# CENG501 – Deep Learning

## Week 9

Fall 2024

Sinan Kalkan

Dept. of Computer Engineering, METU

Previously on CENG501

# In-context Learning (Prompt Engineering)

## Three ways of in-context learning:

**Zero-shot**

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

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

**One-shot**

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

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

## In a single sequence input, the prompted example can learn from previous demonstrations.

**Few-shot**

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

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

Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf

Previously on CENG501

# In-context Learning: How/Why Does it Work?

A Bayesian interpretation:

$$p(\text{output}|\text{prompt}) = \int_{\text{concept}} p(\text{output}|\text{concept}, \text{prompt}) p(\text{concept}|\text{prompt}) d(\text{concept})$$

- During pretraining, the network learns a latent concept space.
- With the prompt, we provide sufficient examples to estimate the most relevant concept – p(concept | prompt).

Xie et al., "An Explanation of In-context Learning as Implicit Bayesian Inference", ICLR 2022.

# In-context learning: Task vectors

Previously in CENG501

Hendel et al., "In-Context Learning Creates Task Vectors", 2023.

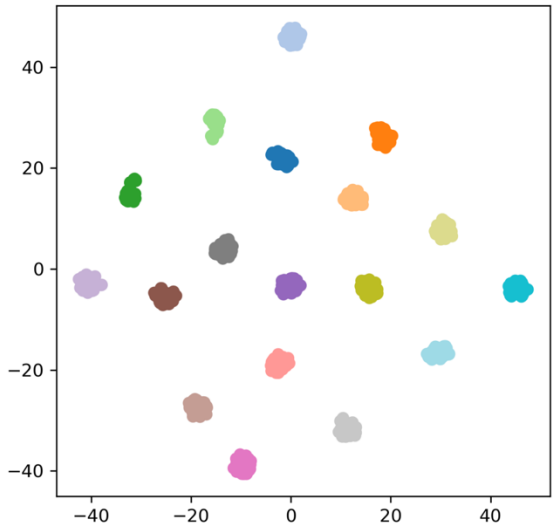



Figure 5: **A t-SNE plot of task vectors.** A 2D t-SNE plot visualizing 50 task vectors for each task, each generated from a different choice of $S$ and $x'$ using LLaMA 7B. Points are color-coded according to the task. Each task can be seen to form its own distinct cluster.

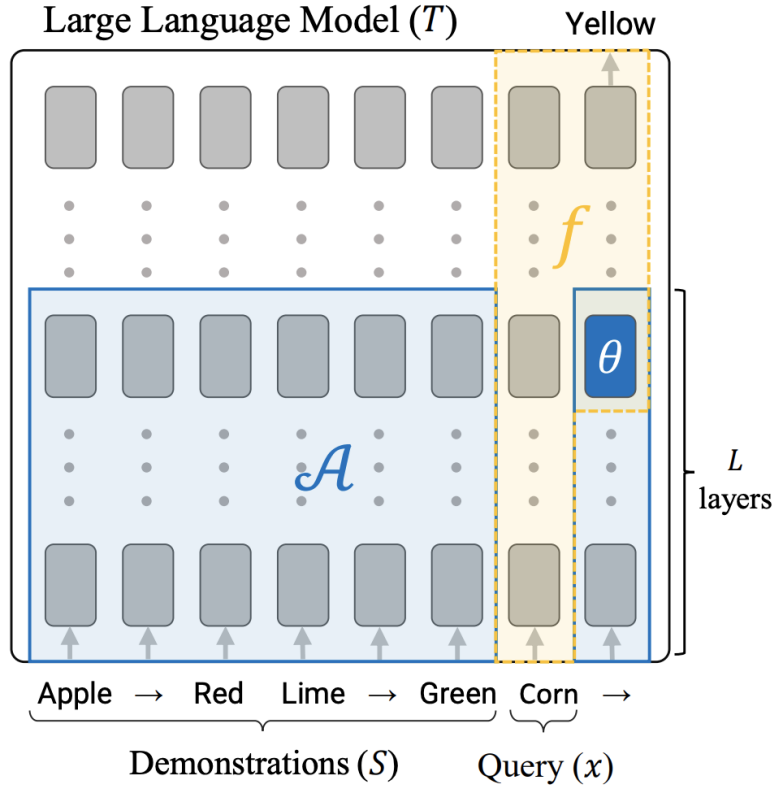



Figure 1: **ICL as learning in a Hypothesis Class.** In ICL, one provides an LLM with a prompt including demonstrations $S$ of some task, and a query $x$. The model generates the output for $x$ (here "Yellow"). We show that the underlying process can be broken down into two parts: $\mathcal{A}$, a "learning algorithm" (marked in blue), computes a query-agnostic vector $\boldsymbol{\theta}(S)$, which we view as a parameter of a function in a hypothesis class. The second part, denoted by $f$ and marked in yellow, is the application of the rule defined by $\boldsymbol{\theta}$ on the query $x$, without direct dependence on $S$.

Previously in CENG501

# In-context Learning: Important Factors

Min et al., “Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?”, 2022.

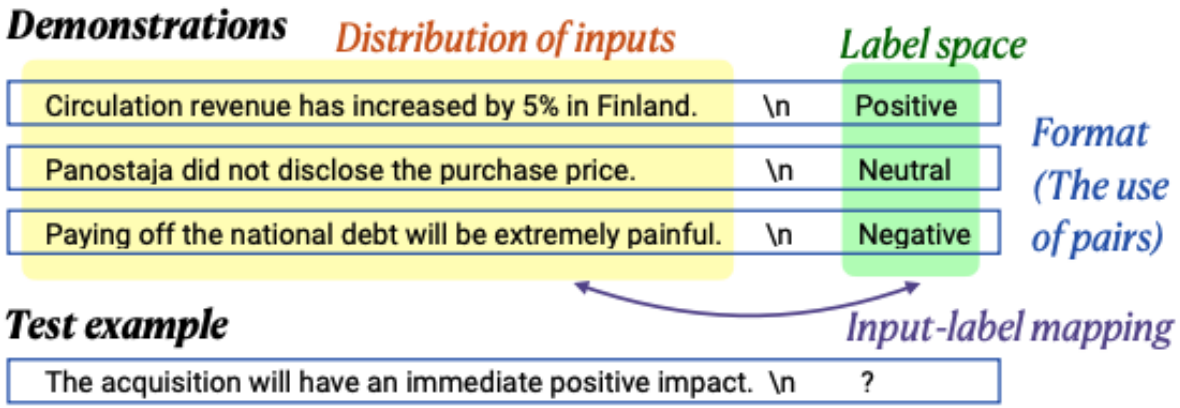



Figure 7: Four different aspects in the demonstrations: the input-label mapping, the distribution of the input text, the label space, and the use of input-label pairing as the format of the demonstrations.


## Abstract

Large language models (LMs) are able to in-context learn—perform a new task via inference alone by conditioning on a few input-label pairs (demonstrations) and making predictions for new inputs. However, there has been little understanding of *how* the model learns and *which* aspects of the demonstrations contribute to end task performance. In this paper, we show that ground truth demonstrations are in fact not required—randomly replacing labels in the demonstrations barely hurts performance on a range of classification and multi-choce tasks, consistently over 12 different models including GPT-3. Instead, we find that other aspects of the demonstrations are the key drivers of end task performance, including the fact that they provide a few examples of (1) the label space, (2) the distribution of the input text, and (3) the overall format of the sequence. Together, our analysis provides a new way of understanding how and why in-context learning works, while opening up new questions about how much can be learned from large language models through inference alone.

Previously on CENG501

# In-context Learning: Limitations

- **Model Parameters and Scale:** Scale helps!
- **Training Data Dependency:** Dataset size matters!
- **Domain Specificity:**
  - ”While LLMs can generalize across various tasks, there might be limitations when dealing with highly specialized domains. Domain-specific data might be required to achieve optimal results.”
- **Model Fine-Tuning:**
  - ”Even with ICL, there might be scenarios where model fine-tuning becomes necessary to cater to specific tasks or correct undesirable emergent abilities.”
- **Ethics and Fairness**
- **Privacy and Security**



https://www.lakera.ai/blog/what-is-in-context-learning

Previously on CSENG501

# Chain of Thought

Wei et al., "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models", NeurIPS 2022.

## Standard Prompting

**Model Input**

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: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

**Model Output**

A: The answer is 27. ✗

## Chain-of-Thought Prompting

**Model Input**

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: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

**Model Output**

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

Previously on CENG501

Kojima et al., “Large Language Models are Zero-Shot Reasoners”, NeurIPS 2022.

# Zero-shot Chain of Thought

(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.* ✓

# Meta Prompting

Previously on CENG501

Zhang et al., “Meta Prompting for AI Systems”, 2024.

Standard Prompting

Problem: Find the domain of the expression $\frac{\sqrt{x-2}}{\sqrt{5-x}}$.
Solution: The expressions inside each square root must be non-negative. Therefore, $x-2 \geq 0$, so $x \geq 2$, and $5-x \geq 0$, so $x \leq 5$. Also, the denominator cannot be equal to zero, so $5-x > 0$, which gives $x < 5$. Therefore, the domain of the expression is $\boxed{[2,5)}$. Final Answer: The final answer is $[2,5)$. I hope it is correct.

---

Problem: If $\det \mathbf{A} = 2$ and $\det \mathbf{B} = 12$, then find $\det(\mathbf{AB})$.
Solution: We have that $\det(\mathbf{AB}) = (\det \mathbf{A})(\det \mathbf{B}) = (2)(12) = \boxed{24}$. Final Answer: The final answer is 24. I hope it is correct.

---

...

Meta Prompting

**Problem Statement**:
- **Problem**: [question to be answered]

**Solution Structure**:
1. Begin the response with "Let's think step by step."
2. Follow with the reasoning steps, ensuring the solution process is broken down clearly and logically.
3. End the solution with the final answer encapsulated in a LaTeX-formatted box, $\boxed{\ldots}$, for clarity and emphasis.
4. Finally, state "The answer is [final answer to the problem].", with the final answer presented in LaTeX notation.

---

Figure 1: A structure meta prompt presented in markdown format for solving MATH [17] problems.

Previously on CENG501

# Chain of Symbol

Hu et al., “Chain-of-Symbol Prompting Elicits Planning in Large Langauge Models”, 2023.

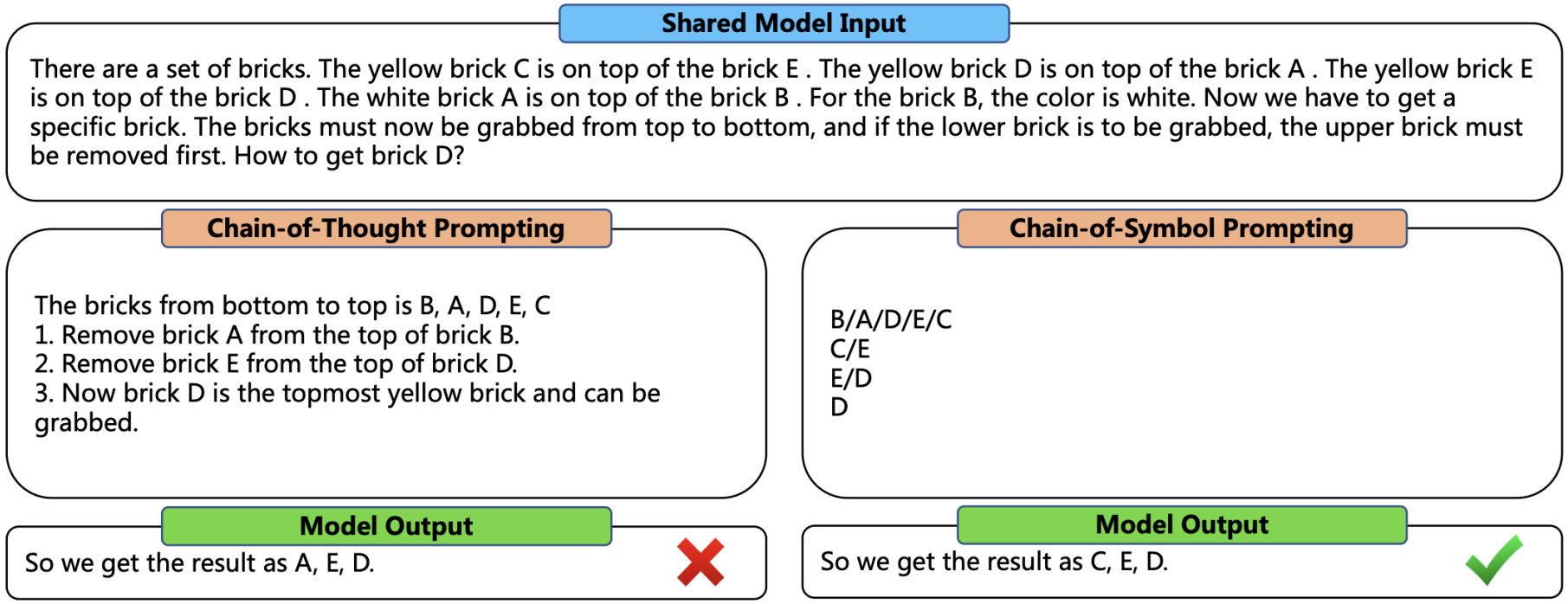



Figure 1: An example for comparison between Chain-of-Thought (CoT) and Chain-of-Symbol (CoS) that elicits large language models in tackling complex planning tasks with higher performance and fewer input tokens. We let the model generate CoT/CoS during inference in a few-shot manner. Results were taken in May 2023 with ChatGPT and can be subject to change.

Liu et al., “Generated Knowledge Prompting for Commonsense Reasoning”, 2022.

Previously on CENG501

# Generated knowledge prompting

It remains an open question whether incorporating external knowledge benefits commonsense reasoning while maintaining the flexibility of pretrained sequence models. To investigate this question, we develop generated knowledge prompting, which consists of generating knowledge from a language model, then providing the knowledge as additional input when answering a question. Our method does not require task-specific supervision for knowledge integration, or access to a structured knowledge base, yet it improves performance of large-scale, state-of-the-art models on four commonsense reasoning tasks, achieving state-of-the-art results on numerical commonsense (NumerSense), general commonsense (CommonsenseQA 2.0), and scientific commonsense (QASC) benchmarks. Generated knowledge prompting highlights large-scale language models as flexible sources of external knowledge for improving common-

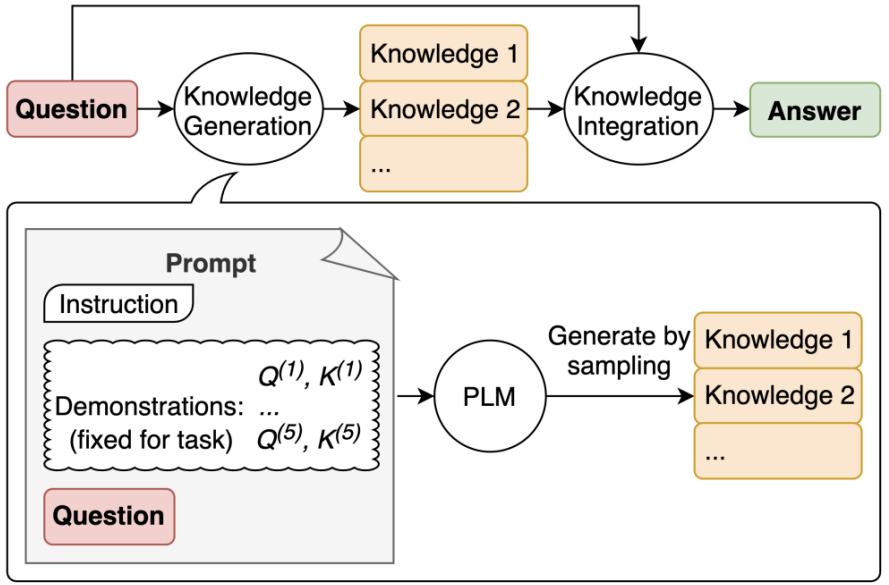



Figure 1: Generated knowledge prompting involves (i) using few-shot demonstrations to generate question-related knowledge statements from a language model; (ii) using a second language model to make predictions with each knowledge statement, then selecting the highest-confidence prediction.

Previously on CENG501

# Self-consistency

Wang et al., "Self-Consistency Improves Chain of Thought Reasoning in Language Models", 2023.

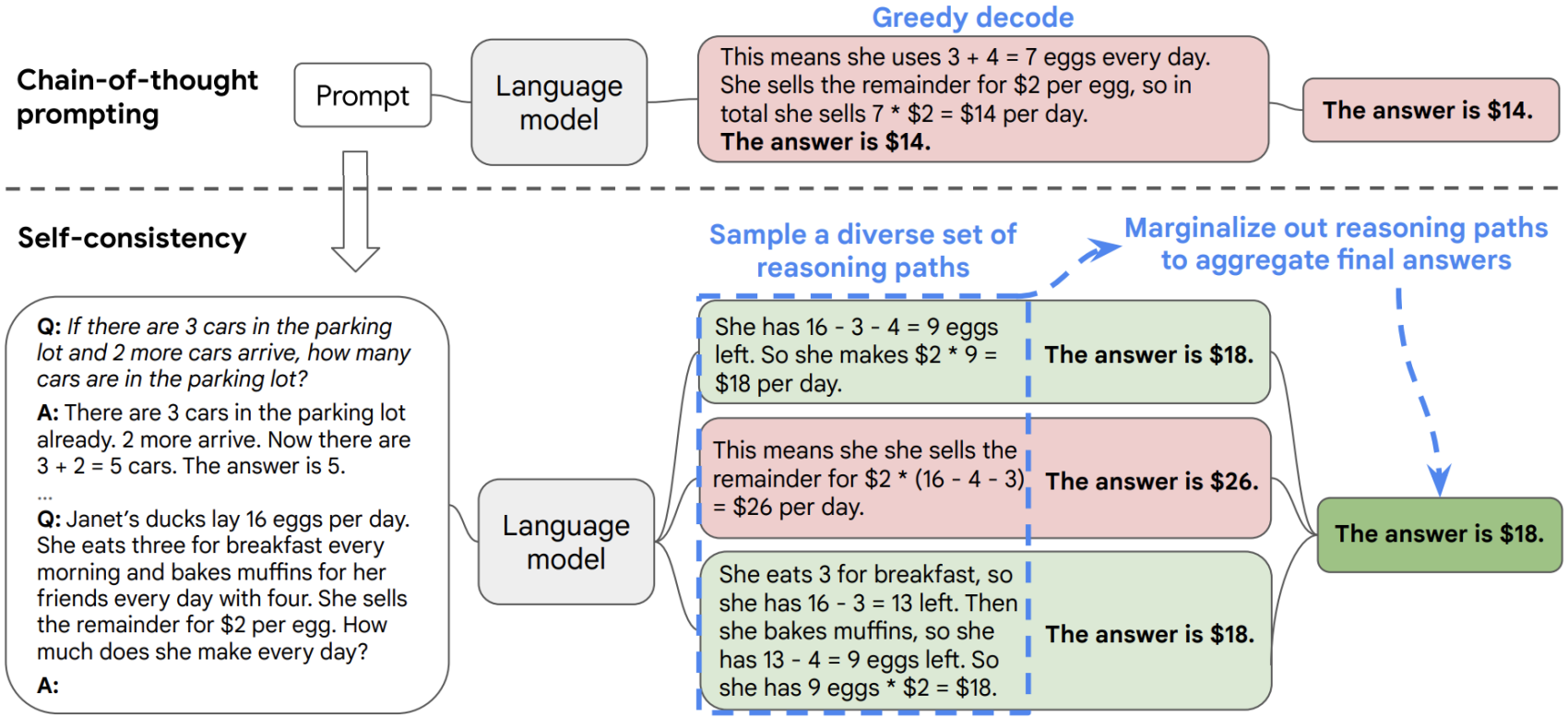
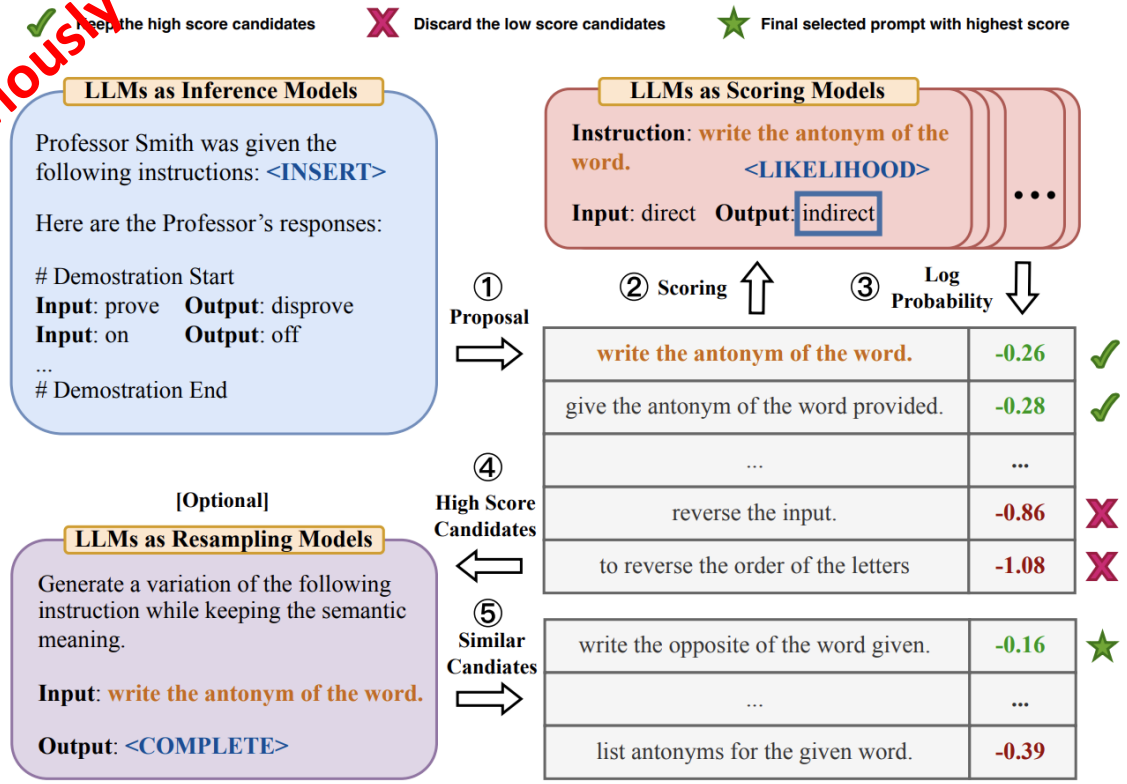



Figure 1: The self-consistency method contains three steps: (1) prompt a language model using chain-of-thought (CoT) prompting; (2) replace the "greedy decode" in CoT prompting by sampling from the language model's decoder to generate a diverse set of reasoning paths; and (3) marginalize out the reasoning paths and aggregate by choosing the most consistent answer in the final answer set.

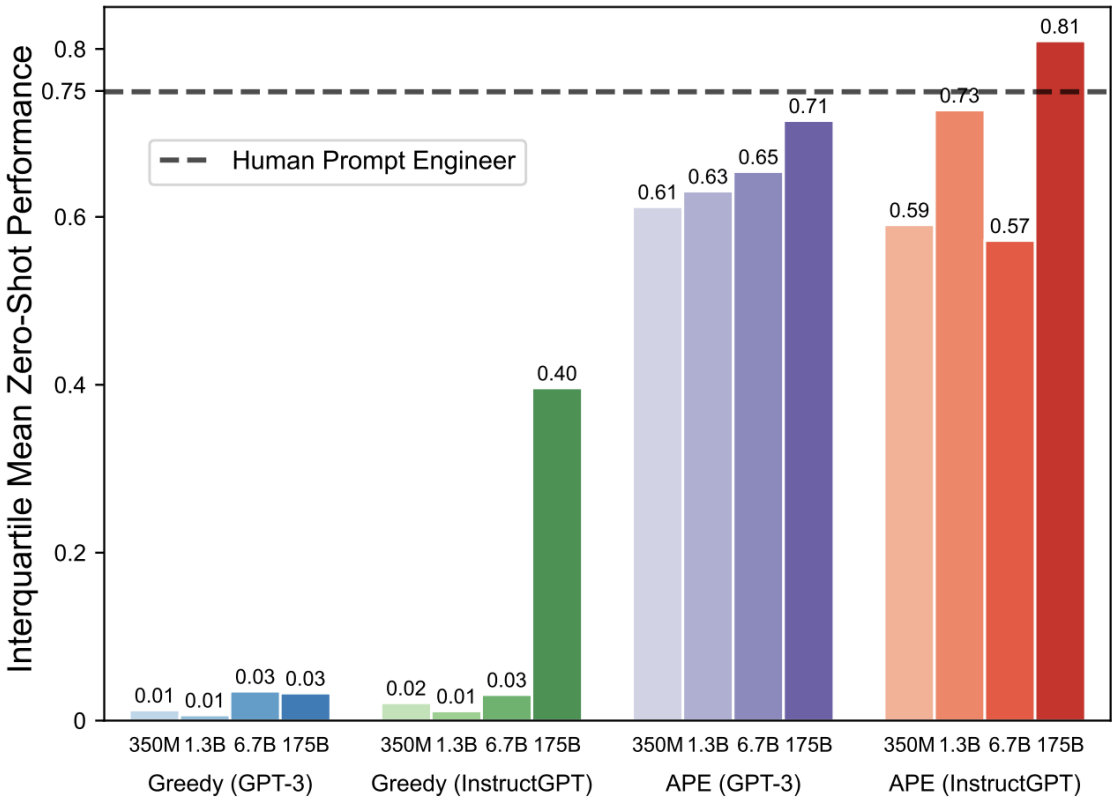
# Automatic Prompt Engineer

Previously on CENG501

Zhou et al., “LARGE LANGUAGE MODELS ARE HUMAN-LEVEL PROMPT ENGINEERS”, 2023.




(a) Automatic Prompt Engineer (APE) workflow




(b) Interquartile mean across 24 tasks

Previously on CENG501

# Tree of Thoughts Prompting

Yao et al., "Tree of Thoughts: Deliberate Problem Solving with Large Language Models", 2023.

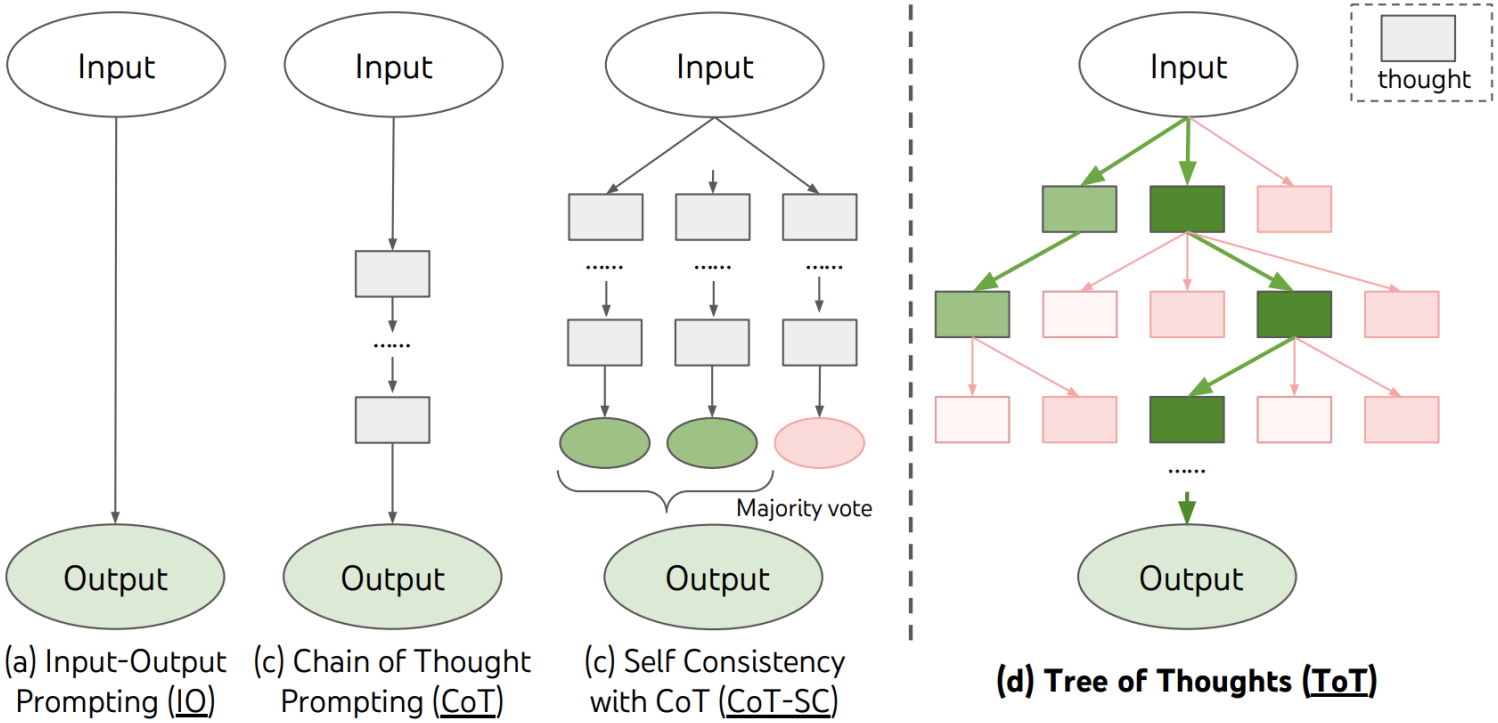



Figure 1: Schematic illustrating various approaches to problem solving with LLMs. Each rectangle box represents a *thought*, which is a coherent language sequence that serves as an intermediate step toward problem solving. See concrete examples of how thoughts are generated, evaluated, and searched in Figures 2,4,6.

Previously on CENG501

# Program-Aided LMs

Gao et al., “PAL: Program-aided Language Models”, 2023.

**Chain-of-Thought (Wei et al., 2022)**

Input

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 tennis balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

A: The bakers started with 200 loaves. They sold 93 in the morning and 39 in the afternoon. So they sold 93 + 39 = 132 loaves. The grocery store returned 6 loaves. So they had 200 - 132 - 6 = 62 loaves left.
The answer is 62.

**Program-aided Language models (this work)**

Input

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 tennis balls.
```
tennis_balls = 5
```
2 cans of 3 tennis balls each is
```
bought_balls = 2 * 3
```
tennis balls. The answer is
```
answer = tennis_balls + bought_balls
```

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

A: The bakers started with 200 loaves
```
loaves_baked = 200
```
They sold 93 in the morning and 39 in the afternoon
```
loaves_sold_morning = 93
loaves_sold_afternoon = 39
```
The grocery store returned 6 loaves.
```
loaves_returned = 6
```
The answer is
```
answer = loaves_baked - loaves_sold_morning
  - loaves_sold_afternoon + loaves_returned
```

```
>>> print(answer)
74
```

Previously on ENG501

# LLMs as Agents

Wang et al., "A Survey on Large Language Model based Autonomous Agents", 2024.

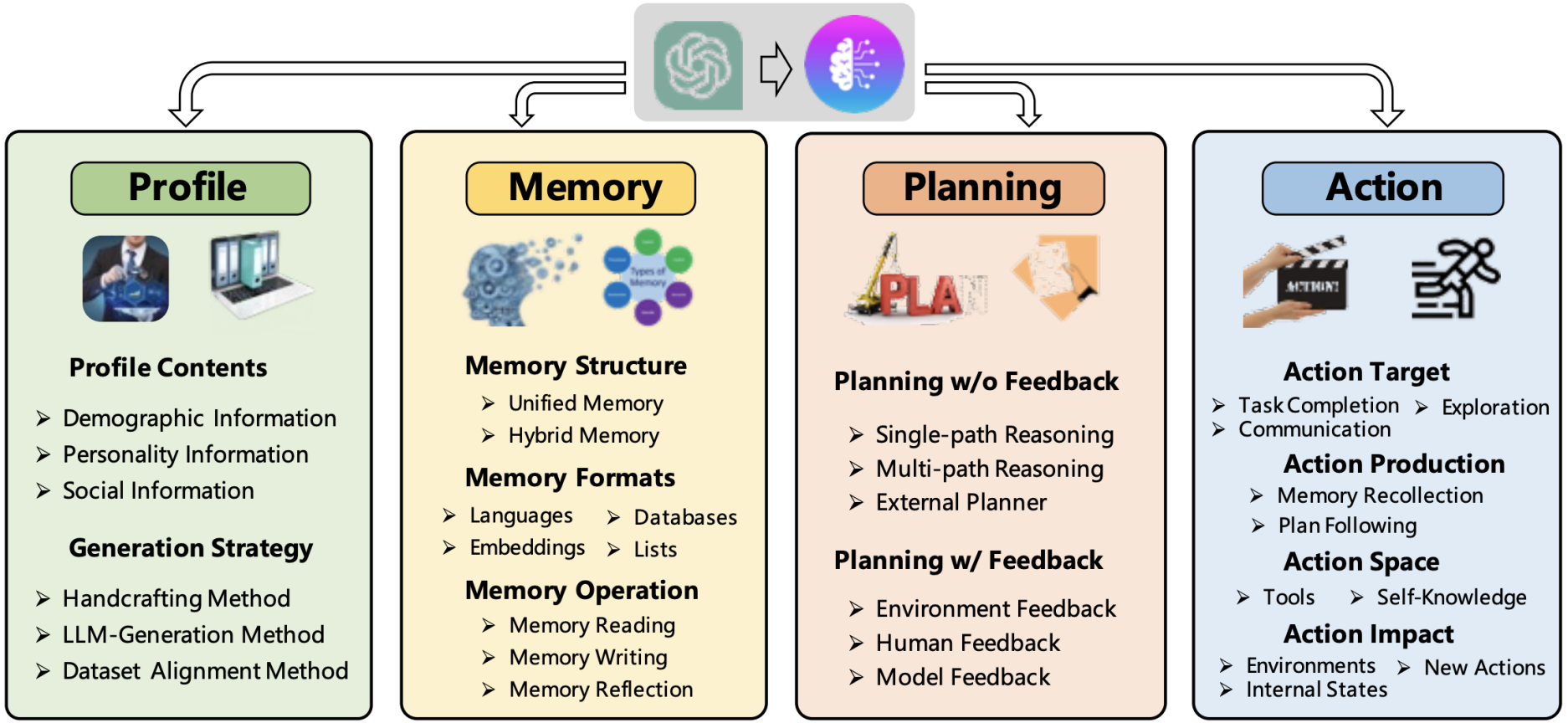



**Fig. 2** A unified framework for the architecture design of LLM-based autonomous agent.

Previously on CENG501

# LLMs as Agents

- Prompting/Planning with feedback

**(1d) ReAct (Reason + Act)**

**Thought 1:** I need to search Apple Remote and find the program it was originally designed to interact with.
**Act 1:** Search[Apple Remote]
**Obs 1:** The Apple Remote is a remote control introduced in October 2005 by Apple … originally designed to control the Front Row media center program …

**Thought 2:** Apple Remote was originally designed to control the Front Row media center program. I need to search Front Row next and find what other device can control it.
**Act 2:** Search[Front Row]
**Obs 2:** Could not find [Front Row]. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports',' Front Row (software) ', …]

**Thought 3:** Front Row is not found. I need to search Front Row (software) .
**Act 3:** Search[Front Row (software) ]
**Obs 3:** Front Row is a discontinued media center software …

**Thought 4:** Front Row (software) is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.
**Act 4:** Finish[keyboard function keys ]

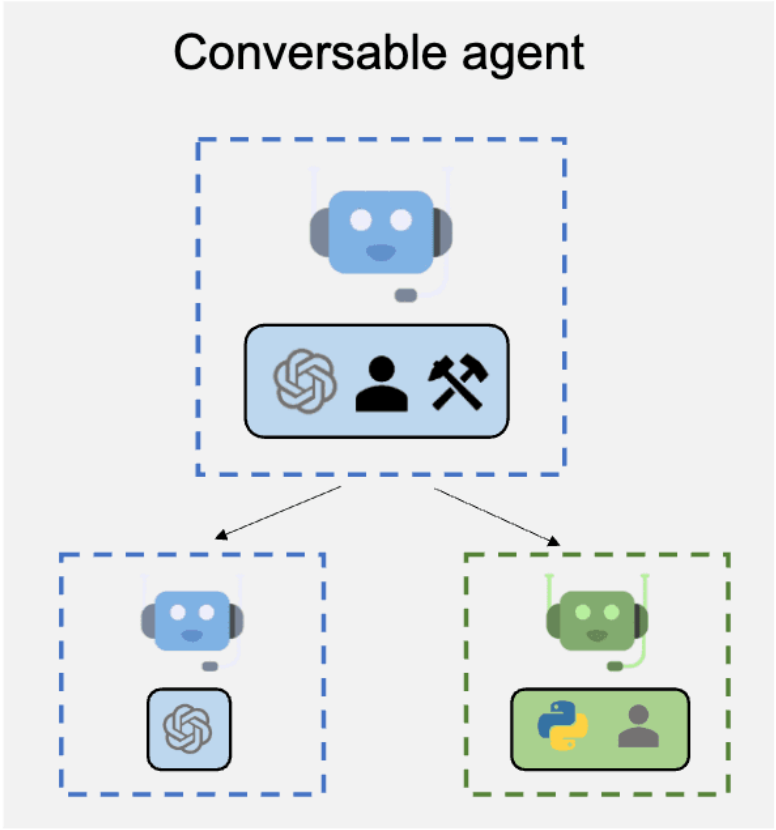
Yao et al., “ReAct: Synergizing Reasoning and Acting in Language Models”, 2023.

Previously on ENG501

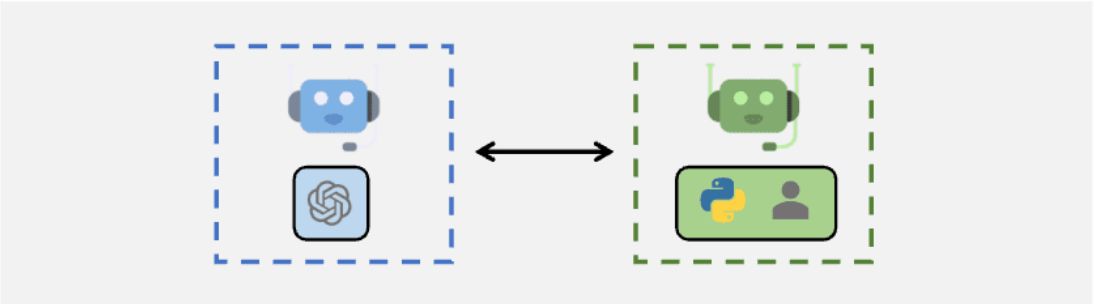
# LLMs as Agents

- Autogen, LangChain, AutoGPT, Langroid, OpenAgents, ….

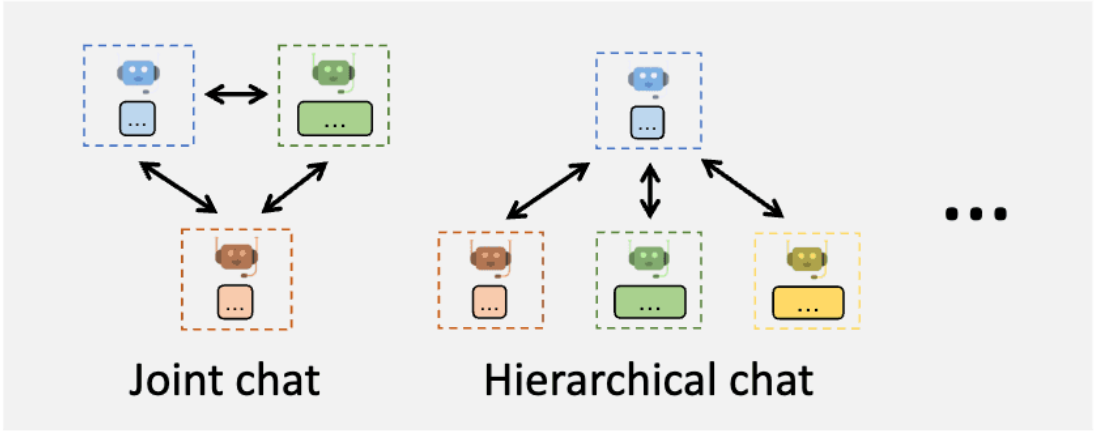
Autogen:




Agent Customization



Multi-Agent Conversations




Flexible Conversation Patterns

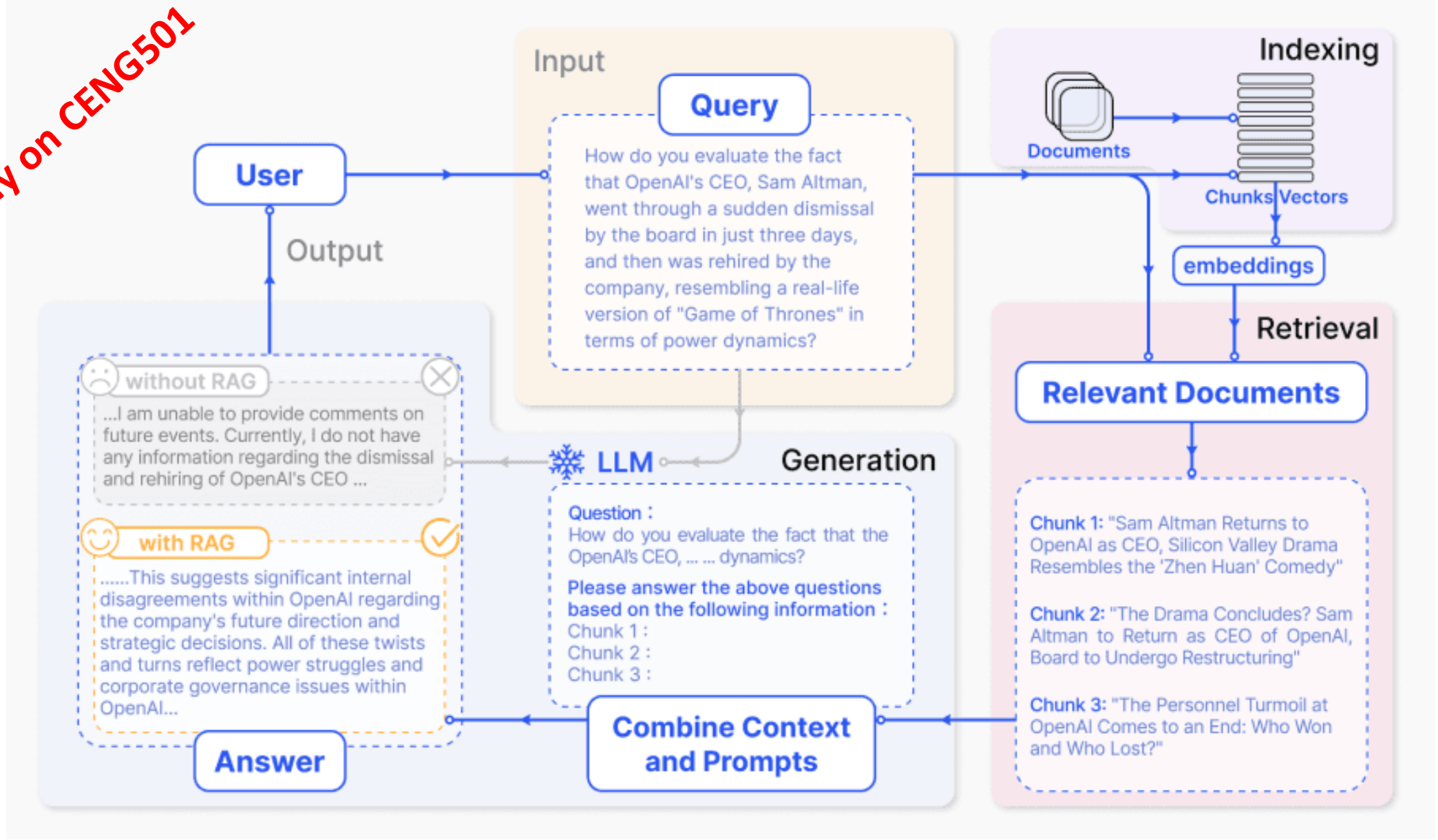
Previously on CENG501

# LLMs as Agents: Challenges

From https://www.promptingguide.ai/research/llm-agents:

- **Role-playing capability**: LLM-based agents typically need to adapt a role to effectively complete tasks in a domain. For roles that the LLM doesn't characterize well, it's possible to fine-tune the LLM on data that represent uncommon roles or psychology characters.
- **Long-term planning and finite context length**: planning over a lengthy history remains a challenge that could lead to errors that the agent may not recover from. LLMs are also limited in context length they can support which could lead to constraints that limit the capabilities of the agent such as leveraging short-term memory.
- **Generalized human alignment**: it's also challenging to align agents with diverse human values which is also common with standard LLMs. A potential solution involves the potential to realign the LLM by designing advanced prompting strategies.
- **Prompt robustness and reliability**: an LLM agent can involve several prompts designed to power the different modules like memory and planning. It's common to encounter reliability issues in LLMs with even the slightest changes to prompts. LLM agents involve an entire prompt framework which makes it more prone to robustness issues. The potential solutions include crafting prompt elements through trial and error, automatically optimizing/tuning prompts, or automatically generating prompts using GPT. Another common issue with LLMs is hallucination which is also prevalent with LLM agents. These agents rely on natural language to interface with external components that could be introducing conflicting information leading to hallucination and factuality issues.
- **Knowledge boundary**: similar to knowledge mismatch issues that could lead to hallucination or factuality issues, it's challenging to control the knowledge scope of LLMs which can significantly impact the effectiveness of simulations. Concretely, an LLM's internal knowledge could introduce biases or utilize user-unknown knowledge that could affect the agent's behavior when operating in specific environments.
- **Efficiency**: LLM agents involve a significant amount of requests that are handled by the LLM which could affect the efficiency of agent actions because it would depend heavily on the LLM inference speed. Cost is also a concern when deploying multiple agents.

Previously on CENG501




Gao et al., “Retrieval-Augmented Generation for Large Language Models: A Survey”, 2024.

Previously on CENG501

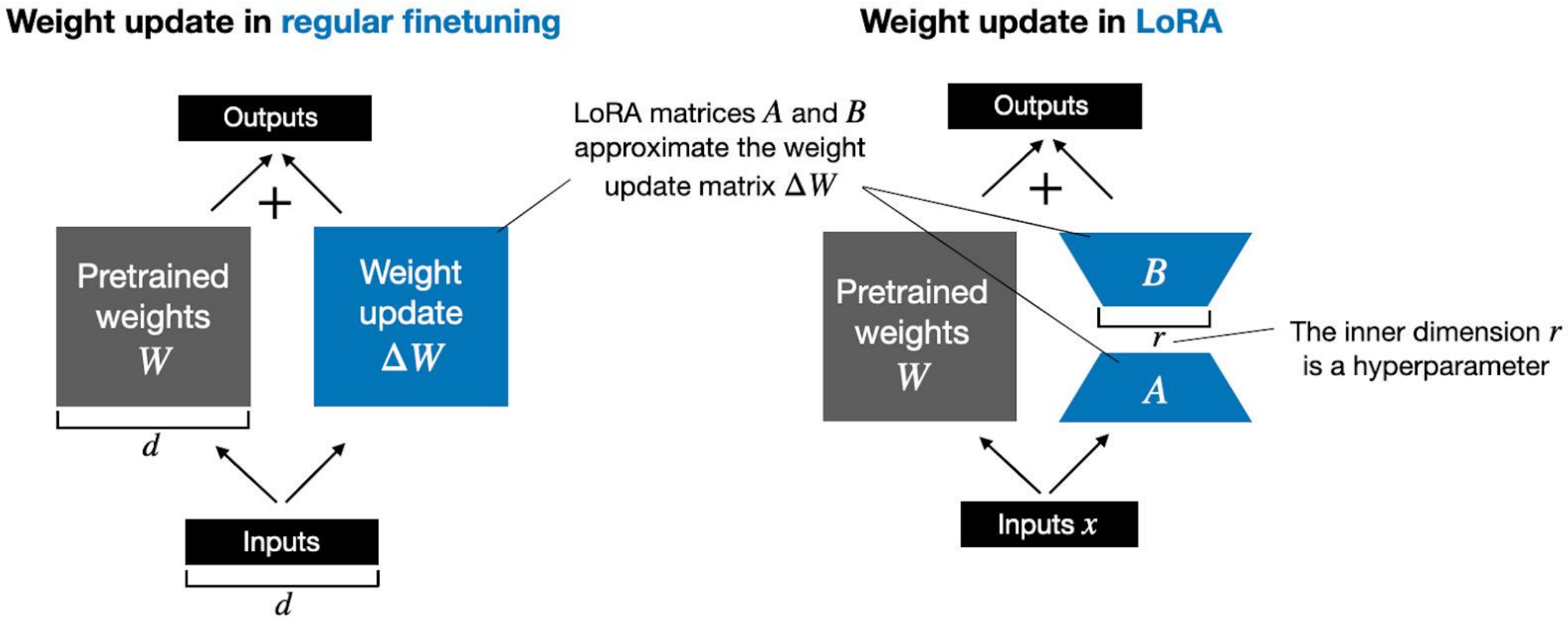
# Finetuning an LLM

- Parameter-efficient Finetuning (vs. Full Finetuning)
  - Update a subset of parameters
- LoRA, LoRA+
- LASER (not finetuning actually)

Previously on ENG501

# LoRA

Hu et al., “LORA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS”, 2021.




Target: Attention blocks

Fig: https://magazine.sebastianraschka.com/p/practical-tips-for-finetuning-llms

Previously on ENG501

LoRA+

## Abstract

In this paper, we show that Low Rank Adaptation (LoRA) as originally introduced in (Hu et al., 2021) leads to suboptimal finetuning of models with large width (embedding dimension). This is due to the fact that adapter matrices $A$ and $B$ in LoRA are updated with the same learning rate. Using scaling arguments for large width networks, we demonstrate that using the same learning rate for $A$ and $B$ does not allow efficient feature learning. We then show that this suboptimality of LoRA can be corrected simply by setting different learning rates for the LoRA adapter matrices $A$ and $B$ with a well-chosen fixed ratio. We call this proposed algorithm LoRA+. In our extensive experiments, LoRA+ improves performance ($1\% - 2\%$ improvements) and finetuning speed (up to $\sim$ 2X SpeedUp), at the same computational cost as LoRA.

Hayou et al., "LoRA+: Efficient Low Rank Adaptation of Large Models", 2024.

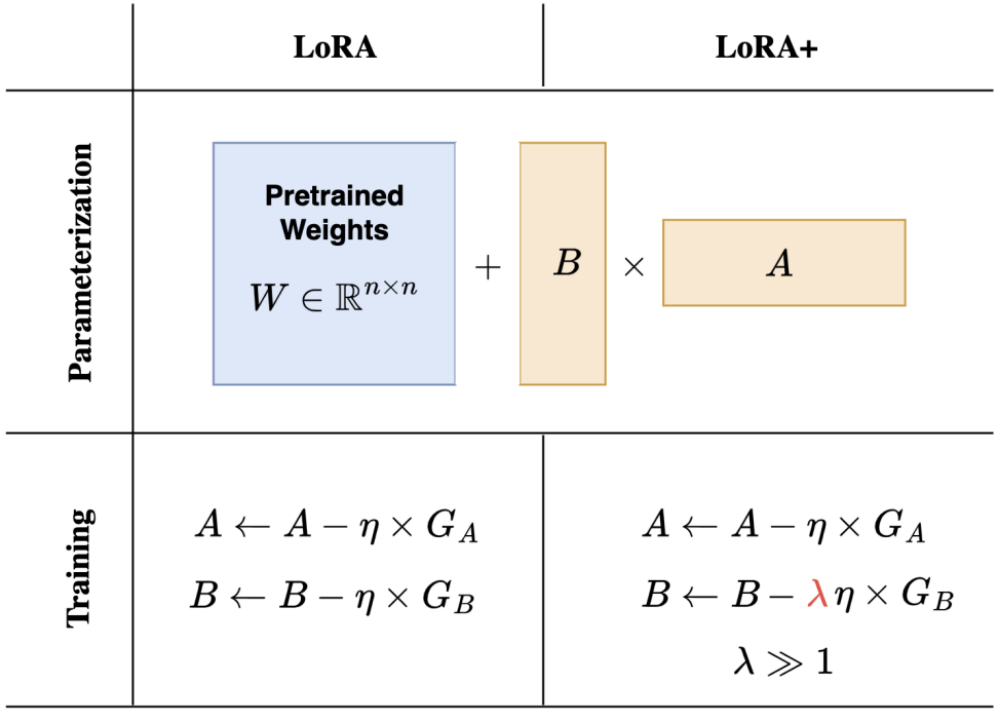



*Figure 1.* The key difference between standard LoRA and LoRA+ is in how learning rates are set (the matrices $G_A$ and $G_B$ are 'effective' gradients from AdamW) With standard LoRA, the learning rate is the same for $A$ and $B$, which provably leads to suboptimal learning when embedding dimension is large. In LoRA+, we set the learning rate of $B$ to be $\lambda\times$ that of $A$, where $\lambda \gg 1$ is fixed. We later provide guidelines on how to set $\lambda$

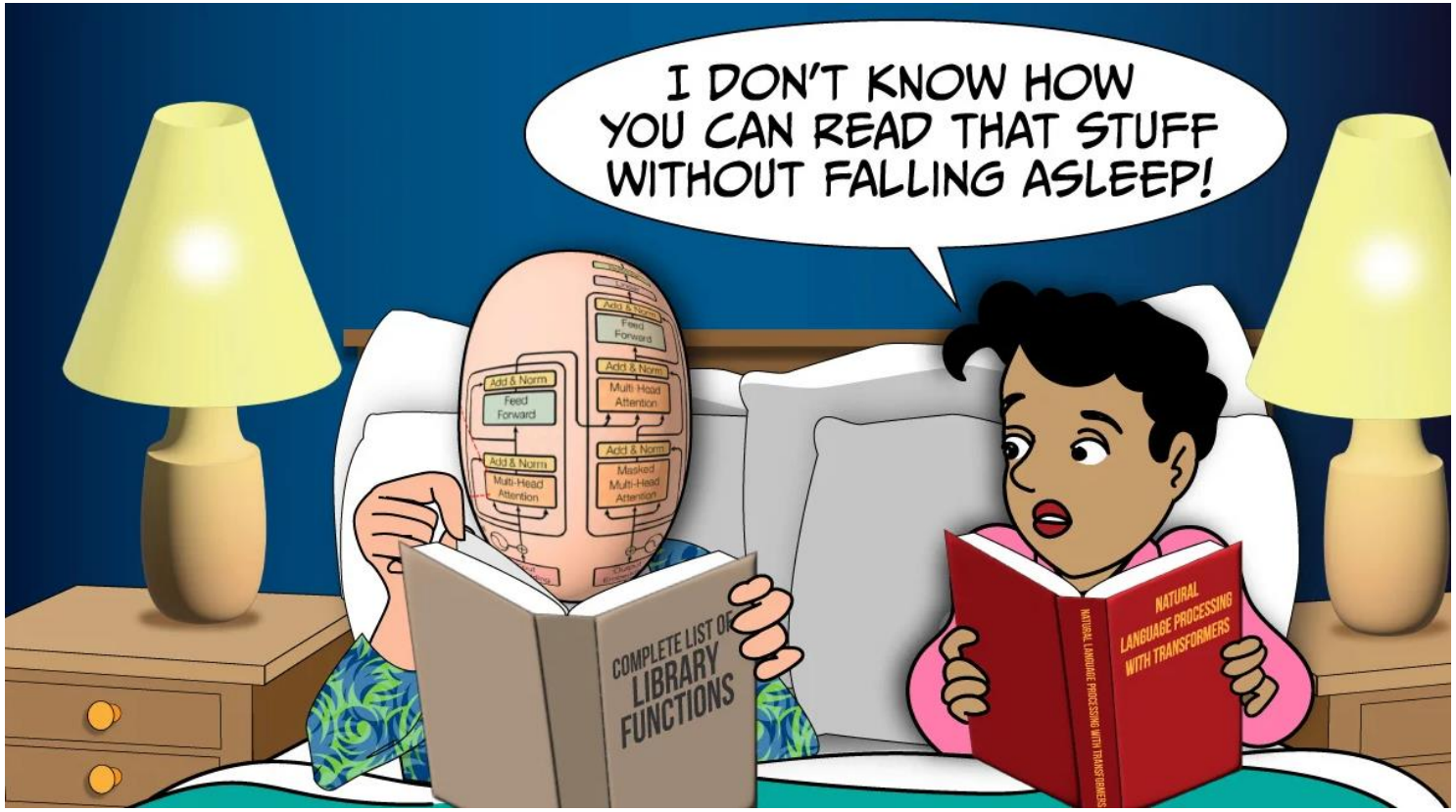
# Fine-tuning an LLM: Challenges

Previously on CENG491

1. **Overfitting:** Fine-tuning can be prone to overfitting, a condition where the model becomes overly specialized on the training data and performs poorly on unseen data. This risk is particularly pronounced when the task-specific dataset is small or not representative of the broader context.
2. **Catastrophic Forgetting:** During fine-tuning for a specific task, the model may forget previously acquired general knowledge. This phenomenon, known as catastrophic forgetting, can impair the model's adaptability to diverse tasks.
3. **Bias Amplification:** Pre-trained models inherit biases from their training data, which fine-tuning can inadvertently amplify when applied to task-specific data. This amplification may lead to biased predictions and outputs, potentially causing ethical concerns.
4. **Generalization Challenges:** Ensuring that a fine tuned model generalizes effectively across various inputs and scenarios is challenging. A model that excels in fine-tuning datasets may struggle when presented with out-of-distribution data.
5. **Data Requirements:** Fine-tuning necessitates task-specific labelled data, which may not always be available or clean. Inadequate or noisy data can negatively impact the model's performance and reliability.
6. **Hyperparameter Tuning Complexity:** Selecting appropriate hyperparameters for fine-tuning can be intricate and time-consuming. Poor choices may result in slow convergence, overfitting, or suboptimal performance.
7. **Domain Shift Sensitivity:** Fine-tuning data significantly different from the pre-training data can lead to domain shift issues. Addressing this problem often requires domain adaptation techniques to bridge the gap effectively.
8. **Ethical Considerations:** Fine tuned large language models may inadvertently generate harmful or inappropriate content, even when designed for benign tasks. Ensuring ethical behaviour and safety is an ongoing challenge, necessitating responsible AI practices.
9. **Resource Intensiveness:** Fine-tuning large models demands substantial computational resources and time, posing challenges for smaller teams or organizations with limited infrastructure and expertise.
10. **Unintended Outputs:** Fine-tuning cannot guarantee that the model consistently produces correct or sensible outputs. It may generate plausible but factually incorrect responses, requiring vigilant post-processing and validation.
11. **Model Drift:** Over time, a fine tuned model's performance can deteriorate due to changes in data distribution or the evolving environment. Regular monitoring and re-fine-tuning may become necessary to maintain optimal performance and adapt to evolving conditions.

From: https://www.lakera.ai/blog/llm-fine-tuning-guide

# The Batch




https://www.deeplearning.ai/the-batch/issue-276/

# Today

- Vision Models
  - Vision Transformers
  - Swin Transformers
  - Fast/Faster ViTs
  - Pretraining

# Administrative Notes

- New quiz this week
  - Deadline: Tomorrow midnight

- Time plan for the projects
  1. Milestone (November 24, midnight):
     - Github repo will be ready
     - Read & understand the paper
     - Download the datasets
     - Prepare the Readme file excluding the results & conclusion
  2. Milestone (December 8, midnight)
     - The results of the first experiment
  3. Milestone (January 5, midnight)
     - Final report (Readme file)
     - Repo with all code & trained models

# AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

**Alexey Dosovitskiy**[*,†], **Lucas Beyer**[*], **Alexander Kolesnikov**[*], **Dirk Weissenborn**[*], **Xiaohua Zhai**[*], **Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby**[*,†]

[*]equal technical contribution, [†]equal advising

Google Research, Brain Team

{adosovitskiy, neilhoulsby}@google.com

# Vision Transformers (ViT)

# ViT: Motivation

- NLP:
  - Successful results of Transformers
  - De facto architecture
- Vision:
  - CNNs are dominantly used
- Prior work
  - Self-attention among pixels. Limit attention to local regions to reduce complexity (Parmar et al., 2018)
  - Self-attention among blocks of different sizes (Weissenborn et al., 2019)
  - Self attention between 2x2 patches (Cordonnier et al., 2020)

# ViT: Motivation

- Prior work
  - Self attention between 2x2 patches (Cordonnier et al., 2020)

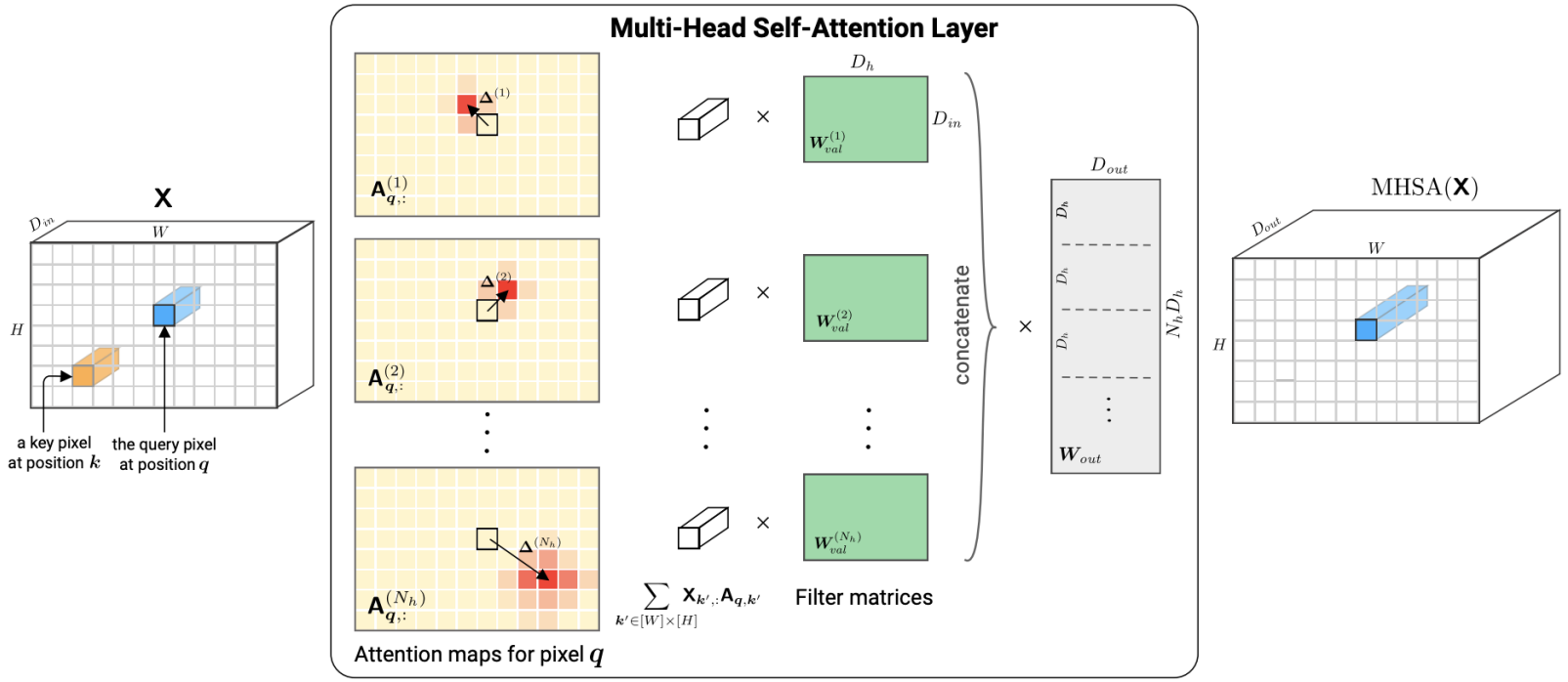



Figure 1: Illustration of a Multi-Head Self-Attention layer applied to a tensor image $\mathbf{X}$. Each head $h$ attends pixel values around shift $\boldsymbol{\Delta}^{(h)}$ and learn a filter matrix $\boldsymbol{W}_{val}^{(h)}$. We show attention maps computed for a query pixel at position $\boldsymbol{q}$.

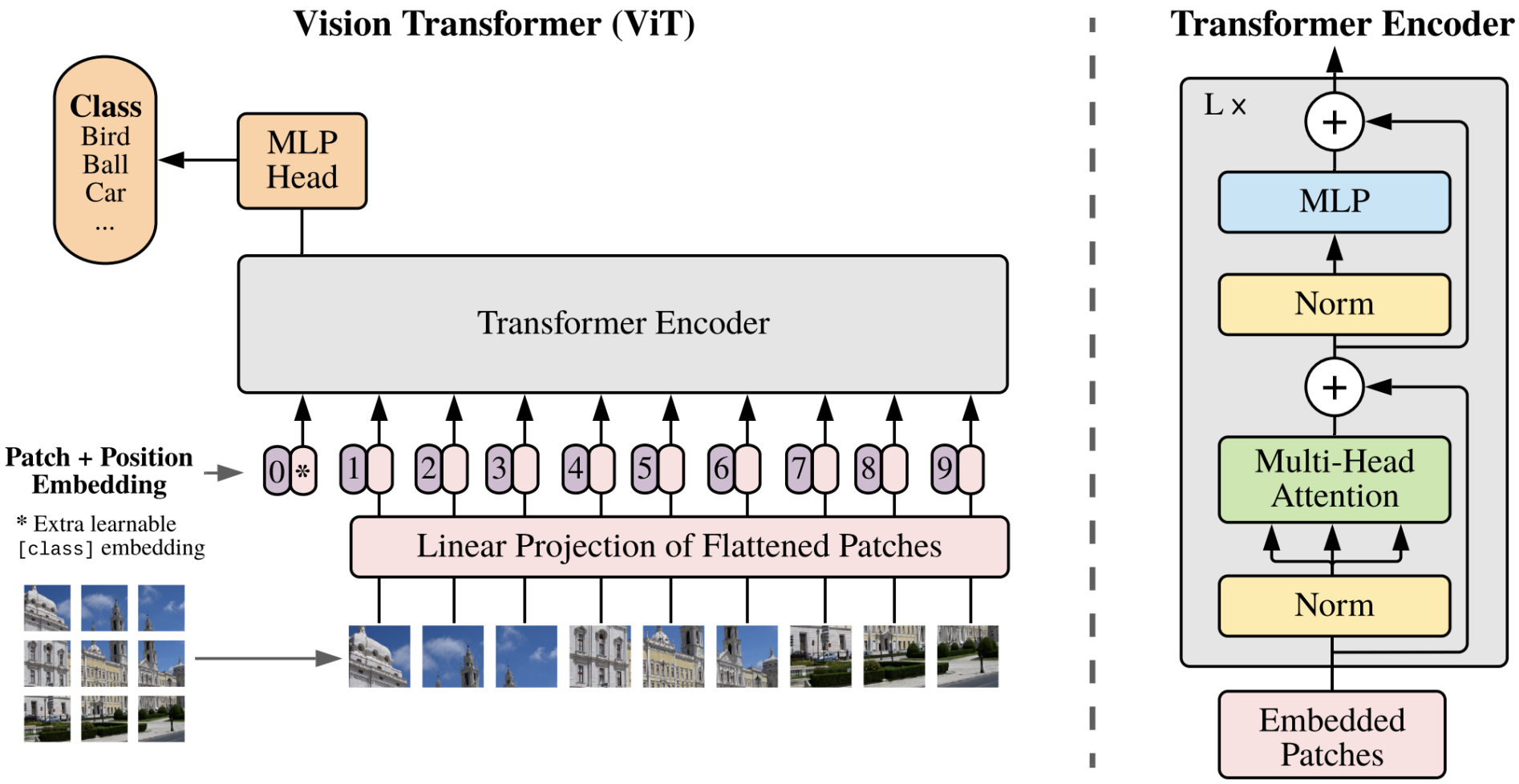
# ViT: Motivation

- Contributions
  - Use plain transformers without modifications
  - Compared to Cordonnier et al. (2020):
    - ViT shows strong results with large datasets & pretraining
    - ViT works with larger patches (generalizes better)

- Challenges in using Transformers with images:
  - How to convert an image into tokens?
  - How to encode position?
  - Transformers lack translation invariance and locality
  - Spoiler: They underperform on “small” datasets

Applied to classification tasks only!

# ViT: Architecture

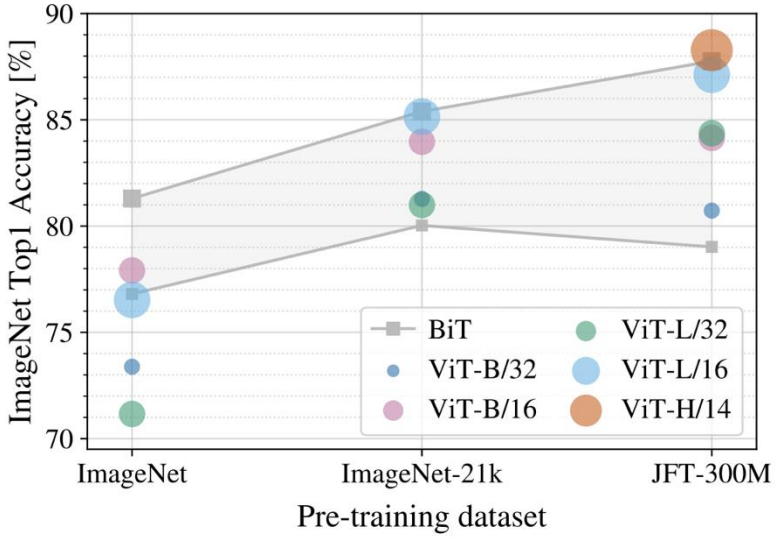


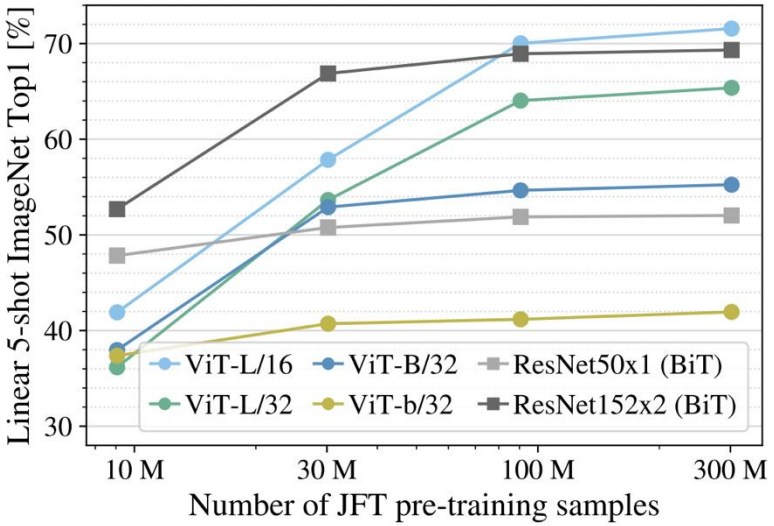
# ViT: Pretraining

- Supervised pretraining on large datasets and finetuning on smaller downstream tasks
- Higher resolutions:
  - The whole network can easily be applied to longer sequences (= higher resolutions)
  - Learned position embeddings are interpolated

# ViT: Results

| | Ours-JFT (ViT-H/14) | Ours-JFT (ViT-L/16) | Ours-I21k (ViT-L/16) | BiT-L (ResNet152x4) | Noisy Student (EfficientNet-L2) |
|---|---|---|---|---|---|
| ImageNet | **88.55** ± 0.04 | 87.76 ± 0.03 | 85.30 ± 0.02 | 87.54 ± 0.02 | 88.4/88.5* |
| ImageNet ReaL | **90.72** ± 0.05 | 90.54 ± 0.03 | 88.62 ± 0.05 | 90.54 | 90.55 |
| CIFAR-10 | **99.50** ± 0.06 | 99.42 ± 0.03 | 99.15 ± 0.03 | 99.37 ± 0.06 | – |
| CIFAR-100 | **94.55** ± 0.04 | 93.90 ± 0.05 | 93.25 ± 0.05 | 93.51 ± 0.08 | – |
| Oxford-IIIT Pets | **97.56** ± 0.03 | 97.32 ± 0.11 | 94.67 ± 0.15 | 96.62 ± 0.23 | – |
| Oxford Flowers-102 | 99.68 ± 0.02 | **99.74** ± 0.00 | 99.61 ± 0.02 | 99.63 ± 0.03 | – |
| VTAB (19 tasks) | **77.63** ± 0.23 | 76.28 ± 0.46 | 72.72 ± 0.21 | 76.29 ± 1.70 | – |
| TPUv3-core-days | 2.5k | 0.68k | 0.23k | 9.9k | 12.3k |



2021

**Swin Transformer: Hierarchical Vision Transformer using Shifted Windows**

Ze Liu[1,2†*] Yutong Lin[1,3†*] Yue Cao[1*] Han Hu[1*‡] Yixuan Wei[1,4†]
Zheng Zhang[1] Stephen Lin[1] Baining Guo[1]
[1]Microsoft Research Asia [2]University of Science and Technology of China
[3]Xian Jiaotong University [4]Tsinghua University
{v-zeliu1,v-yutlin,yuecao,hanhu,v-yixwe,zhez,stevelin,bainguo}@microsoft.com



2022

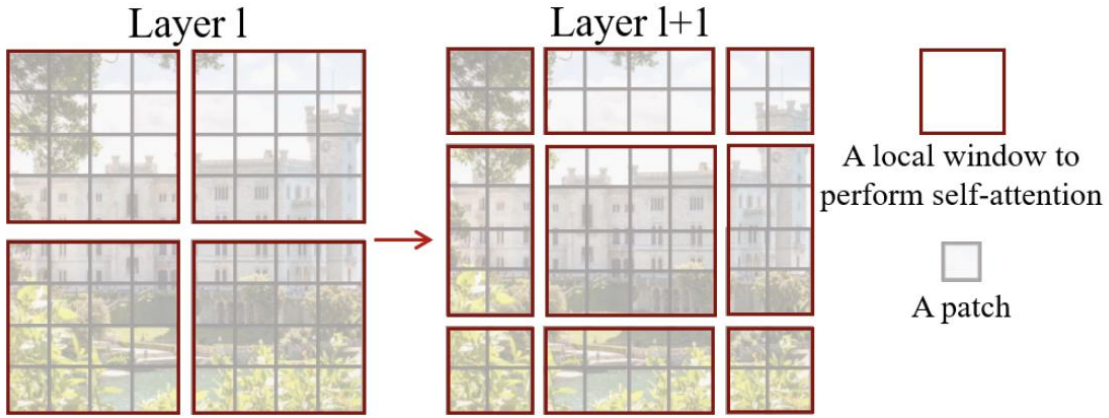
**Swin Transformer V2: Scaling Up Capacity and Resolution**

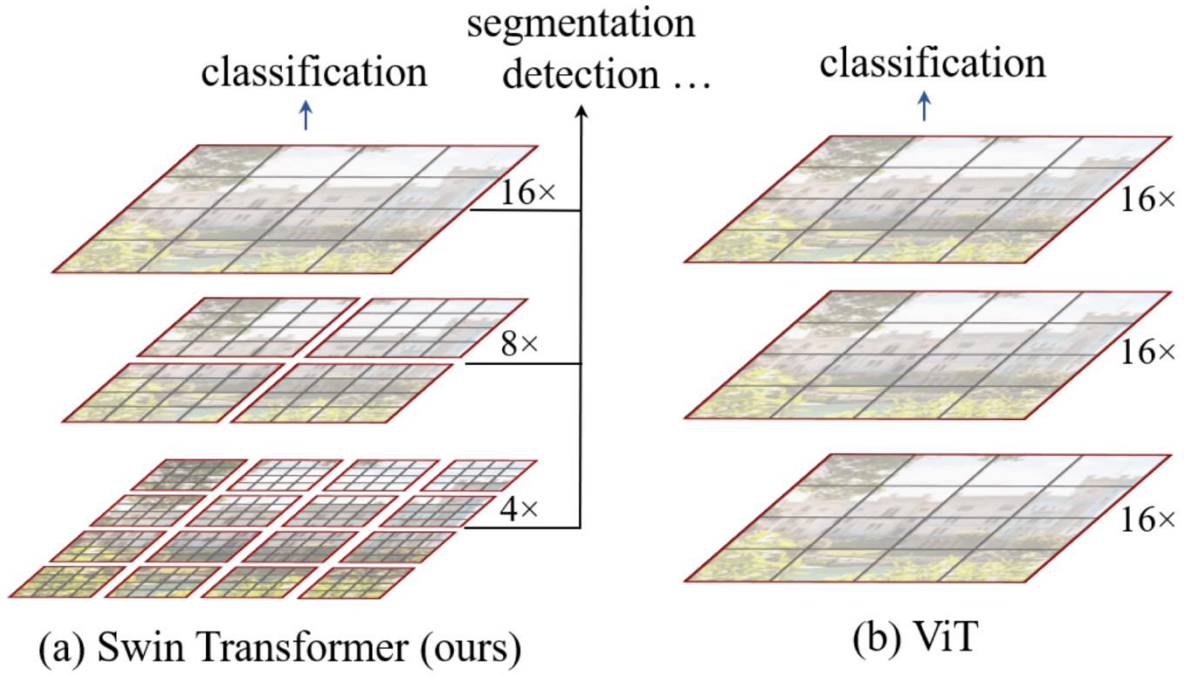
Ze Liu[2*] Han Hu[1*†] Yutong Lin[3] Zhuliang Yao[4] Zhenda Xie[4] Yixuan Wei[4] Jia Ning[5]
Yue Cao[1] Zheng Zhang[1] Li Dong[1] Furu Wei[1] Baining Guo[1]
[1]Microsoft Research Asia [2]University of Science and Technology of China
[3]Xian Jiaotong University [4]Tsinghua University [5]Huazhong University of Science and Technology
{t-liuze,hanhu,t-yutonglin,t-zhuyao,t-zhxie,t-yixuanwei,v-jianing}@microsoft.com
{yuecao,zhez,lidong1,fuwei,bainguo}@microsoft.com

# Swin Transformers

# Swin Transformer v1

- Motivation:
  - ViT is promising but limited to classification
  - Challenges in using Tranformers:
    - large variations in scales of visual entities,
    - more pixels compared to words in text
  - Existing Transformers use fixed token size across layers
- Contributions:
  - Limit self-attention to non-overlapping windows while allowing cross-window attention
  - Change token size across layers



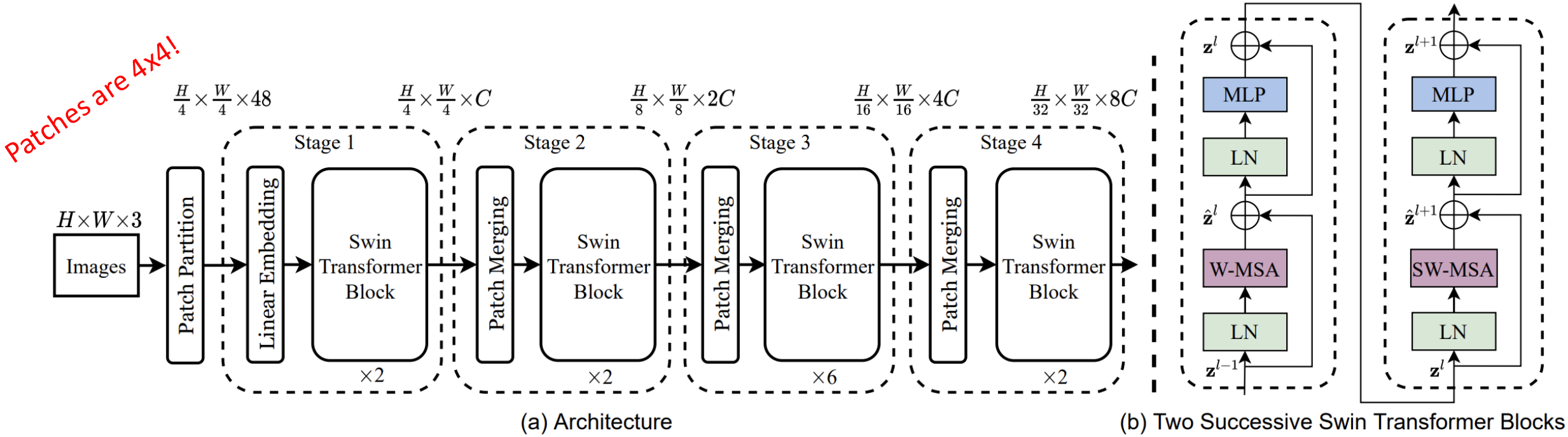



Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

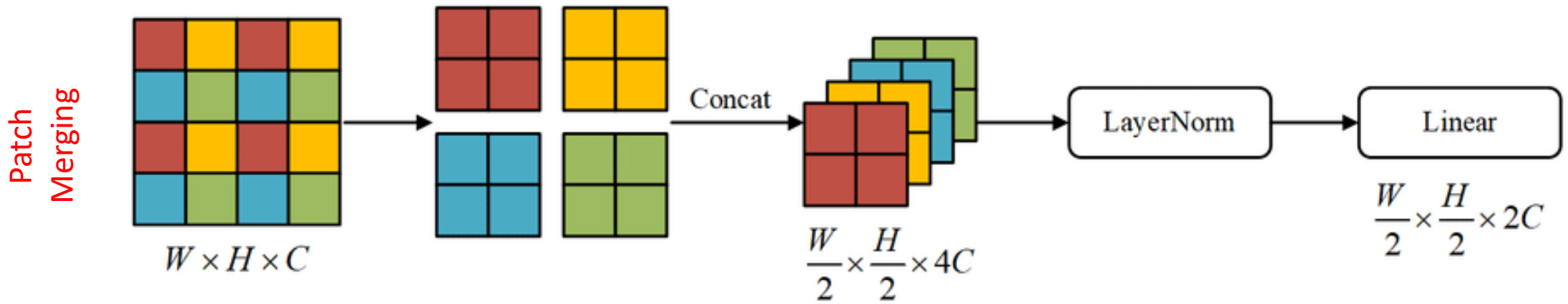



Image: http://dx.doi.org/10.48550/arXiv.2206.03336

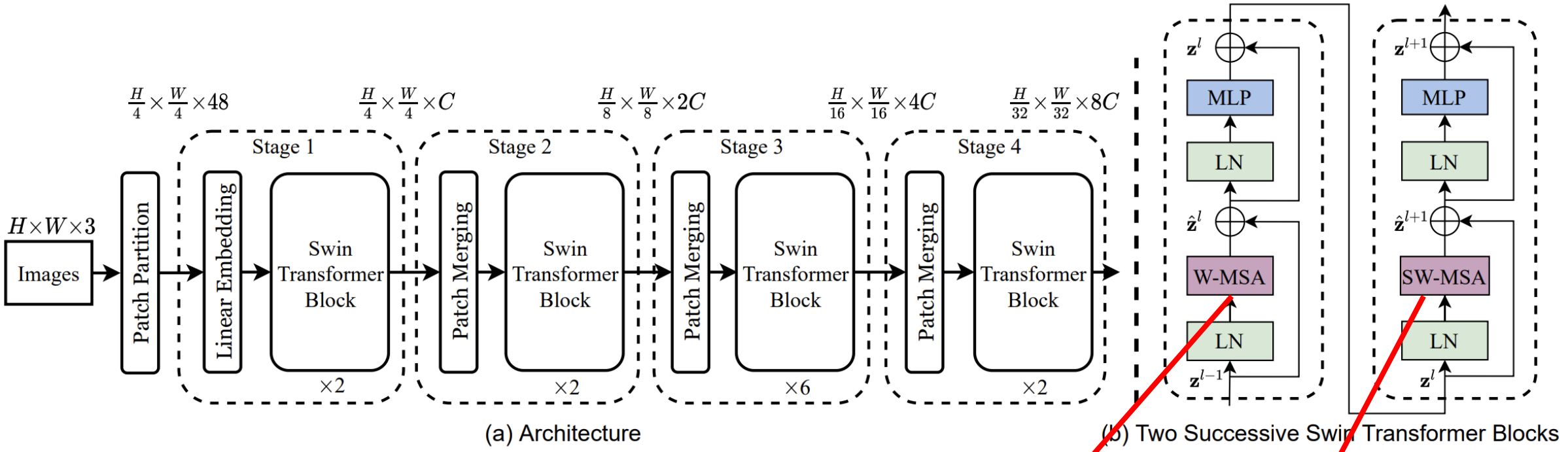
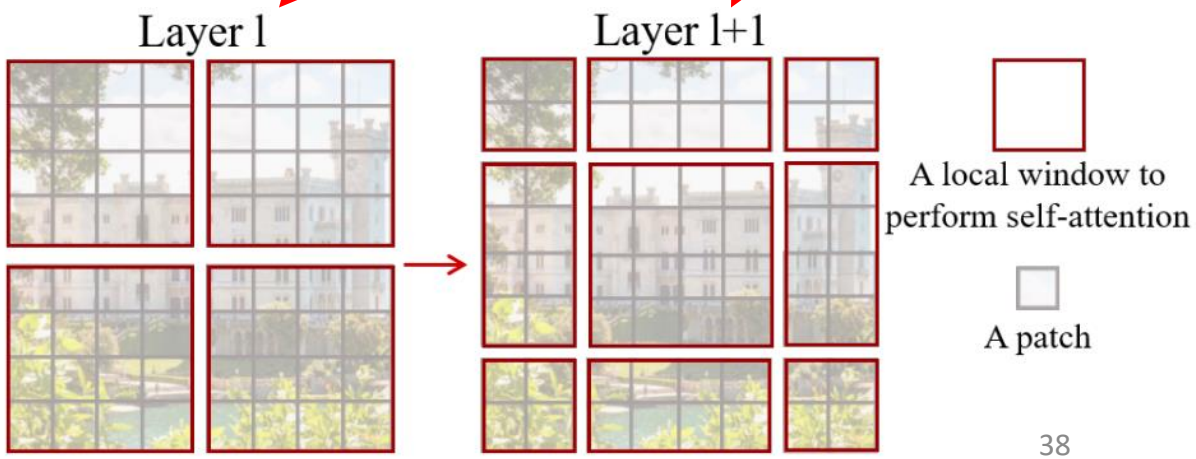



Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 |
| 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 |
| 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 |

A: Attention in local windows

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 |
| 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 |
| 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 |

B: Attention in shifted windows (2,2)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 19 | 20 | 21 | 22 | 23 | 24 | 17 | 18 |
| 27 | 28 | 29 | 30 | 31 | 32 | 25 | 26 |
| 35 | 36 | 37 | 38 | 39 | 40 | 33 | 34 |
| 43 | 44 | 45 | 46 | 47 | 48 | 41 | 42 |
| 51 | 52 | 53 | 54 | 55 | 56 | 49 | 50 |
| 59 | 60 | 61 | 62 | 63 | 64 | 57 | 58 |
| 3 | 4 | 5 | 6 | 7 | 8 | 1 | 2 |
| 11 | 12 | 13 | 14 | 15 | 16 | 9 | 10 |

C: Batched windows after cyclic shift

Figure: https://amaarora.github.io/posts/2022-07-04-swintransformerv1.html

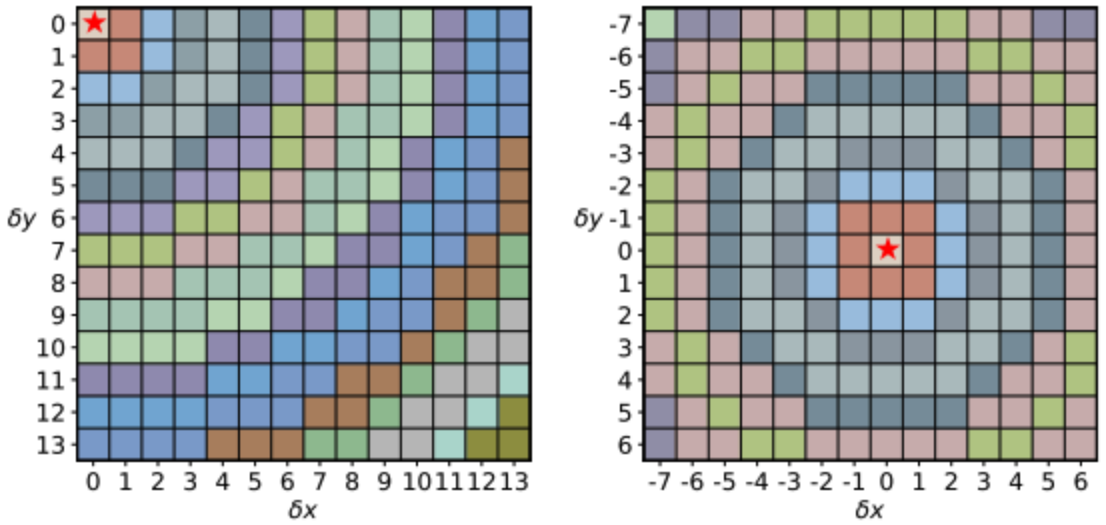
# Swin Transformer v1

**Relative position bias** In computing self-attention, we follow [49, 1, 32, 33] by including a relative position bias $B \in \mathbb{R}^{M^2 \times M^2}$ to each head in computing similarity:

$$\text{Attention}(Q, K, V) = \text{SoftMax}(QK^T/\sqrt{d} + B)V, \quad (4)$$

where $Q, K, V \in \mathbb{R}^{M^2 \times d}$ are the *query*, *key* and *value* matrices; $d$ is the *query/key* dimension, and $M^2$ is the number of patches in a window. Since the relative position along each axis lies in the range $[-M+1, M-1]$, we parameterize a smaller-sized bias matrix $\hat{B} \in \mathbb{R}^{(2M-1)\times(2M-1)}$, and values in $B$ are taken from $\hat{B}$.

Plot from https://arxiv.org/pdf/2107.14222:




(a) top-left (b) center

Figure 6: Visualization of Euclidean method. The red star ★ presents the reference position. Different color means different bucket. The relative positions with the same color share the same encoding.

# Swin Transformer v1

**Relative position bias** In computing self-attention, we follow [49, 1, 32, 33] by including a relative position bias $B \in \mathbb{R}^{M^2 \times M^2}$ to each head in computing similarity:

$$\text{Attention}(Q, K, V) = \text{SoftMax}(QK^T/\sqrt{d} + B)V, \quad (4)$$

where $Q, K, V \in \mathbb{R}^{M^2 \times d}$ are the *query*, *key* and *value* matrices; $d$ is the *query/key* dimension, and $M^2$ is the number of patches in a window. Since the relative position along each axis lies in the range $[-M+1, M-1]$, we parameterize a smaller-sized bias matrix $\hat{B} \in \mathbb{R}^{(2M-1)\times(2M-1)}$, and values in $B$ are taken from $\hat{B}$.

| | ImageNet | | COCO | | ADE20k |
|---|---|---|---|---|---|
| | top-1 | top-5 | $AP^{box}$ | $AP^{mask}$ | mIoU |
| w/o shifting | 80.2 | 95.1 | 47.7 | 41.5 | 43.3 |
| shifted windows | **81.3** | **95.6** | **50.5** | **43.7** | **46.1** |
| no pos. | 80.1 | 94.9 | 49.2 | 42.6 | 43.8 |
| abs. pos. | 80.5 | 95.2 | 49.0 | 42.4 | 43.2 |
| abs.+rel. pos. | 81.3 | 95.6 | 50.2 | 43.4 | 44.0 |
| rel. pos. w/o app. | 79.3 | 94.7 | 48.2 | 41.9 | 44.1 |
| rel. pos. | **81.3** | **95.6** | **50.5** | **43.7** | **46.1** |

Table 4. Ablation study on the *shifted windows* approach and different position embedding methods on three benchmarks, using the Swin-T architecture. w/o shifting: all self-attention modules adopt regular window partitioning, without *shifting*; abs. pos.: absolute position embedding term of ViT; rel. pos.: the default settings with an additional relative position bias term (see Eq. (4)); app.: the first scaled dot-product term in Eq. (4).

# Swin Transformer v1: Results

**(a) Regular ImageNet-1K trained models**

| method | image size | #param. | FLOPs | throughput (image / s) | ImageNet top-1 acc. |
|---|---|---|---|---|---|
| RegNetY-4G [44] | $224^2$ | 21M | 4.0G | 1156.7 | 80.0 |
| RegNetY-8G [44] | $224^2$ | 39M | 8.0G | 591.6 | 81.7 |
| RegNetY-16G [44] | $224^2$ | 84M | 16.0G | 334.7 | 82.9 |
| ViT-B/16 [19] | $384^2$ | 86M | 55.4G | 85.9 | 77.9 |
| ViT-L/16 [19] | $384^2$ | 307M | 190.7G | 27.3 | 76.5 |
| DeiT-S [57] | $224^2$ | 22M | 4.6G | 940.4 | 79.8 |
| DeiT-B [57] | $224^2$ | 86M | 17.5G | 292.3 | 81.8 |
| DeiT-B [57] | $384^2$ | 86M | 55.4G | 85.9 | 83.1 |
| Swin-T | $224^2$ | 29M | 4.5G | 755.2 | 81.3 |
| Swin-S | $224^2$ | 50M | 8.7G | 436.9 | 83.0 |
| Swin-B | $224^2$ | 88M | 15.4G | 278.1 | 83.5 |
| Swin-B | $384^2$ | 88M | 47.0G | 84.7 | 84.5 |

**(b) ImageNet-22K pre-trained models**

| method | image size | #param. | FLOPs | throughput (image / s) | ImageNet top-1 acc. |
|---|---|---|---|---|---|
| R-101x3 [34] | $384^2$ | 388M | 204.6G | - | 84.4 |
| R-152x4 [34] | $480^2$ | 937M | 840.5G | - | 85.4 |
| ViT-B/16 [19] | $384^2$ | 86M | 55.4G | 85.9 | 84.0 |
| ViT-L/16 [19] | $384^2$ | 307M | 190.7G | 27.3 | 85.2 |
| Swin-B | $224^2$ | 88M | 15.4G | 278.1 | 85.2 |
| Swin-B | $384^2$ | 88M | 47.0G | 84.7 | 86.4 |
| Swin-L | $384^2$ | 197M | 103.9G | 42.1 | 87.3 |

Table 1. Comparison of different backbones on ImageNet-1K classification. Throughput is measured using the GitHub repository of [62] and a V100 GPU, following [57].

**(a) Various frameworks**

| Method | Backbone | $AP^{box}$ | $AP^{box}_{50}$ | $AP^{box}_{75}$ | #param. | FLOPs | FPS |
|---|---|---|---|---|---|---|---|
| Cascade Mask R-CNN | R-50 | 46.3 | 64.3 | 50.5 | 82M | 739G | 18.0 |
| | Swin-T | **50.5** | **69.3** | **54.9** | 86M | 745G | 15.3 |
| ATSS | R-50 | 43.5 | 61.9 | 47.0 | 32M | 205G | 28.3 |
| | Swin-T | **47.2** | **66.5** | **51.3** | 36M | 215G | 22.3 |
| RepPointsV2 | R-50 | 46.5 | 64.6 | 50.3 | 42M | 274G | 13.6 |
| | Swin-T | **50.0** | **68.5** | **54.2** | 45M | 283G | 12.0 |
| Sparse R-CNN | R-50 | 44.5 | 63.4 | 48.2 | 106M | 166G | 21.0 |
| | Swin-T | **47.9** | **67.3** | **52.3** | 110M | 172G | 18.4 |

**(b) Various backbones w. Cascade Mask R-CNN**

| | $AP^{box}$ | $AP^{box}_{50}$ | $AP^{box}_{75}$ | $AP^{mask}$ | $AP^{mask}_{50}$ | $AP^{mask}_{75}$ | param | FLOPs | FPS |
|---|---|---|---|---|---|---|---|---|---|
| DeiT-S† | 48.0 | 67.2 | 51.7 | 41.4 | 64.2 | 44.3 | 80M | 889G | 10.4 |
| R50 | 46.3 | 64.3 | 50.5 | 40.1 | 61.7 | 43.4 | 82M | 739G | 18.0 |
| Swin-T | **50.5** | **69.3** | **54.9** | **43.7** | **66.6** | **47.1** | 86M | 745G | 15.3 |
| X101-32 | 48.1 | 66.5 | 52.4 | 41.6 | 63.9 | 45.2 | 101M | 819G | 12.8 |
| Swin-S | **51.8** | **70.4** | **56.3** | **44.7** | **67.9** | **48.5** | 107M | 838G | 12.0 |
| X101-64 | 48.3 | 66.4 | 52.3 | 41.7 | 64.0 | 45.1 | 140M | 972G | 10.4 |
| Swin-B | **51.9** | **70.9** | **56.5** | **45.0** | **68.4** | **48.7** | 145M | 982G | 11.6 |

**(c) System-level Comparison**

| Method | mini-val $AP^{box}$ | mini-val $AP^{mask}$ | test-dev $AP^{box}$ | test-dev $AP^{mask}$ | #param. | FLOPs |
|---|---|---|---|---|---|---|
| RepPointsV2* [12] | - | - | 52.1 | - | - | - |
| GCNet* [7] | 51.8 | 44.7 | 52.3 | 45.4 | - | 1041G |
| RelationNet++* [13] | - | - | 52.7 | - | - | - |
| DetectoRS* [42] | - | - | 55.7 | 48.5 | - | - |
| YOLOv4 P7* [4] | - | - | 55.8 | - | - | - |
| Copy-paste [23] | 55.9 | 47.2 | 56.0 | 47.4 | 185M | 1440G |
| X101-64 (HTC++) | 52.3 | 46.0 | - | - | 155M | 1033G |
| Swin-B (HTC++) | 56.4 | 49.1 | - | - | 160M | 1043G |
| Swin-L (HTC++) | 57.1 | 49.5 | 57.7 | 50.2 | 284M | 1470G |
| Swin-L (HTC++)* | **58.0** | **50.4** | **58.7** | **51.1** | 284M | - |

Table 2. Results on COCO object detection and instance segmentation. †denotes that additional decovolution layers are used to produce hierarchical feature maps. * indicates multi-scale testing.

# Swin Transformer v2

Motivation:

- “Large-scale NLP models have been shown to significantly improve the performance on language tasks with no signs of saturation”.
- “They also demonstrate amazing few-shot capabilities like that of human beings”

Contribution:

- “Explore large-scale models in computer vision”
- Three major issues & solutions:
  - Training stability: “1) a residual-post-norm method combined with cosine attention to improve training stability”
  - resolution gaps between pre-training and fine-tuning: “2) A log-spaced continuous position bias method to effectively transfer models pre-trained using low-resolution images to downstream tasks with high-resolution inputs”
  - hunger on labelled data: “3) A self-supervised pretraining method, SimMIM, to reduce the needs of vast labeled images”

# Swin Transformer v2

- Issues with v1
  - “An instability issue when scaling up model capacity.”
  - “Degraded performance when transferring models across window resolutions.”

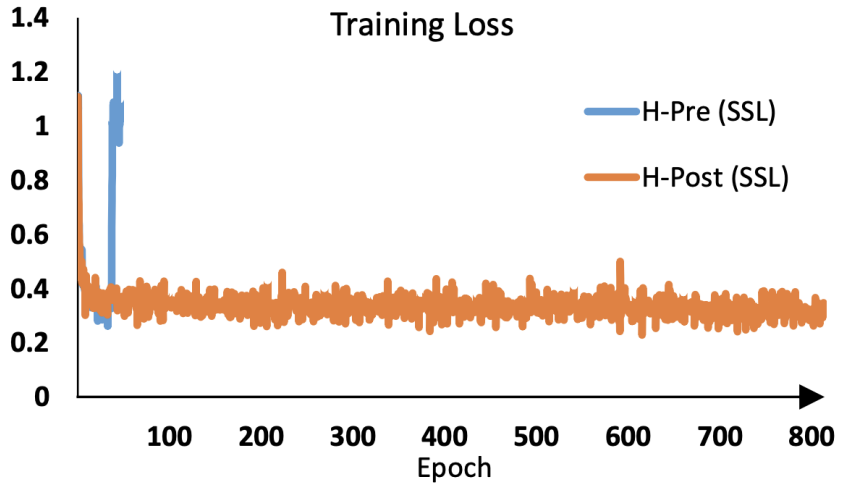


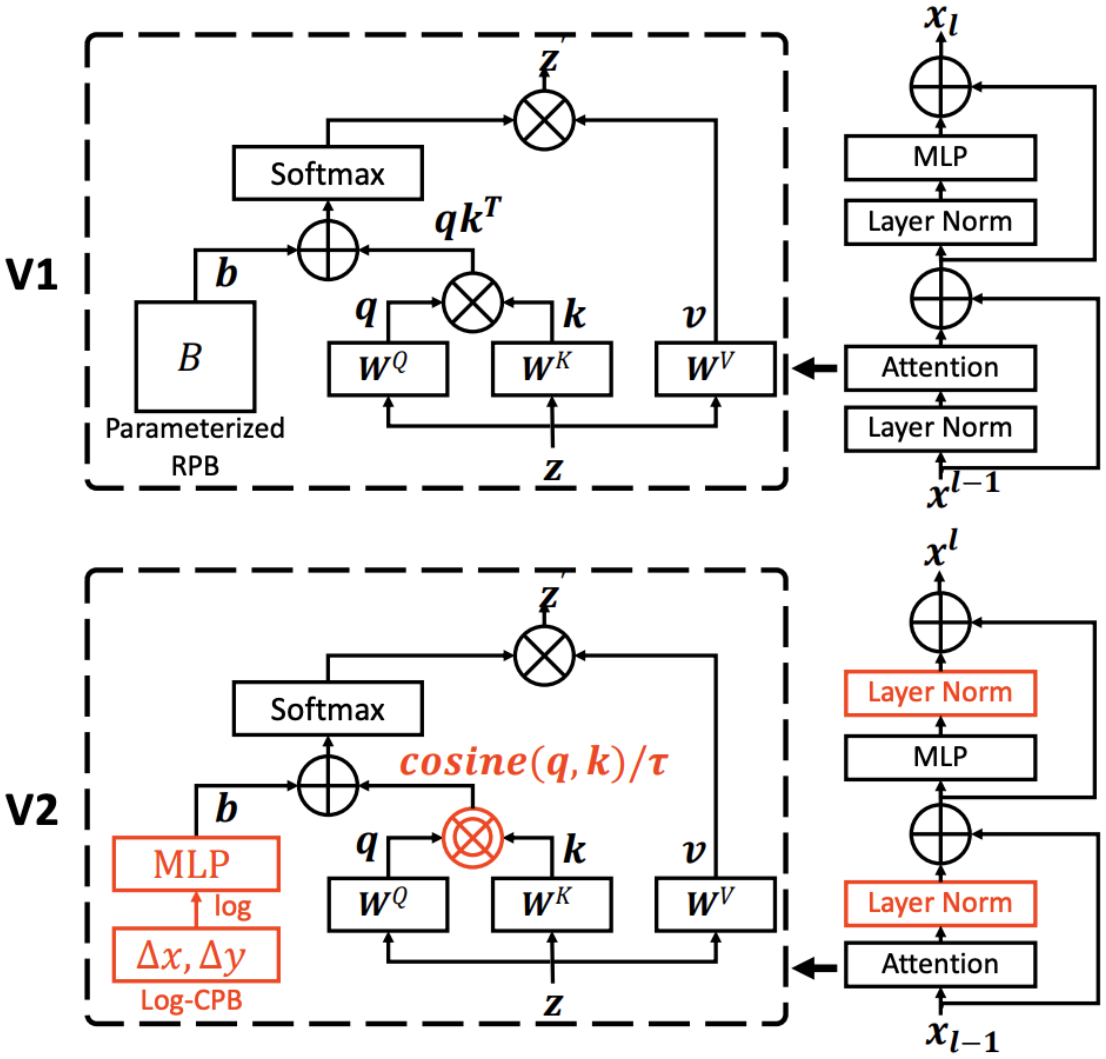

Figure 3. SwinV1-H versus SwinV2-H in training [72].

| | Trained on | Tested/Finetuned on | | | | Finetuned on | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | ImageNet* | ImageNet† | | | | COCO | | ADE20k | | |
| method | W8, I256 top-1 acc | W12, I384 top-1 acc | W16, I512 top-1 acc | W20, I640 top-1 acc | W24, I768 top-1 acc | W16 $AP^{box}$ | W32 $AP^{box}$ | W16 mIoU | W20 mIoU | W32 mIoU |
| Parameterized position bias [46] | 81.7 | 79.4/82.7 | 77.2/83.0 | 73.2/83.2 | 68.7/83.2 | 50.8 | 50.9 | 45.5 | 45.8 | 44.5 |
| Linear-Spaced CPB | 81.7 (+0.0) | 82.0/82.9 (+2.6/+0.2) | 81.2/83.3 (+4.0/+0.3) | 79.8/83.6 (+6.6/+0.4) | 77.6/83.7 (+8.9/+0.5) | 50.9 (+0.1) | 51.7 (+0.8) | 47.0 (+1.5) | 47.4 (+1.6) | 47.2 (+2.7) |
| Log-Spaced CPB | 81.8 (+0.1) | 82.4/83.2 (+3.0/+0.5) | 81.7/83.8 (+4.5/+0.8) | 80.4/84.0 (+7.2/+0.8) | 79.1/84.2 (+10.4/+1.0) | 51.1 (+0.3) | 51.8 (+0.9) | 47.0 (+1.5) | 47.7 (+1.9) | 47.8 (+3.3) |

# Swin Transformer v2

## Scaling up model capacity

- Residual post normalization: Prevents output to diverge
- Scaled cosine attention:
  - learnt attention maps of some blocks and heads are frequently dominated by a few pixel pairs, especially in the res-post-norm configuration => cosine yields values in a smaller range



# Scaling Up Window Resolution

- Continuous relative position bias

$$B(\Delta x, \Delta y) = \mathcal{G}(\Delta x, \Delta y), \tag{3}$$

where $\mathcal{G}$ is a small network, e.g., a 2-layer MLP with a ReLU activation in between by default.

The meta network $\mathcal{G}$ generates bias values for arbitrary relative coordinates, and thus can be naturally transferred to fine-tuning tasks with arbitrarily varying window sizes.

**Log-spaced coordinates**

**Log-spaced coordinates** When transferring across largely varying window sizes, a large portion of the relative coordinate range needs to be extrapolated. To ease this issue, we propose using log-spaced coordinates instead of the original linear-spaced ones:

$$\begin{aligned} \widehat{\Delta x} &= \operatorname{sign}(x) \cdot \log(1 + |\Delta x|), \\ \widehat{\Delta y} &= \operatorname{sign}(y) \cdot \log(1 + |\Delta y|), \end{aligned} \tag{4}$$

where $\Delta x$, $\Delta y$ and $\widehat{\Delta x}$, $\widehat{\Delta y}$ are the linear-scaled and log-spaced coordinates, respectively.

| | Trained on | Tested/Finetuned on | | | | Finetuned on | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| method | ImageNet* | ImageNet† | | | | COCO | | ADE20k | | |
| | W8, I256 top-1 acc | W12, I384 top-1 acc | W16, I512 top-1 acc | W20, I640 top-1 acc | W24, I768 top-1 acc | W16 $AP^{box}$ | W32 $AP^{box}$ | W16 mIoU | W20 mIoU | W32 mIoU |
| Parameterized position bias [46] | 81.7 | 79.4/82.7 | 77.2/83.0 | 73.2/83.2 | 68.7/83.2 | 50.8 | 50.9 | 45.5 | 45.8 | 44.5 |
| Linear-Spaced CPB | 81.7 (+0.0) | 82.0/82.9 (+2.6/+0.2) | 81.2/83.3 (+4.0/+0.3) | 79.8/83.6 (+6.6/+0.4) | 77.6/83.7 (+8.9/+0.5) | 50.9 (+0.1) | 51.7 (+0.8) | 47.0 (+1.5) | 47.4 (+1.6) | 47.2 (+2.7) |
| Log-Spaced CPB | 81.8 (+0.1) | 82.4/83.2 (+3.0/+0.5) | 81.7/83.8 (+4.5/+0.8) | 80.4/84.0 (+7.2/+0.8) | 79.1/84.2 (+10.4/+1.0) | 51.1 (+0.3) | 51.8 (+0.9) | 47.0 (+1.5) | 47.7 (+1.9) | 47.8 (+3.3) |

# Swin Transformer v2

## Self-Supervised Pre-training

- Prior work used JFT-3B
- Use SimMIM to train 3B-size Swin Transformer

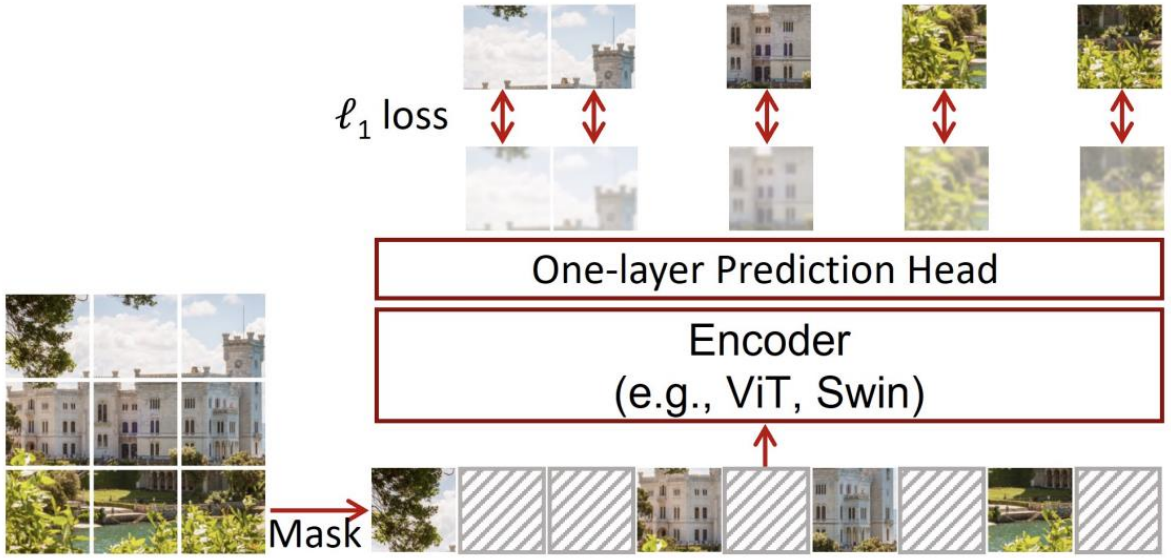



Figure 1. An illustration of our simple framework for masked language modeling, named *SimMIM*. It predicts raw pixel values of the randomly masked patches by a lightweight one-layer head, and performs learning using a simple $\ell_1$ loss.

From SimMIM paper.

| Method | param | pre-train images | pre-train length (#im) | pre-train im size | pre-train time | fine-tune im size | ImageNet-1K-V1 top-1 acc | ImaegNet-1K-V2 top-1 acc |
|---|---|---|---|---|---|---|---|---|
| SwinV1-B | 88M | IN-22K-14M | 1.3B | $224^2$ | $<30^\dagger$ | $384^2$ | 86.4 | 76.58 |
| SwinV1-L | 197M | IN-22K-14M | 1.3B | $224^2$ | $<10^\dagger$ | $384^2$ | 87.3 | 77.46 |
| ViT-G [80] | 1.8B | JFT-3B | 164B | $224^2$ | >30k | $518^2$ | 90.45 | 83.33 |
| V-MoE [56] | 14.7B* | JFT-3B | - | $224^2$ | 16.8k | $518^2$ | 90.35 | - |
| CoAtNet-7 [17] | 2.44B | JFT-3B | - | $224^2$ | 20.1k | $512^2$ | **90.88** | - |
| SwinV2-B | 88M | IN-22K-14M | 1.3B | $192^2$ | $<30^\dagger$ | $384^2$ | 87.1 | 78.08 |
| SwinV2-L | 197M | IN-22K-14M | 1.3B | $192^2$ | $<20^\dagger$ | $384^2$ | 87.7 | 78.31 |
| SwinV2-G | 3.0B | IN-22K-ext-70M | 3.5B | $192^2$ | $<0.5k^\dagger$ | $640^2$ | 90.17 | **84.00** |

Table 2. Comparison with previous largest vision models on ImageNet-1K V1 and V2 classification. * indicates the sparse model; the "pre-train time" column is measured by the TPUv3 core days with numbers copied from the original papers. † That of SwinV2-G is estimated according to training iterations and FLOPs.

| Method | train I(W) size | test I(W) size | views | top-1 |
|---|---|---|---|---|
| ViViT [2] | -(-) | -(-) | 4×3 | 84.8 |
| SwinV1-L [47] | $480(12)^2 \times 16(8)$ | $480(12)^2 \times 16(8)$ | 10×5 | 84.9 |
| TokenLearner [57] | $256(8)^2 \times 64(64)$ | $256(8)^2 \times 64(64)$ | 4×3 | 85.4 |
| Video-SwinV2-G | $320(20)^2 \times 8(8)$ | $320(20)^2 \times 8(8)$ | 1×1 | 83.2 |
| | | $384(24)^2 \times 8(8)$ | 1×1 | 83.4 |
| | | $384(24)^2 \times 8(8)$ | 4×5 | **86.8** |

Table 5. Comparison with previous best results on Kinetics-400 video action classification.

2022

## A ConvNet for the 2020s

Zhuang Liu[1,2*] Hanzi Mao[1] Chao-Yuan Wu[1] Christoph Feichtenhofer[1] Trevor Darrell[2] Saining Xie[1†]
[1]Facebook AI Research (FAIR) [2]UC Berkeley

Code: https://github.com/facebookresearch/ConvNeXt

# ConvNeXt



2023

## ConvNeXt V2: Co-designing and Scaling ConvNets with Masked Autoencoders

Sanghyun Woo[1*] Shoubhik Debnath[2] Ronghang Hu[2]
Xinlei Chen[2] Zhuang Liu[2] In So Kweon[1] Saining Xie[3†]
[1]KAIST [2] Meta AI, FAIR [3]New York University

Code: https://github.com/facebookresearch/ConvNeXt-V2

# ConvNeXt v1

- "Modernized" ResNet

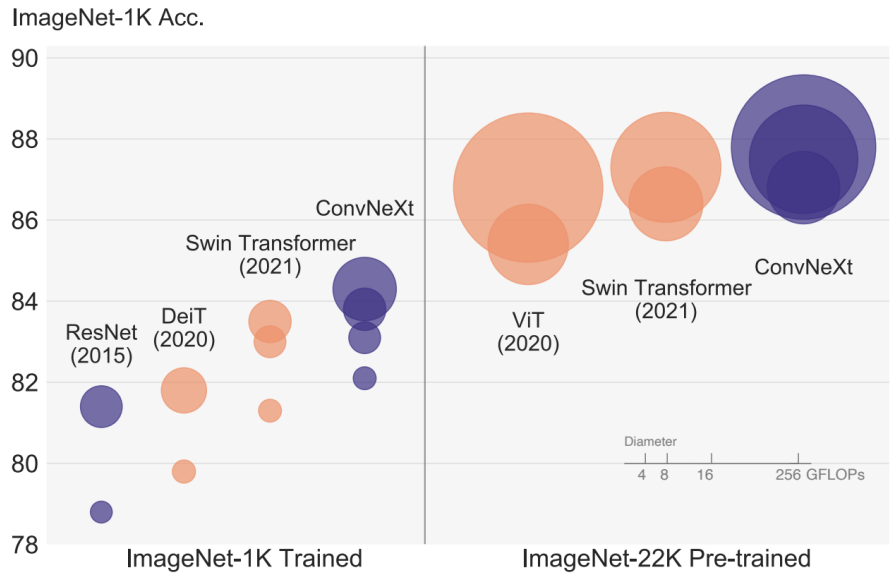
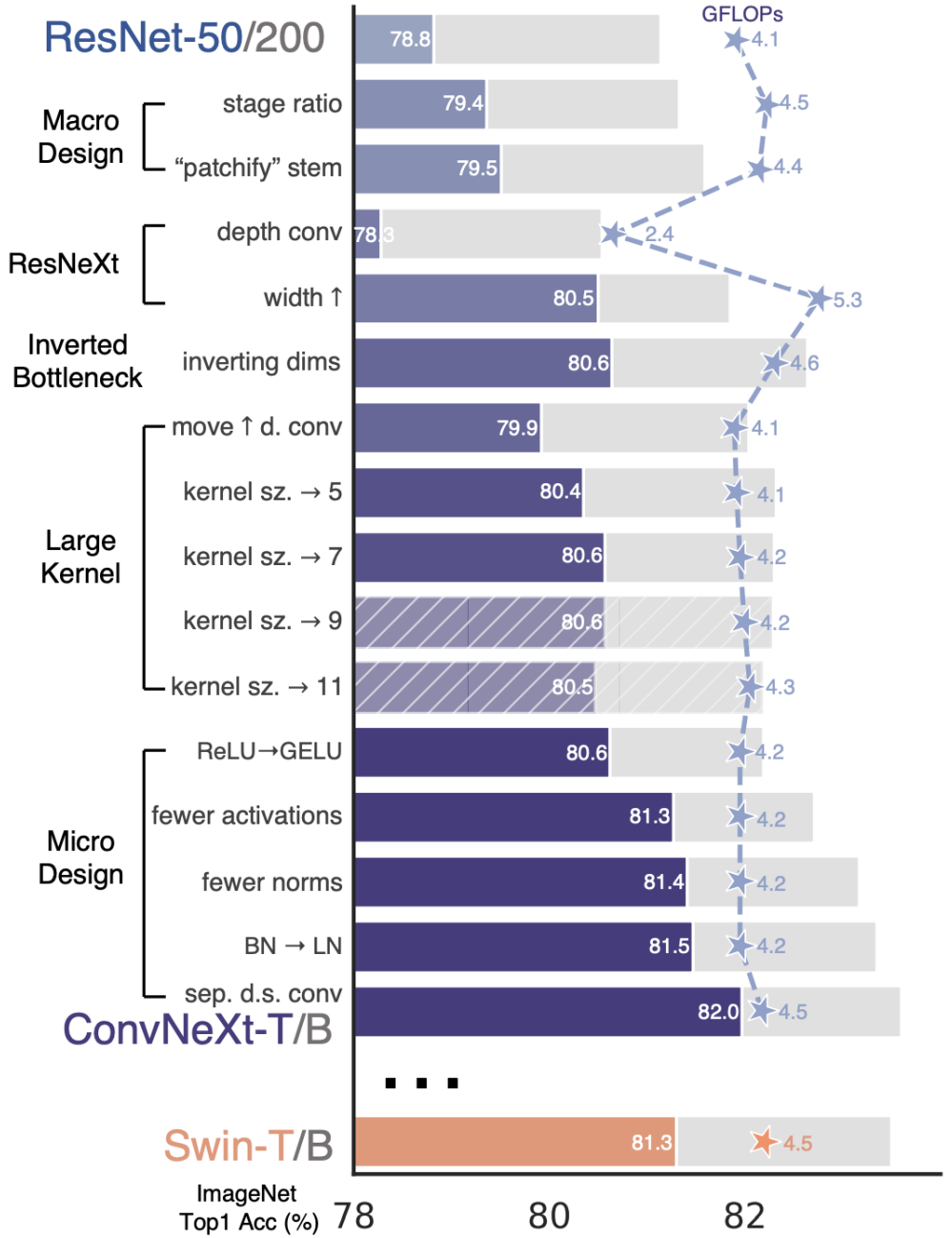
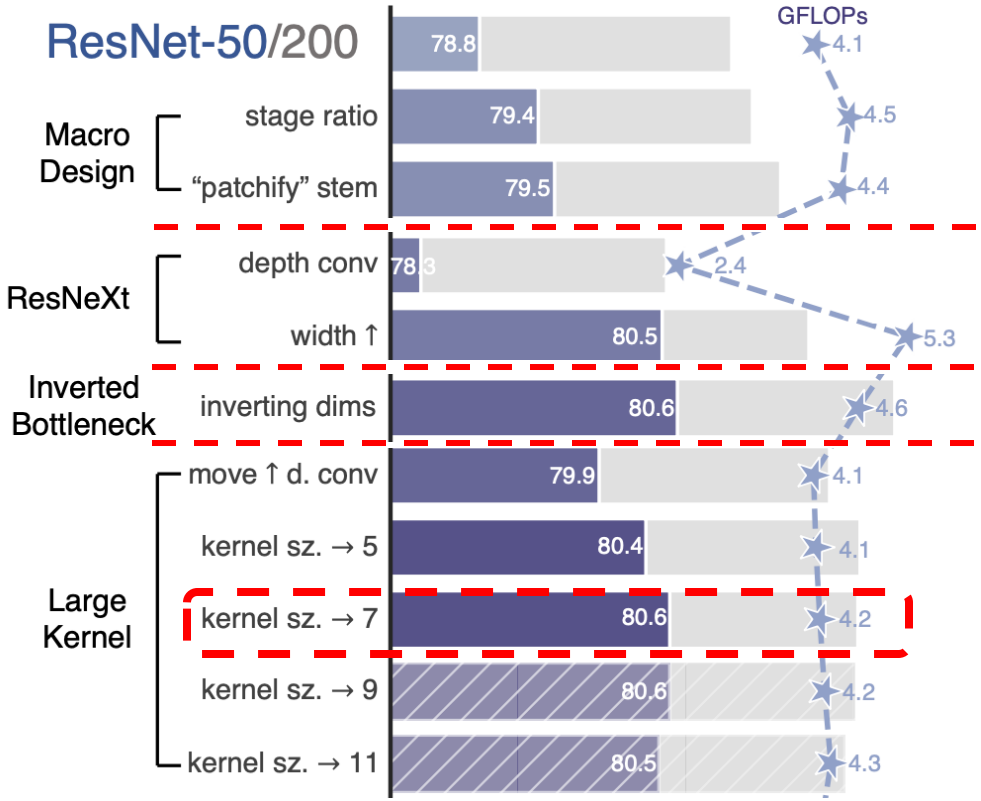



Figure 1. **ImageNet-1K classification** results for • ConvNets and ○ vision Transformers. Each bubble's area is proportional to FLOPs of a variant in a model family. ImageNet-1K/22K models here take $224^2$/$384^2$ images respectively. ResNet and ViT results were obtained with improved training procedures over the original papers. We demonstrate that a standard ConvNet model can achieve the same level of scalability as hierarchical vision Transformers while being much simpler in design.






# of blocks in each stage from [3,4,6,3] to [3,3,9,3]

7x7 conv with stride 2 + max-pool (4x downsamp) => 4x4 conv with stride 4

# of channels increased to match Swin-T

Increase embedding dim

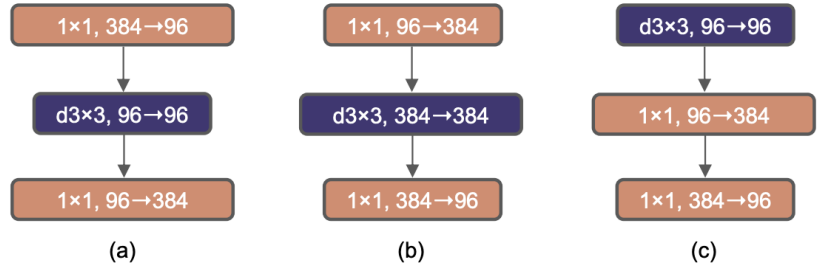
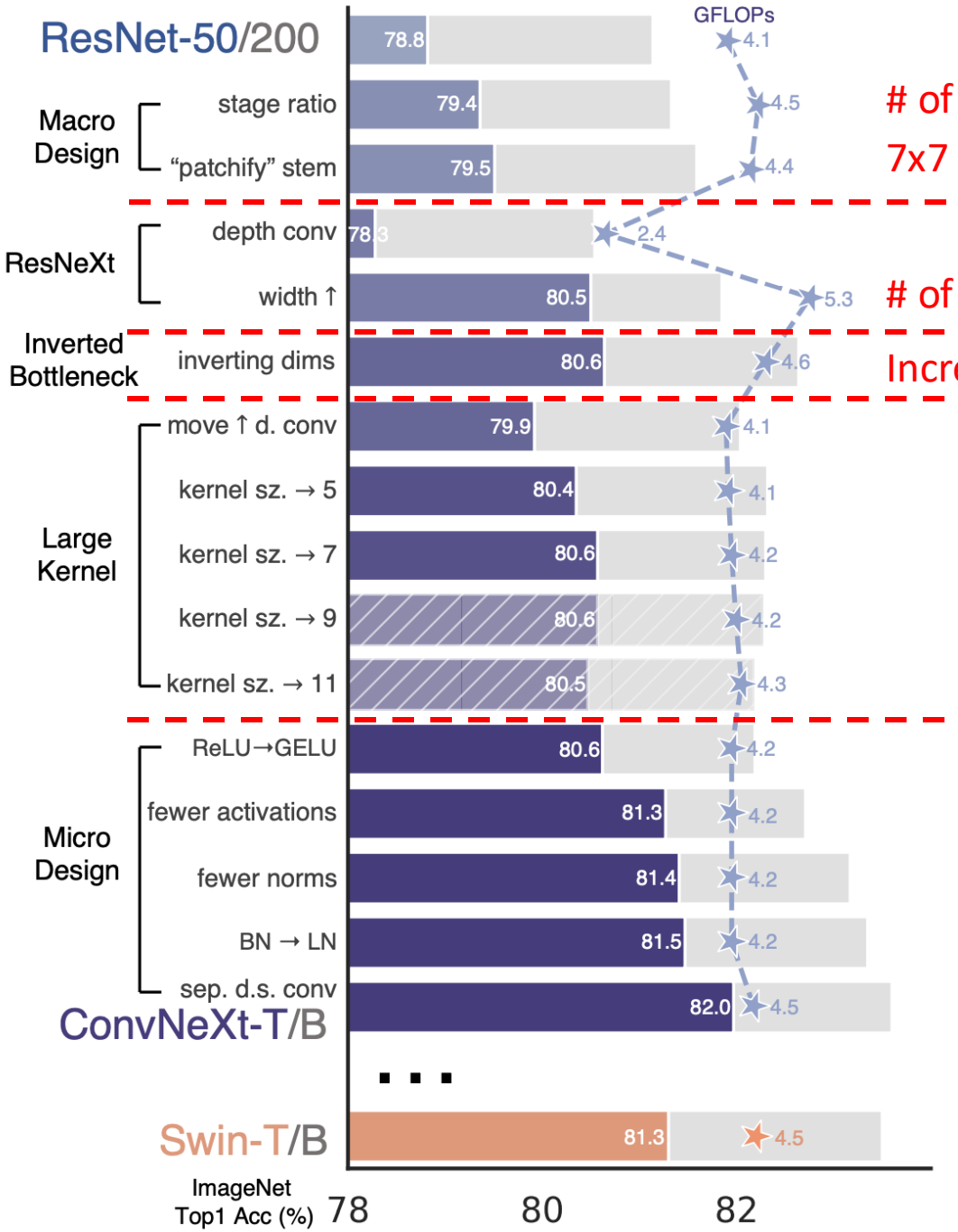



Figure 3. **Block modifications and resulted specifications. (a)** is a ResNeXt block; in **(b)** we create an inverted bottleneck block and in **(c)** the position of the spatial depthwise conv layer is moved up.

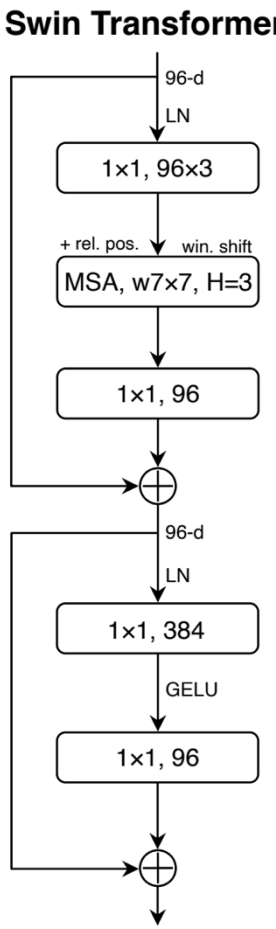


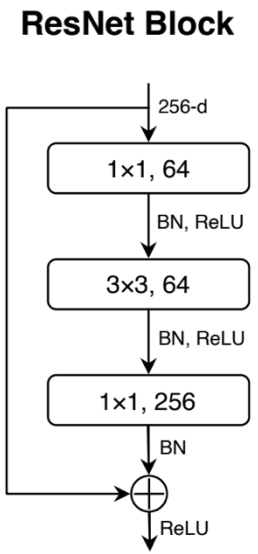

# of blocks in each stage from [3,4,6,3] to [3,3,9,3]

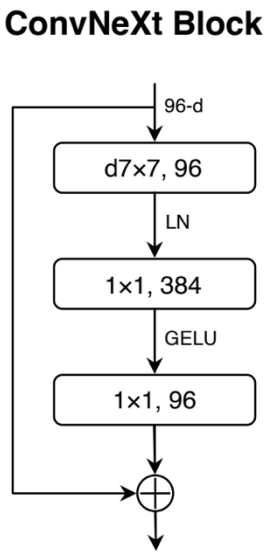
7x7 conv with stride 2 + max-pool (4x downsamp) => 4x4 conv with stride 4

# of channels increased to match Swin-T

Increase embedding dim



# ConvNeXt v2

- Training ConvNeXt v1 with Masked AE performs poorly
- Introduce
  - fully convolutional masked autoencoder framework and
  - a new Global Response Normalization to enhance inter-channel competition

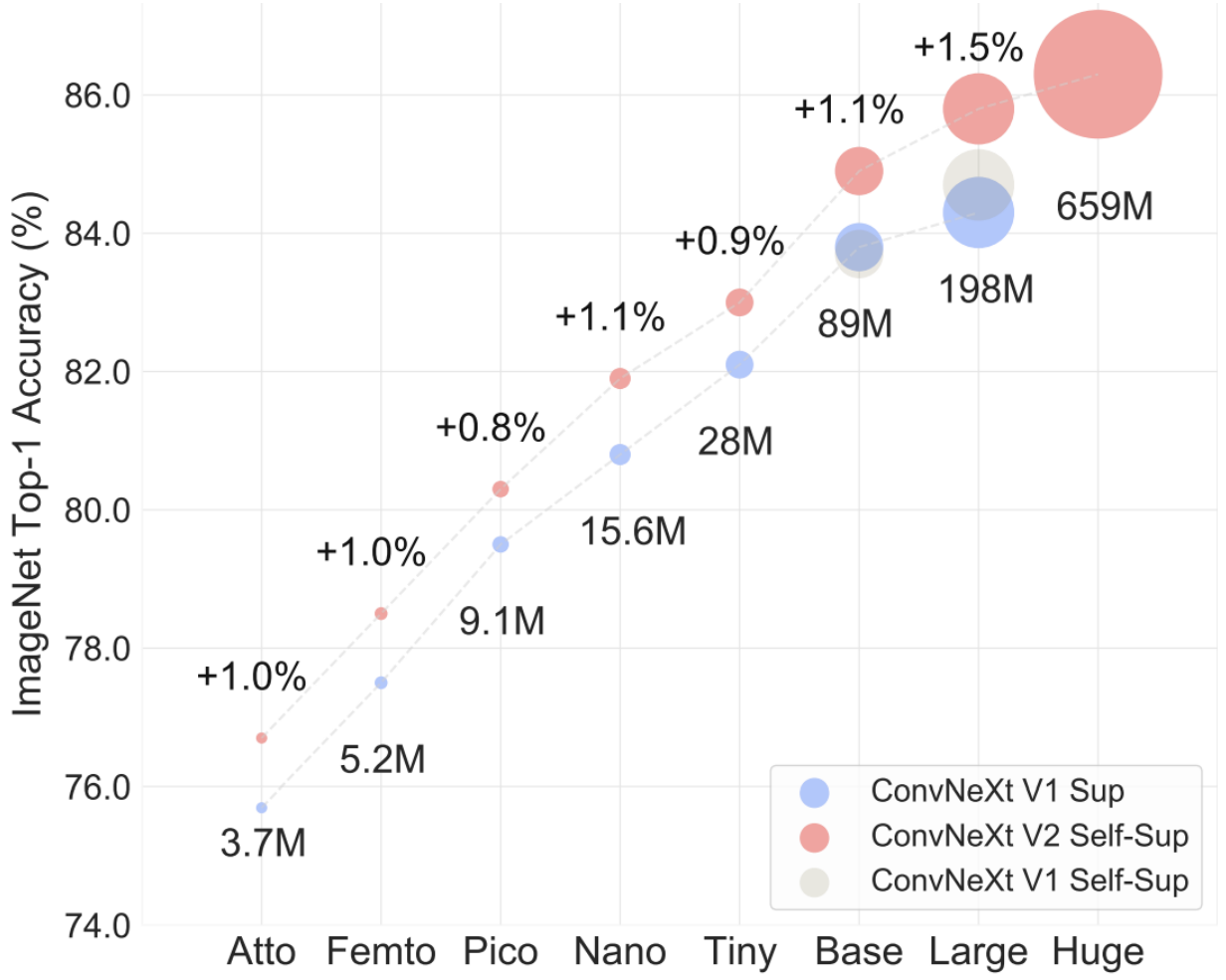



Figure 1. **ConvNeXt V2 model scaling**. The ConvNeXt V2 model, which has been pre-trained using our fully convolutional masked autoencoder framework, performs significantly better than the previous version across a wide range of model sizes.

# ConvNeXt v2

- Training ConvNext v1 with Masked AE performs poorly
- Introduce
  - fully convolutional masked autoencoder framework and
  - a new Global Response Normalization to enhance inter-channel competition

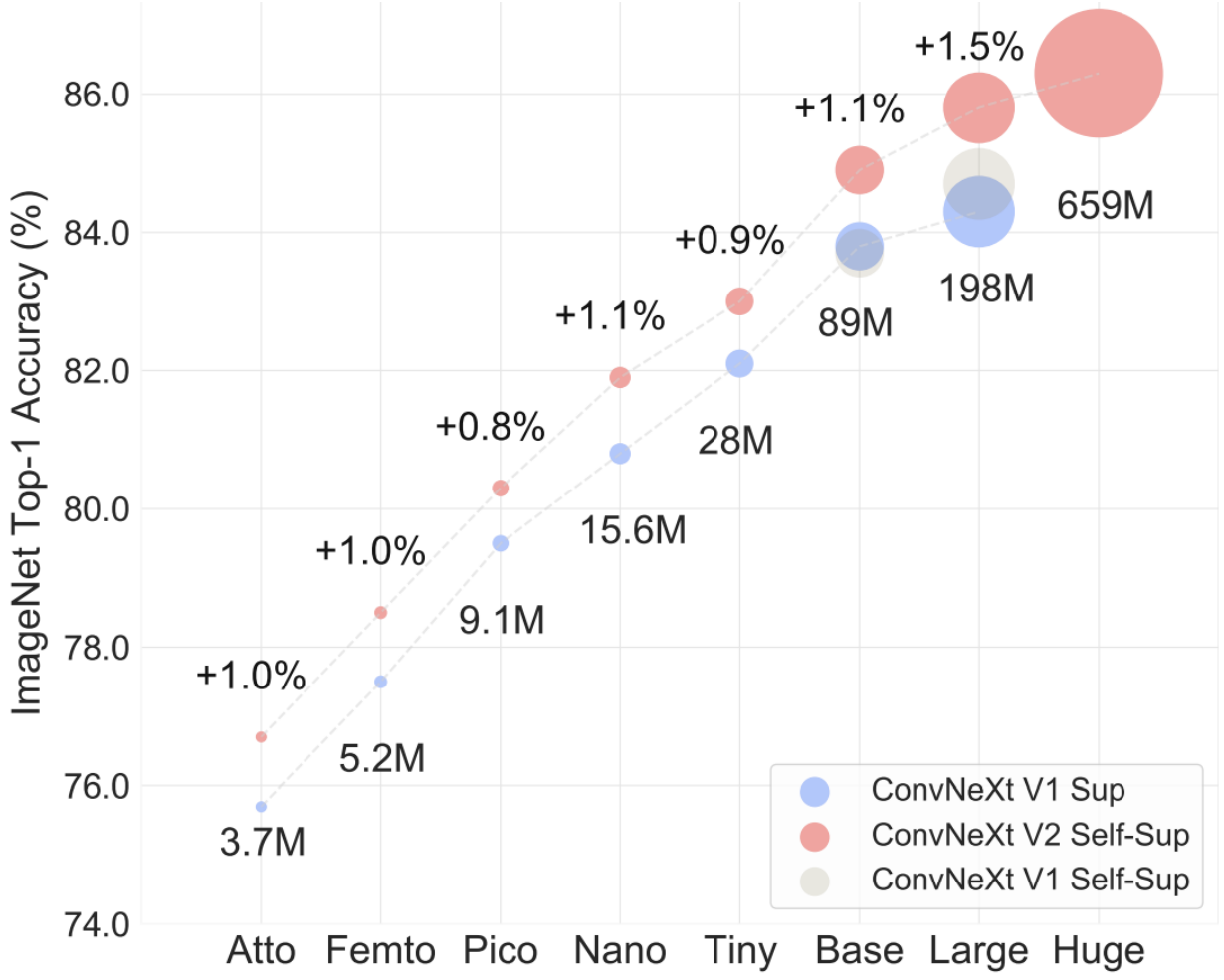



Figure 1. **ConvNeXt V2 model scaling**. The ConvNeXt V2 model, which has been pre-trained using our fully convolutional masked autoencoder framework, performs significantly better than the previous version across a wide range of model sizes.

# ConvNeXt v2

Fully Convolutional Masked Autoencoder

- Encoder:
  - Sparse conv to prevent simple copy-paste of input (not a problem in MAE with transformers)
  - Consider image as a 2D array of pixels
- Decoder: ConvNeXt block

| w/o Sparse conv. | w/ Sparse conv. |
|---|---|
| 79.3 | 83.7 |

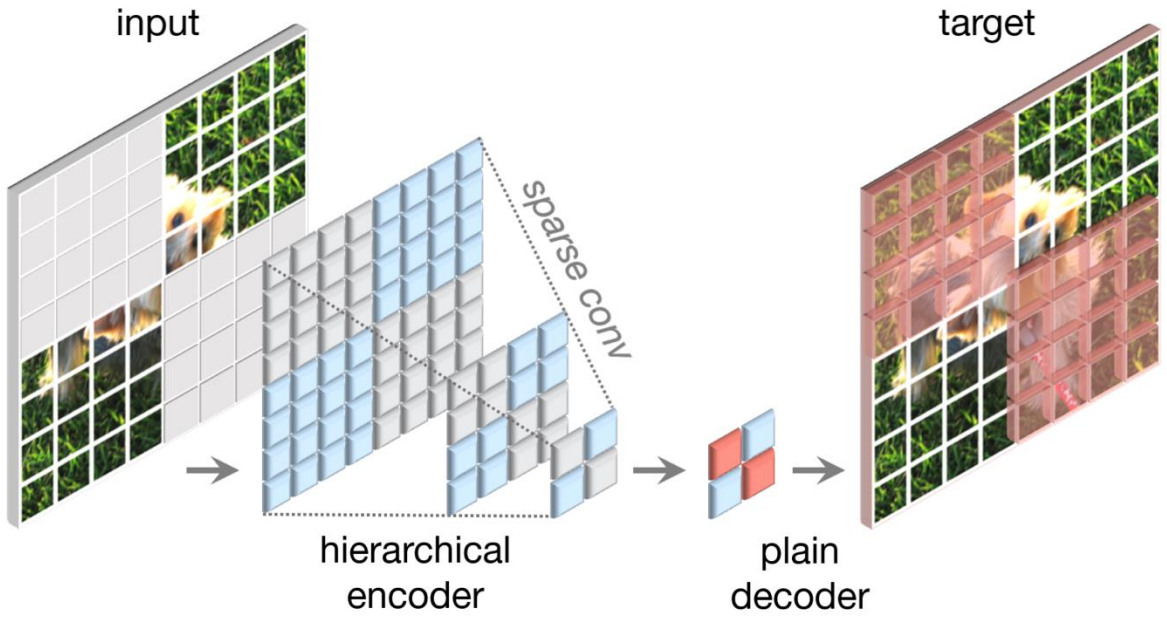
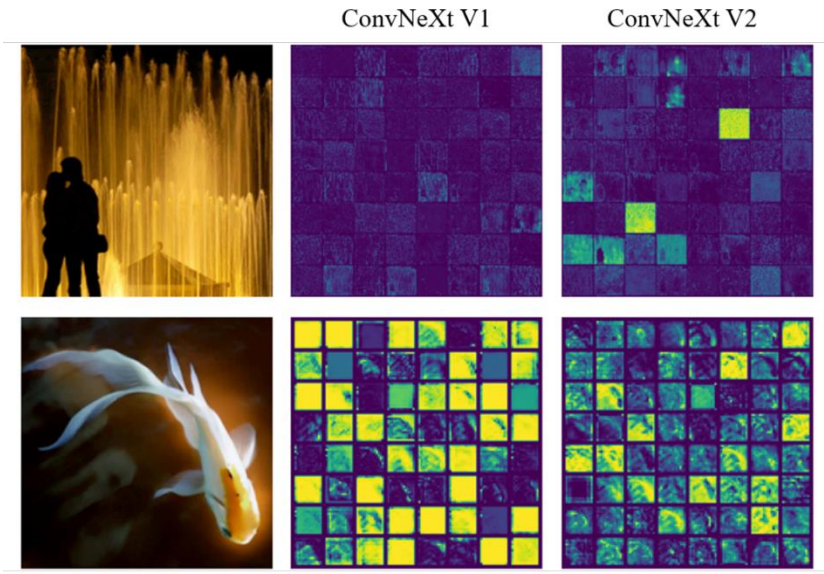


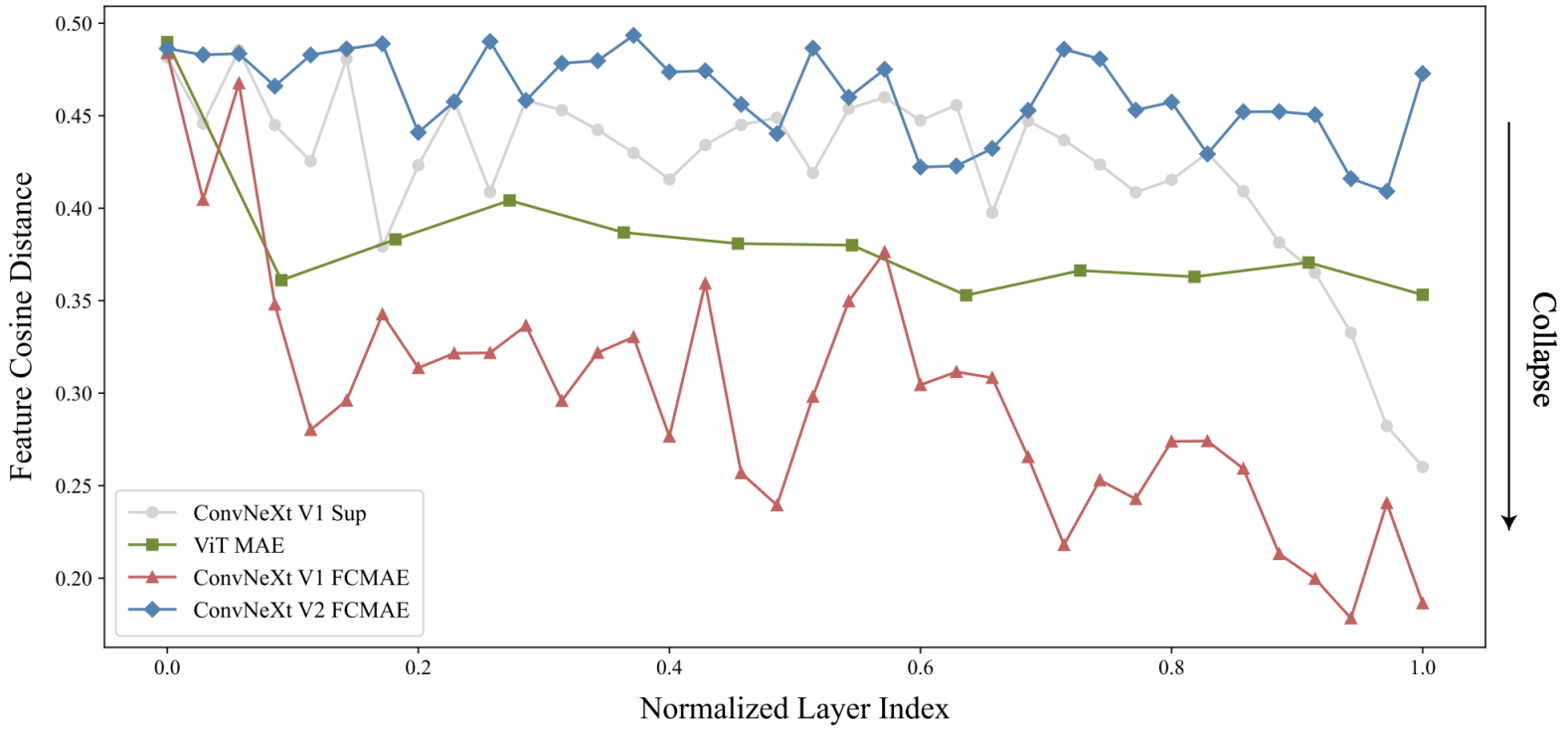

Figure 2. **Our FCMAE framework.** We introduce a fully convolutional masked autoencoder (FCMAE). It consists of a sparse convolution-based ConvNeXt encoder and a lightweight ConvNeXt block decoder. Overall, the architecture of our autoencoder is asymmetric. The encoder processes only the visible pixels, and the decoder reconstructs the image using the encoded pixels and mask tokens. The loss is calculated only on the masked region.

# ConvNeXt v2

Global Response Normalization

- Feature collapse: many dead/saturated feature maps



# ConvNeXt v2

Global Response Normalization

- Inspiration from lateral inhibition in brain
- Three steps:
  1. global feature aggregation,
  2. feature normalization, and
  3. feature calibration.

First, we aggregate a spatial feature map $X_i$ into a vector $gx$ with a global function $\mathcal{G}(\cdot)$:

$$\mathcal{G}(X) := X \in \mathcal{R}^{H\times W\times C} \rightarrow gx \in \mathcal{R}^{C}. \quad (1)$$

Next, we apply a response normalization function $\mathcal{N}(\cdot)$ to the aggregated values. Concretely, we use a standard divisive normalization as follows,

$$\mathcal{N}(||X_i||) := ||X_i|| \in \mathcal{R} \rightarrow \frac{||X_i||}{\sum_{j=1,\ldots,C} ||X_j||} \in \mathcal{R}, \quad (2)$$

where $||X_i||$ is the L2-norm of the $i$-th channel. [1] Intu-

Finally, we calibrate the original input responses using the computed feature normalization scores:

$$X_i = X_i * \mathcal{N}(\mathcal{G}(X)_i) \in \mathcal{R}^{H\times W} \quad (3)$$

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

```
# gamma, beta: learnable affine transform parameters
# X: input of shape (N,H,W,C)

gx = torch.norm(X, p=2, dim=(1,2), keepdim=True)
nx = gx / (gx.mean(dim=-1, keepdim=True)+1e-6)
return gamma * (X * nx) + beta + X
```

# ConvNeXt v2

## Global Response Normalization

| case | ft |
|---|---|
| g.avg. | 83.7 |
| L1 | 84.3 |
| L2 | **84.6** |

(a) **Global aggregation** $G(\cdot)$. L2 Norm-based aggregation function produces the best result.

| case | ft |
|---|---|
| $(\|\|X_i\|\| - \mu)/\sigma$ | 84.5 |
| $1/\sum \|\|X_i\|\|$ | 83.8 |
| $\|\|X_i\|\|/\sum \|\|X_i\|\|$ | **84.6** |

(b) **Normalization operator,** $N(\cdot)$. Divisive normalization is an effective channel importance calibrator.

| case | ft |
|---|---|
| w/o skip | 84.0 |
| w/ skip | **84.6** |

(c) **Residual connection** helps with GRN optimization and leads to better performance.

| case | ft |
|---|---|
| Baseline | 83.7 |
| LRN [26] | 83.2 |
| BN [22] | 80.5 |
| LN [1] | 83.8 |
| GRN | **84.6** |

(d) **Feature normalization**. GRN outperforms other normalizations through global contrasting.

| case | ft | #param |
|---|---|---|
| Baseline | 83.7 | 89M |
| SE [19] | 84.4 | 109M |
| CBAM [48] | 84.5 | 109M |
| GRN | **84.6** | **89M** |

(e) **Feature re-weighting**. GRN does effective and efficient feature re-weighting without parameter overhead.

| case | ft |
|---|---|
| Baseline | 83.7 |
| drop at ft. | 78.8 |
| add at ft. | 80.6 |
| both | **84.6** |

(f) **GRN in pre-training/fine-tuning**. To be effective, GRN should be used in both stages.

Table 2. **GRN ablations** with ConvNeXt-Base. We report fine-tuning accuracy on ImageNet-1K. Our final proposal is marked in gray.

# ConvNeXt v2

| Backbone | Method | #param | FLOPs | Val acc. |
|---|---|---|---|---|
| ConvNeXt V1-B | Supervised | 89M | 15.4G | 83.8 |
| ConvNeXt V1-B | FCMAE | 89M | 15.4G | 83.7 |
| ConvNeXt V2-B | Supervised | 89M | 15.4G | 84.3 (+0.5) |
| ConvNeXt V2-B | FCMAE | 89M | 15.4G | **84.6** (+0.8) |
| ConvNeXt V1-L | Supervised | 198M | 34.4G | 84.3 |
| ConvNeXt V1-L | FCMAE | 198M | 34.4G | 84.4 |
| ConvNeXt V2-L | Supervised | 198M | 34.4G | 84.5 (+0.2) |
| ConvNeXt V2-L | FCMAE | 198M | 34.4G | **85.6** (+1.3) |

Table 3. **Co-design matters**. When the architecture and the learning framework are co-designed and used together, masked image pre-training becomes effective for ConvNeXt. We report the fine-tuning performance from 800 epoch FCMAE pre-trained models. The relative improvement is bigger with a larger model.

| Backbone | Method | #param | PT epoch | FT acc. |
|---|---|---|---|---|
| ViT-B | BEiT | 88M | 800 | 83.2 |
| ViT-B | MAE | 88M | 1600 | 83.6 |
| Swin-B | SimMIM | 88M | 800 | 84.0 |
| ConvNeXt V2-B | FCMAE | 89M | 800 | 84.6 |
| ConvNeXt V2-B | FCMAE | 89M | 1600 | **84.9** |
| ViT-L | BEiT | 307M | 800 | 85.2 |
| ViT-L | MAE | 307M | 1600 | 85.9 |
| Swin-L | SimMIM | 197M | 800 | 85.4 |
| ConvNeXt V2-L | FCMAE | 198M | 800 | 85.6 |
| ConvNeXt V2-L | FCMAE | 198M | 1600 | **85.8** |
| ViT-H | MAE | 632M | 1600 | 86.9 |
| Swin V2-H | SimMIM | 658M | 800 | 85.7 |
| ConvNeXt V2-H | FCMAE | 659M | 800 | 85.8 |
| ConvNeXt V2-H | FCMAE | 659M | 1600 | **86.3** |

Table 4. **Comparisons with previous masked image modeling approaches**. The pre-training data is the IN-1K training set. All self-supervised methods are benchmarked by the end-to-end fine-tuning performance with an image size of 224. We underline the highest accuracy for each model size and bold our best results.

2023

**FastViT: A Fast Hybrid Vision Transformer using Structural Reparameterization**

Pavan Kumar Anasosalu Vasu† James Gabriel Jeff Zhu Oncel Tuzel Anurag Ranjan†

Apple

Published as a conference paper at ICLR 2024

**FASTERVIT: FAST VISION TRANSFORMERS WITH HIERARCHICAL ATTENTION**

**Ali Hatamizadeh, Greg Heinrich, Hongxu Yin, Andrew Tao, Jose M. Alvarez, Jan Kautz, Pavlo Molchanov**
NVIDIA
{ahatamizadeh, pmolchanov}@nvidia.com

# Fast/Faster ViTs

# Fast ViT

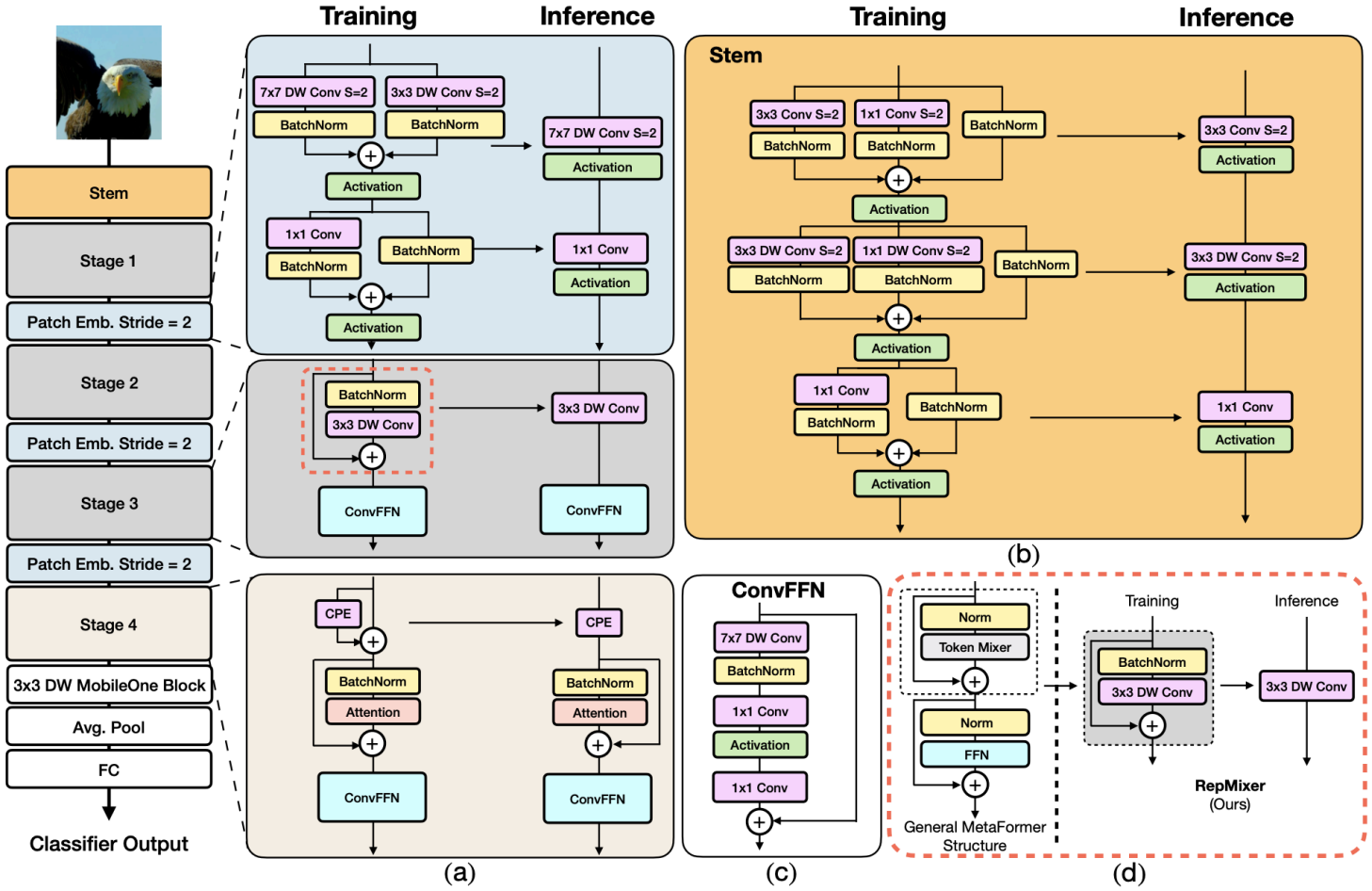
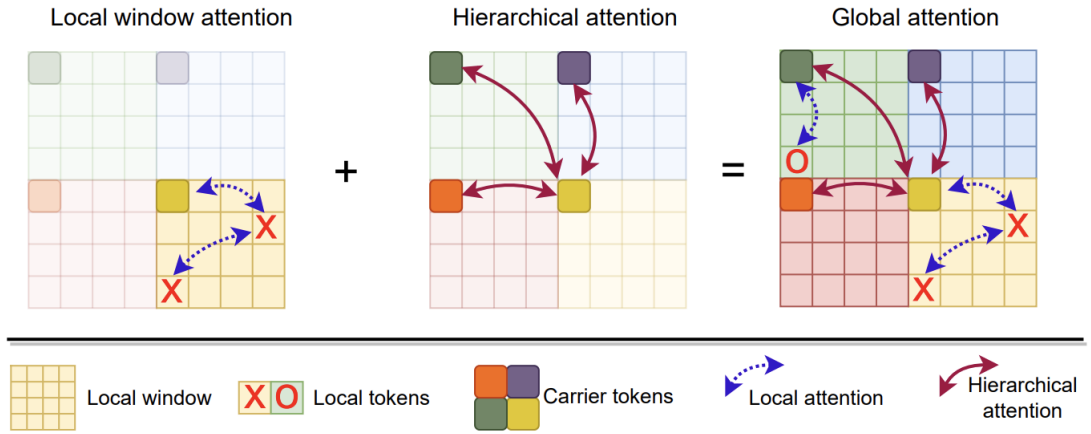



Figure 2: (a) Overview of FastViT architecture which decouples train-time and inference-time architecture. Stages 1, 2, and 3 have the same architecture and uses RepMixer for token mixing. In stage 4, self attention layers are used for token mixing. (b) Architecture of the convolutional stem. (c) Architecture of convolutional-FFN (d) Overview of *RepMixer* block, which reparameterizes a skip connection at inference.

# Faster ViT

Figure 2: Visualization of the proposed Hierarchical Attention in the feature space. By performing local window attention and hierarchical attention we can achieve global information propagation at reduced costs.



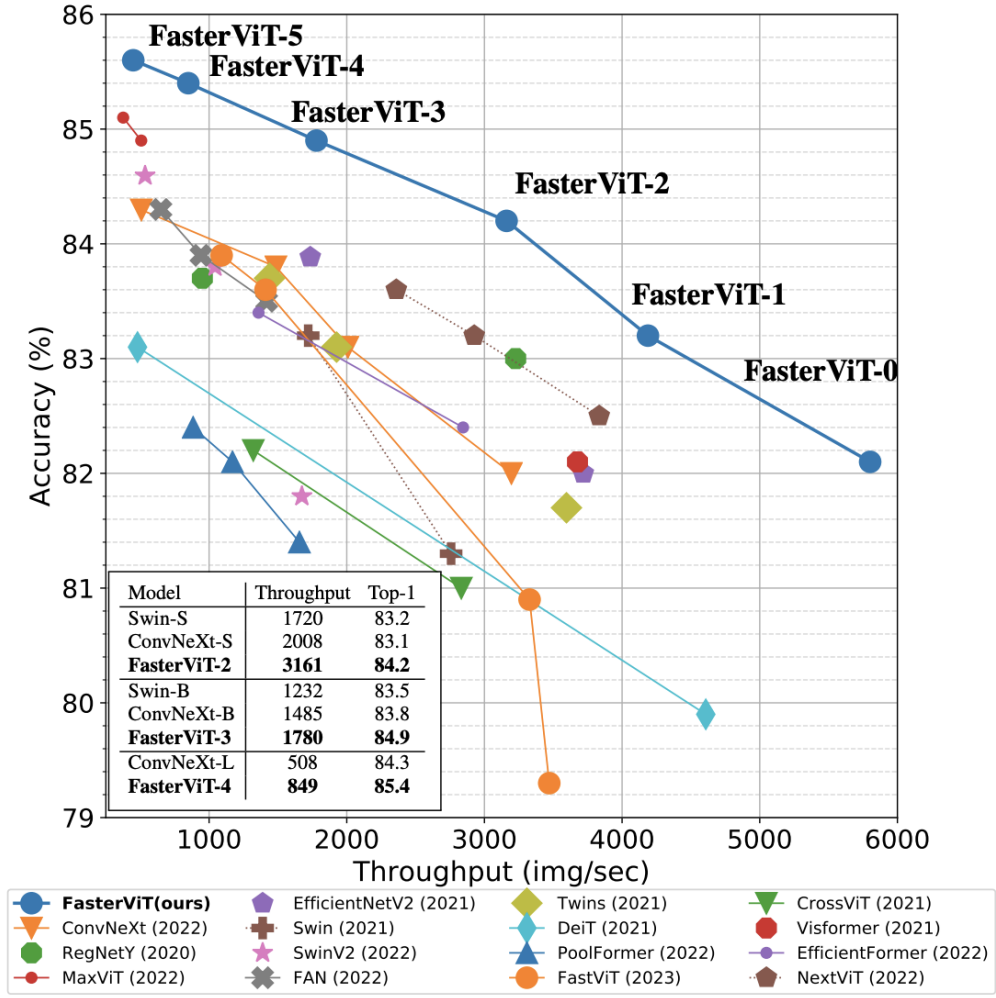




Figure 1: Comparison of image throughput and ImageNet-1K Top-1 accuracy. Throughput is measured on A100 GPU with batch size of 128.

# Pretraining Vision Transformers

# Masked Autoencoders

**Masked Autoencoders Are Scalable Vision Learners**

Kaiming He*,† Xinlei Chen* Saining Xie Yanghao Li Piotr Dollár Ross Girshick
*equal technical contribution †project lead
Facebook AI Research (FAIR)

CVPR 2022

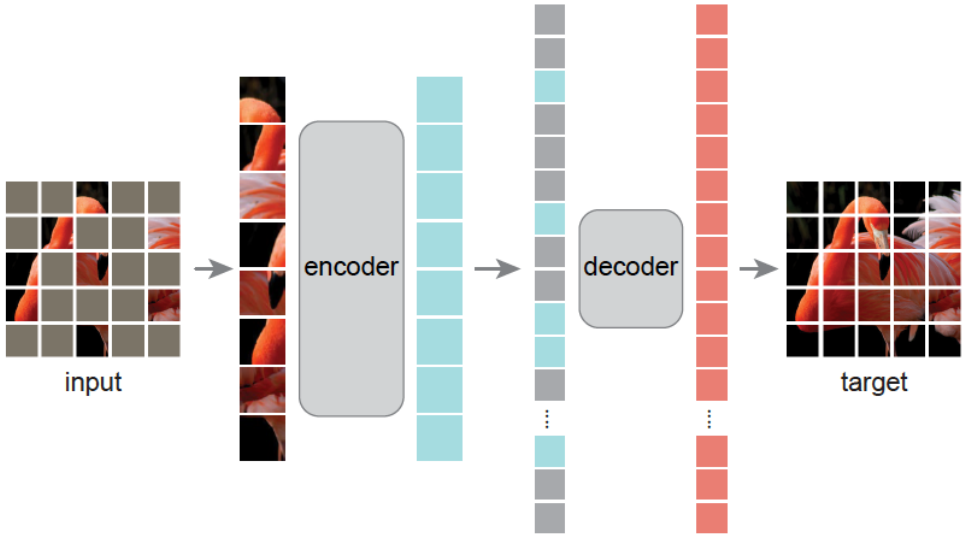



Figure 1. **Our MAE architecture**. During pre-training, a large random subset of image patches (*e.g.*, 75%) is masked out. The encoder is applied to the small subset of *visible patches*. Mask tokens are introduced *after* the encoder, and the full set of encoded patches and mask tokens is processed by a small decoder that reconstructs the original image in pixels. After pre-training, the decoder is discarded and the encoder is applied to uncorrupted images (full sets of patches) for recognition tasks.

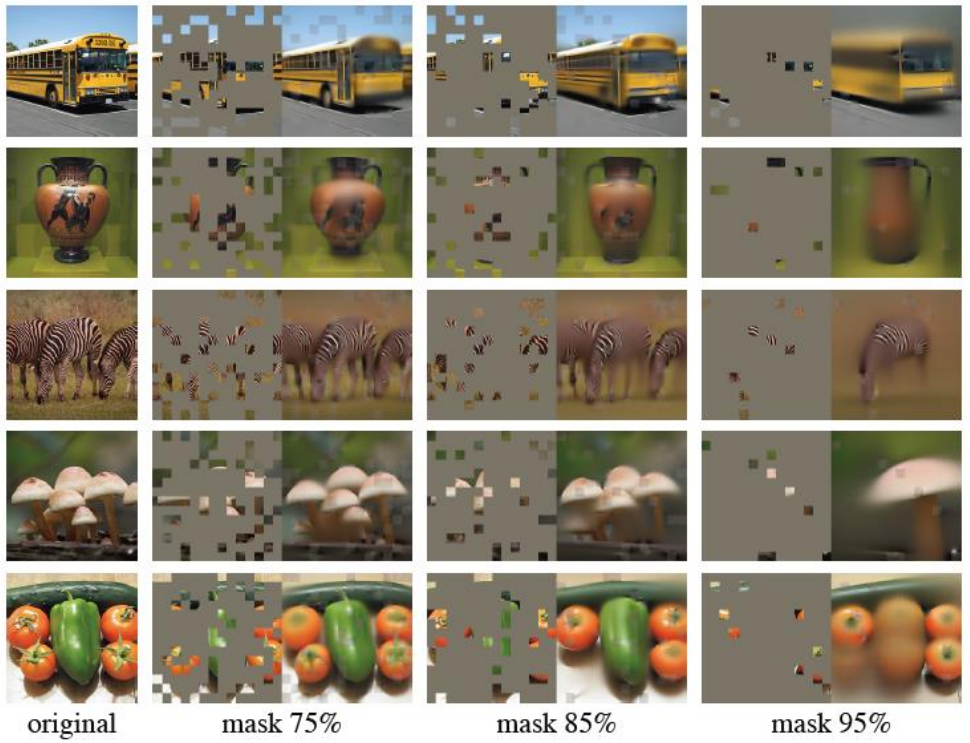



Figure 4. Reconstructions of ImageNet *validation* images using an MAE pre-trained with a masking ratio of 75% but applied on inputs with higher masking ratios. The predictions differ plausibly from the original images, showing that the method can generalize.

MSE loss between pixels for masked tokens only!

# Masked Autoencoders

**Masked Autoencoders Are Scalable Vision Learners**

Kaiming He*,† Xinlei Chen* Saining Xie Yanghao Li Piotr Dollár Ross Girshick

*equal technical contribution †project lead

Facebook AI Research (FAIR)

CVPR 2022

| blocks | ft | lin |
|---|---|---|
| 1 | 84.8 | 65.5 |
| 2 | **84.9** | 70.0 |
| 4 | **84.9** | 71.9 |
| 8 | **84.9** | **73.5** |
| 12 | 84.4 | 73.3 |

(a) **Decoder depth**. A deep decoder can improve linear probing accuracy.

| dim | ft | lin |
|---|---|---|
| 128 | **84.9** | 69.1 |
| 256 | 84.8 | 71.3 |
| 512 | **84.9** | **73.5** |
| 768 | 84.4 | 73.1 |
| 1024 | 84.3 | 73.1 |

(b) **Decoder width**. The decoder can be narrower than the encoder (1024-d).

| case | ft | lin | FLOPs |
|---|---|---|---|
| encoder w/ `[M]` | 84.2 | 59.6 | 3.3× |
| encoder w/o `[M]` | **84.9** | **73.5** | **1**× |

(c) **Mask token**. An encoder without mask tokens is more accurate and faster (Table 2).

| case | ft | lin |
|---|---|---|
| pixel (w/o norm) | 84.9 | 73.5 |
| pixel (w/ norm) | **85.4** | **73.9** |
| PCA | 84.6 | 72.3 |
| dVAE token | 85.3 | 71.6 |

(d) **Reconstruction target**. Pixels as reconstruction targets are effective.

| case | ft | lin |
|---|---|---|
| none | 84.0 | 65.7 |
| crop, fixed size | 84.7 | 73.1 |
| crop, rand size | **84.9** | **73.5** |
| crop + color jit | 84.3 | 71.9 |

(e) **Data augmentation**. Our MAE works with minimal or no augmentation.

| case | ratio | ft | lin |
|---|---|---|---|
| random | 75 | **84.9** | **73.5** |
| block | 50 | 83.9 | 72.3 |
| block | 75 | 82.8 | 63.9 |
| grid | 75 | 84.0 | 66.0 |

(f) **Mask sampling**. Random sampling works the best. See Figure 6 for visualizations.

Table 1. **MAE ablation experiments** with ViT-L/16 on ImageNet-1K. We report fine-tuning (ft) and linear probing (lin) accuracy (%). If not specified, the default is: the decoder has depth 8 and width 512, the reconstruction target is unnormalized pixels, the data augmentation is random resized cropping, the masking ratio is 75%, and the pre-training length is 800 epochs. Default settings are marked in gray.

# Masked Autoencoders

**Masked Autoencoders Are Scalable Vision Learners**

Kaiming He*,† Xinlei Chen* Saining Xie Yanghao Li Piotr Dollár Ross Girshick

*equal technical contribution †project lead

Facebook AI Research (FAIR)

CVPR 2022

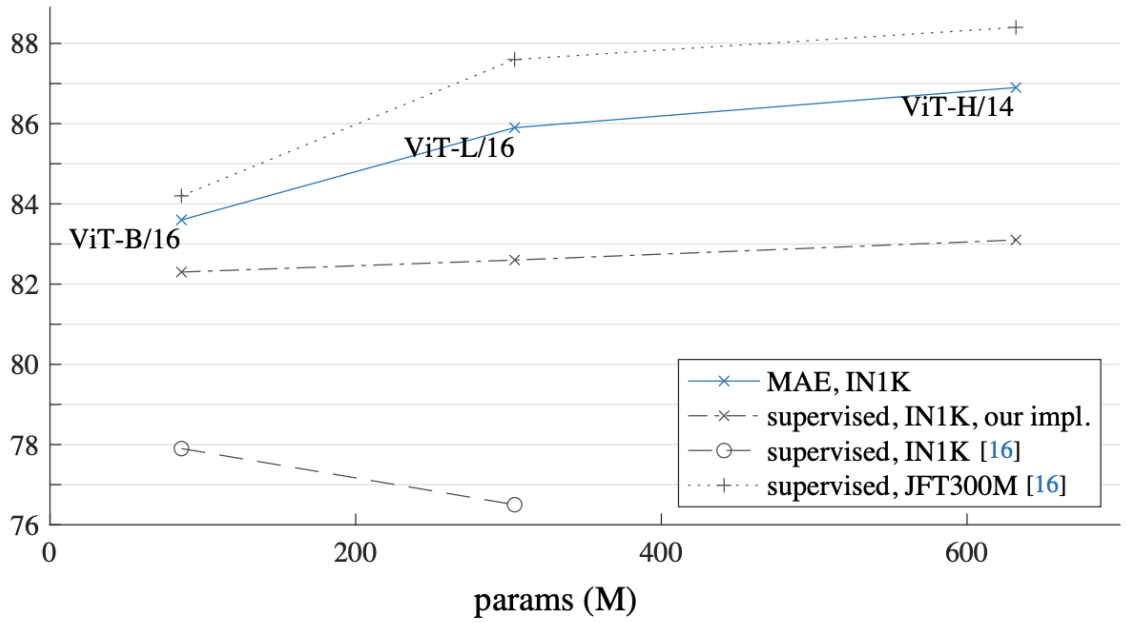



Figure 8. **MAE pre-training *vs*. supervised pre-training**, evaluated by fine-tuning in ImageNet-1K (224 size). We compare with the original ViT results [16] trained in IN1K or JFT300M.

| method | pre-train data | ViT-B | ViT-L | ViT-H | ViT-$H_{448}$ |
|---|---|---|---|---|---|
| scratch, our impl. | - | 82.3 | 82.6 | 83.1 | - |
| DINO [5] | IN1K | 82.8 | - | - | - |
| MoCo v3 [9] | IN1K | 83.2 | 84.1 | - | - |
| BEiT [2] | IN1K+DALLE | 83.2 | 85.2 | - | - |
| MAE | IN1K | 83.6 | 85.9 | 86.9 | **87.8** |

Table 3. **Comparisons with previous results on ImageNet-1K**. The pre-training data is the ImageNet-1K training set (except the tokenizer in BEiT was pre-trained on 250M DALLE data [50]). All self-supervised methods are evaluated by end-to-end fine-tuning. The ViT models are B/16, L/16, H/14 [16]. The best for each column is underlined. All results are on an image size of 224, except for ViT-H with an extra result on 448. Here our MAE reconstructs normalized pixels and is pre-trained for 1600 epochs.

# SimMIM

**_SimMIM_: a Simple Framework for Masked Image Modeling**

CVPR 2022

Zhenda Xie[1*] Zheng Zhang[2*] Yue Cao[2*]
Yutong Lin[3] Jianmin Bao[2] Zhuliang Yao[1] Qi Dai[2] Han Hu[2*]
[1]Tsinghua University [2]Microsoft Research Asia [3]Xi'an Jiaotong University
{t-zhxie,zhez,yuecao,t-yutonglin,jianmin.bao,t-zhuyao,qid,hanhu}@microsoft.com

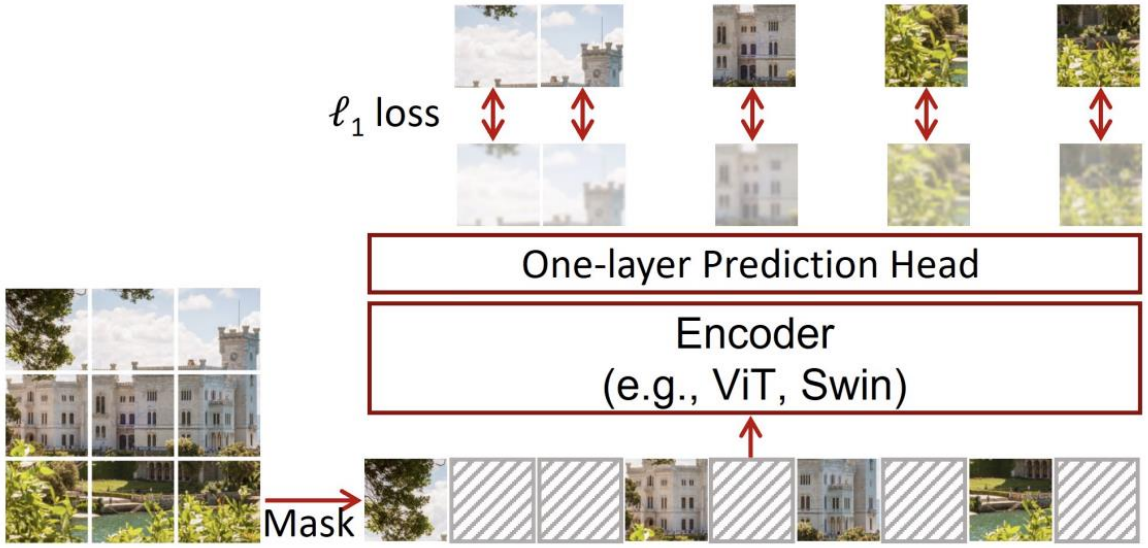



Figure 1. An illustration of our simple framework for masked language modeling, named *SimMIM*. It predicts raw pixel values of the randomly masked patches by a lightweight one-layer head, and performs learning using a simple $\ell_1$ loss.

| Methods | Input Size | Fine-tuning Top-1 acc (%) | Linear eval Top-1 acc (%) | Pre-training costs |
|---|---|---|---|---|
| Sup. baseline [44] | $224^2$ | 81.8 | - | - |
| DINO [5] | $224^2$ | 82.8 | 78.2 | 2.0× |
| MoCo v3 [9] | $224^2$ | 83.2 | 76.7 | 1.8× |
| ViT [15] | $384^2$ | 79.9 | - | ∼4.0× |
| BEiT [1] | $224^2$ | 83.2 | 56.7 | 1.5×† |
| Ours | $224^2$ | **83.8** | 56.7 | 1.0× |

Table 6. System-level comparison using ViT-B as the encoder. Training costs are counted in relative to our approach. † BEiT requires an additional stage to pre-train dVAE, which is not counted.

# DINO v1 & v2 (ICCV'21 & TMLR'24)

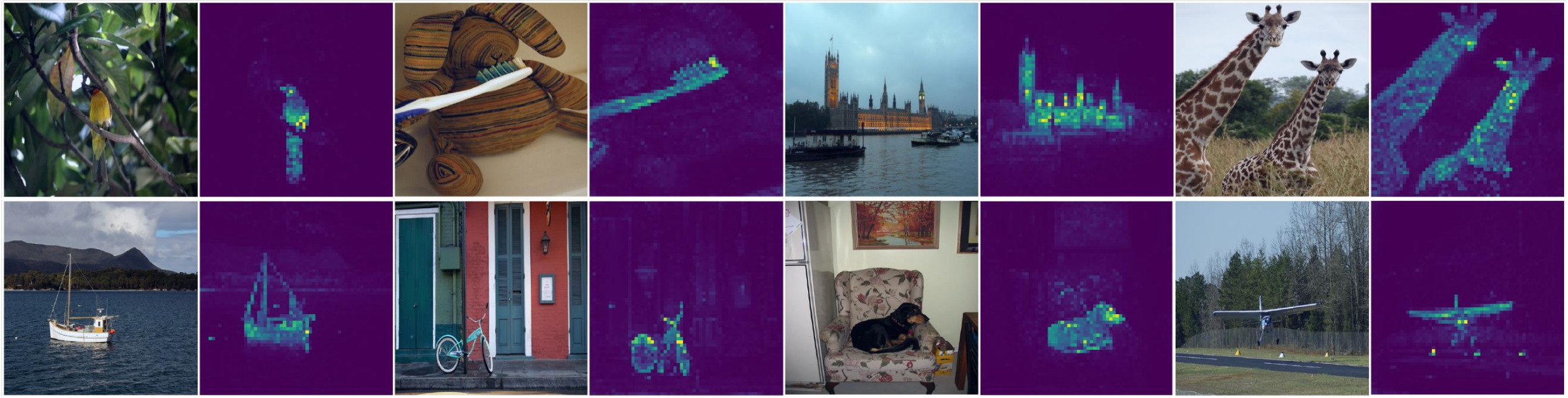


Figure 1: **Self-attention from a Vision Transformer with $8 \times 8$ patches trained with no supervision.** We look at the self-attention of the `[CLS]` token on the heads of the last layer. This token is not attached to any label nor supervision. These maps show that the model automatically learns class-specific features leading to unsupervised object segmentations.

# DINO v1 & v2 (ICCV'21 & TMLR'24)

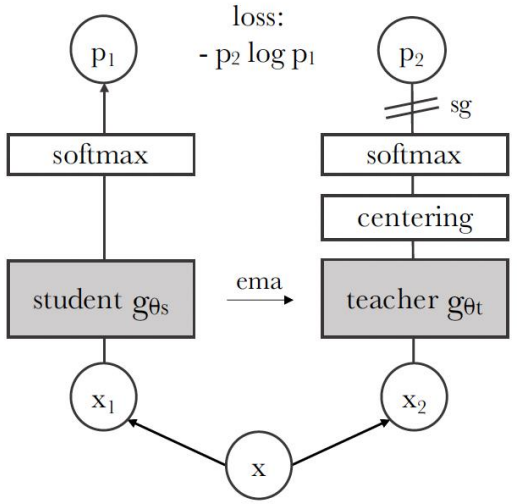



Figure 2: **Self-distillation with no labels.** We illustrate DINO in the case of one single pair of views ($x_1$, $x_2$) for simplicity. The model passes two different random transformations of an input image to the student and teacher networks. Both networks have the same architecture but different parameters. The output of the teacher network is centered with a mean computed over the batch. Each networks outputs a $K$ dimensional feature that is normalized with a temperature softmax over the feature dimension. Their similarity is then measured with a cross-entropy loss. We apply a stop-gradient (sg) operator on the teacher to propagate gradients only through the student. The teacher parameters are updated with an exponential moving average (ema) of the student parameters.

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

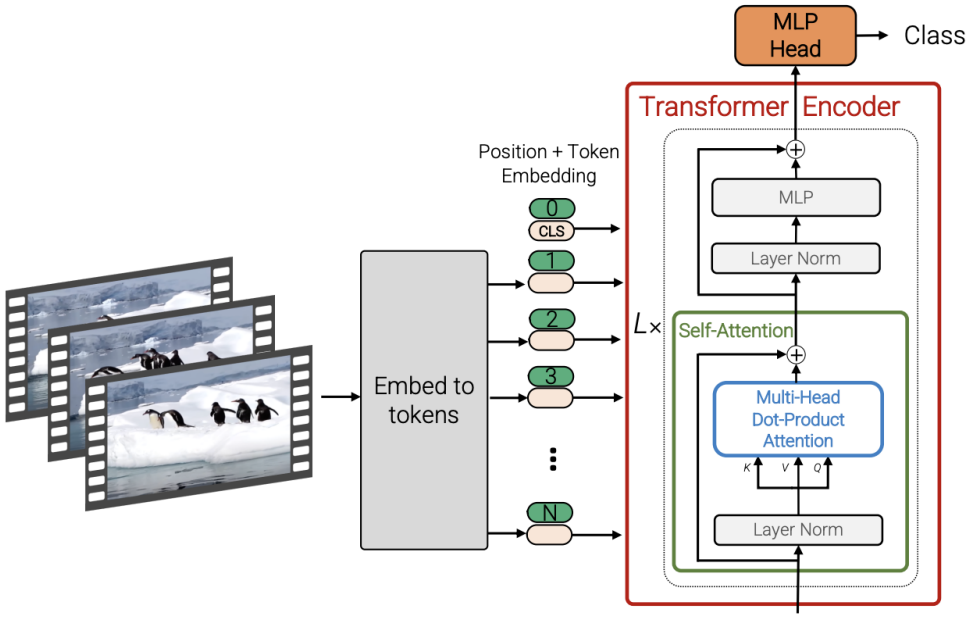
```
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# l, m: network and center momentum rates
gt.params = gs.params
for x in loader: # load a minibatch x with n samples
    x1, x2 = augment(x), augment(x) # random views

    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
    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
    return - (t * log(s)).sum(dim=1).mean()
```

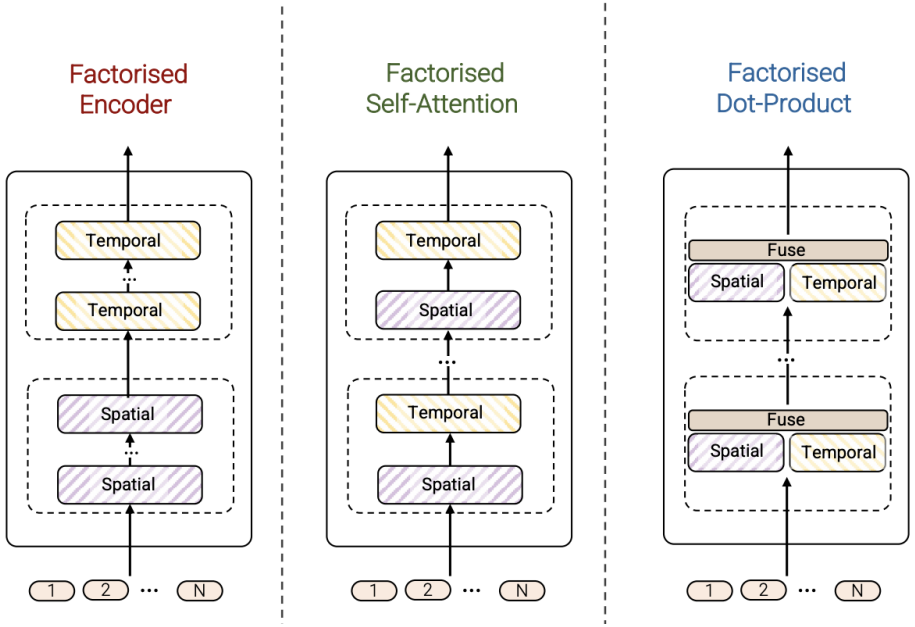
# ViViT

**ViViT: A Video Vision Transformer**

Anurag Arnab* Mostafa Dehghani* Georg Heigold Chen Sun Mario Lučić† Cordelia Schmid†
Google Research
{aarnab, dehghani, heigold, chensun, lucic, cordelias}@google.com

ICCV 2021



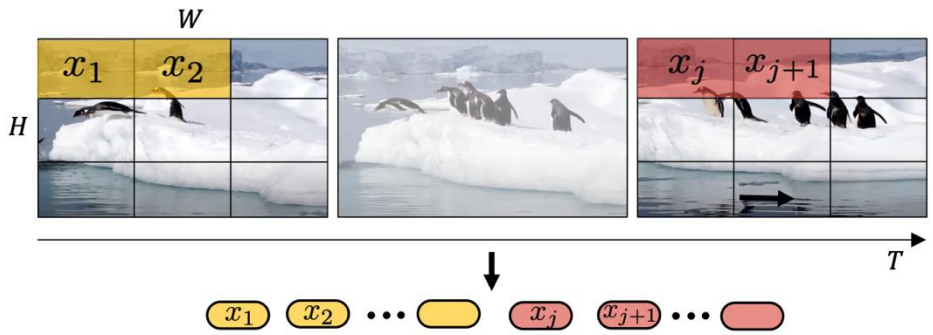




Figure 2: Uniform frame sampling: We simply sample $n_t$ frames, and embed each 2D frame independently following ViT [18].

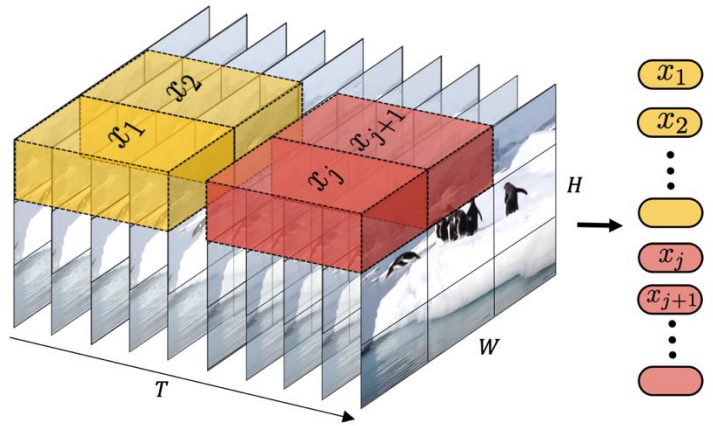


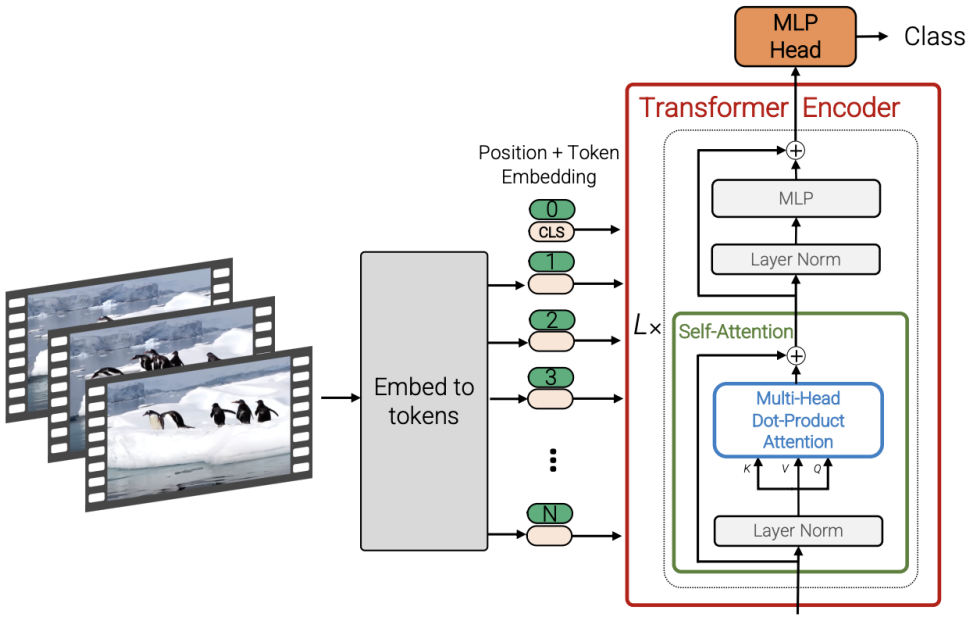

Figure 3: Tubelet embedding. We extract and linearly embed non-overlapping tubelets that span the spatio-temporal input volume.

# ViViT

**ViViT: A Video Vision Transformer**

Anurag Arnab* Mostafa Dehghani* Georg Heigold Chen Sun Mario Lučić† Cordelia Schmid†
Google Research
{aarnab, dehghani, heigold, chensun, lucic, cordelias}@google.com

ICCV 2021



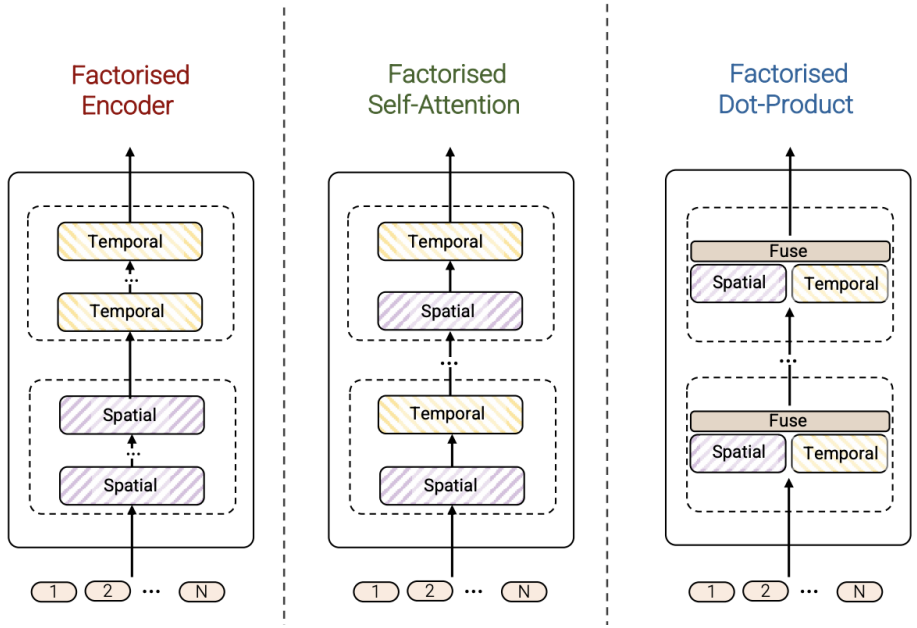




Table 1: Comparison of input encoding methods using ViViT-B and spatio-temporal attention on Kinetics. Further details in text.

| | Top-1 accuracy |
|---|---|
| Uniform frame sampling | 78.5 |
| *Tubelet embedding* | |
| Random initialisation [25] | 73.2 |
| Filter inflation [8] | 77.6 |
| Central frame | 79.2 |

Table 2: Comparison of model architectures using ViViT-B as the backbone, and tubelet size of $16 \times 2$. We report Top-1 accuracy on Kinetics 400 (K400) and action accuracy on Epic Kitchens (EK). Runtime is during inference on a TPU-v3.

| | K400 | EK | FLOPs ($\times 10^9$) | Params ($\times 10^6$) | Runtime (ms) |
|---|---|---|---|---|---|
| Model 1: Spatio-temporal | 80.0 | 43.1 | 455.2 | 88.9 | 58.9 |
| Model 2: Fact. encoder | 78.8 | 43.7 | 284.4 | 115.1 | 17.4 |
| Model 3: Fact. self-attention | 77.4 | 39.1 | 372.3 | 117.3 | 31.7 |
| Model 4: Fact. dot product | 76.3 | 39.5 | 277.1 | 88.9 | 22.9 |
| Model 2: Ave. pool baseline | 75.8 | 38.8 | 283.9 | 86.7 | 17.3 |

# VideoMAE

**VideoMAE: Masked Autoencoders are Data-Efficient Learners for Self-Supervised Video Pre-Training**

NeurIPS 2022

**Zhan Tong** [1,2*] **Yibing Song** [2] **Jue Wang** [2] **Limin Wang** [1,3†]
[1]State Key Laboratory for Novel Software Technology, Nanjing University
[2]Tencent AI Lab [3]Shanghai AI Lab
tongzhan@smail.nju.edu.cn {yibingsong.cv, arphid}@gmail.com lmwang@nju.edu.cn

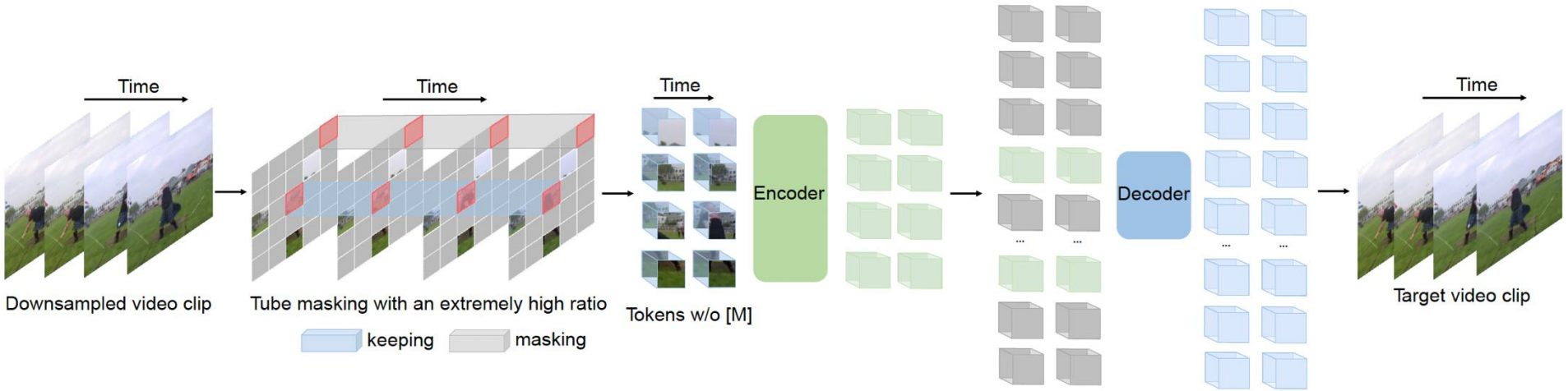



Figure 1: **VideoMAE** performs the task of masking random cubes and reconstructing the missing ones with an asymmetric encoder-decoder architecture. Due to high redundancy and temporal correlation in videos, we present the customized design of tube masking with an extremely high ratio (90% to 95%). This simple design enables us to create a more challenging and meaningful self-supervised task to make the learned representations capture more useful spatiotemporal structures.