more data/parameters
 - Setting hyperparameters is easier in this setting. We don't need as much hyperparameter selection

Vision Transformer

Attention



- $\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$
- $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$
 - Transformers use multiple heads, each transformed by different linear projections
- $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$
 - Scaled dot-product attention
 - Q query
 - K keys
 - V values

Vision Transformer

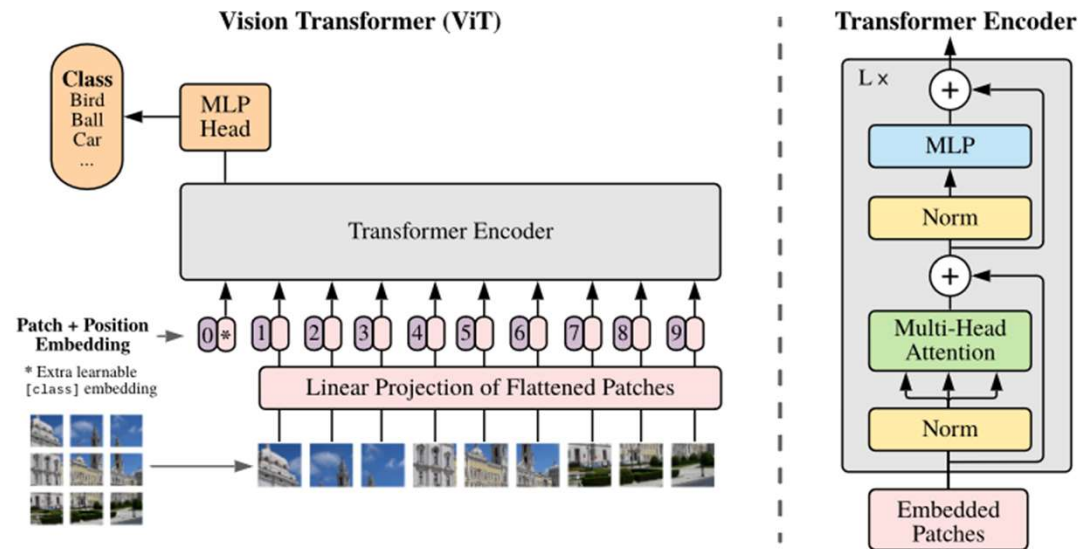


Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable “classification token” to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017).

Embedding Image Patches



- Split images in 16x16 patches
- Add positional embedding
 - Learnable 1D embeddings
 - Needed because attention is invariant to the patch position
- Once we have the 1D embeddings, we can use the Transformer like we do for text

ViT Block



- MSA: Multiheaded Self-Attention
- LN: Layer Normalization
- GELU nonlinearity
- $\mathbf{z}_0 = [\mathbf{x}_{\text{class}} ; \mathbf{x}_p^1 \mathbf{E}; \mathbf{x}_p^2 \mathbf{E}; \dots ; \mathbf{x}_p^N \mathbf{E}] + \mathbf{E}_{\text{pos}}$
- $\mathbf{z}'_\ell = \text{MSA}(\text{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \ell = 1 \dots L$
- $\mathbf{z}_\ell = \text{MLP}(\text{LN}(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell, \ell = 1 \dots L$
- $\mathbf{y} = \text{LN}(\mathbf{z}_L^0)$

- $\mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D}, \mathbf{E}_{\text{pos}} \in \mathbb{R}^{(N+1) \times D}$
- \mathbf{E} projects embedding into latent space
- Image $H \times W \times C$
- Patch $P^2 C$ (flattened patch)
- D latent size dimension
- $\mathbf{x}_{\text{class}}$ is a learned embedding and provides the final image representation at the last layer (equivalent to `BERT[class]` token)

Results

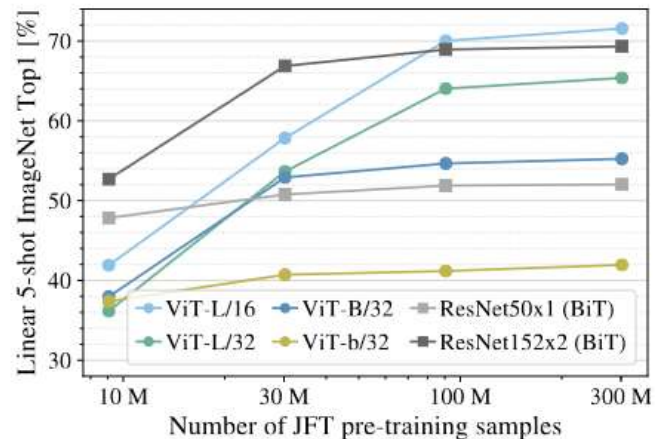


Figure 4: Linear few-shot evaluation on ImageNet versus pre-training size. ResNets perform better with smaller pre-training datasets but plateau sooner than ViT, which performs better with larger pre-training. ViT-b is ViT-B with all hidden dimensions halved.

Vision-Language Models

Motivation



- With LLM we can do zero-shot training with prompting
- We want to exploit language to encode visual information
 - Language encodes a high-level understanding of images
 - We have lots of captioned images that provide much more information than using the image alone
- We need to combine vision and language models

Vision-Language Pretraining with CLIP

- **pre-training task:** learn to map text captions with visual images
- **zero-shot transfer:** natural language is used to reference learned visual concepts (or describe new ones)

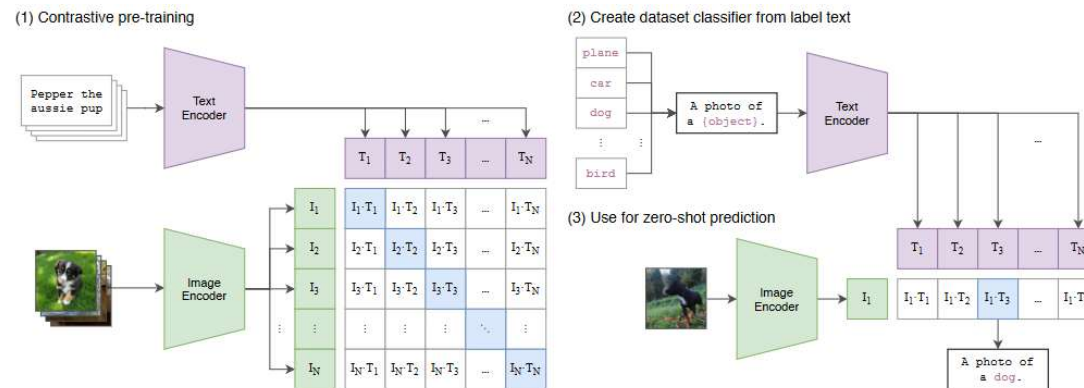
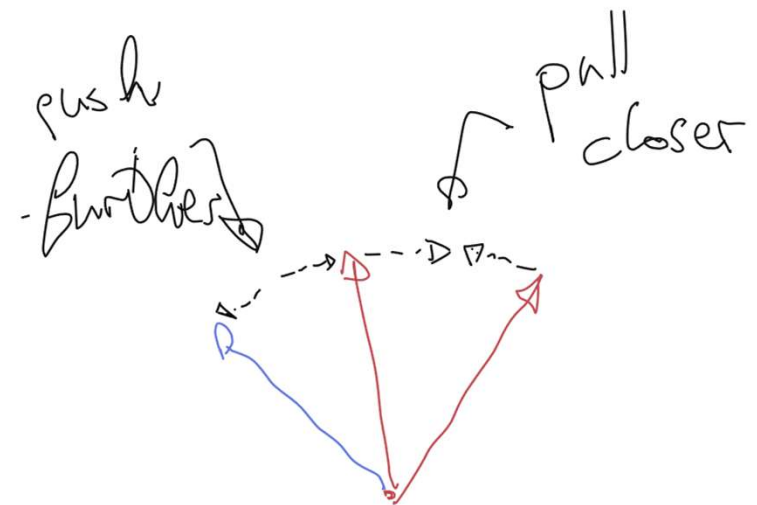


Figure 1. Summary of our approach. While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's classes.

Contrastive Learning

- **Objective:** learn a latent representation space that is semantically meaningful (for downstream tasks)
- **Example:** in a downstream classification problem, we would like the examples to be linearly separable
- **IDEA:** similar images are close to each other, diverse images are far
- **contrastive learning:** compare and contrast pairs of images during training to learn good representations



Contrastive Pretraining



- **Supervised training** is inefficient if we try to learn from $\langle \textit{caption}, \textit{image} \rangle$ pairs
 - There is too much diversity in the possible images and captions
 - It is very difficult to predict the exact image/captions of the current pair
- Solution: **contrastive training**
 - Given a batch of $\langle \textit{caption}, \textit{image} \rangle$ pairs, predict which text caption maps with which image among the $N \times N$ possibilities
 - Learns a multi-modal embedding space where text and images are aligned

CLIP Pretraining

```

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

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

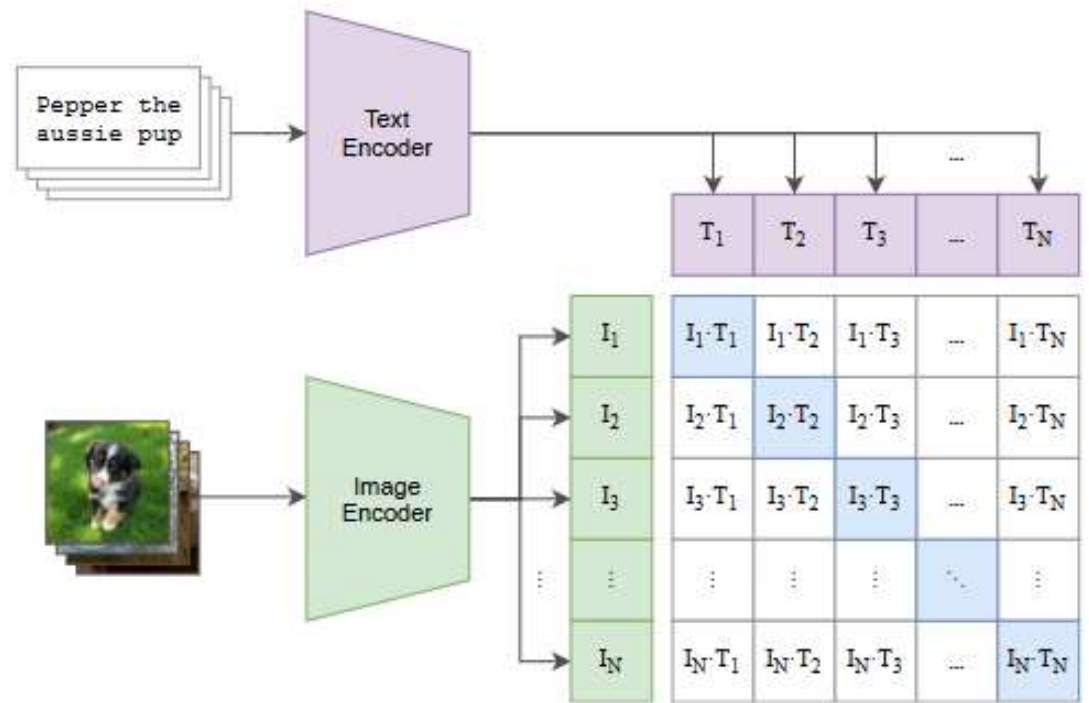
# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

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

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

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

(1) Contrastive pre-training



Zero-Shot Image Classification



- Given a number of classes {«dog», «cat», ... }
- Create prompts: «this is an image of {class}»
- For each image, find the closest prompt
 - You can encode the text prompt and use its embedding to get a linear classification head

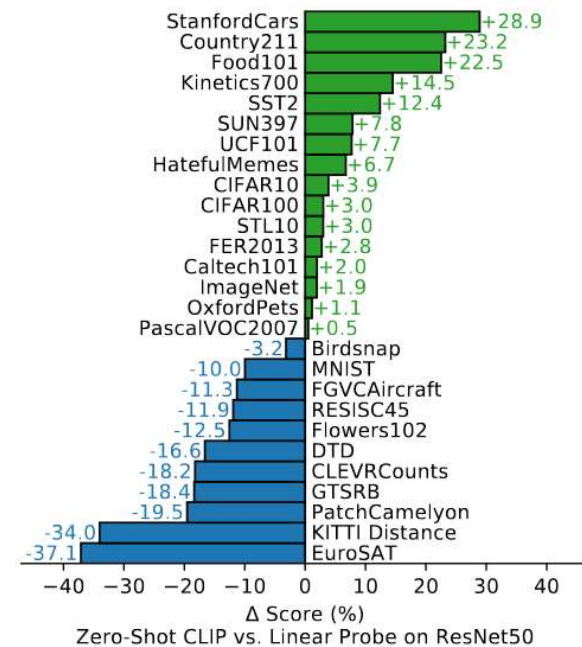


Figure 5. Zero-shot CLIP is competitive with a fully supervised baseline. Across a 27 dataset eval suite, a zero-shot CLIP classifier outperforms a fully supervised linear classifier fitted on ResNet-50 features on 16 datasets, including ImageNet.

Zero-Shot Image Classification



- **Ensembling:** You can use multiple prompts for the same class:
 - «a photo of a small {class}»
 - «a photo of a big {class}»
- **Prompt Engineering:** different prompts can have different results
 - E.g. «{class}» vs «a photo of {class}»
- **Paper Results:** (on ImageNet)
 - ensemble 80 different context prompts (+3.5%) w.r.t. single default prompt
 - prompt engineering and ensembling improve (+5%)
 - In general, ensembling and prompt engineering have a big impact

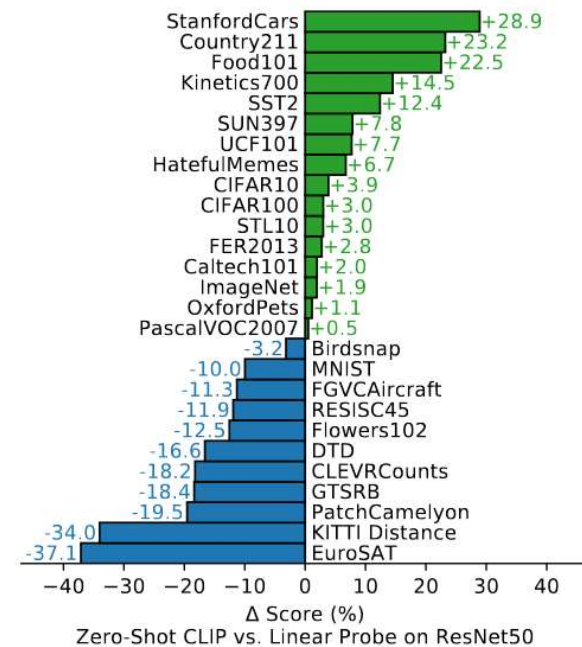


Figure 5. Zero-shot CLIP is competitive with a fully supervised baseline. Across a 27 dataset eval suite, a zero-shot CLIP classifier outperforms a fully supervised linear classifier fitted on ResNet-50 features on 16 datasets, including ImageNet.

CLIP Pretraining is More Sample-Efficient

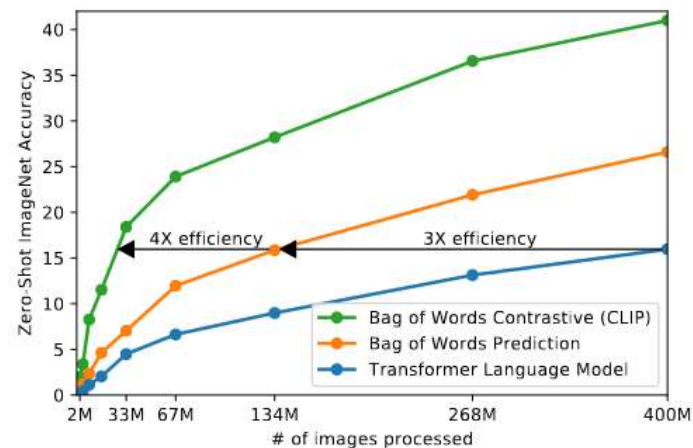


Figure 2. CLIP is much more efficient at zero-shot transfer than our image caption baseline. Although highly expressive, we found that transformer-based language models are relatively weak at zero-shot ImageNet classification. Here, we see that it learns 3x slower than a baseline which predicts a bag-of-words (BoW) encoding of the text (Joulin et al., 2016). Swapping the prediction objective for the contrastive objective of CLIP further improves efficiency another 4x.

Robustness to Dataset Shift

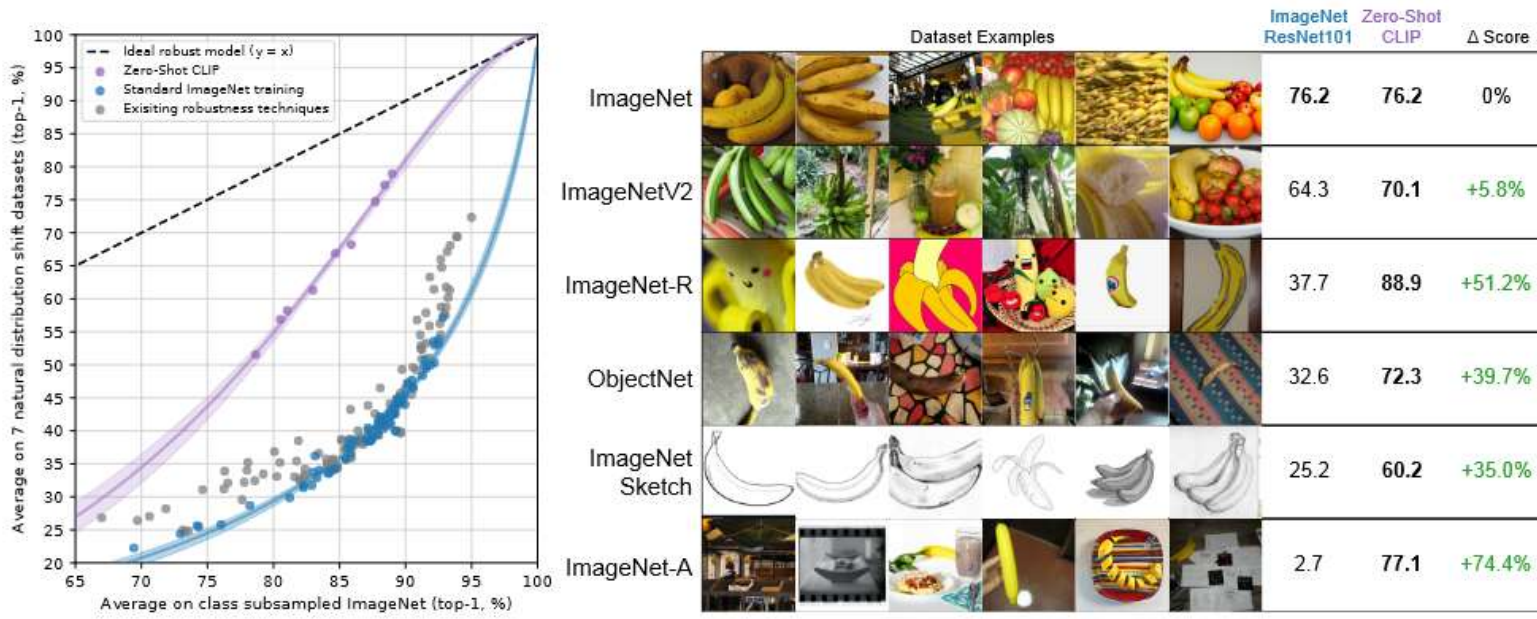


Figure 13. Zero-shot CLIP is much more robust to distribution shift than standard ImageNet models. (Left) An ideal robust model (dashed line) performs equally well on the ImageNet distribution and on other natural image distributions. Zero-shot CLIP models shrink this “robustness gap” by up to 75%. Linear fits on logit transformed values are shown with bootstrap estimated 95% confidence intervals. (Right) Visualizing distribution shift for bananas, a class shared across 5 of the 7 natural distribution shift datasets. The performance of the best zero-shot CLIP model, ViT-L/14@336px, is compared with a model that has the same performance on the ImageNet validation set, ResNet-101.

Efficient Finetuning

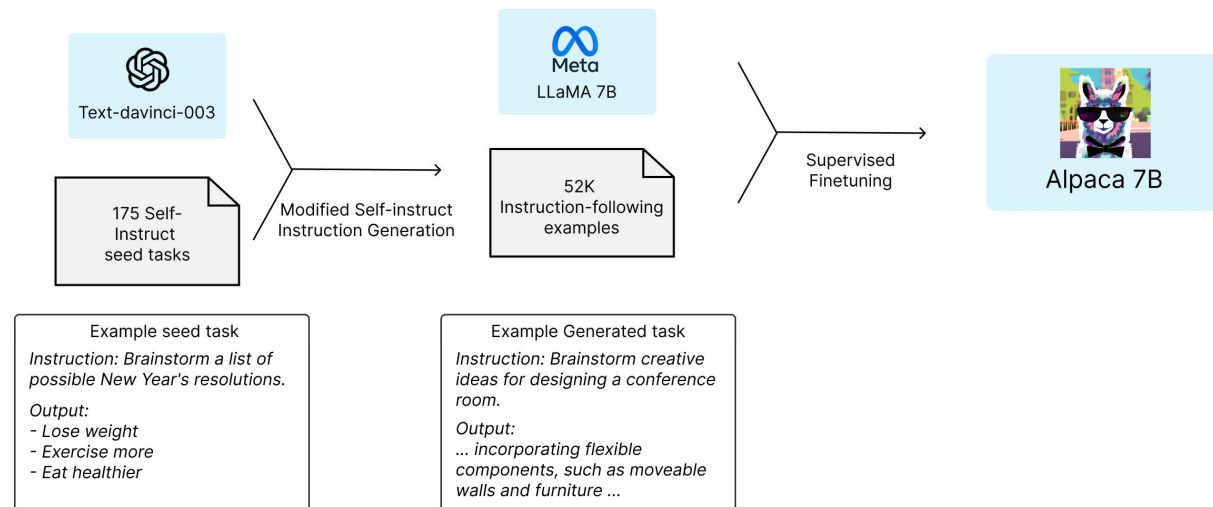
Distillation - Alpaca



We can use large models to create data to train the smaller models

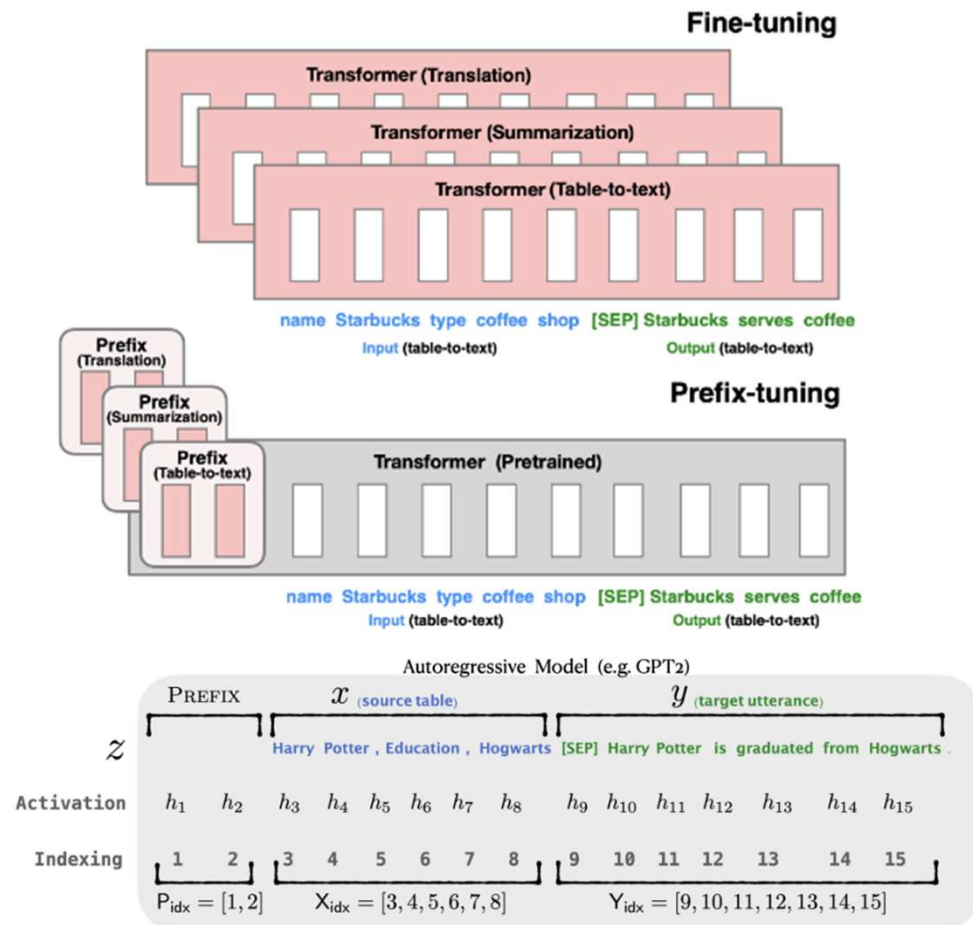
- 7B model trained from GPT3.5 inputs
- Cheap to train
 - 3 hours on 8 80GB A100s
 - <600\$ cloud cost
- Start from LLaMA (7B LLM)
- Finetune with
 - 175 human-written instruction-output pairs
 - 52K additional pairs generated by GPT 3.5 using the human-written samples as context

Stanford Alpaca



Learned Prompts – Prefix Tuning

- Keep the backbone frozen
- Add learnable token at the beginning of the input
 - embeddings that are learned via backpropagation
 - They do not need to be a real word, they are free parameters
- Automates the prompt engineering



Prompt Tuning – Learning To Prompt (L2P)



- For classification problems, the prompts are fixed embedding vectors that we compare to the image embedding
- Can we learn the prompt?
- **Learning2Prompt:**
 - Continual learning method
 - Fixed pretrained backbone
 - Keep a prompt pool with $\langle \text{key}, \text{value} \rangle$ pairs
 - Uses the key to select the prompt

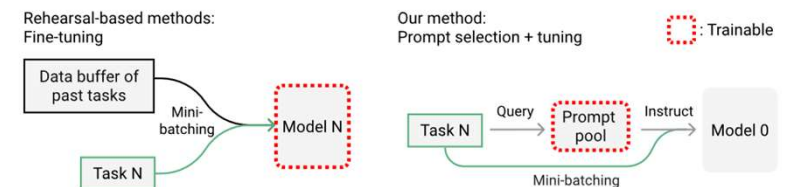


Figure 1. Overview of the L2P framework. Compared with typical methods that adapt entire or partial model weights to tasks sequentially with a rehearsal buffer to avoid forgetting, L2P uses a single backbone model and learns a prompt pool to instruct the model conditionally. Task-specific knowledge is stored inside a prompt pool, thus a rehearsal buffer is no longer mandatory to mitigate forgetting. L2P automatically selects and updates prompts from the pool in an instance-wise fashion, thus task identity is not required at test time. Notably, our largest prompt space is smaller than the size of one 224×224 image.

L2P – Prompt Selection

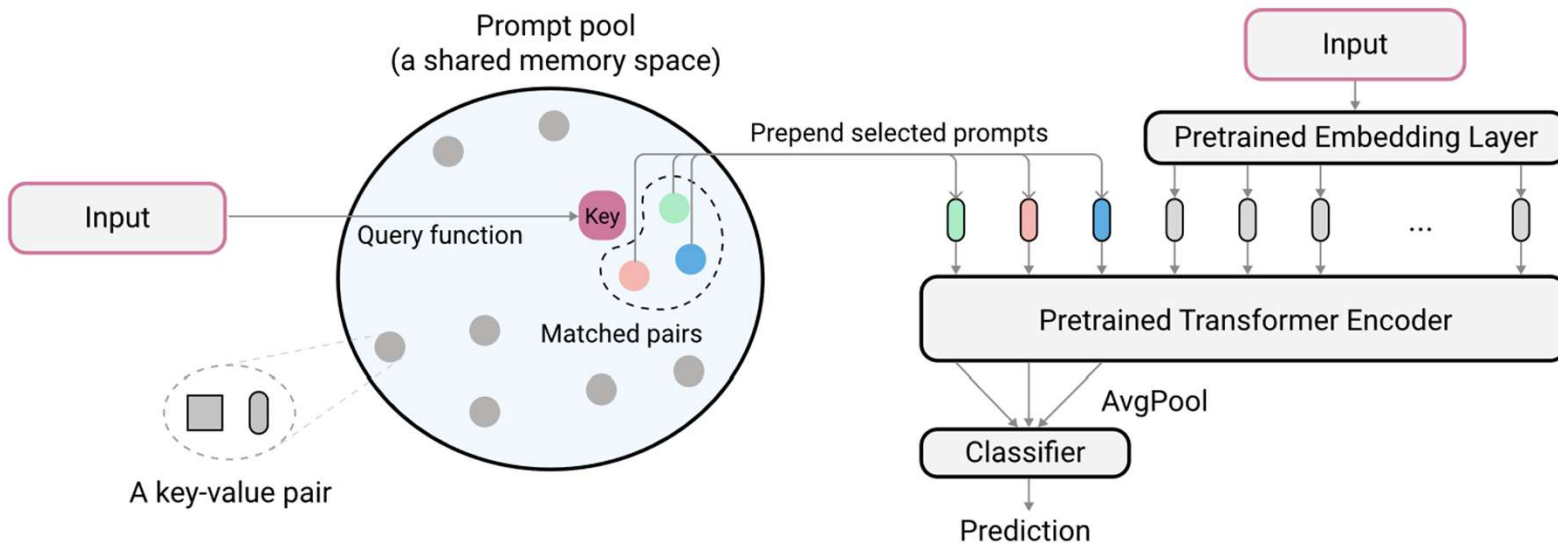


Figure 2. Illustration of L2P at test time. We follow the same procedure at training time: First, L2P selects a subset of prompts from a key-value paired *prompt pool* based on our proposed instance-wise query mechanism. Then, L2P prepends the selected prompts to the input tokens. Finally, L2P feeds the extended tokens to the model, and optimize the prompt pool through the loss defined in equation 5. The objective is learning to select and update prompts to instruct the prediction of the pre-trained backbone model.

L2P – Prompt Selection



- $\mathbf{K}_x = \operatorname{argmin}_{\{s_i\}_{i=1}^N \subseteq [1, M]} \sum_{i=1}^N \gamma(q(\mathbf{x}), \mathbf{k}_{s_i})$ **Prompt Selection**
 - $q(\mathbf{x})$ pretrained feature extractor gets query features
 - \mathbf{k}_{s_i} key
 - γ scoring function that matches image and prompts
- $\min_{\mathbf{P}, \mathbf{K}, \phi} \mathcal{L} \left(g_{\phi} \left(f_r^{\text{avg}}(\mathbf{x}_p) \right), y \right) + \lambda \sum_{\mathbf{K}_x} \gamma(q(\mathbf{x}), \mathbf{k}_{s_i})$ **Optimization Objective**
 - f_r^{avg} ViT encoder
 - g_{ϕ} classifier

Efficient Finetuning – Low Rank Adaptation (LoRA)



- We can finetune only a small part of the model
- Represent the finetuned models as $\Phi_0 + \Delta\Phi$ with small (in memory size) $\Delta\Phi$
 - This allows us to easily train and store hundreds of finetuned models
- Low-Rank Parameter Update
 - $h = W_0x + \Delta Wx = W_0x + BAx$
 - With $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$, $r \ll d$

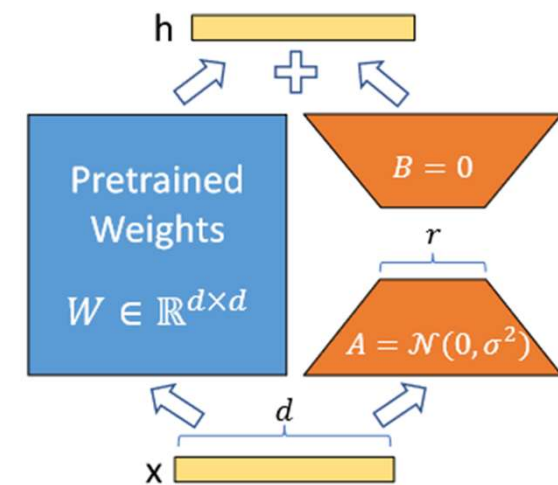
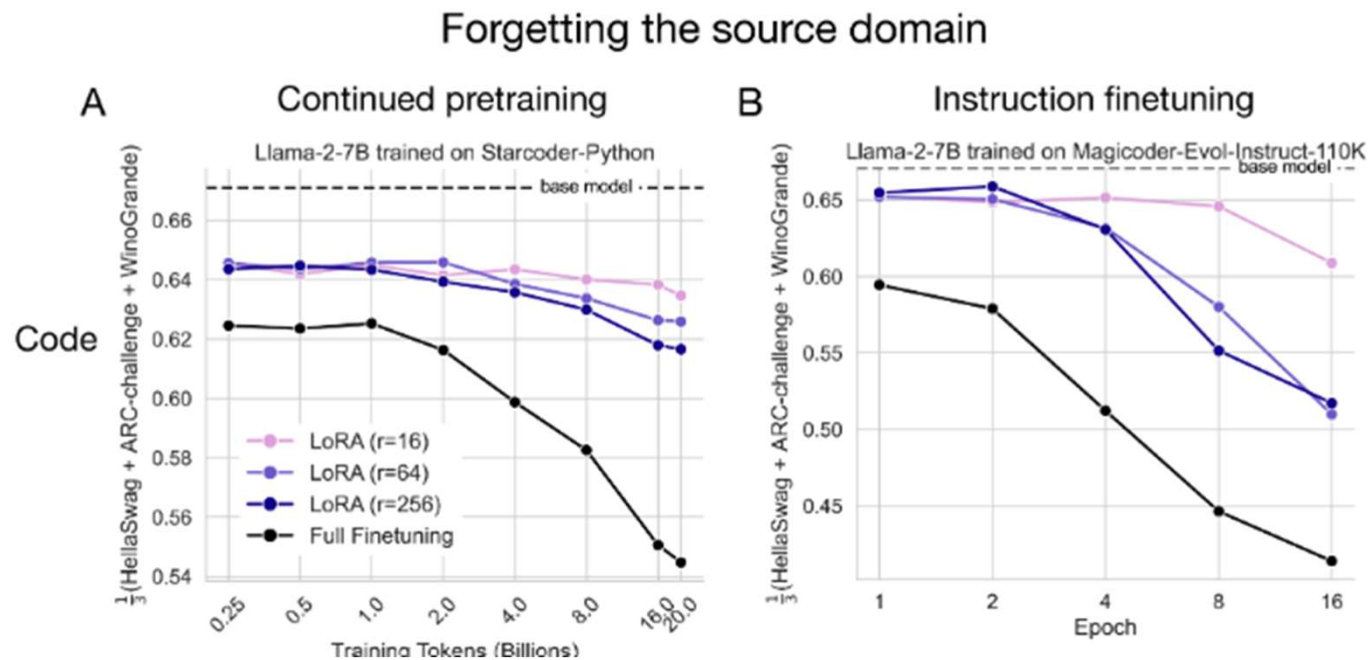


Figure 1: Our reparametrization. We only train A and B .

LoRA learns less and forgets less



- Lower ranks learn less but also have less forgetting

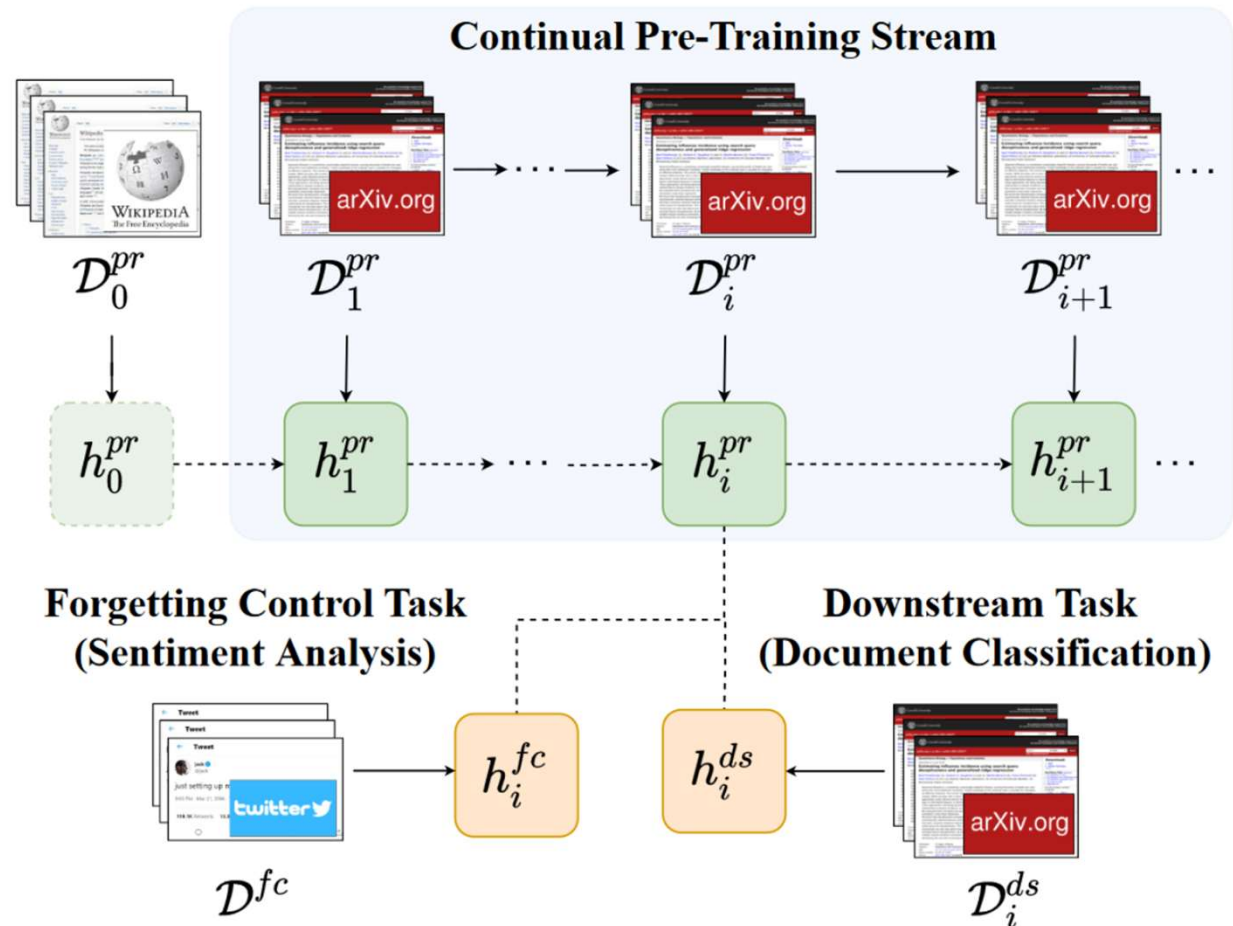


Biderman, Dan, et al. "Lora learns less and forgets less." *TMLR* (2024).

Continual Pretraining

Continual Pretraining

- **Continual Pretraining** is the problem of efficiently updating a large pretrained model
- **Forgetting Control Task**: we don't want to forget general knowledge
- **Downstream Task**: we want to improve on domain-specific tasks



Pretraining Results



Evaluation on the
Forgetting Control Task

Table 2: Accuracy on the entire dataset of **sentiment analysis** with RoBERTa model. Continual pre-training has been performed sequentially over each experience of scientific abstracts. Base refers to the model pre-trained on Wikipedia, while NT refers to the model with vocabulary expansion.

RoBERTa		Accuracy					1-epoch Accuracy				
Base		93.40					92.40				
Exp.	e1	e2	e3	e4	e5	e1	e2	e3	e4	e5	
Pretr	93.40	93.15	93.35	93.20	92.90	92.40	91.80	92.30	91.85	92.20	
Pretr. NT	93.75	93.70	93.75	93.60	94.10	91.75	91.15	92.00	92.30	92.45	

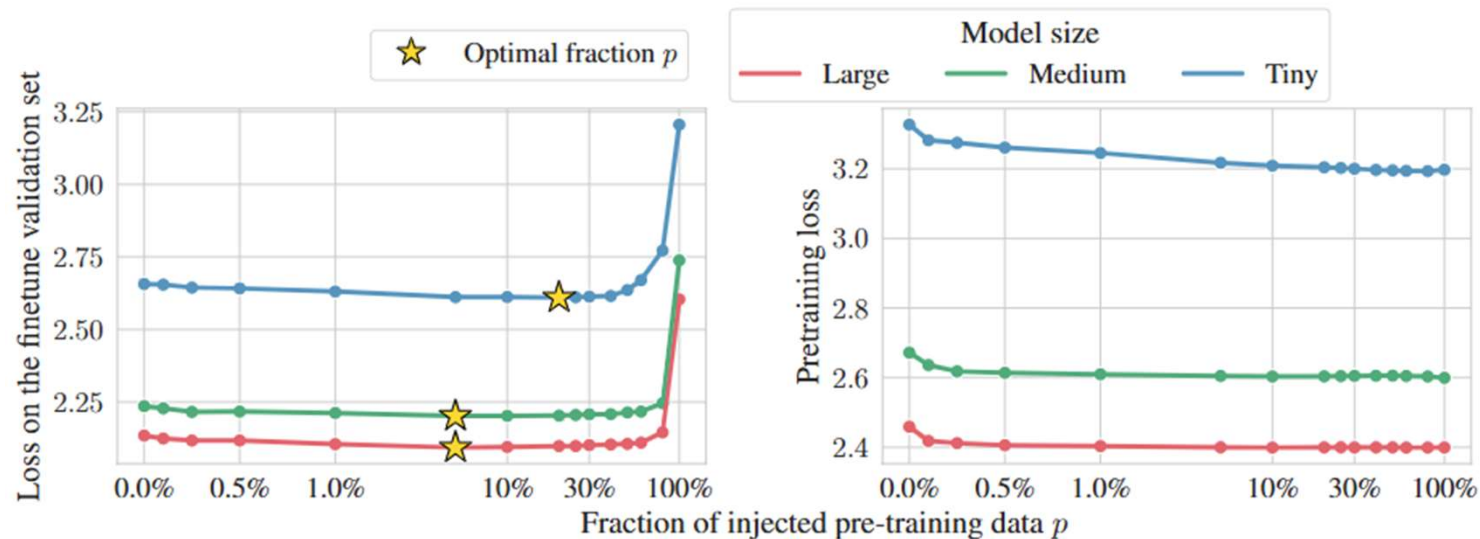
Forgetting is limited even with finetuning.
Dynamic vocabulary expansion (NT) slightly
improves the performance.

Self-supervised pretraining is more
robust than supervised methods
(result for vision in the paper)

Replay at scale



- “injecting as little as 1% of pretraining data in the finetuning data mixture prevents the model from forgetting the pretraining set”



Self-Supervised CL

- Distillation loss maps old representations in a new projected space
- SSL tricks such as heavy augmentations and SSL losses
- Linear probing evaluation

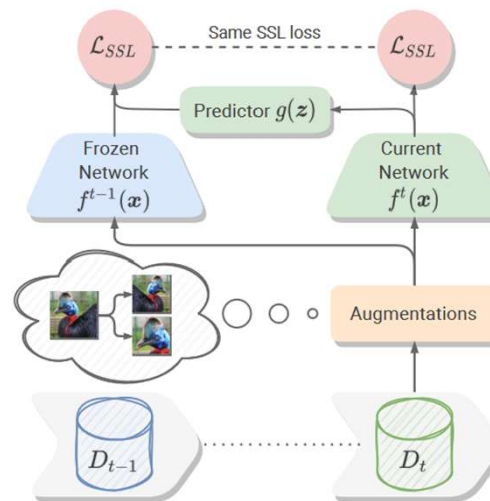


Figure 2. Overview of the CaSSLe framework.

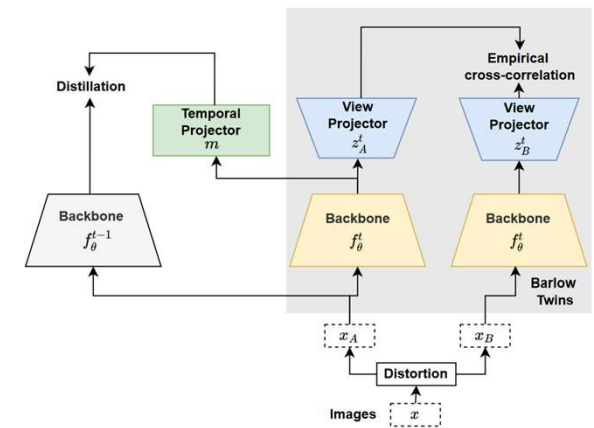


Figure 1. Self-supervised continual learning with Projected Functional Regularization. Instead of performing feature distillation directly between the previous task backbone and the new one, we use a *learned temporal projection* between the two feature spaces.

CaSSLe – Self-Supervised CL



- We can also use KD to train SSL models
- SSL loss on new data + KD loss on old embeddings

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_{SSL}(\mathbf{z}^A, \mathbf{z}^B) + \mathcal{L}_D(\mathbf{z}^A, \bar{\mathbf{z}}^A) \\ &= \mathcal{L}_{SSL}(\mathbf{z}^A, \mathbf{z}^B) + \mathcal{L}_{SSL}(g(\mathbf{z}^A), \bar{\mathbf{z}}^A).\end{aligned}$$

- A and B images, $\mathbf{z}^A, \mathbf{z}^B$ embeddings of new model
- $\bar{\mathbf{z}}^A$ embedding of old model
- \mathcal{L}_{SSL} is used as the distillation loss
- g is a projection network that maps from the new representation to the old ones
 - Allows to update the representations and change them

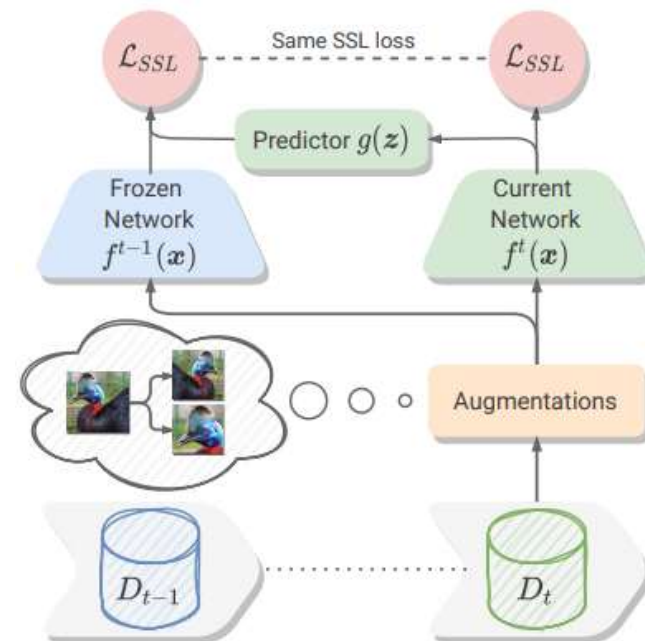


Figure 2. Overview of the CaSSLe framework.

Large-Scale Pretraining



- self-supervised pretraining provides very general knowledge about large domains (vision, language)
 - We can adapt them to solve new tasks with few (or zero) examples
 - They are more robust compared to supervised models
 - They are even more robust than finetuned models

Are Transformers Meta-Learning Methods?

In-Context Learning



- What is in-context learning (ICL) and why does it work so well?
- Is ICL even learning?
- Is it just an emergent property of large pretraining/data/architecture?
- **IDEA:** If ICL is a learning algorithm, then pretraining is a «meta-learning» algorithm

Motivation – Learning to Learn



- **Meta-learning:** Can we optimize the learning algorithm to solve novel tasks (transfer, low-shot and fast adaptation, hyperparameter search)? *a.k.a. Learning to learn*
 - $\mathcal{A}^{META}: \{\mathcal{D}_1, \dots, \mathcal{D}_N\} \rightarrow \mathcal{A}^*, \mathcal{A}^*: \mathcal{D}_{N+1}^{\text{train}} \rightarrow \theta^{N+1}$
- **Example: Optimization-based meta learning (MAML)**
 - $\theta^* = \min_{\theta} \mathbb{E}_{p(\mathcal{T})} [\mathcal{L}_{\mathcal{T}}(U_k(\theta, D_{\tau}^{\text{Supp}}), D_{\tau}^{\text{query}})]$
 - Learn the best init

Hypothesis: ICL is a form of SGD

- training Transformers on auto-regressive objectives is closely related to gradient-based metalearning formulations
- trained Transformers become mesa-optimizers i.e. learn models by gradient descent in their forward pass.
- The paper provides a constructive proof of this behavior for regression problems
 - Note that this means that maybe ICL is not a consequence of large-scale training, but of the Transformer architecture

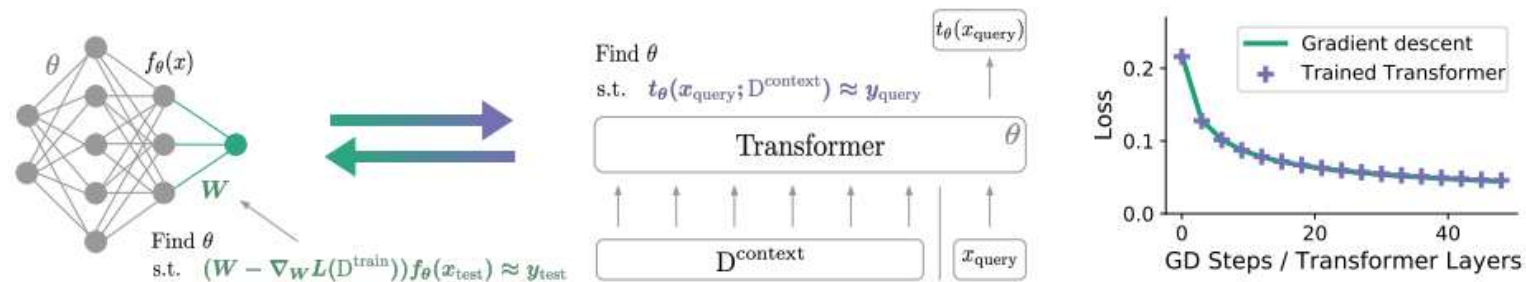


Figure 1. Illustration of our hypothesis: gradient-based optimization and attention-based in-context learning are equivalent. *Left:* Learning a neural network output layer by gradient descent on a dataset D^{train} . The task-shared meta-parameters θ are obtained by meta-learning with the goal that after adjusting the neural network output layer, the model generalizes well on unseen data. *Center:* Illustration of a Transformer that adjusts its query prediction on the data given in-context i.e. $t_\theta(x_{\text{query}}; D^{\text{context}})$. The weights of the Transformer are optimized to predict the next token y_{query} . *Right:* Our results confirm the hypothesis that learning with K steps of gradient descent on a dataset D^{train} (green part of the left plot) matches trained Transformers with K linear self-attention layers (central plot) when given D^{train} as in-context data D^{context} .

Hypothesis: ICL is a form of SGD

- “We reverse-engineer Transformers trained on simple sequence modeling tasks, and find strong evidence that their forward pass implements two-step algorithms: (i) early self-attention layers construct internal training datasets by grouping and copying tokens, and therefore implicitly define internal objective functions, (ii) deeper layers optimize these objectives to generate predictions.”

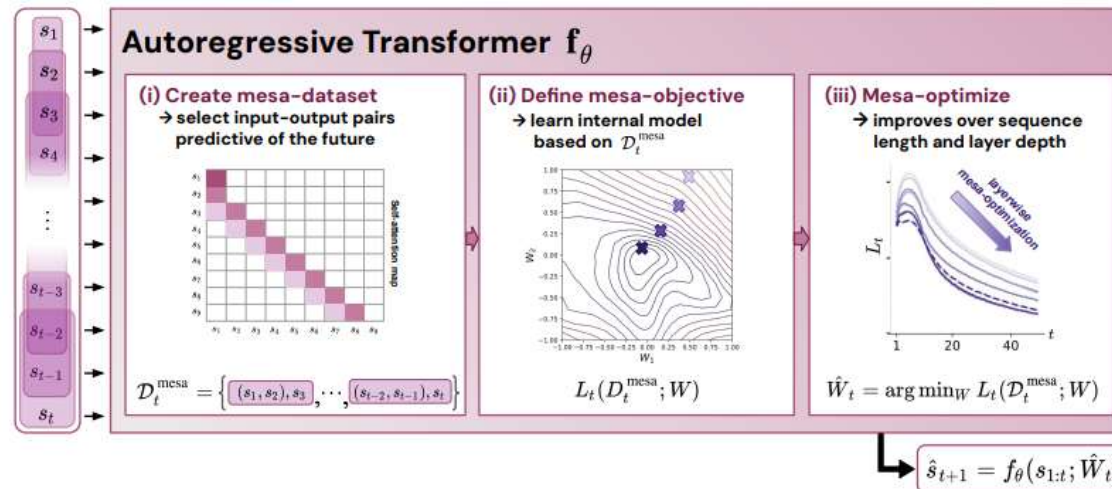


Figure 1: **Illustration of our hypothesis:** Optimizing the weights θ of an autoregressive Transformer f_θ gives rise to mesa-optimization algorithms implemented in the forward pass of the model. As a sequence of inputs s_1, \dots, s_t is processed up to timestep t , the Transformer (i) creates an internal training set consisting of pairs of input-target associations, (ii) defines an internal objective function through the resulting dataset, used to measure the performance of an internal model with weights W , (iii) optimizes this objective and uses the learned model to generate a prediction \hat{s}_{t+1} of the future.

Learnable Modular Memories


Tradeoffs of different memory types



Memory Type	Per-item Storage	Capacity	Modality	Update Speed	Interference Risk	Selective Forgetting	Retrieval	Generalization
Slot-based Memory ↔ Raw Data (summaries, exemplars, coresets)	Low	Unbounded	Uni-modal	Fast	Low	Easy	Grows with #items	Limited
↔ Embeddings (KV cache, activations, learned)	Medium	Unbounded	Uni/ Multi-modal	Fast	Low	Easy	Grows with #items	Implicit (learned encodings)
Neural Memory (e.g., DNNs, Hopfield, SDM)	High	Bounded	Uni/ Multi-modal	Slow/ Fast	High	Difficult	Depends on network size	Explicit (shared params)

Table 1. Desiderata and assessment of current memory representations (as defined in Sec. 4.1). Storage is more compact for neural memory while Capacity is unbounded for slot-based, raw data is typically unimodal, update speed is varied based on specific design choices, interference is high in neural memory, selective forgetting is easier for slot based. Finally generalization is poor in raw-data, implicitly encoded in embeddings, while explicitly optimized in neural memory.

Types of Memory







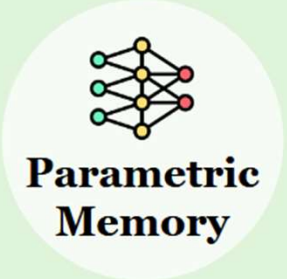
Token-level Memory

Features:

- Symbolic, addressable, transparent
- Swift adding/deleting/updating

Suitable Applications:

-  Multi-turn chatbot
-  Personalized agents
-  Recommender system
-  High-stake domains (law, finance, medical)






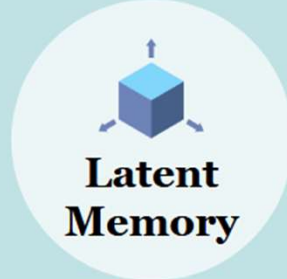
Parametric Memory

Features:

- Implicit, abstract, and generalizable
- Slower memory update
- (Typically) better performance gain
- More severe catastrophic forgetting

Suitable Applications:

-  Role-playing
-  Reasoning-intensive task
-  Tasks that require fundamentally new capabilities






Latent Memory

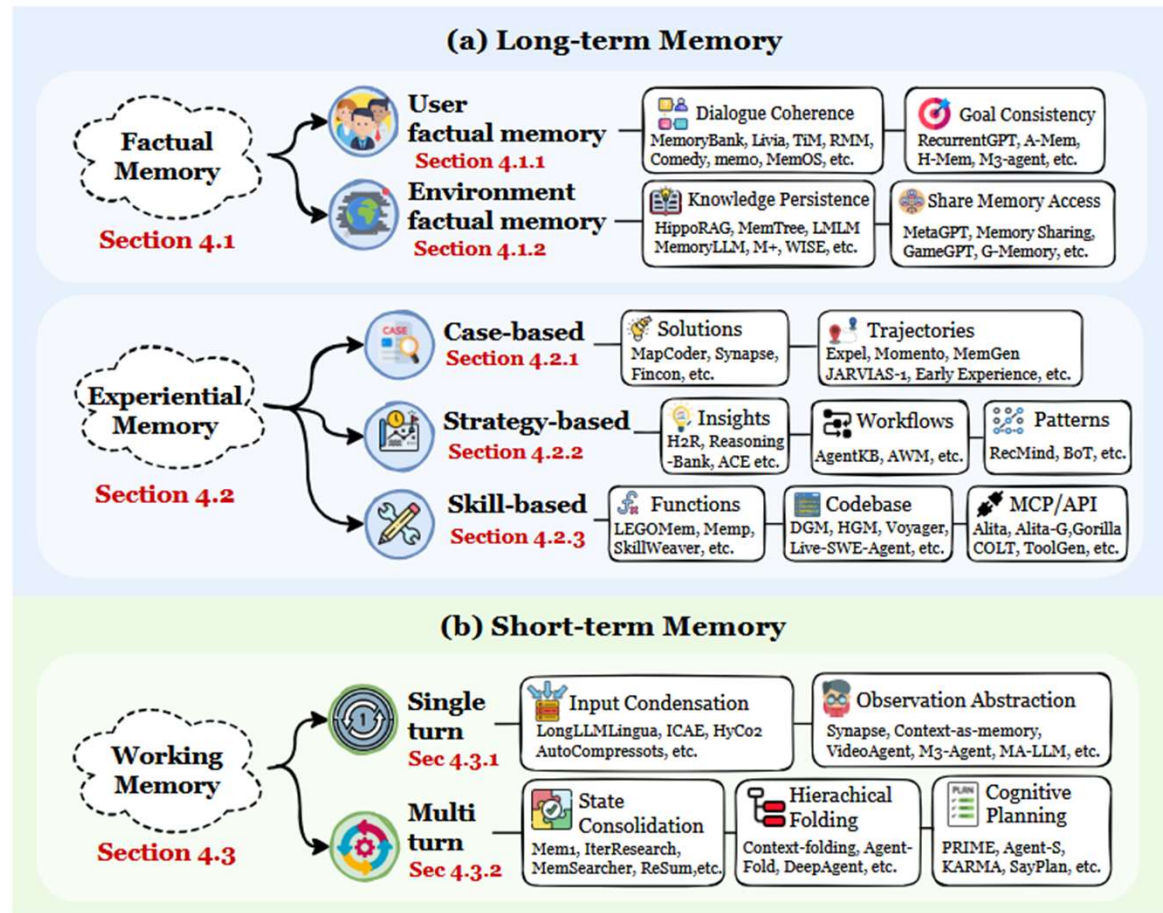
Features:

- Implicit, human-unreadable
- Trade-off between efficiency/flexibility
- Convenient modality fusion
- Machine-native and token-efficient

Suitable Applications:

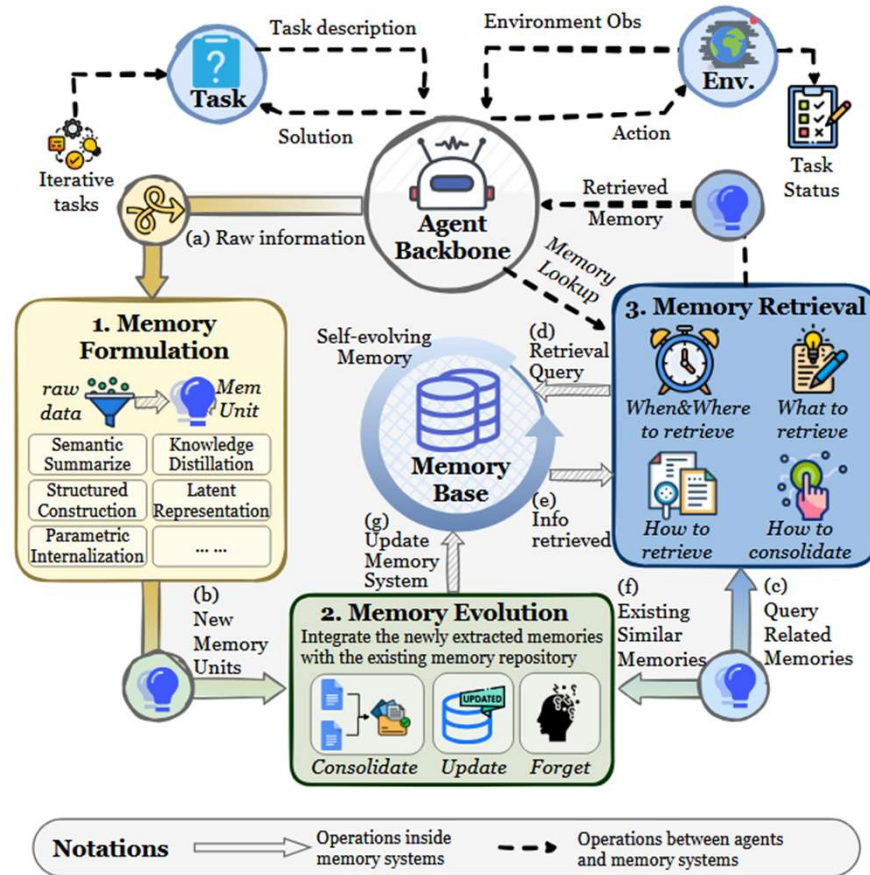
-  Multimodal memory
-  On-device or edge delopy
-  Low-resource or small-data setting

Types of Knowledge



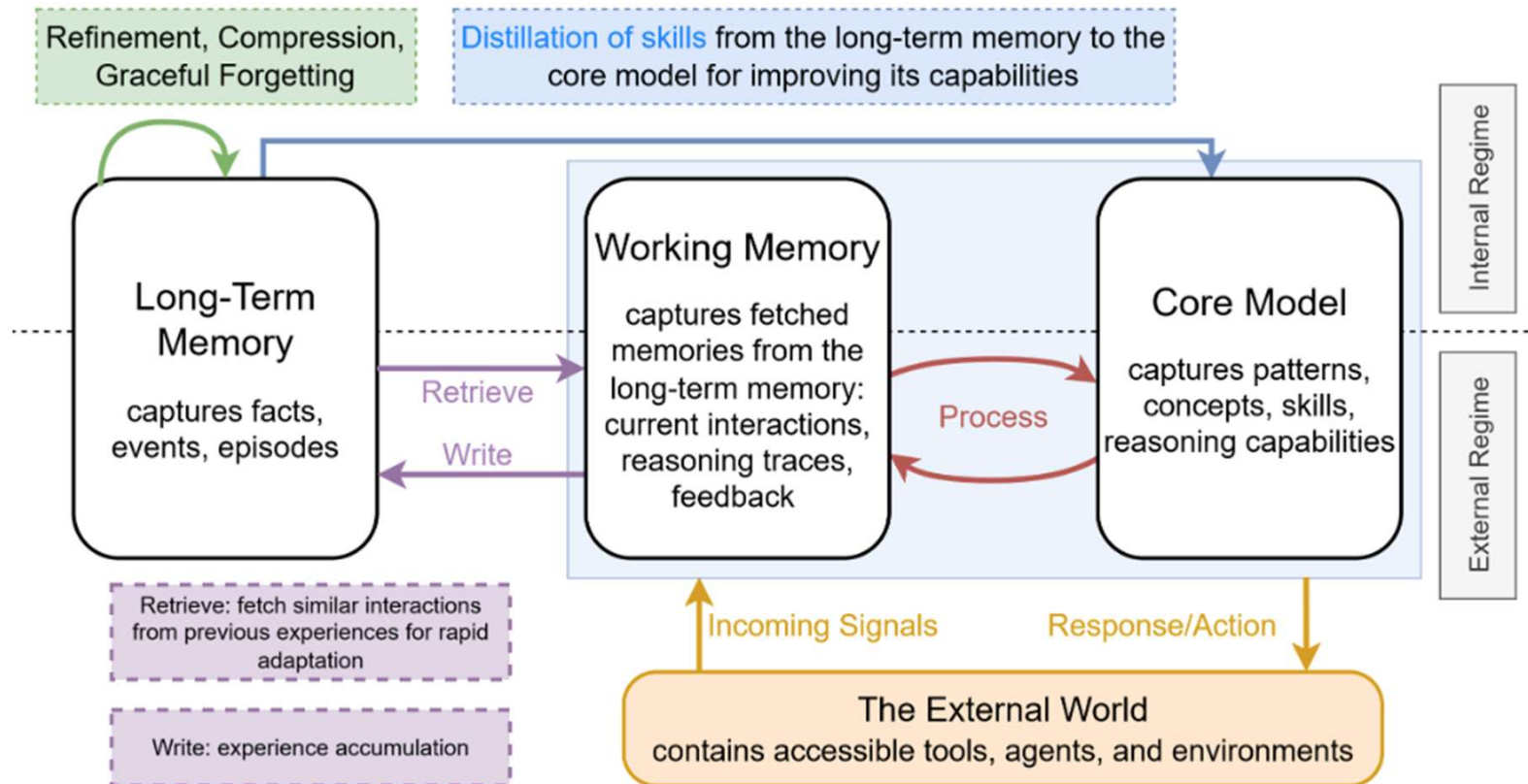
Hu, Yuyang, et al. "Memory in the age of ai agents." *arXiv preprint arXiv:2512.13564* (2025).

Memory Dynamics



Hu, Yuyang, et al. "Memory in the age of ai agents." *arXiv preprint arXiv:2512.13564* (2025).

The future of CL (?) – Modular Incremental Memory



Conclusion

Take-Home Messages



- scale matters
- Be careful about the evaluation
 - It's easy to evaluate on the training set
 - companies are not sharing many details about the models (evaluation, training data, number of parameters)
- methods that we saw in the course are used (in slightly different forms)
 - distillation to reduce the size of the model
 - zero-shot and few-shot techniques (e.g. prompting)
 - finetuning (e.g. LoRA)
 - Shift from in-weight-learning to in-context-learning

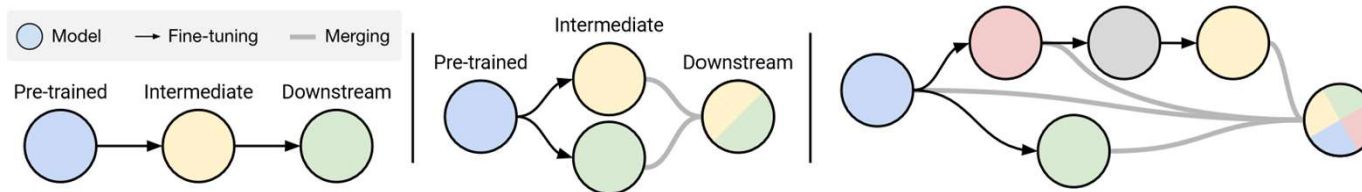
Model Patching

Building Models like OSS



Community-developed and continually-improved models

- Incremental and cheaply-communicable updates
- Merging models
- Vetting community contributions
- Versioning and backward compatibility
- Modularity and distribution



<https://colinraffel.com/blog/a-call-to-build-models-like-we-build-open-source-software.html>

Model Patching



GOAL: we want to consolidate two pretrained models

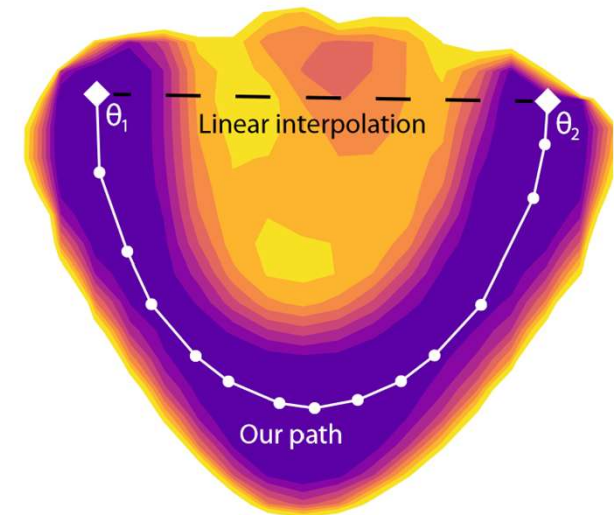
- Given two models f_{θ_1} and f_{θ_2} trained on tasks T1 and T2
- Find a model f_{θ_M} that solves both T1 and T2
- The two models can have different architectures, initialization, training algorithm, tasks...

(nonlinear) Mode Connectivity



Mode Connectivity: the models (modes) are connected by a low-loss path.

- we **construct continuous paths between minima of recent neural network architectures on CIFAR10 and CIFAR100.**
- “We observe the following trend: The **deeper and wider** an architecture, **the lower are the saddles between minima until they essentially vanish** for current-day deep architectures.
- The **more complex dataset CIFAR100 raises the barriers.**”
- **AutoNEB:** method for connecting minima from molecular statistical mechanics



Simple Model Patching



Linear Mode Connectivity:

- Naive average: $\theta^M = \frac{1}{2}(\theta^1 + \theta^2)$
- Weighted average: $\theta_i^M = \alpha_i \theta_i^1 + (1 - \alpha_i) \theta_i^2$

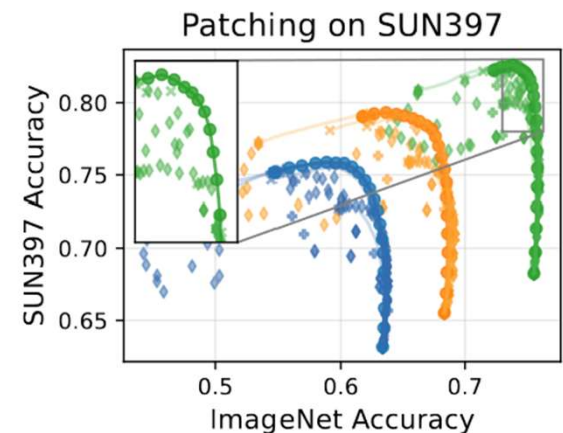
How do we find the weights?

- (normalized) Fisher diagonal
- (normalized) gradients
- Optimizing on a validation set

Patching Vision-Language Models



- Open-vocabulary models are characterized by their ability to perform any image classification task (example: CLIP) based on text descriptions of the classes
- Example: measure the similarity between an image and the embedding of the text *“this is an image of a dog”*
- PAINT also allows a single model to be **patched on multiple tasks** and **improves with model scale**.



Step 1. Fine-tune θ_{zs} on training data from $\mathcal{D}_{\text{patch}}$ to produce a model with weights θ_{ft} .

Step 2. For mixing coefficient $\alpha \in [0, 1]$, linearly interpolate between θ_{zs} and θ_{ft} to produce $\theta_{\text{patch}} = (1 - \alpha) \cdot \theta_{zs} + \alpha \cdot \theta_{\text{ft}}$. The mixing coefficient is determined via held-out validation sets for $\mathcal{D}_{\text{supp}}$ and $\mathcal{D}_{\text{patch}}$. We refer to the resulting model as θ_{patch} .

Git-Rebasin



- **IDEA:** we can exploit DNN symmetries to find a «better alignment» between the models that allows to patch them via weight averaging (remember: no barriers in loss)
- **DNN Units can be permuted** without changing the DNN output:

$$W'_\ell = P_\ell W_\ell^{(B)} P_{\ell-1}^\top, \quad b'_\ell = P_\ell b_\ell^{(B)}$$

- **Git-Rebasin:** find the best unit permutations by matching weights or activations. Then, we can merge with the naïve average.

What are we missing?



Necessary conditions for mode connectivity are unclear

- **initialization?**
 - Pretrained models are the best choice -> if the first part of the training is shared patching becomes much easier.
 - Same init is better. Different init works for simple datasets?
- **Width and depth?** Wider is better, but is it always enough? How wide should it be? How quickly is it growing with dataset complexity?
- **Optimizer?** Adaptive vs non-adaptive
- **Architecture?** Transformers, residual connections, batchnorm...