

Research Topics

Andreas Marfurt
NLP

Information Technology
16.05.2025

Overview

- Mixture-of-experts architecture (FS24)
- Multimodality/Vision & Language (FS23)
- Agents (FS25)

Not discussed in FS25:

- Efficiency techniques (FS24)
- Interpretability & explainability (FS24)
- Linear RNNs (HS24)
- Transformer variants (HS24)

Mixture-of-Experts (MoE)

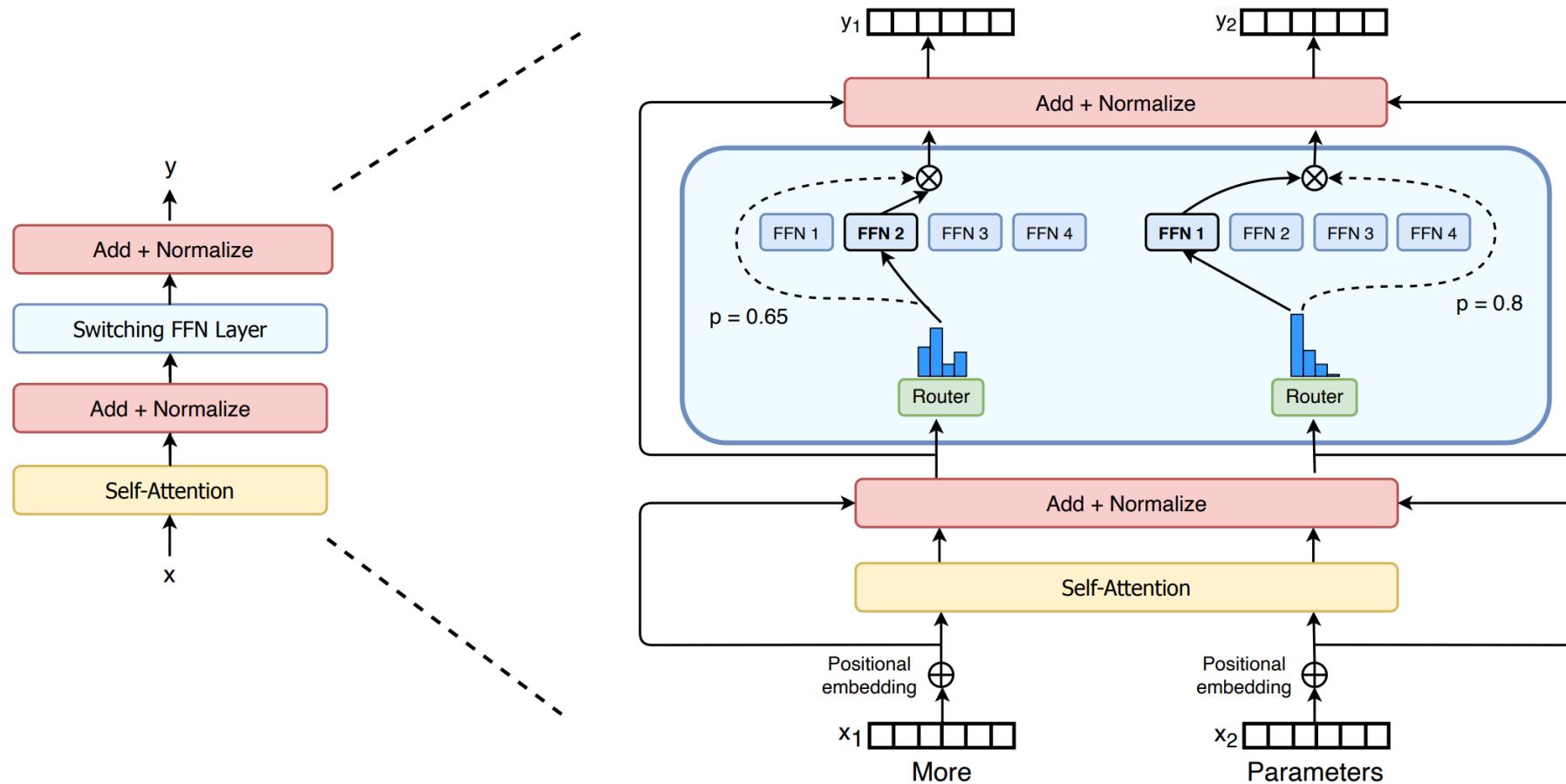
Mixture-of-Experts

- Sparsely-Gated Mixture-of-Experts Layer ([Shazeer et al., 2017](#))
- Switch Transformers ([Fedus et al., 2021](#))
 - Extension of the Transformer architecture with the above layer

Idea:

- Instead of one feed-forward network (FFN), use many “experts”
- Only pass each input to a subset of experts
- Allows specialization of experts to different functions
 - Similar to attention heads

Mixture-of-Experts Architecture



Routing

- Determines weights ("gates", $G(x)$) for each expert
- Multiplies outputs of expert $E(x)$ with that weight

$$y = \sum_{i=1}^n G(x)_i E_i(x)$$

- Weight = 0 → don't have to compute the output of that expert ("Sparse MoE")

Sparse Routing

- Dense routing (all experts are active):

$$G(x) = \text{Softmax}(xW_g)$$

- Sparse routing (only K experts are active):

$$G(x) = \text{Softmax}(\text{TopK}(xW_g))$$

TopK: set rest of logits to $-\infty$

Mixture-of-Experts Implications

Allows for more total parameters, but fewer active parameters:

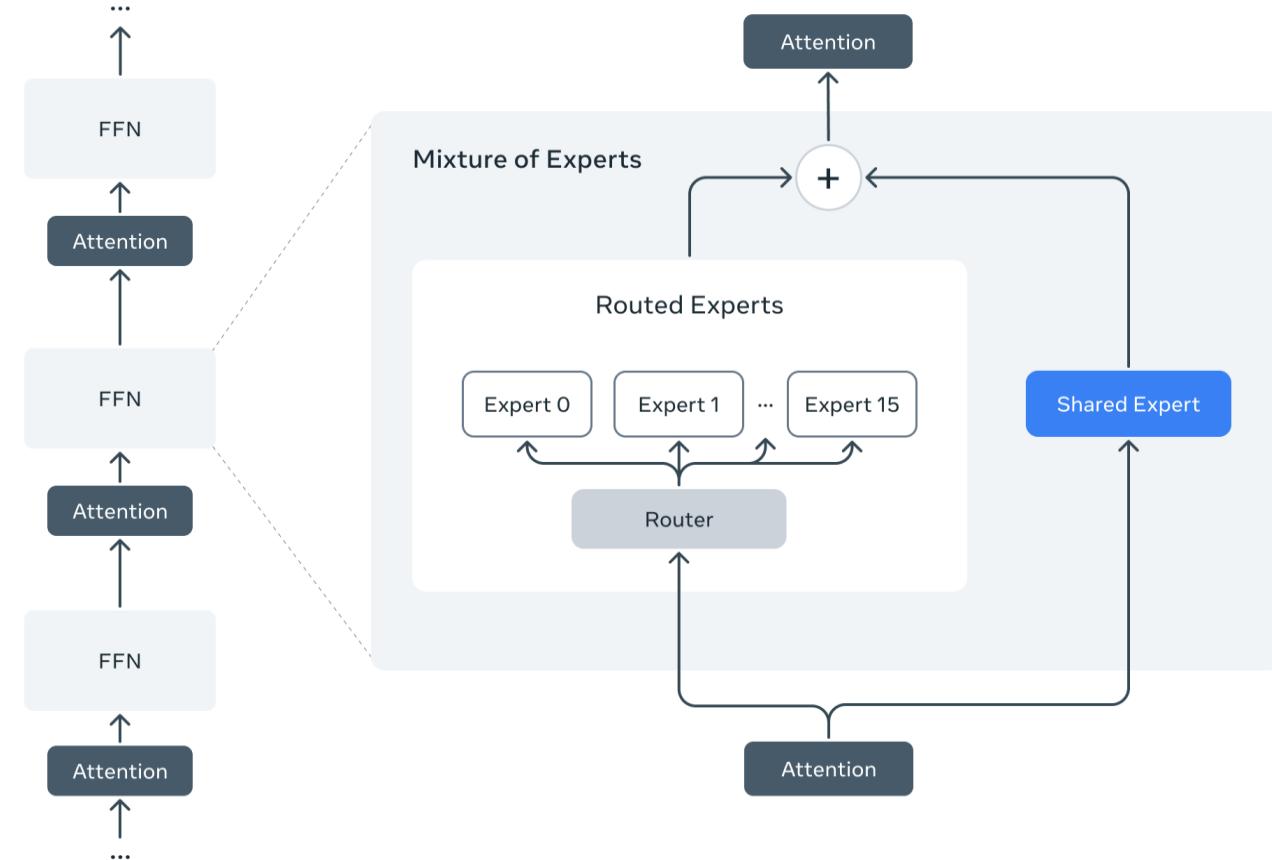
- Cheaper training
- Cheaper inference
- Faster convergence
- More memory

Architecture Choices

Model	# experts	# active experts	# parameters	# active params
GPT-4 (rumored)	16	2	1.8T	280B
Mixtral 8x7B	8	2	47B	13B
Mixtral 8x22B	8	2	141B	39B
Grok-1	8	2	314B	86B?
DBRX	16	4	132B	36B
Snowflake Arctic	128	2	480B	17B
Llama 4 Scout	16	2	109B	17B
Llama 4 Maverick	128	2	400B	17B
Llama 4 Behemoth	16	2	2T	288B

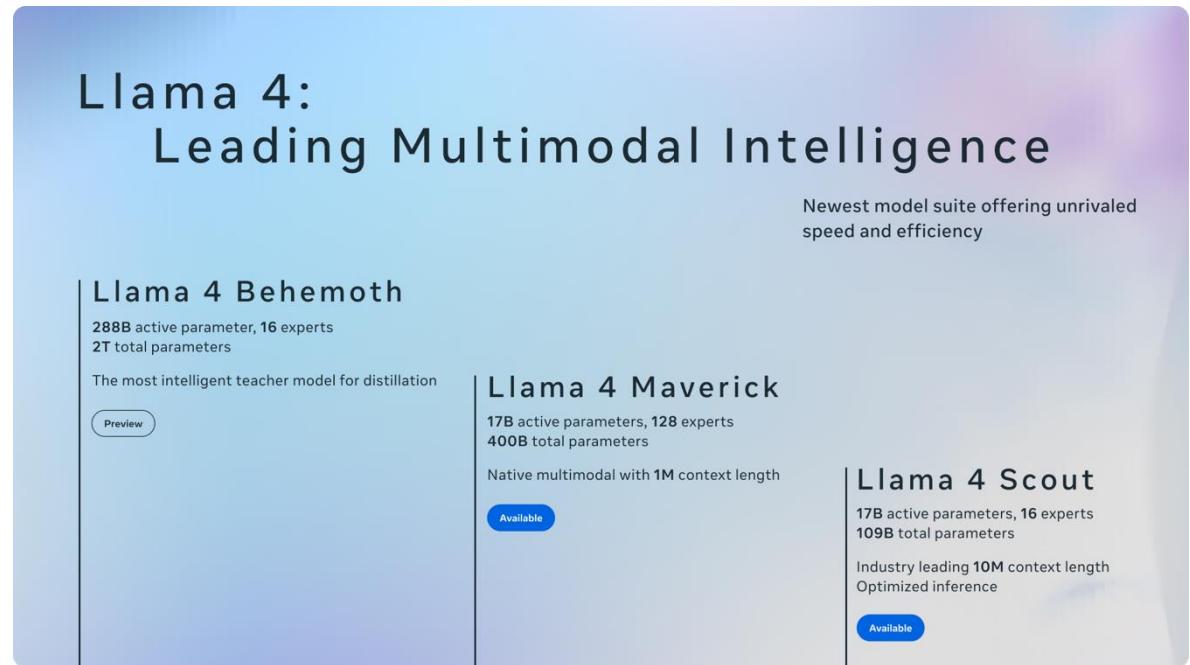
Llama 4: Shared Expert

- 2 active experts
 - Shared expert (always used)
 - Router selects one of 16/128 routed experts
- Alternate dense and MoE layers for inference efficiency



Llama 4

- 30T tokens
 - ~20T English
 - ~10T multilingual
- 200 languages
 - 100 languages with >1B tokens
- FP8 training for efficiency (this is not so easy)
- 10M input context length
- Multimodal
 - Vision & text inputs/outputs
 - Uses early fusion: interleave images and text in pre-training [1][2]



Multimodality

Multimodality Overview

- Models
 - VisualBERT
 - CLIP
 - Flamingo
 - LLaMA-Adapter
- Data Generation
 - LLaVA
 - MiniGPT-4
- Datasets & Evaluations

VisualBERT

- Encoder-only model for image classification
- Uses a CNN (ResNet) to obtain image features for objects in bounding boxes (from an object detector)
- Train with COCO captions dataset
 - Masked language modeling on captions
 - Model has to distinguish if 2 captions both describe the image or if one is about a different image (uses CLS embedding)
- [Paper](#)
- [Hugging Face model description](#)

VisualBERT Architecture

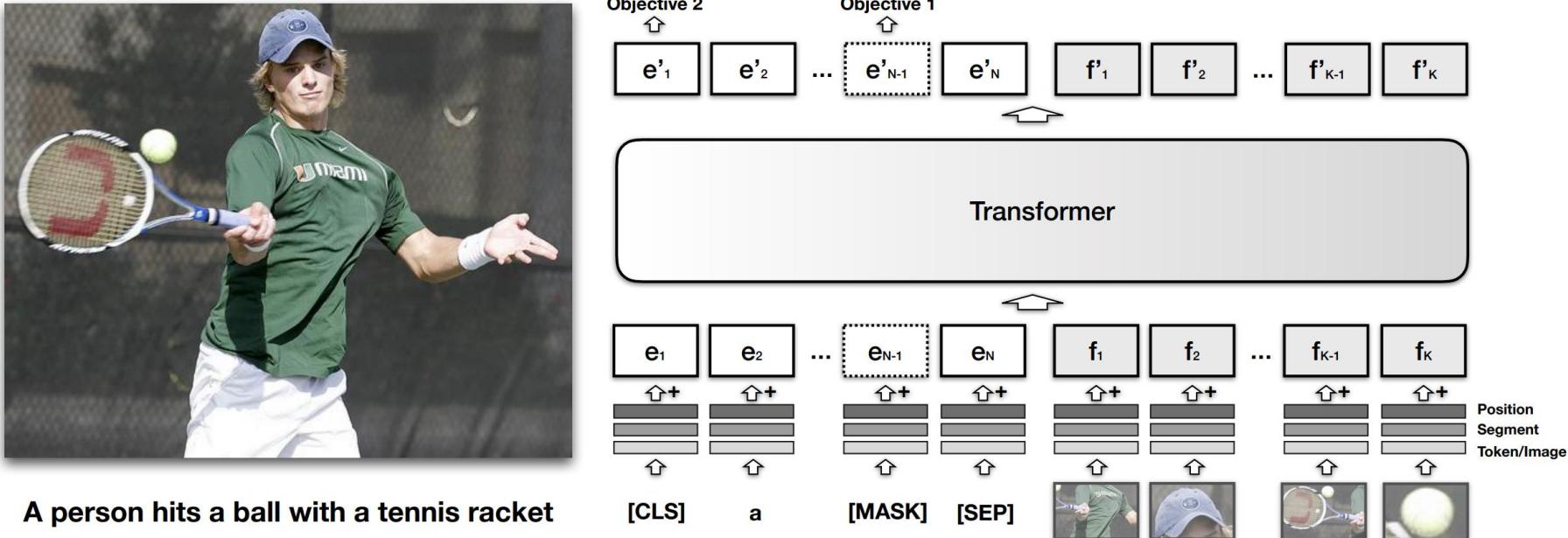
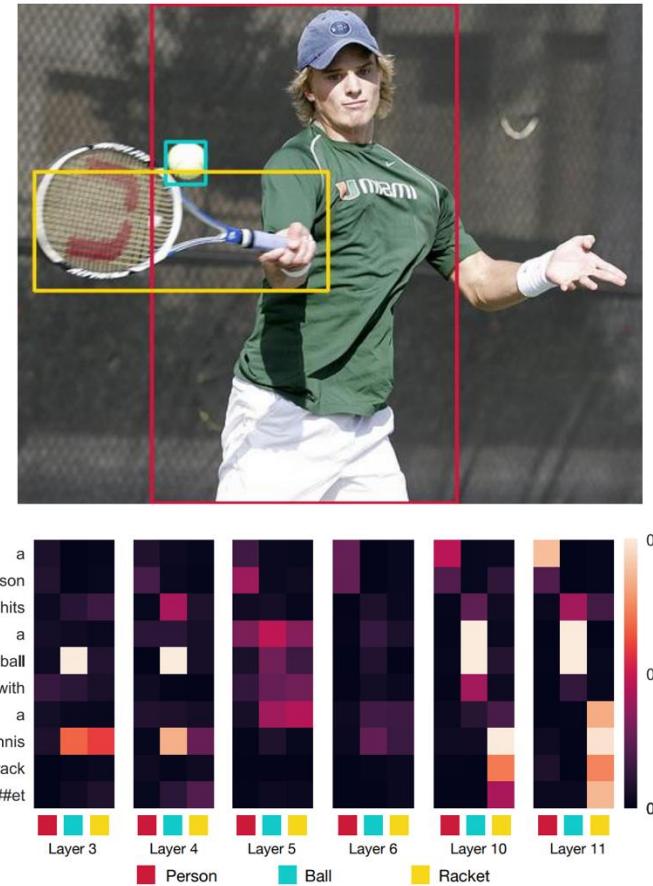


Figure 2: The architecture of VisualBERT. Image regions and language are combined with a Transformer to allow the self-attention to discover implicit alignments between language and vision. It is pre-trained with a masked language modeling (Objective 1), and sentence-image prediction task (Objective 2), on caption data and then fine-tuned for different tasks. See §3.3 for more details.

VisualBERT Attention Visualization



CLIP

- Encoder-decoder model
- Pretraining: Find correct caption for image in a batch
 - 400M (image, text) pairs
- Inference: Create caption for each class, find most likely caption (like verbalizers in PET)
- [Blog post](#)
- [Paper](#)
- [Hugging Face model documentation](#)

CLIP Training & Inference

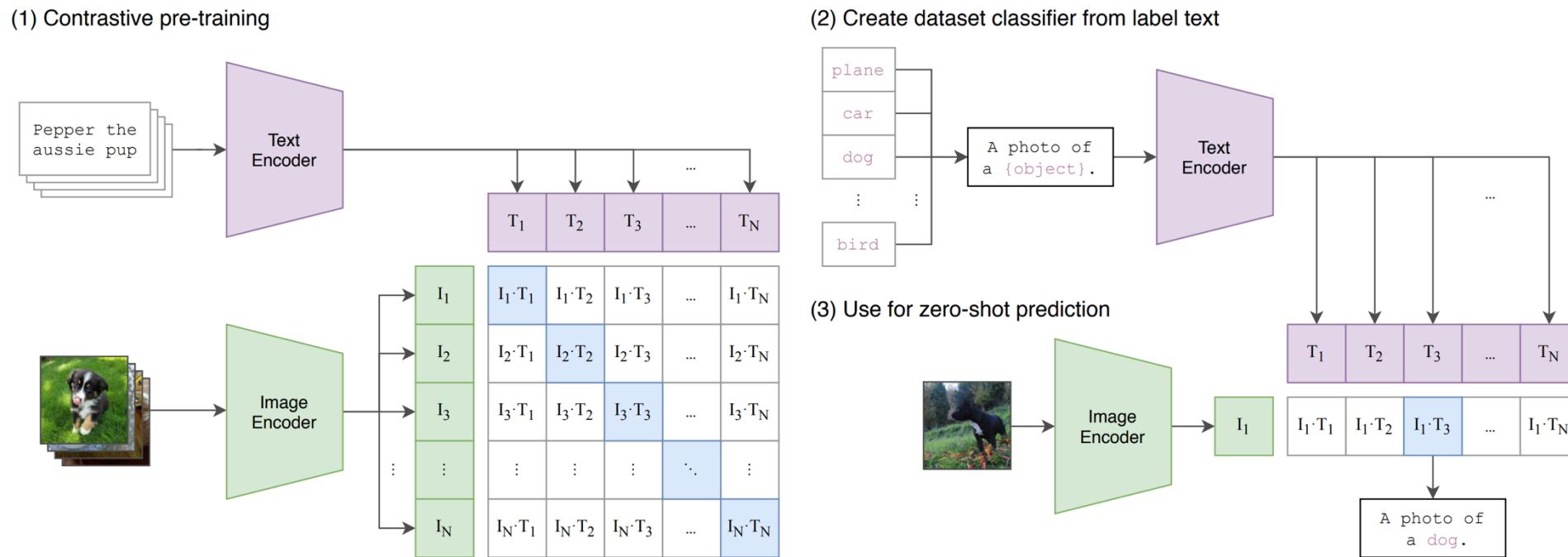


Figure 1. Summary of our approach. While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time the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

CLIP Architecture

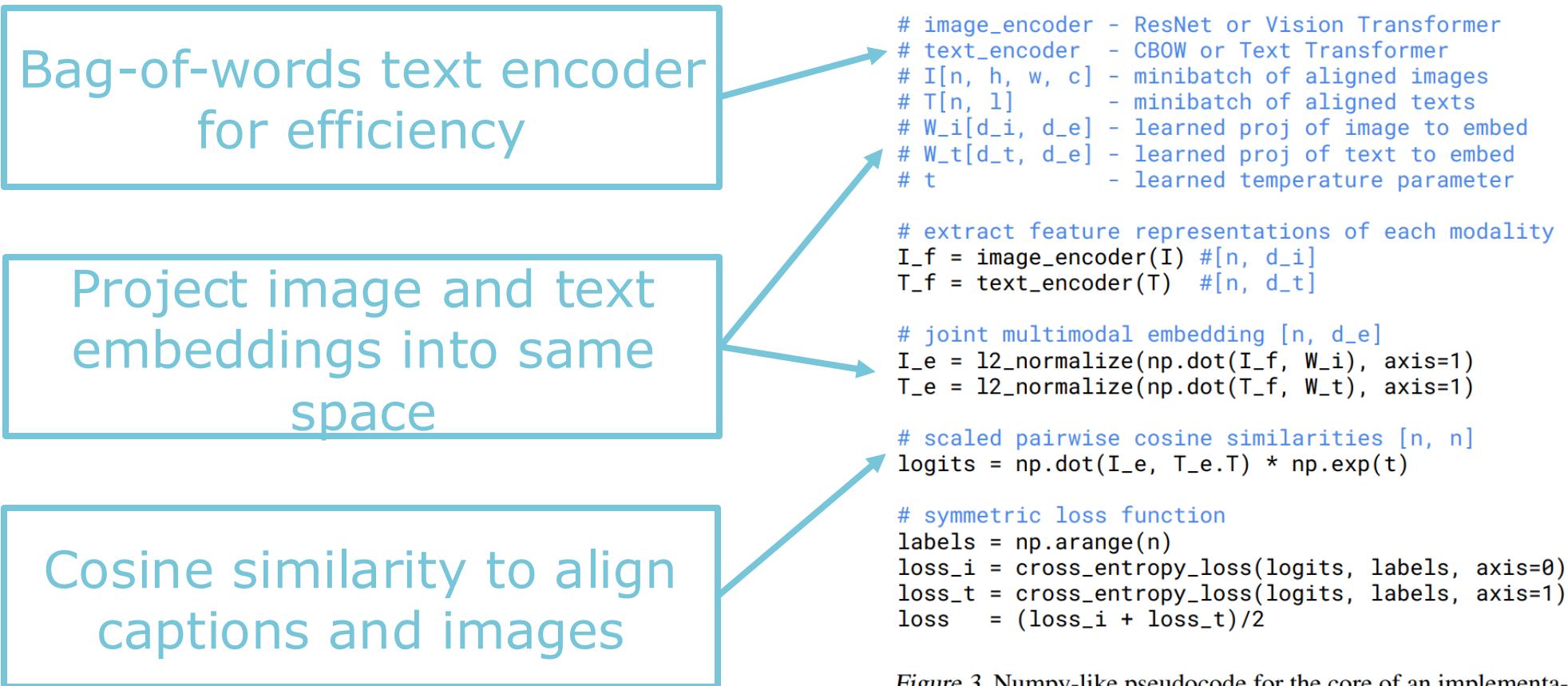


Figure 3. Numpy-like pseudocode for the core of an implementation of CLIP.

CLIP Showcase

Food101

guacamole (90.1%) Ranked 1 out of 101 labels



- a photo of **guacamole**, a type of food.
- a photo of **ceviche**, a type of food.
- a photo of **edamame**, a type of food.
- a photo of **tuna tartare**, a type of food.
- a photo of **hummus**, a type of food.

SUN397

television studio (90.2%) Ranked 1 out of 397 labels



- a photo of a **television studio**.
- a photo of a **podium indoor**.
- a photo of a **conference room**.
- a photo of a **lecture room**.
- a photo of a **control room**.

CLIP's Influence

CLIP provides a similarity of image and caption

→ Can use its objective in generative models to create images from user prompts:

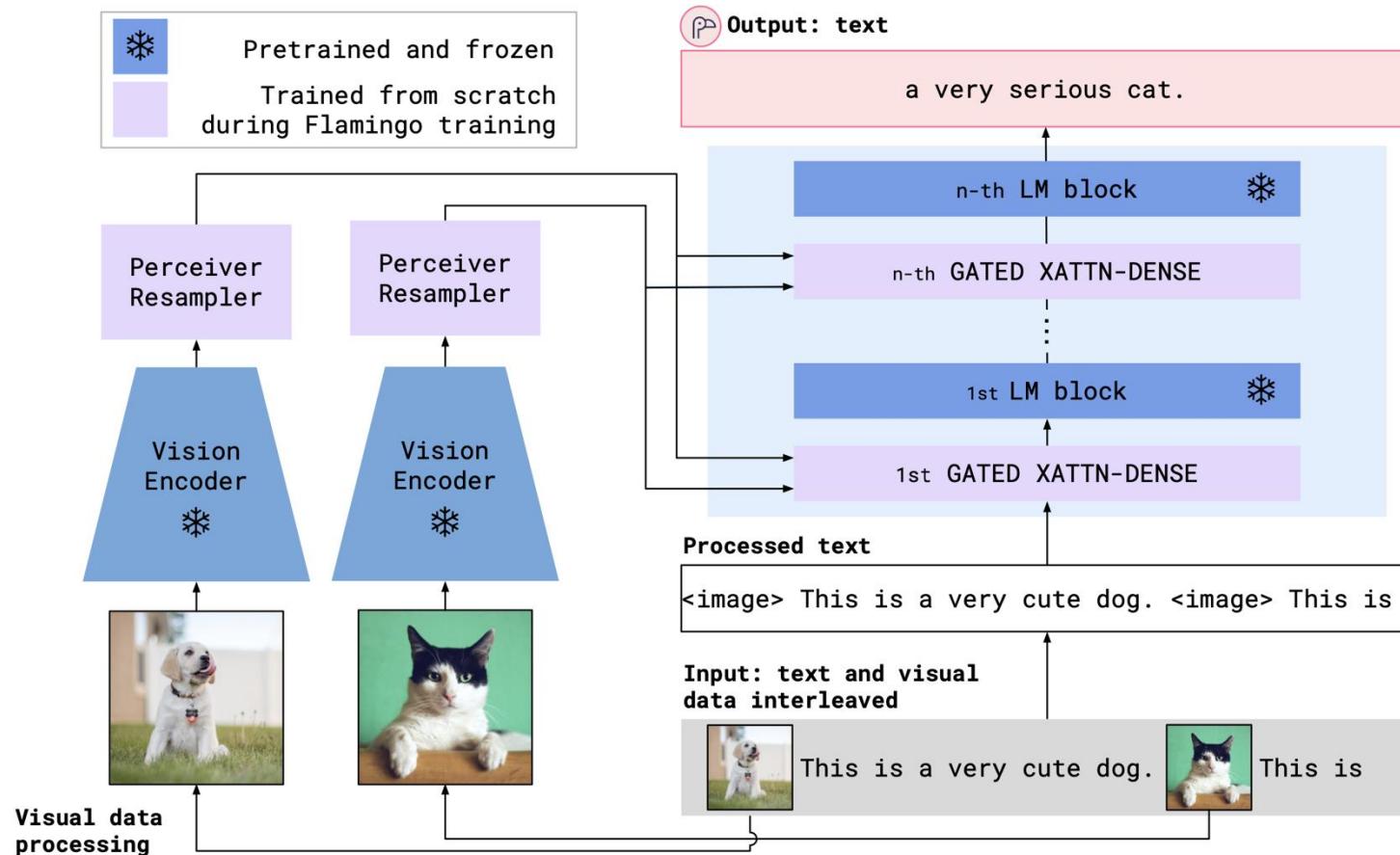
1. Encode text prompt with text encoder --> text representation (= image representation, they are in the same latent space)
2. Use image generator (e.g. GAN, diffusion model) to generate an image from the latent representation

This has also influenced later models such as [GLIDE](#) or [DALL-E](#)

Flamingo

- Flamingo (80B) uses a pretrained vision encoder (NFNet) and language model (Chinchilla, 70B)
 - Freeze pretrained weights
 - Only train mapping between the two (slowly increase their impact with a "gate")
- Training
 - Input: Interleaved images and text captions
 - Output: text (continuation of caption)
- [Deepmind blog post](#) & [paper overview blog post](#)
- [Paper](#)

Flamingo Architecture

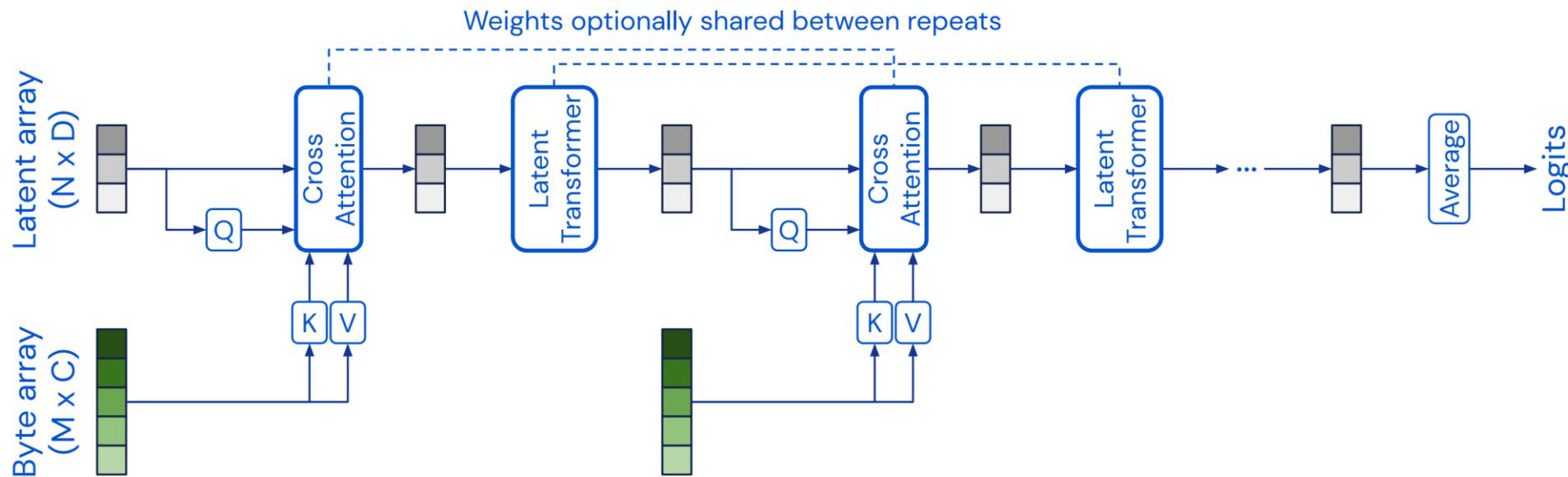


Flamingo: Image Encoder

- Normalizer-Free ResNet (NFNet)
- Pretrained with CLIP objective
 - BERT model as text encoder, discarded after training
- 2 datasets
 - ALIGN: 1.8B (image, alt-text) pairs
 - LTIP: 312M (image, description) pairs
- Flatten 2D images into 1D sequence
 - Problem: Sequence length blows up even at moderate resolution (288x288 image = 82944 pixels)

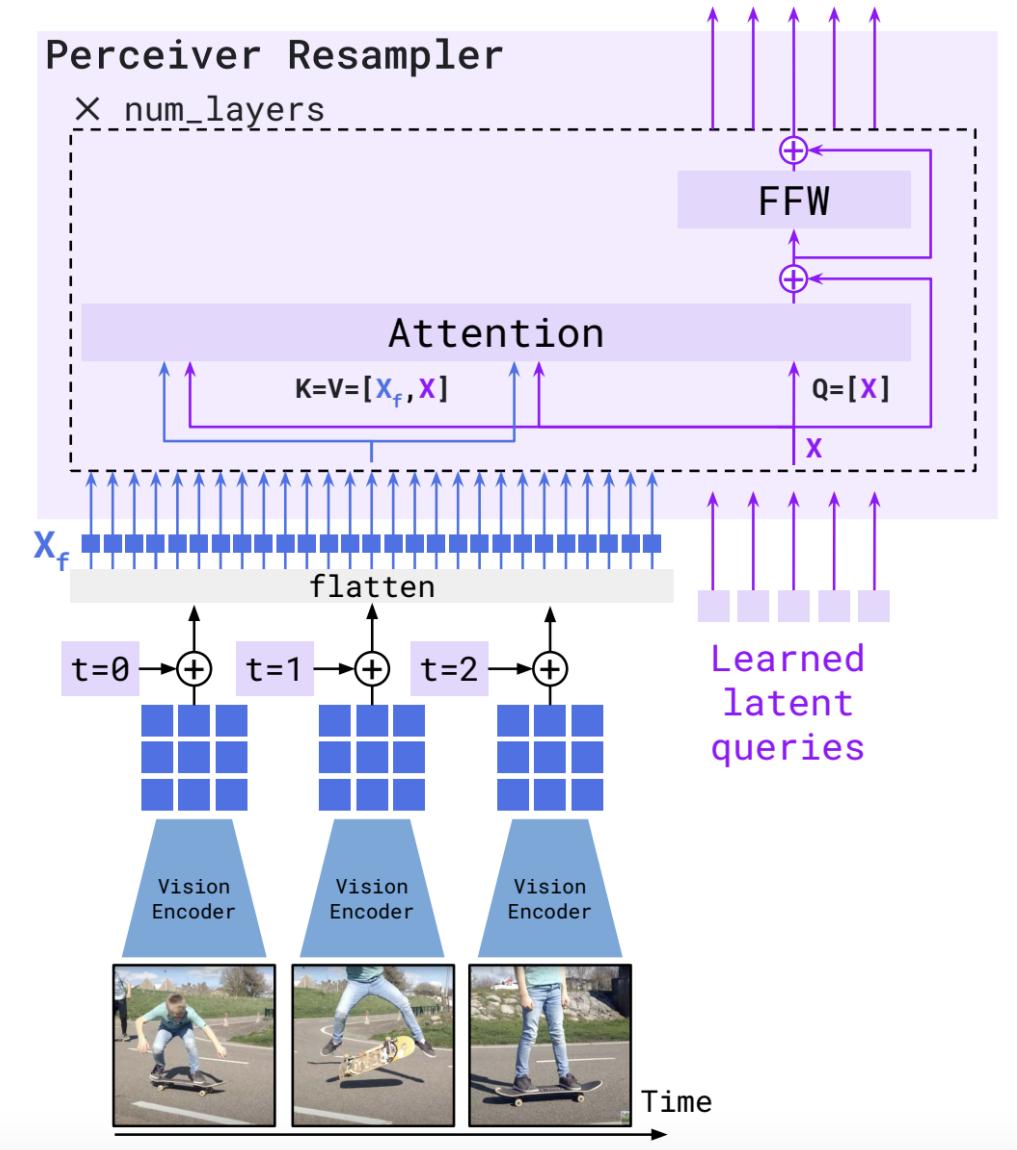
Aside: Perceiver

- [Jaegle et al., 2021](#)
- Cross-attention layers (costly): integrate image information
- Self-attention on (*fixed-sized*) latent representation (cheap)

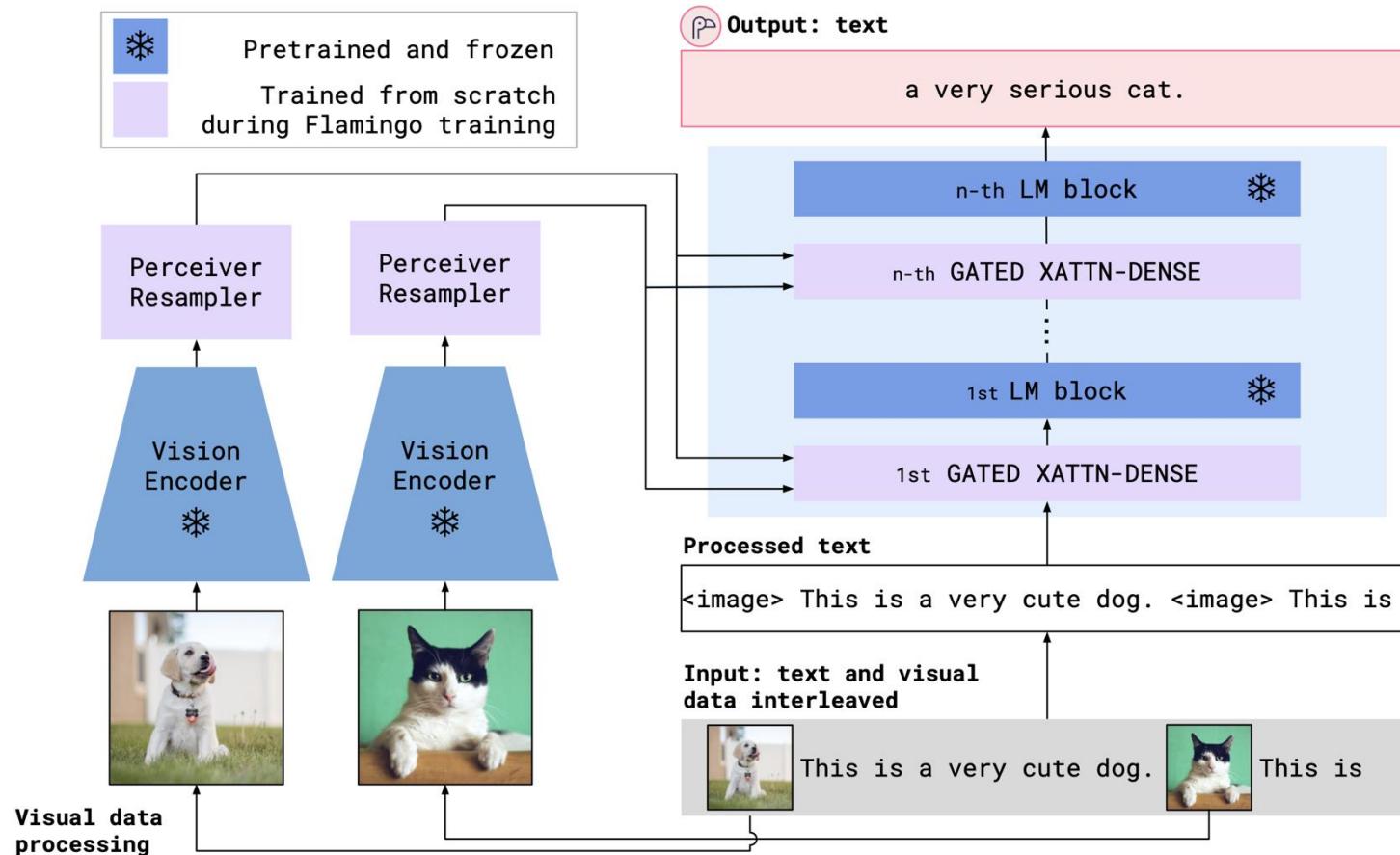


Perceiver in Flamingo

- 64 learned latent queries
- Integrate vision information with attention
- For videos: Add time t as position embedding
 - Down-sampling during training: 1 frame per second



Flamingo Architecture



Flamingo Training

- Datasets
 - Image-text pairs (ALIGN, LTIP)
 - Video-text pairs (VTP, 27M short videos)
 - Multi-Modal Massive Web dataset (text and images from 43M websites)



Image-Text Pairs dataset
[N=1, T=1, H, W, C]



Video-Text Pairs dataset
[N=1, T>1, H, W, C]



Multi-Modal Massive Web (M3W) dataset
[N>1, T=1, H, W, C]

- Predict next caption word based on all previous tokens and the preceding image
 - Worked better than all preceding images
 - Mask all images except the previous

Flamingo Model Sizes

Requires model sharding	Frozen		Trainable		Total count
	Language	Vision	GATED	XATTN-DENSE	
<i>Flamingo-3B</i>	✗	1.4B	435M	1.2B (every)	194M 3.2B
<i>Flamingo-9B</i>	✗	7.1B	435M	1.6B (every 4th)	194M 9.3B
<i>Flamingo</i>	✓	70B	435M	10B (every 7th)	194M 80B

Flamingo Showcases

Input Prompt



Question: What do you think the capacities of these are? Answer:

Completion

The floppy disk is 1.44MB and the CD is 700MB.



Question: What is odd about this image? Explain why it is unusual. Answer:



Question: What country is this? Why do you think so? Answer:

The image is odd because the elephant is in the back of the truck. It is unusual because elephants are not usually transported in the back of a truck.

It is Canada. I think so because the flag is the Canadian flag.

Flamingo Showcases

- [Architecture GIF](#)
- Multimodal dialogue
 - [Image description](#)
 - [Stroop test](#) (identify text/font color; example: **blue**)

OpenFlamingo

- Open-source version by LAION (March 28, 2023)
- 9B parameter model based on LLaMA 7B and CLIP ViT/L-14
 - Can also use OPT, GPT-Neo, GPT-J, and Pythia for the LM
 - Can use any CLIP encoder for the vision model
- Trained with interleaved text and images (mmc4) and image-caption pairs (LAION-2B)
- [Blog post](#)
- [Code](#)
- [Demo](#)
- [Model on Hugging Face](#)

Future plans

- Add support for video input
- Release better performing and larger OpenFlamingo models
- Expand our evaluation suite
- Add support for FSDP training

OpenFlamingo Showcases

Captioning

Input Prompt



Two cats are sleeping next to each other on a sofa.



A bathroom with a sink, mirror and cabinets.



Completion

A table full of food for a holiday party.

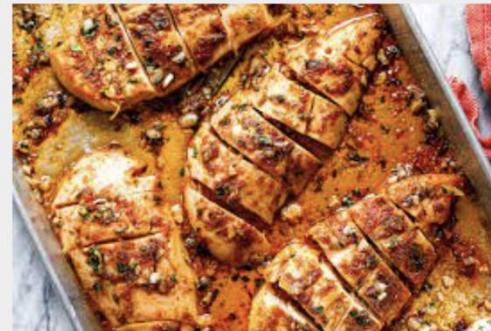
OpenFlamingo Showcases

Recipe generation

Input Prompt



Step 1. Fill a large pot with lightly salted water and bring to a rolling boil. Stir in pasta and return to a boil. Cook pasta uncovered, stirring occasionally, until tender yet firm to the bite, about 8 to 10 minutes. Drain and transfer into a large bowl. Step 2. Meanwhile, heat oil in a frying pan over medium-low heat. Add onion; cook and stir until softened, about 3 minutes. Stir in pesto, salt, and pepper until warmed through. Step 3. Add pesto mixture to hot pasta; stir in grated cheese and toss well to coat.



Completion

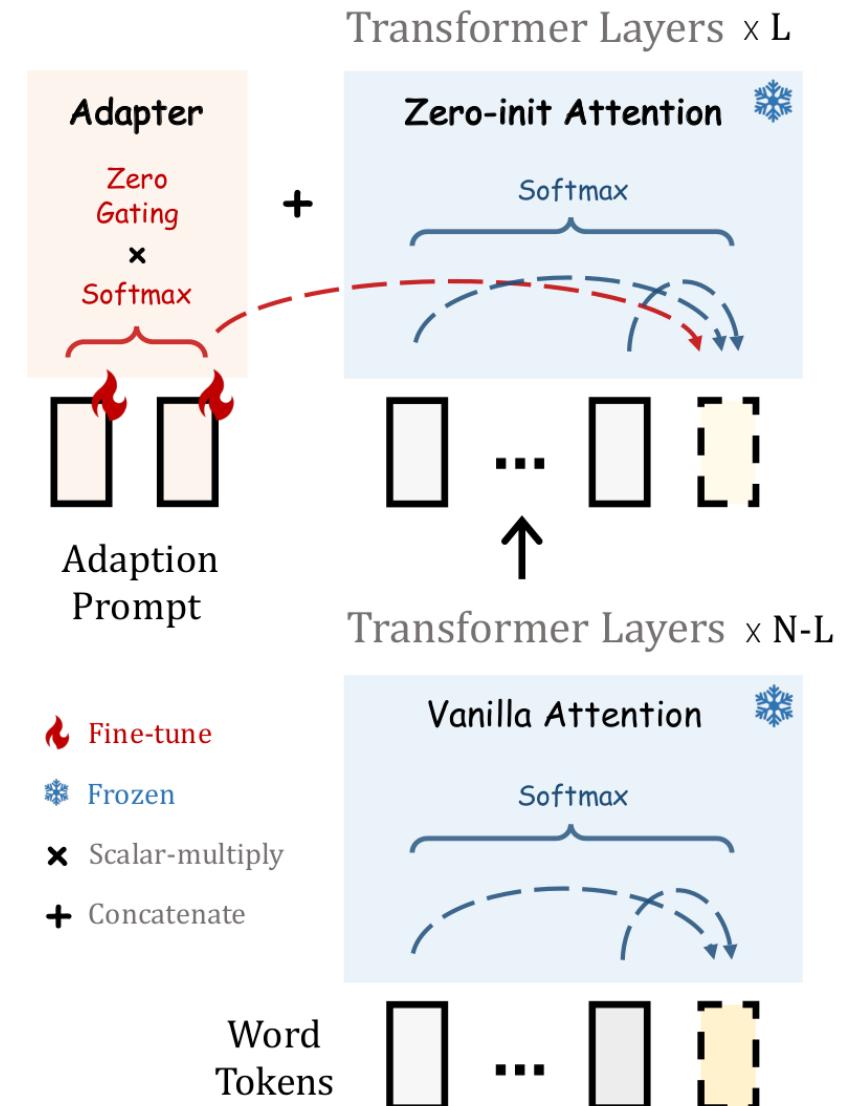
Step 1. Preheat oven to 400°F. Line a large rimmed baking sheet with foil. Step 2. Place chicken in a large resealable plastic bag. Add lemon juice, garlic, 1 teaspoon salt, and 1/2 teaspoon pepper. Seal bag and shake to coat chicken. Step 3. Place chicken in a single layer on prepared baking sheet. Step 4. Bake until chicken is cooked through and juices run clear, 20 to 25 minutes. Step 5. Meanwhile, in a small bowl, whisk together remaining 1/2 teaspoon salt, remaining 1/2 teaspoon pepper, and lemon zest. Step 6. Remove chicken from oven and brush with lemon juice mixture.

LLaMA-Adapter

- [Zhang et al., 2023](#) (March 28, 2023)
- *Adaption prompts*: PEFT from combination of prompt tuning and adapter layers
- Adds instruction tuning and multimodality
 - in one hour of training
 - with just 1.2M learnable parameters
- Can answer questions from multimodal ScienceQA
- [Code](#)

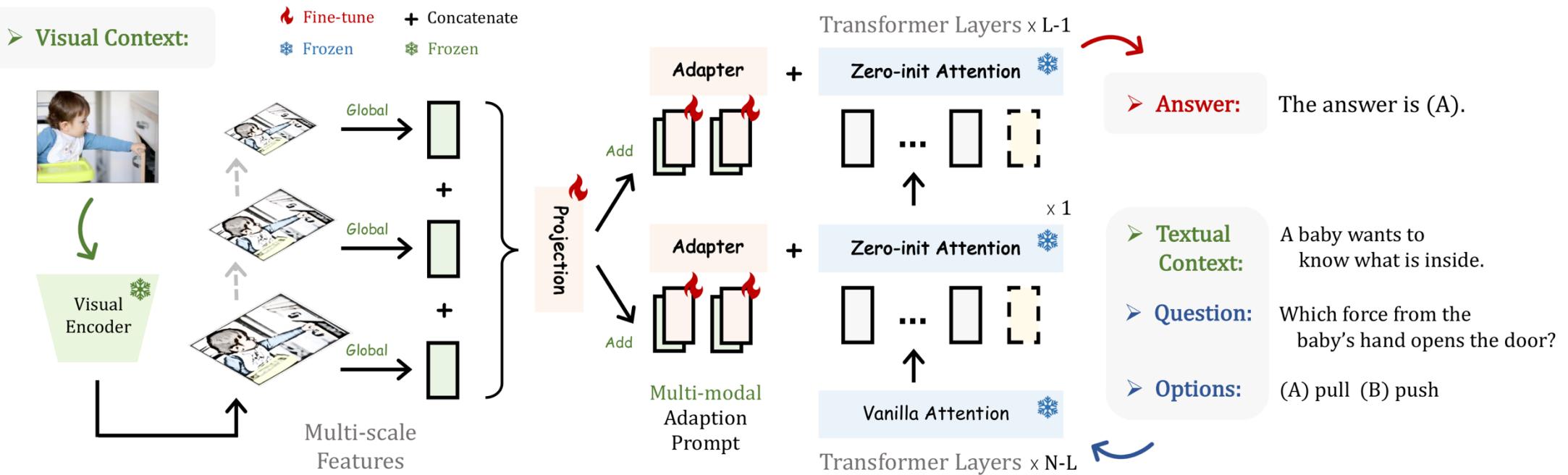
LLaMA-Adapter

- LLaMA base model (7B params, N=32 layers)
- First N-L=2 layers untouched
- Last L=30 layers with learnable adaption prompts
 - Separate softmax for adaption prompts and word tokens
 - Learned gate initialized with 0, controls weight of adaption prompts



LLaMA-Adapter Multimodality

- Extract image features at varying resolutions, concatenate
- Down-project to hidden dim, add to adaption prompts
- Use for any modality (video, audio) with a respective encoder



LLaMA-Adapter Showcase

Question: Select the fish below.

Context: Fish live underwater. They have fins, not limbs. Fish are cold-blooded. The body temperature of cold-blooded animals depends on their environment. A Banggai cardinalfish is an example of a fish.

Choices: (A) green moray eel (B) rabbit (C) woodpecker (D) bald eagle

Answer: The answer is (A)



Question: Think about the magnetic force between the magnets in each pair. Which of the following statements is true?

Context: The images below show two pairs of magnets. The magnets in different pairs do not affect each other.

All the magnets shown are made of the same material.

Choices:

- (A) The magnitude of the magnetic force is the same in both pairs.
- (B) The magnitude of the magnetic force is greater in Pair 1.
- (C) The magnitude of the magnetic force is greater in Pair 2.

Answer: The answer is (C)



Figure 4. Two examples of **Multi-modal Reasoning on ScienceQA [27]** by LLaMA-Adapter.

LLaMA-Adapter Interesting Results

- Use adaption prompts in most layers
- Validation loss increases, but validation accuracy still improves (loss is not measuring our target)

Layers	Params	Val Acc (%)
10	0.97	55.95
20	1.37	73.36
30	1.79	83.85

Table 4. Ablation on Inserted Layers.

Epoch	Train Loss	Val Loss	Val Acc (%)
15	0.022	0.136	82.08
30	0.004	0.241	83.85
60	0.001	0.282	83.94

Table 6. Loss Values vs. Validation Accuracy.

LLaMA-Adapter v2

- [Gao et al., 2023](#) (April 28, 2023)
- LLaMA-Adapter v1 has no visual instructions
 - Cannot do complex visual reasoning/QA
- Tried to finetune v1 with image-text pairs
 - Visual features dominate
 - Instruction following gets worse
- In v2: Add visual features to first K layers
 - Instruction following at last L layers (no overlap)
 - Train the two sets of parameters separately
 - No need for multimodal instruction data

LLaMA-Adapter v2: Bias Tuning

- Train more parameters
 - All normalization layers
 - For each linear layer: Add a bias and a scaling factor

$$\mathbf{y} = \mathbf{W} \cdot \mathbf{x} \rightarrow \mathbf{y} = s \cdot (\mathbf{W} \cdot \mathbf{x} + b),$$

where $b = \text{Init}(0)$, $s = \text{Init}(1)$.

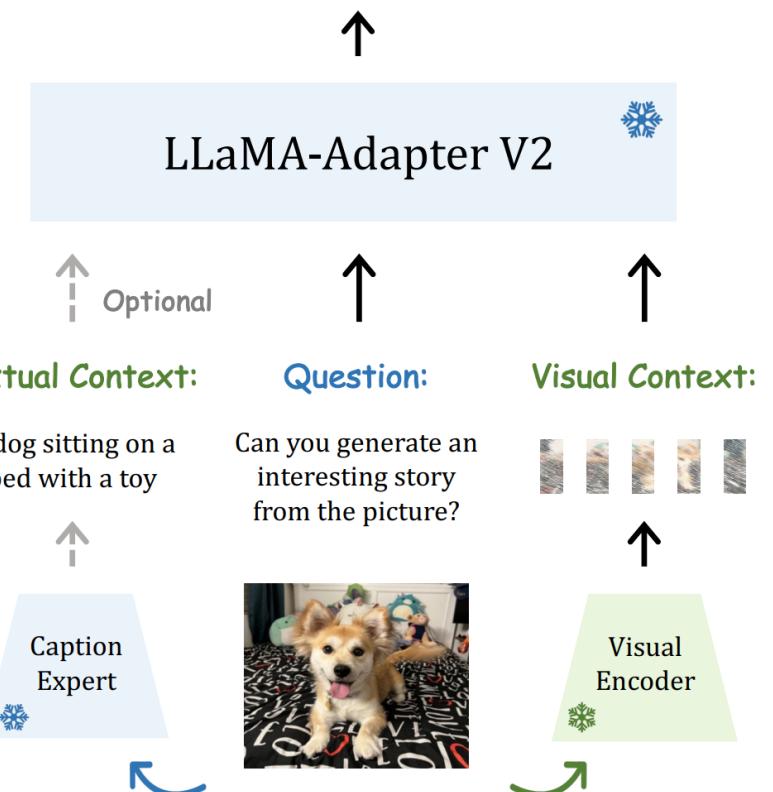
- Makes 5M (0.04%) of parameters trainable

LLaMA-Adapter v2: External Models

- Integrate external expert models
 - Image captioning
 - Object detection
 - OCR
- Experts output text
- Optional input to LLM

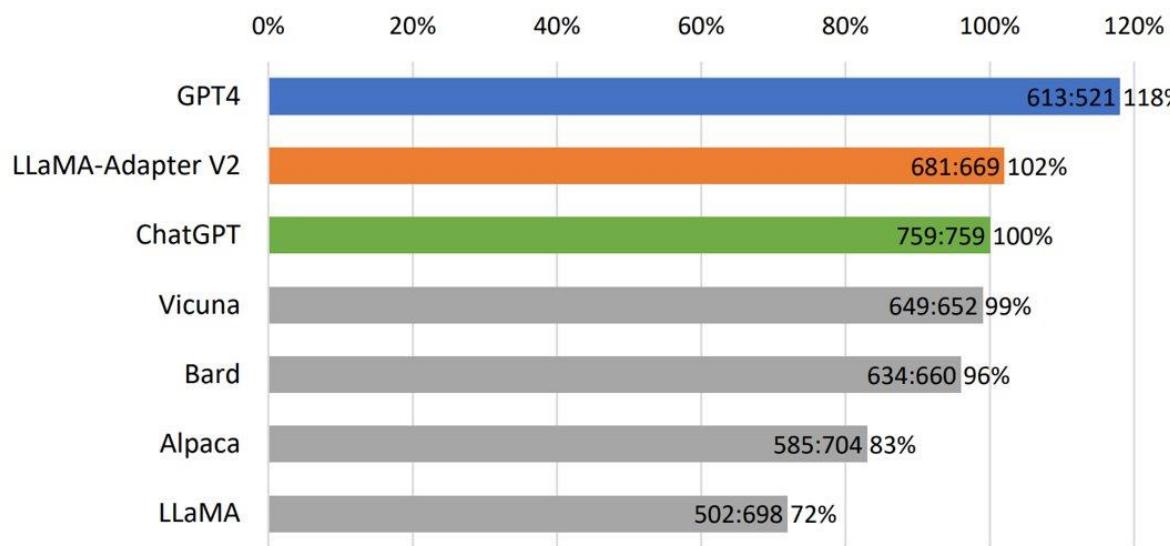
Answer:

Once upon a time, there was a dog who lived in a cozy little house with a big bed. The dog loved to curl up on the bed and take a nap, but he always got lonely ...

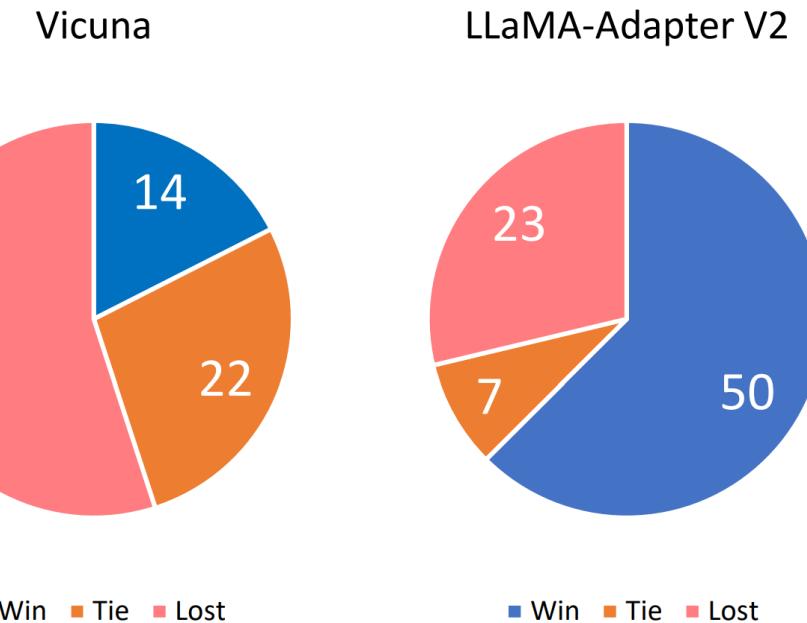


LLaMA-Adapter v2: Results

- LLaMA-Adapter v2 is based on LLaMA 65B
- Evaluated by GPT-4



(a) The total quality scores compared with ChatGPT.



(b) Left: Vicuna vs. ChatGPT. Right: LLaMA-Adapter V2 vs. ChatGPT.

Multimodal Data Generation

Large Language and Vision Assistant (LLaVA)

- Pre-trained image encoder (CLIP ViT-L/14) and LLaMA
 - Project visual features into word embedding space
 - Stage 1: freeze ViT and LLaMA, train projection W on image-text pairs
 - Stage 2: unfreeze LLaMA, train LLaMA and W on multimodal instruction following
- [Website](#)
- [Demo](#)
- [Paper \(April 17, 2023\)](#)

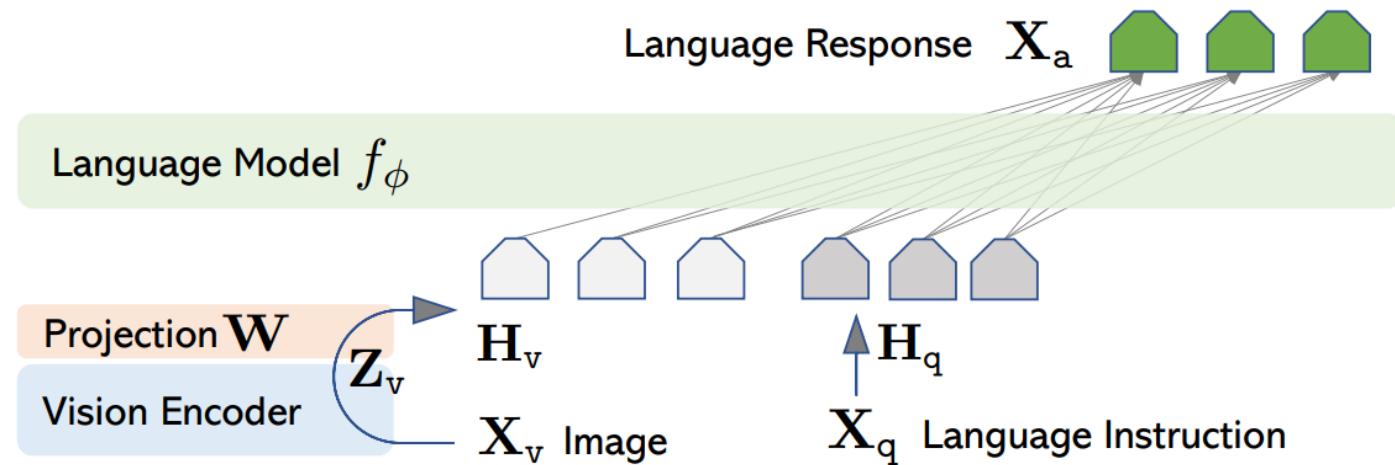


Figure 1: LLaVA network architecture.

LLaVA Data Generation

- Generate multimodal instruction following data
- Use captions and bounding boxes (in text form) as input to GPT-4/ChatGPT
- Output synthetic data (158K examples)
 - Conversations (58K)
 - Detailed descriptions (23K)
 - Complex reasoning Q&A (77K)

LLaVA Data Generation

- Input is just the text

Context type 1: Captions

A group of people standing outside of a black vehicle with various luggage.

Luggage surrounds a vehicle in an underground parking area

People try to fit all of their luggage in an SUV.

The sport utility vehicle is parked in the public garage, being packed for a trip

Some people with luggage near a van that is transporting it.

Context type 2: Boxes

person: [0.681, 0.242, 0.774, 0.694], person: [0.63, 0.222, 0.686, 0.516], person: [0.444, 0.233, 0.487, 0.34], backpack: [0.384, 0.696, 0.485, 0.914], backpack: [0.755, 0.413, 0.846, 0.692], suitcase: [0.758, 0.413, 0.845, 0.69], suitcase: [0.1, 0.497, 0.173, 0.579], bicycle: [0.282, 0.363, 0.327, 0.442], car: [0.786, 0.25, 0.848, 0.322], car: [0.783, 0.27, 0.827, 0.335], car: [0.86, 0.254, 0.891, 0.3], car: [0.261, 0.101, 0.787, 0.626]



LLaVA Data Generation

- Output: conversation, description or reasoning task
 - Use a few manually crafted outputs as few-shot examples
 - Input is just the text...
 - Quality?
 - Easy shortcuts?

Response type 1: conversation

Question: What type of vehicle is featured in the image?

Answer: The image features a black sport utility vehicle (SUV).

Question: Where is the vehicle parked?

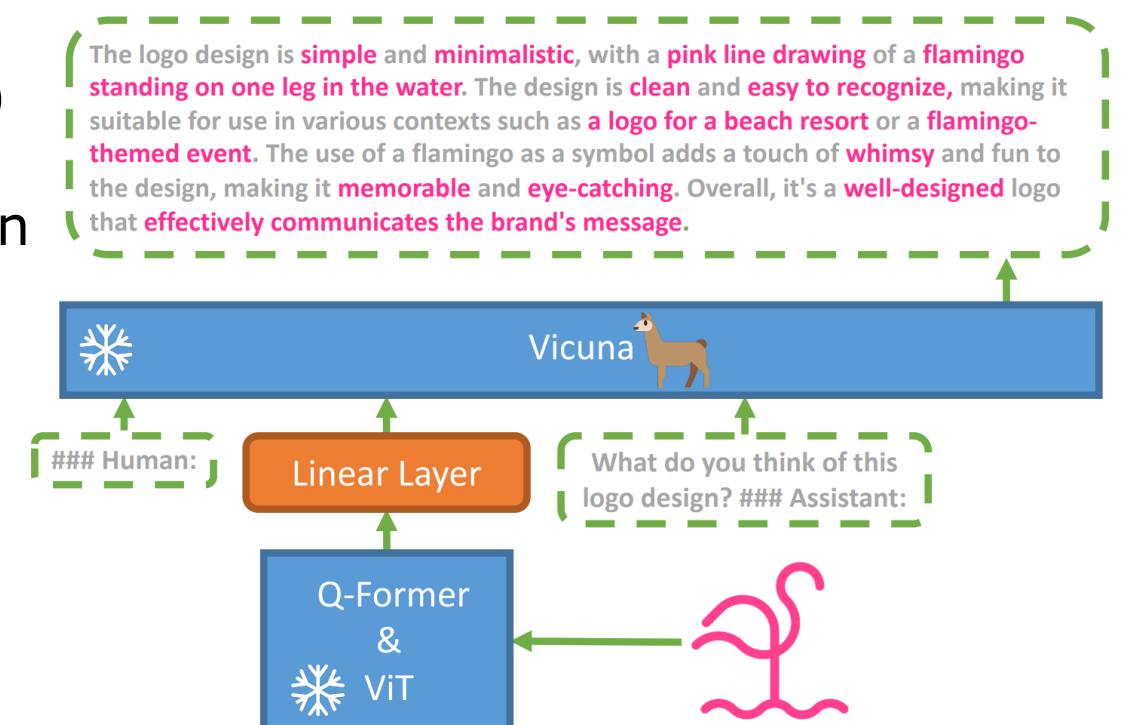
Answer: The vehicle is parked in an underground parking area, likely in a public garage.

Question: What are the people in the image doing?

Answer: The people are trying to fit all of their luggage into the SUV, likely preparing for a trip.

MiniGPT-4

- Use pretrained Vicuna LLM and BLIP-2 (ViT with Q-Former)
- Freeze models, train linear projection layer
- Training
 - Stage 1: 5M image-caption pairs (10h) [model doesn't follow instructions yet]
 - Stage 2: 3.5K filtered image-instruction pairs (7 min), created with a template and ChatGPT
- [Website with examples](#)
- [Paper](#) (April 20, 2023)



MiniGPT-4 Data Generation

- Detailed image description (by image captioning model)
 - If too short (<80 tokens), ask it to continue → concatenate
- Postprocessing #1: ChatGPT refines descriptions
 - Remove repetitions, correct grammar
- Postprocessing #2: Humans refine descriptions
 - Correct mistakes ChatGPT missed, filter bad examples (5K → 3.5K)
- Create image-instruction pairs from template:

```
###Human: <Img><ImageFeature></Img> <Instruction>
###Assistant: <Description>
```

 - Instructions look like “Describe this image in detail” or “Could you describe the contents of this image for me”

MiniGPT-4 Showcase



Describe the image



The image depicts a wolf and a lamb lying in a field of grass. The wolf is lying down with its head resting on its paws, while the lamb is lying next to it, looking up at the wolf. The sun is shining down on the scene, casting a warm glow on the animals and the grass around them. The overall mood of the image is peaceful and serene.

Can this happen in real world?



In the wild, it is not uncommon for wolves and lambs to coexist in the same area, although it is rare for them to interact in the way depicted in the image. In most cases, wolves will prey on lambs and other young livestock, so it is unlikely that they would lie down together in a peaceful manner like in the image. However, it is possible that they may come into contact with each other in other ways, such as when the wolf is defending its territory or when the lamb is separated from its flock.



Describe the image



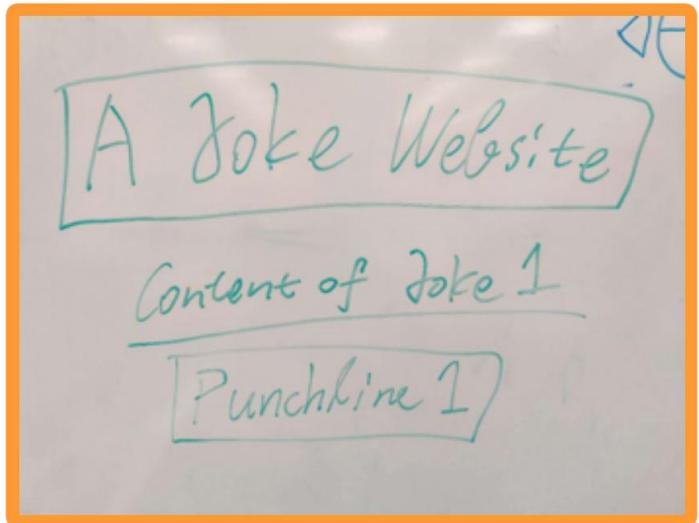
The image shows a cactus plant standing in the middle of a frozen lake. The cactus is surrounded by large ice crystals that have formed on the surface of the water. In the background, there is a range of mountains with snow-capped peaks. The sky is clear and blue, with a few fluffy clouds floating in the distance. The overall mood of the image is peaceful and serene, with the cactus standing tall and proud in the center of the frozen lake.

Is this image common in real world?



No, this image is not common in the real world. While it is possible for cacti to grow in cold climates, it is rare to see one standing in the middle of a frozen lake. Additionally, the formation of large ice crystals on the surface of the water is also not a common occurrence. This image is likely a digital creation or manipulation.

MiniGPT-4 Showcase



Write brief HTML/JS to turn this mock-up into a colorful website, where the jokes are replaced by a real joke



```
HTML:  
```php  
<!DOCTYPE html>
<html>
<head>
<title>A Joke Website</title>
<link rel="stylesheet" type="text/css" href="style.css"/>
</head>
<body>
<div id="header">
<h1>A Joke Website</h1>
<nav>

Home
About
Contact

</nav>
</div>
<div id="main">
<h2>Today's Joke</h2>
<p>Why did the tomato turn red?</p>
<p>Because it saw the salad dressing!</p>
</div>
<div id="footer">
<p>© 2023 A Joke Website</p>
</div>
<script src="script.js"></script>
</body>
</html>

CSS:
```css  
body {  
background-color: #f2f2f2;  
font-family: Arial, sans-serif;  
}  
  
#header {  
background-color: #333;  
color: #fff;  
padding: 20px;  
text-align: center;  
}  
  
#header h1 {  
margin: 0;  
}  
  
#header nav ul {  
list-style: none;  
margin: 0;  
padding: 0;  
}
```

A Joke Website

Home About Contact

Today's Joke

Why did the tomato turn red?

Because it saw the salad dressing!

© 2023 A Joke Website

How well do multimodal models really work?

Describing Images



Describing Images: LLaVA

In the image, there is a large house with various objects placed inside, including a refrigerator, **an oven**, a sink, a bed, and a dining table. The house also features a green backyard with a lawn and a couple of trees. There are two small figures, possibly people, standing in the house. The backyard **has a couple of chairs and a bench**, along with **a dog and two cows near the fence**. The image is a graphical drawing, possibly a floor plan or a visual representation of a video game. As such, the dog and cows might be elements of a game, **while the people** and objects like the refrigerator, oven, and sink **are part of the overall interior layout**. The focus of the image is on the house and its outdoor area, showcasing both the interior and exterior elements in a single, cohesive display.

Describing Images



Describing Images: MiniGPT-4

This is a screenshot of a map in a game. It shows a neighborhood with houses, trees, and roads. The houses are different colors and have different shapes. There is a park in the center of the map.

User: what kind of game could it be?

It could be a city building simulation game, where the player has to build and manage a city by constructing buildings, roads, and other infrastructure. The game could also be a real-time strategy game, where the player has to manage resources and units to defeat enemies and complete objectives.

Multimodal Datasets & Evaluations

Datasets & Evaluations

- Visual question-answering ([VQAv2](#), [OK-VQA](#))
- Image captioning ([COCO](#), [Flickr30k](#))
- Image classification ([ImageNet](#))

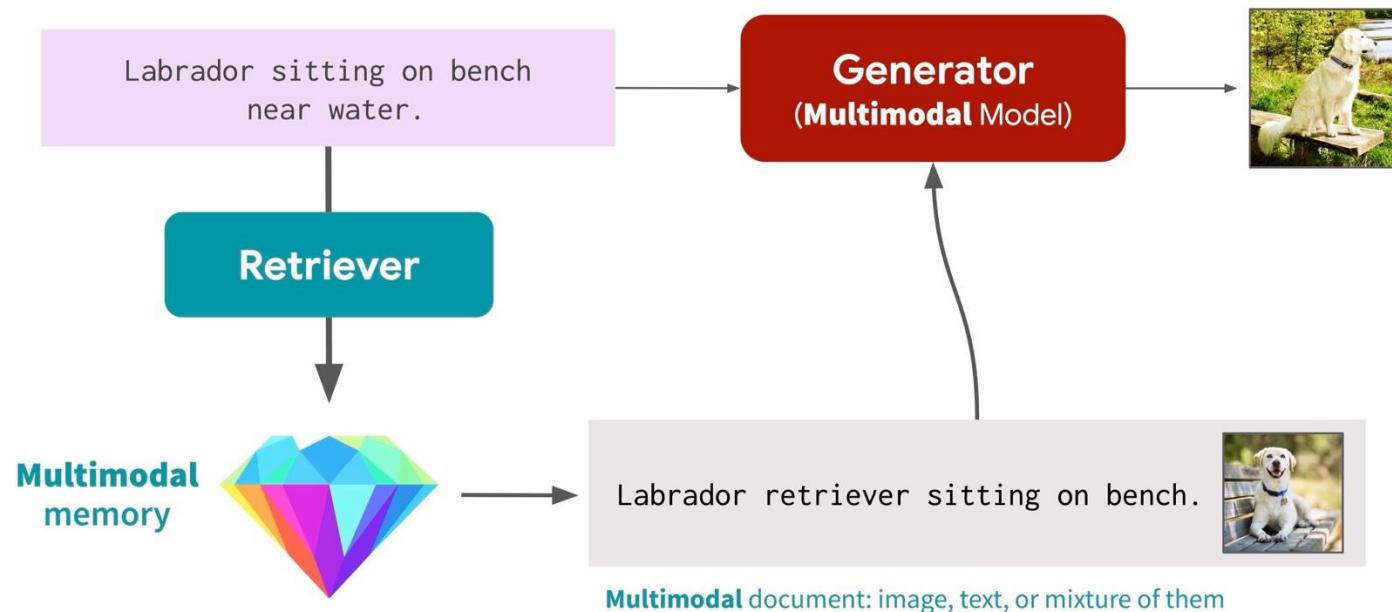
Dataset: Multimodal C4

- Open corpus of images interleaved with text by Allen Institute for AI
- 103 million documents, encompassing around 585M images and 43B tokens
- Improves OpenFlamingo image captioning when added to COCO caption dataset
- [Scripts to collect the data](#)
- [Paper](#) (April 14, 2023)

Bonus: Combine RAG and multimodal model

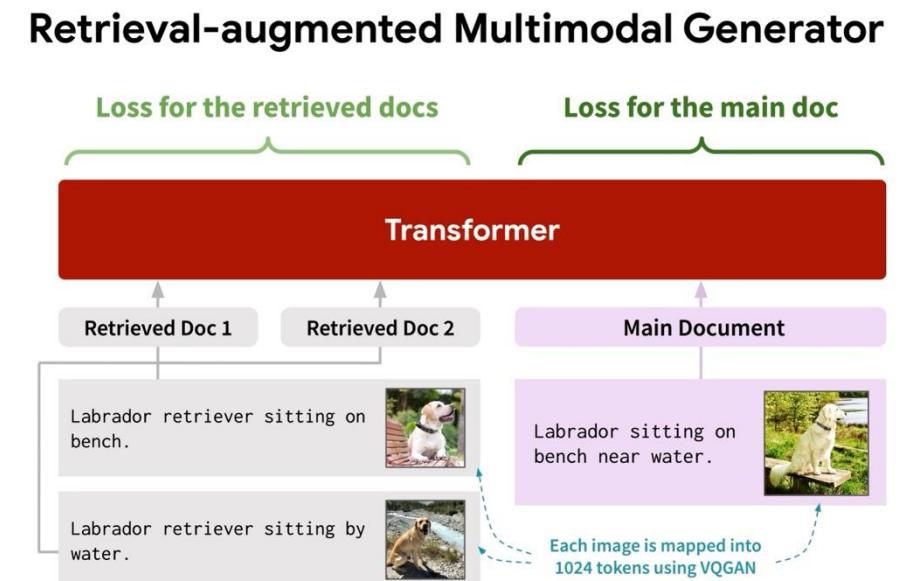
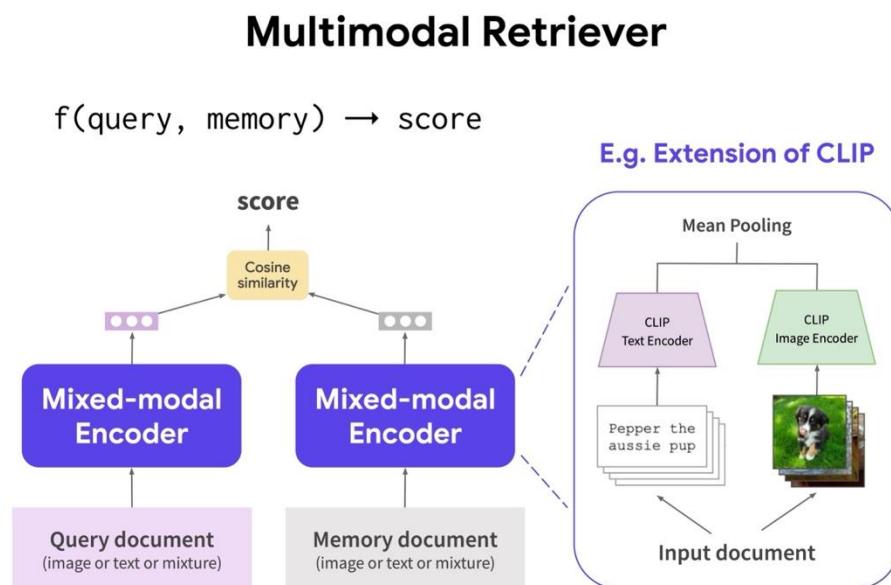
Retrieval-Augmented Multimodal Model

- RA-CM3 (Yasunaga et al., 2023)
- Adds retrieval-augmentation to CM3 multimodal model



Retrieval-Augmented Multimodal Model

- Uses a multimodal retriever
- Retrieve 1 or 2 multimodal documents
- Give as context to multimodal generator



Agents

Agent Definition

Anthropic:

Distinction between workflows and agents:

- Workflows are systems where LLMs and tools are orchestrated through predefined code paths.
- Agents, on the other hand, are systems where LLMs dynamically direct their own processes and tool usage, maintaining control over how they accomplish tasks.

Agent Definition

[OpenAI](#):

Agents are systems that independently accomplish tasks on your behalf.

[...]

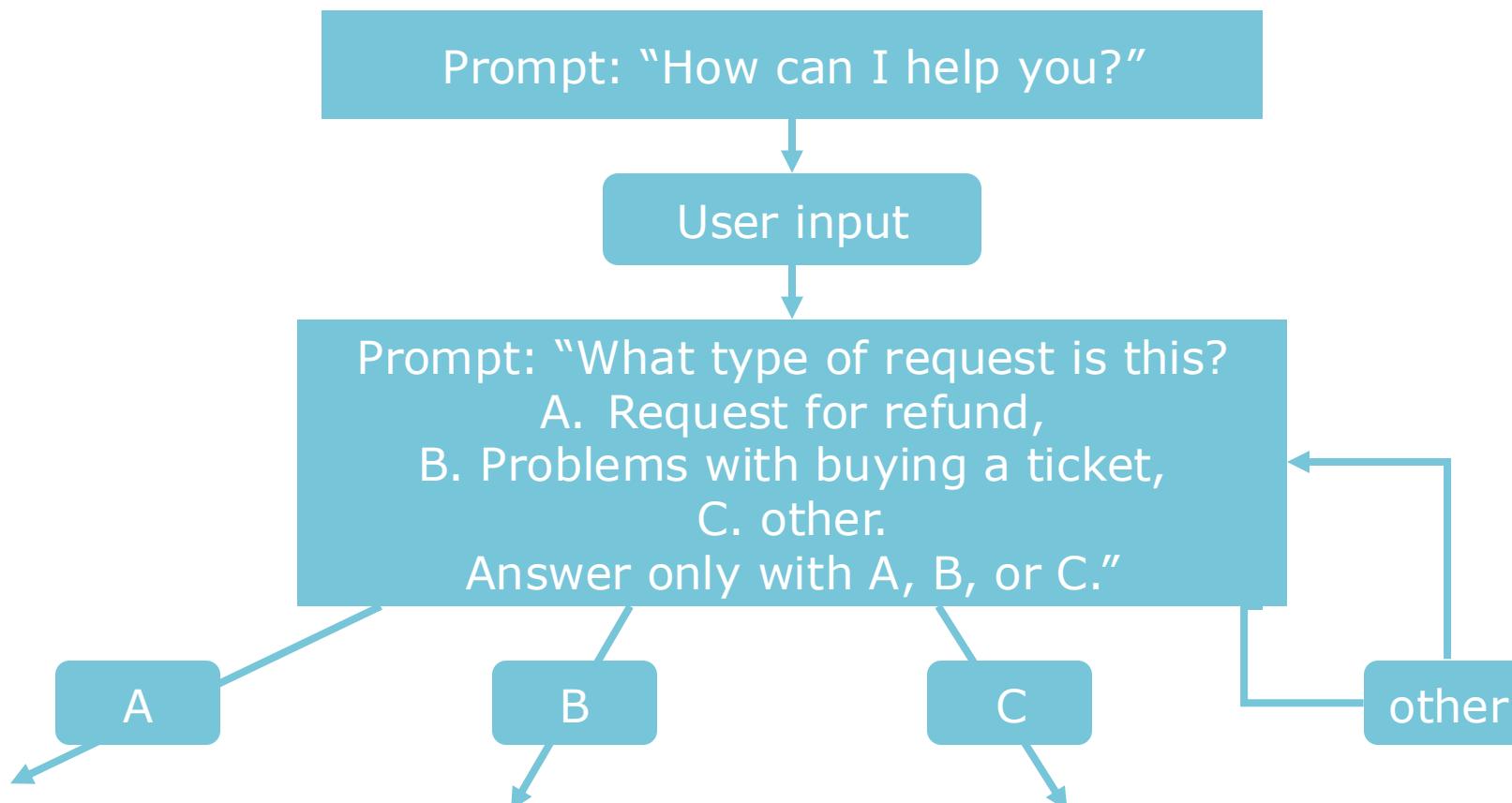
Applications that integrate LLMs but don't use them to control workflow execution—think simple chatbots, single-turn LLMs, or sentiment classifiers—are not agents.

More concretely, an agent possesses core characteristics that allow it to act reliably and consistently on behalf of a user:

1. It leverages an LLM to manage workflow execution and make decisions. It recognizes when a workflow is complete and can proactively correct its actions if needed. In case of failure, it can halt execution and transfer control back to the user.
2. It has access to various tools to interact with external systems—both to gather context and to take actions—and dynamically selects the appropriate tools depending on the workflow's current state, always operating within clearly defined guardrails.

Control Flow

- Determines the series of steps an LLM must execute



Tool Use

- OpenAI built-in tools
 - Web search
 - File search
 - Computer use
 - Function calling (custom code)
- Anthropic tools:
 - Client tools
 - Text editor
 - Computer use
 - Server tools
 - Web search

Tool Use Benchmarks

- [ToolTalk](#) (tool usage through dialogue)
- [Berkeley Function-Calling Leaderboard](#) (mostly coding, last updated in August 2024)
- [GTA: A benchmark for general tool agents](#) (based on real data, multimodal questions)
- [SWE-bench](#) (GitHub issues, tested with pull request's unit tests)
- [SWE-bench verified](#) (human-verified to be solvable)
- [SWE-bench multimodal](#) (user interfaces built with JavaScript)
- [τ-bench](#) (simulated user conversations)
- [BrowseComp](#) (locate hard-to-find information on the web)
 - [SimpleQA](#) (short, fact-seeking queries, already saturated by GPT-4o)

Model Context Protocol (MCP)

- Open standard to connect agents with data sources
- Developed by Anthropic [[blog](#)][[GitHub](#)]
- Discovery through the [registry](#)
- Implementation of data sources as MCP [servers](#)
- Example servers
 - Brave Search - Web and local search using Brave's Search API
 - GitHub - Repository management, file operations, and GitHub API integration
 - Google Drive - File access and search capabilities for Google Drive
 - Google Maps - Location services, directions, and place details
 - Slack - Channel management and messaging capabilities
 - Sqlite - Database interaction and business intelligence capabilities

Multi-agent System

- Divide a task among multiple agents that interact with each other
 - Designate one agent as the “manager” or run decentralized
 - Can use the MCP to exchange information
 - Alternative: Use a central knowledge base
 - Pros: Parallel execution, intuitive to separate concerns
 - Cons: Additional complexity, communication overhead
- If in doubt, start with a single-agent system

When to Use Agents

Anthropic:

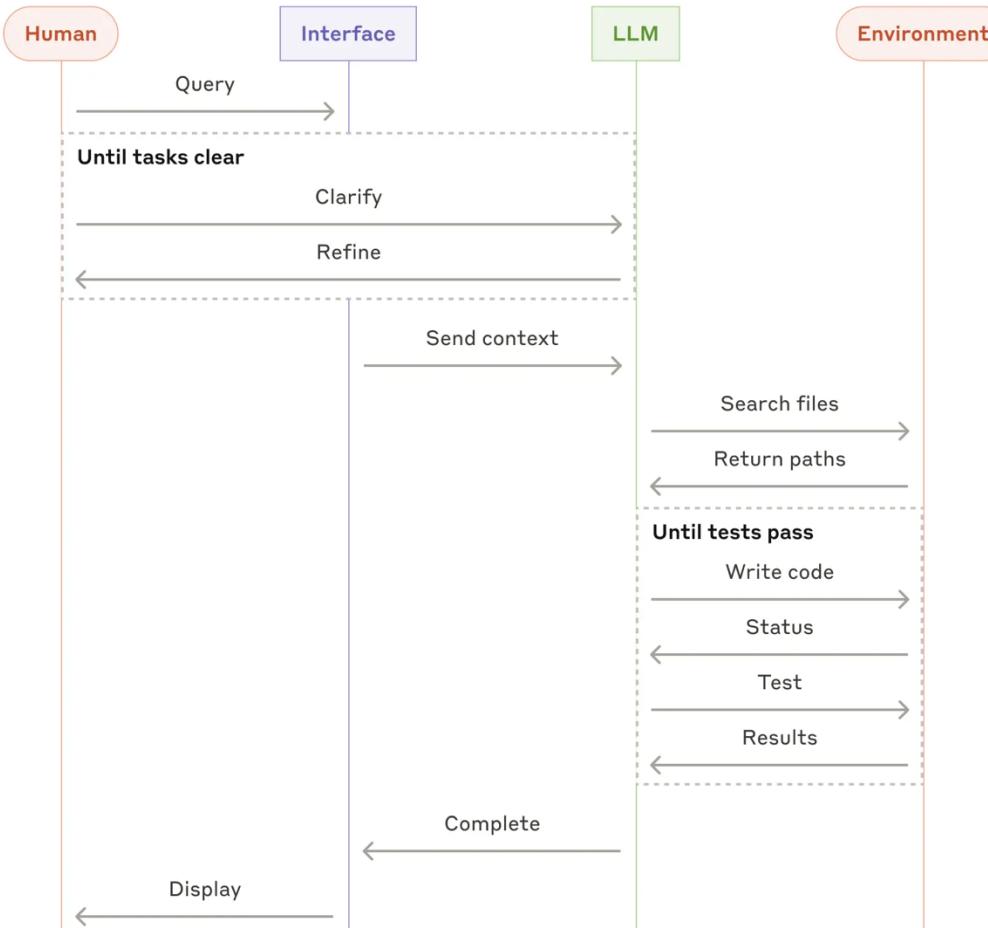
"When building applications with LLMs, we recommend finding the **simplest solution possible**, and only increasing complexity when needed. This might mean not building agentic systems at all. Agentic systems often **trade latency and cost for better task performance**, and you should consider when this tradeoff makes sense.

When more complexity is warranted, workflows offer predictability and consistency for well-defined tasks, whereas agents are the better option when **flexibility and model-driven decision-making are needed at scale**. For many applications, however, optimizing single LLM calls with retrieval and in-context examples is usually enough."

Use Cases

- Customer support
 - Support interactions naturally follow a conversation flow while requiring access to external information and actions
 - Tools can be integrated to pull customer data, order history, and knowledge base articles
 - Actions such as issuing refunds or updating tickets can be handled programmatically
 - Success can be clearly measured through user-defined resolutions
- Coding
 - Code solutions are verifiable through automated tests
 - Agents can iterate on solutions using test results as feedback
 - The problem space is well-defined and structured
 - Output quality can be measured objectively

Coding Agent Control Flow



Code Examples

- Anthropic examples
 - [Calculator tool](#)
 - [Customer service agent](#)
 - [JSON extractor](#)
- OpenAI [examples](#), include:
 - Agent patterns
 - Airline customer service
 - Financial research agent
 - Handoffs (passing control between agents)
 - MCP

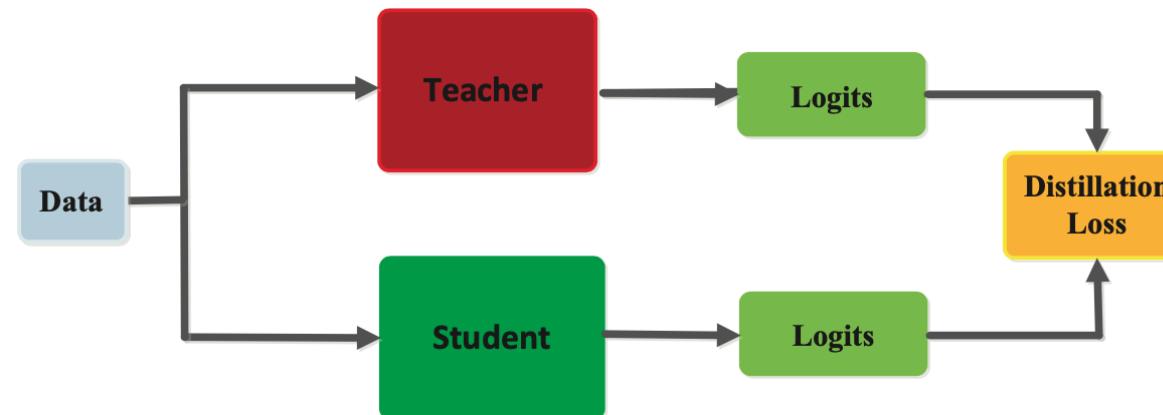
Further Reading (and Watching)

- OpenAI's [practical guide to building agents](#)
 - [Function calling](#) video
 - [Assistants & agents](#) video
- Anthropic's [blog post on building effective agents](#)
 - [YouTube video](#) for the above blog post
 - YouTube video on [Claude plays Pokemon](#)
- [AI tinkerers YouTube](#) channel has recent agent videos

Efficiency Techniques

Knowledge Distillation

- [Hinton et al., 2015](#)
- Train a smaller student model on the *predictions* of a large teacher model
 - Much more training signal in a single example than with cross-entropy loss to a ground-truth class label
- Student model gets scores almost as good as the teacher



Knowledge Distillation

- ChatGPT most likely same size as GPT-3 (175B parameters)
- gpt-3.5-turbo apparently 20B
- Speculation: OpenAI uses this technique for “turbo” models
 - Smaller models: Much cheaper to run
 - Similar performance

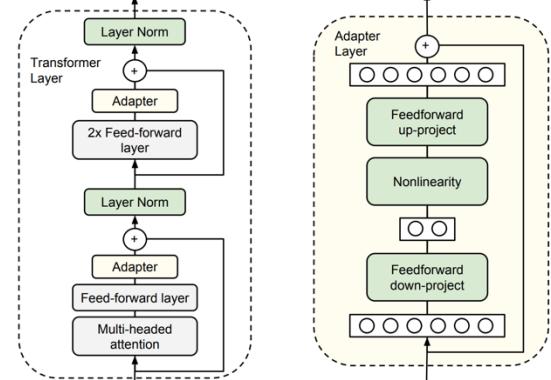
Felix (@felix_red_panda) · ... Microsoft paper claims ChatGPT 3.5 has ~20 billion parameters arxiv.org/abs/2310.17680

System description		Python (0)	
System	Model	#P	top-1
T5	t5-large	770M	80.4
CodeT5	codet5-large	770M	80.5
GPT-3	text-davinci-003	175B	82.5
ChatGPT	gpt-3.5-turbo	20B	80.6
StarCoder	starcoder	15.5B	79.2
CodeT5+	codet5p-16b	16B	79.6
CodeGen	codegen-350m	350M	80.1
Diffusion-LM	Custom	50M	70.4
GENIE	Custom	93M	73.2
CODEFUSION	Custom	75M	80.7

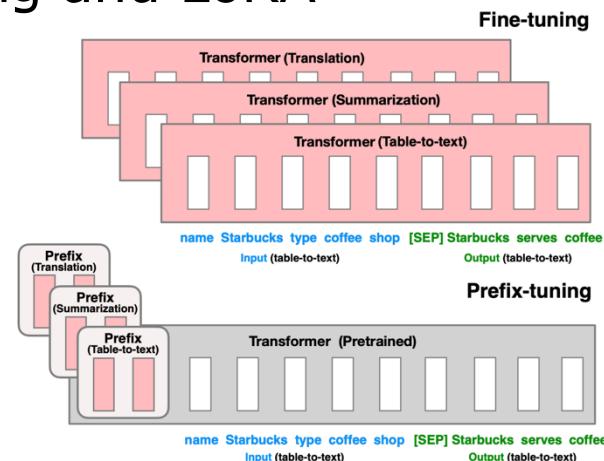
10:03 · 30.10.23 aus Earth · 285 Mal angezeigt

Parameter-Efficient Finetuning (PEFT)

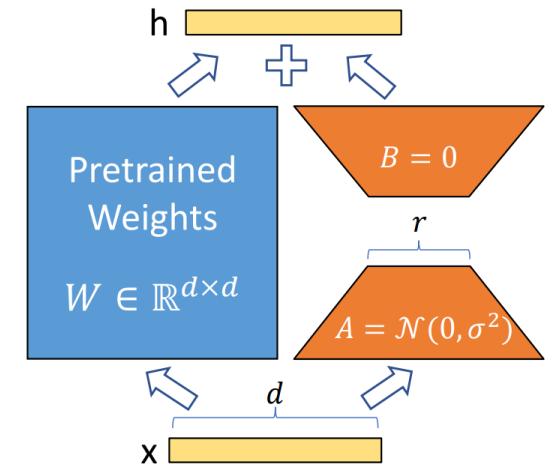
- Adapter layers ([Houlsby et al., 2019](#); [Pfeiffer et al., 2020](#))
- Prompt tuning ([Lester et al., 2021](#)), Prefix tuning ([Li and Liang, 2021](#)), P-tuning ([Liu et al., 2021](#))
- Low-rank adaptation (LoRA; [Hu et al., 2022](#))
→ Hugging Face has the [adapter-transformers](#) library for adapter layers, and the [PEFT](#) library for prompt tuning and LoRA



Adapter layers ([Houlsby et al., 2019](#))



Prefix tuning ([Li and Liang, 2021](#))

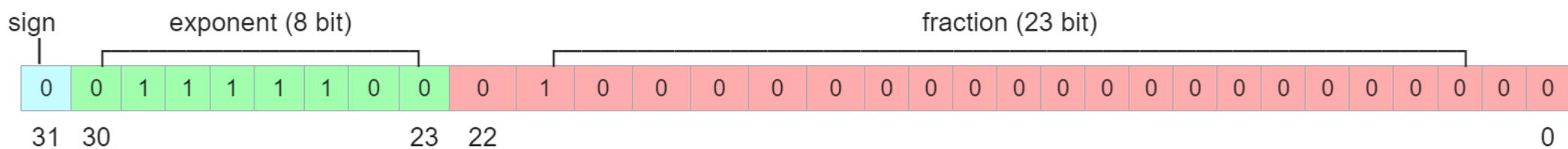


LoRA ([Hu et al., 2022](#))

Quantization

- 32-bit floating point
 - Sign bit: positive/negative number
 - 8 exponent bits, 23 fraction/mantissa/significand bits

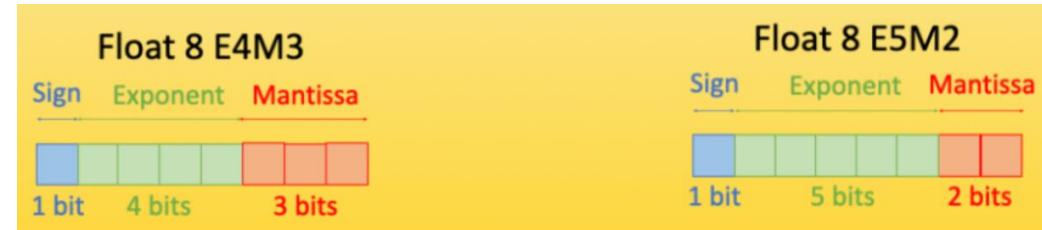
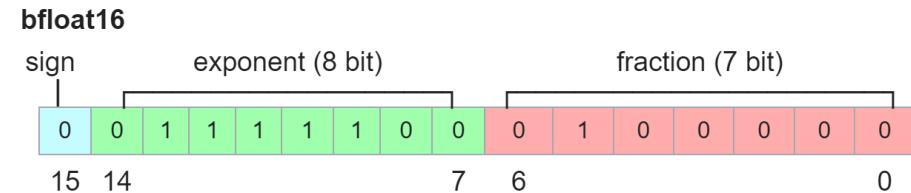
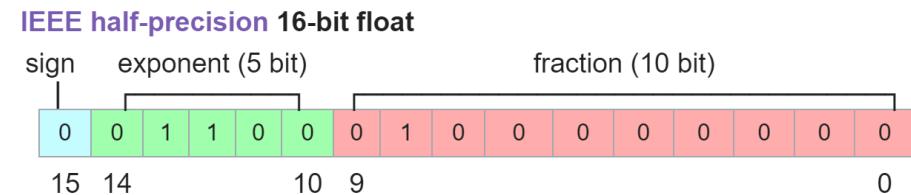
IEEE 754 single-precision 32-bit float



- Number = $(-1)^{\text{signbit}} * 2^{\text{exponentbits}-127} * 1.\text{significandbits}$

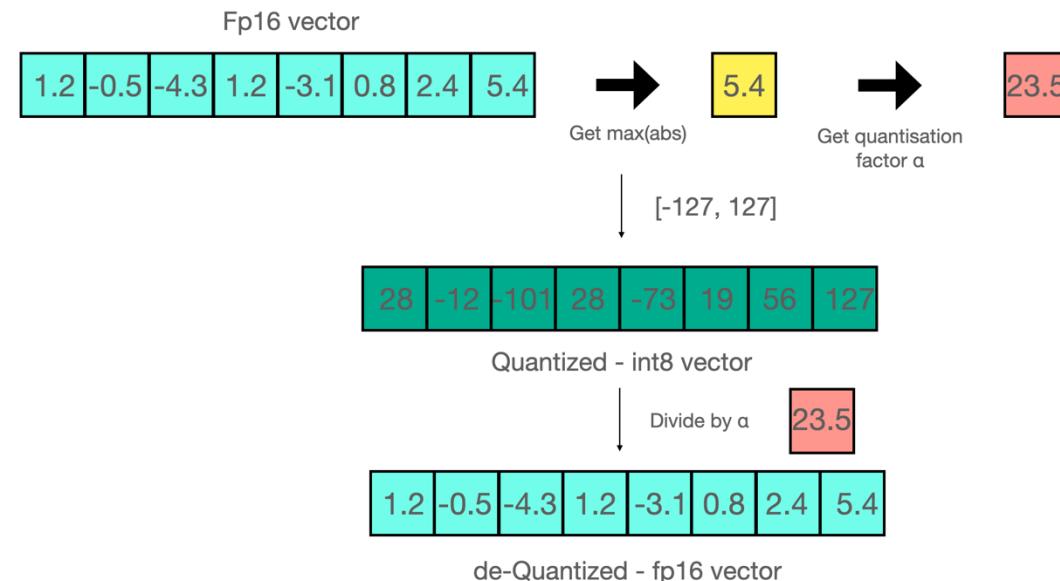
Quantization Data Types

- 16-bit floating point (fp16)
 - 16-bit BrainFloat (bfloat16)
 - 8-bit integer (int8)
 - 8-bit floating point (fp8)
 - E4M3 or E5M2 formats



Quantization Data Types

- Can improve precision with a *quantization constant/factor*
- Quantizing and dequantizing numbers:



Quantization Data Types

- 4-bit floating point (fp4)
 - E2M1 or E3M0 format
- 4-bit NormalFloat (NF4; [Dettmers et al., 2023](#))
 - Uses quantile quantization: Each quantization bin has an equal number of elements from the input tensor
- 2-bit integer (INT2; [Chai et al., 2023](#))
- 2-bit floating point ([Chee et al., 2023](#))
- Sub-1-bit for mixture-of-experts model ([Frantar and Alistarh, 2023](#))

Post Training Quantization

- Train with 32-bit floating points, then quantize the weights
 - Current (end of 2023) consensus is that this loses 1-2 points of accuracy on benchmarks
 - Better to use quantization already during training (see QLoRA)
- Quantization methods:
 - AWQ ([Lin et al., 2023](#))
 - GPTQ ([Frantar et al., 2023](#))
 - GGUF (format for inference on laptop: [llama.cpp](#))
- Some quantized models are available on [Hugging Face](#)

Quantized LoRA (QLoRA)

- [Dettmers et al., 2023](#): Combine quantization with LoRA
 - Uses 4-bit NormalFloat for storage and 16-bit bfloat for computation
 - Uses double quantization (quantizes the quantization constants) to save another 0.5 bits per parameter
 - Can achieve equal performance to full finetuning when finetuning with 4-bit quantization
 - Just applying quantization at inference time can hurt performance

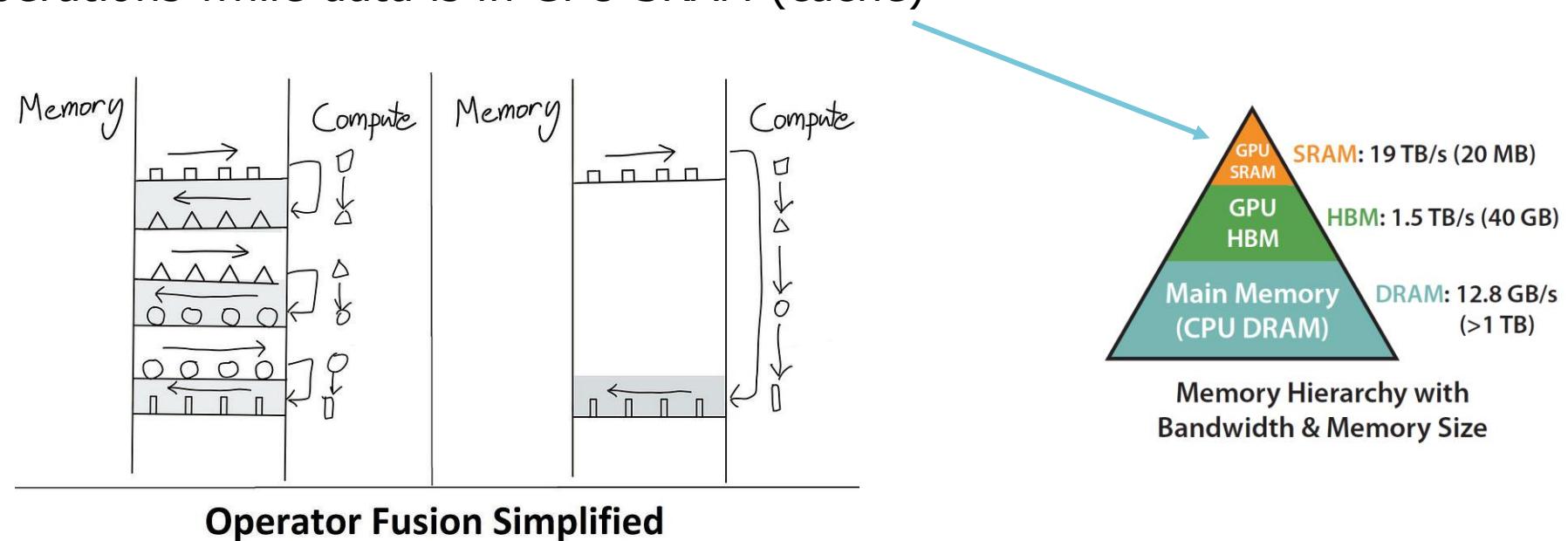
QLoRA GPU Memory

- Allows training 65B models on GPUs with 48GB memory (Nvidia A6000)
- 13B models on Google Colab and the GPUHub (16GB GPUs)
- [Guanaco demo](#) from QLoRA paper
 - Uses Hugging Face's PEFT library

Model / Dataset	Params	Model bits	Memory
GPT-4	-	-	-
Bard	-	-	-
Guanaco	65B	4-bit	41 GB
Alpaca	65B	4-bit	41 GB
FLAN v2	65B	4-bit	41 GB
Guanaco	33B	4-bit	21 GB
Open Assistant	33B	16-bit	66 GB
Alpaca	33B	4-bit	21 GB
FLAN v2	33B	4-bit	21 GB
Vicuna	13B	16-bit	26 GB
Guanaco	13B	4-bit	10 GB
Alpaca	13B	4-bit	10 GB
HH-RLHF	13B	4-bit	10 GB
Unnatural Instr.	13B	4-bit	10 GB
Chip2	13B	4-bit	10 GB
Longform	13B	4-bit	10 GB
Self-Instruct	13B	4-bit	10 GB
FLAN v2	13B	4-bit	10 GB
Guanaco	7B	4-bit	5 GB
Alpaca	7B	4-bit	5 GB
FLAN v2	7B	4-bit	5 GB

FlashAttention

- Version 1 ([Dao et al., 2022](#)) [[blog post](#)][[integration in ML frameworks](#)]
- Reduces the number of times the GPU memory needs to be accessed during the attention computation
 - Operator fusion: Instead of writing intermediate results back to memory, perform all operations while data is in GPU SRAM (cache)



Running Models Locally

- [llama.cpp](#): Inference in C/C++
- [llamafile](#): Distribute LLMs as a single executable file
- [GPT4All](#): Desktop application for all 3 major OS
- [Ollama](#): Customizable

How should you decide?

- Compare available models, UIs, integration into codebase
 - Best choice depends on your use case
- Try them out yourself!

Interpretability and Explainability

Definition

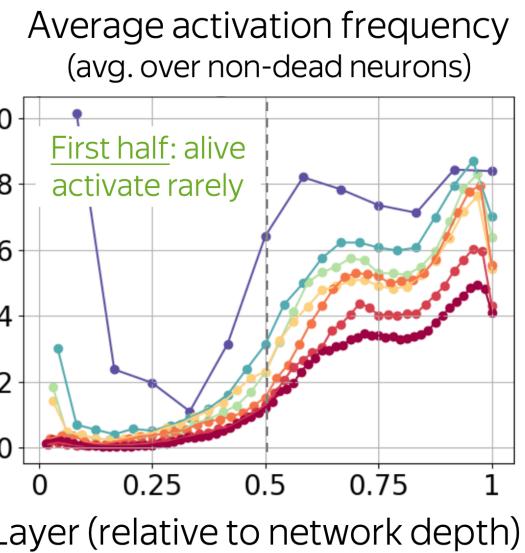
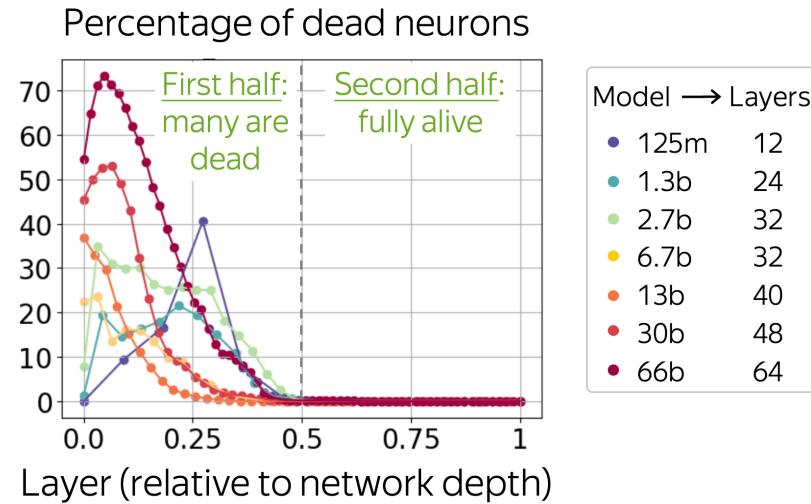
- Interpretability: Understand how the model works
- Explainability: Explain the predictions

Interpretability

How does the model work?

Neuron Activation

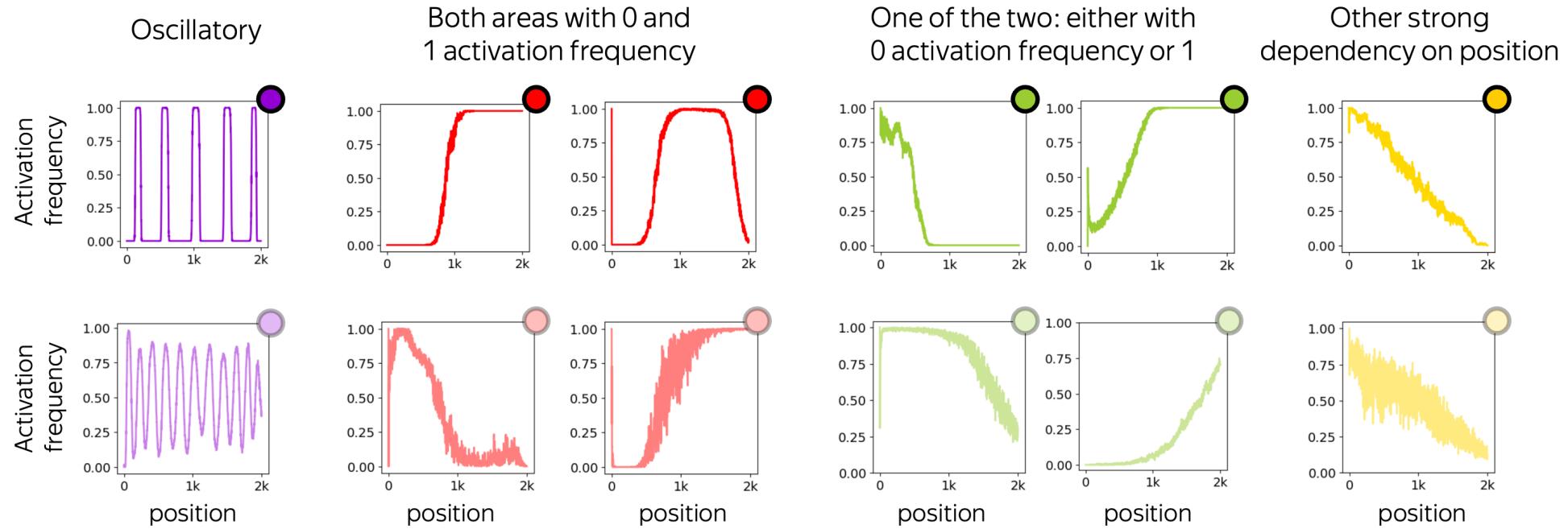
- Look at activations (absolute value) across layers:



- Hypothesis: Lower layers encode few “simple” concepts, higher layers encode many semantic concepts

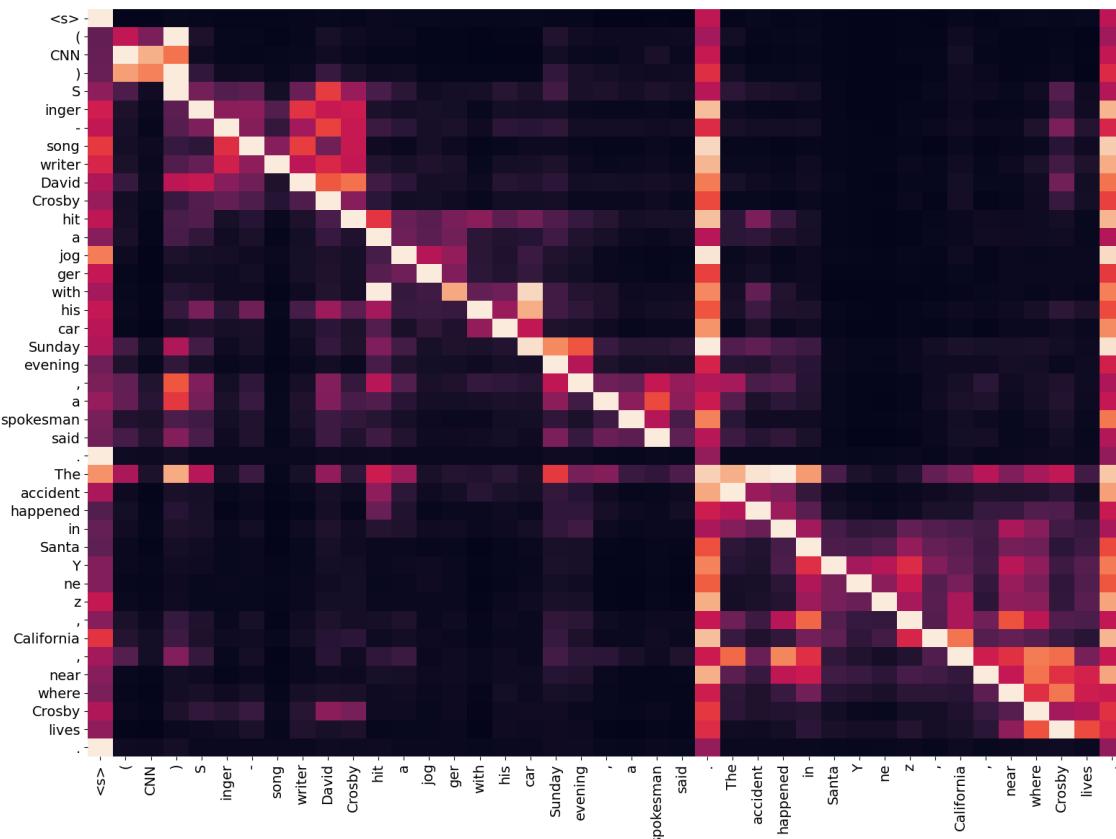
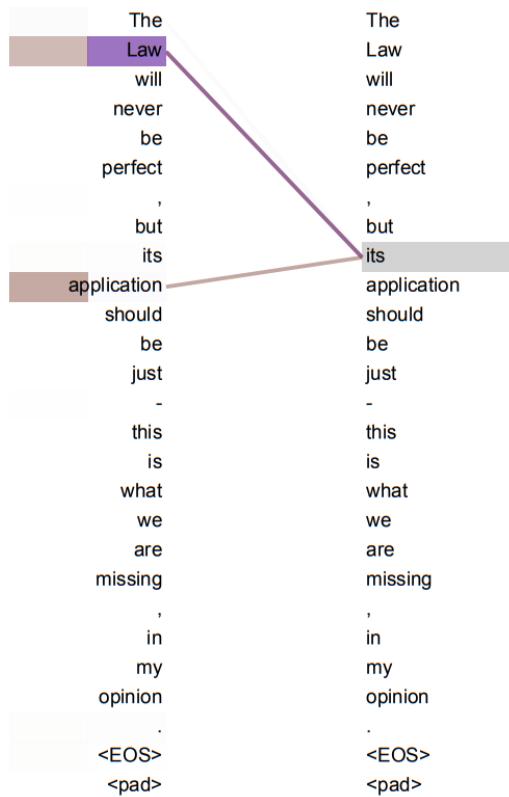
Neuron Activation

- For which input positions is a neuron highly active?



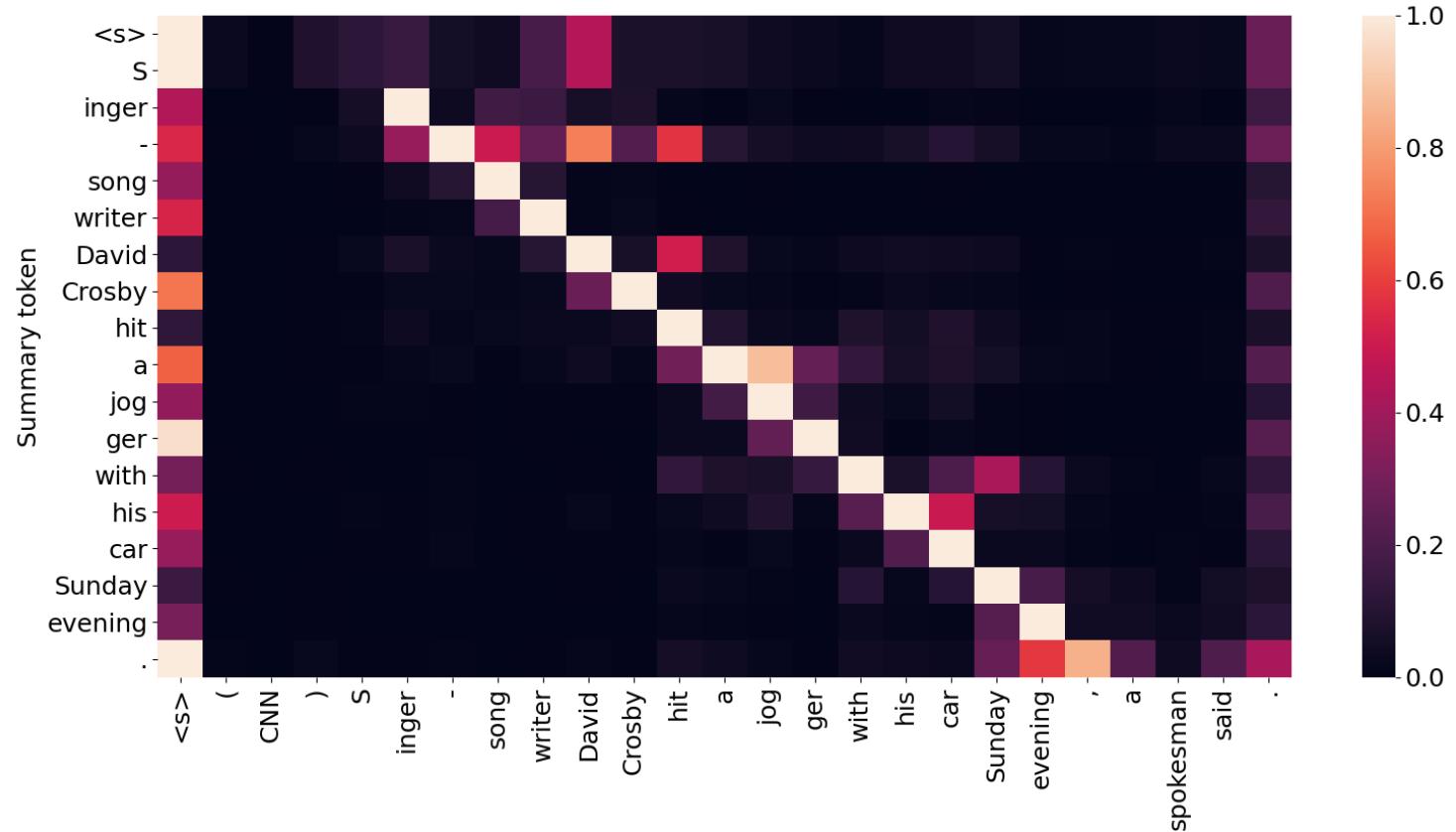
Attention Visualization

- Self-attention: Find dependencies between words and sentences



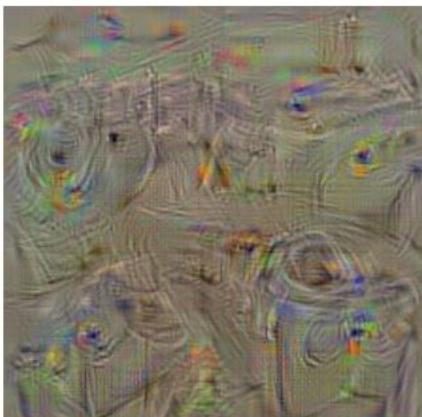
Attention Visualization

- Cross-attention: Does the summary copy from the input?



Gradient Attribution

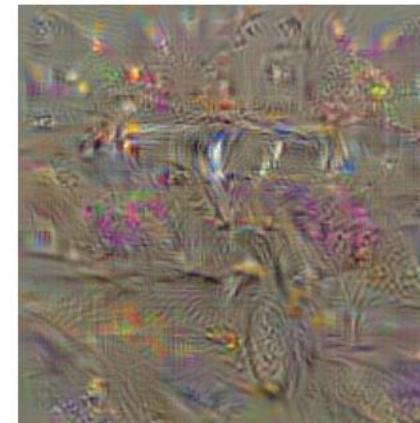
- Which input maximally activates a class neuron?
 - [Deconvnet](#) (start with the output, reverse NN operations)
 - Use numerical optimization (gradient descent) to find the optimal input image for a class ([Erhan and Bengio](#), 2009):



washing machine



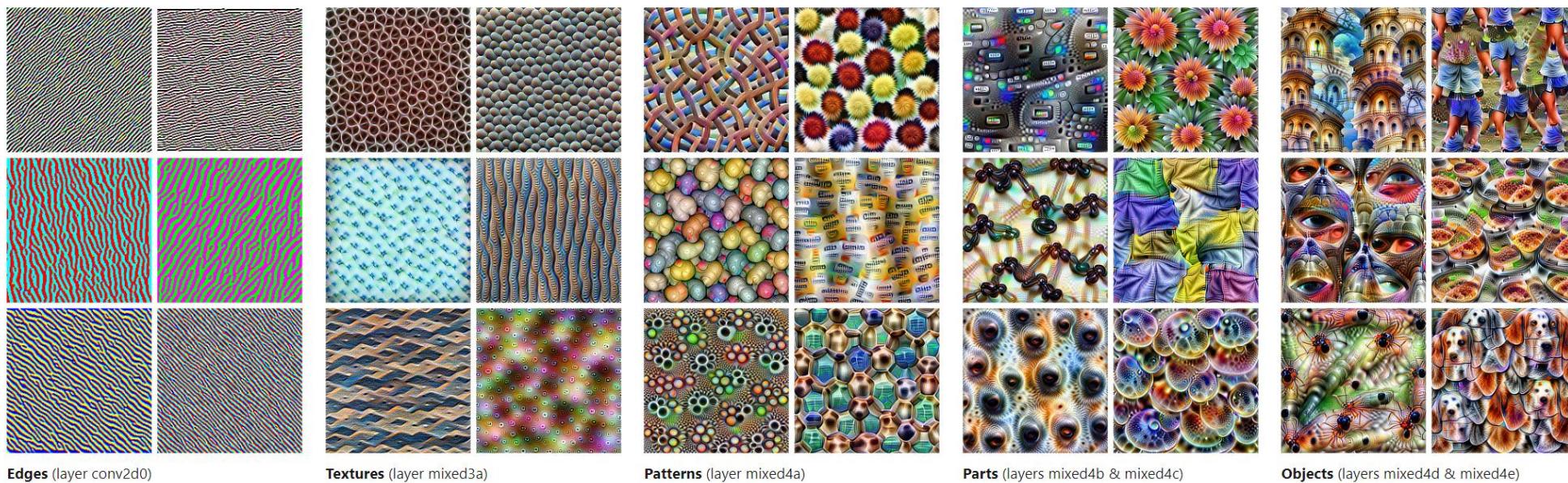
goose



limousine

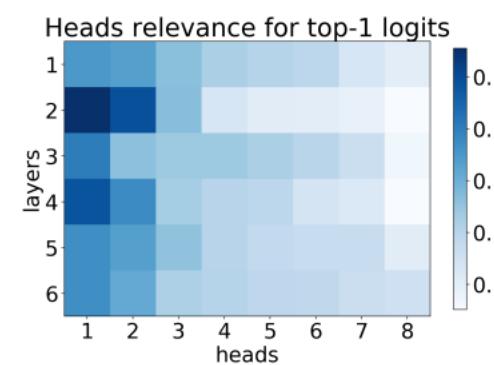
Feature Neuron Visualization

- What input maximally activates a neuron in an intermediate layer (as opposed to class neurons)?
 - Early layers: edge detectors
 - Higher layers: objects

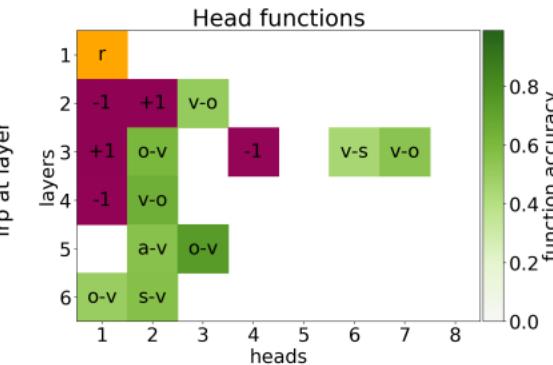


Layer-Wise Relevance Propagation (LRP)

- Alternative to gradient attribution
- Recursively propagates importance of each neuron back through the layers
- Used in Voita et al., 2019: Only few attention heads contribute substantially to the final predictions
 - Heads learn positional and syntactic functions, and attend to rare words
 - Can prune other heads without losing performance



(a) LRP (EN-DE)



(b) head functions

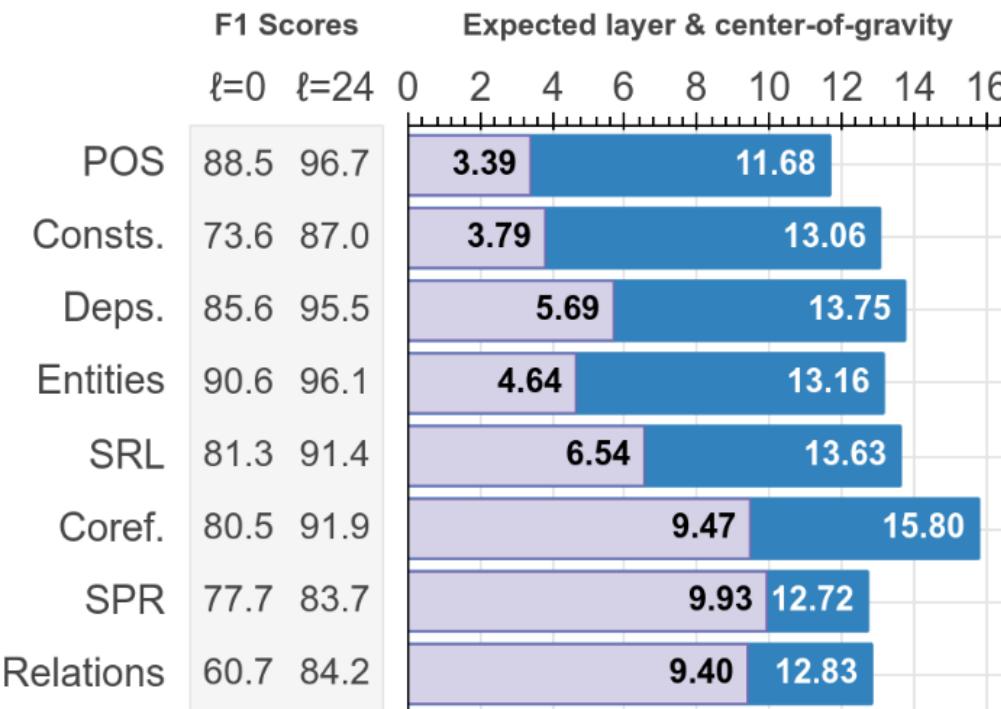
- Best practices for LRP: [Kohlbrenner et al., 2020](#)

Probing

- Goal: Find if a model encodes certain information in its activations
- Idea: If this is the case, we should be able to use them for classification
- Implementation: Add a linear layer as classifier and evaluate on linguistic tasks

Probing

- BERT encodes linguistic functions in different layers

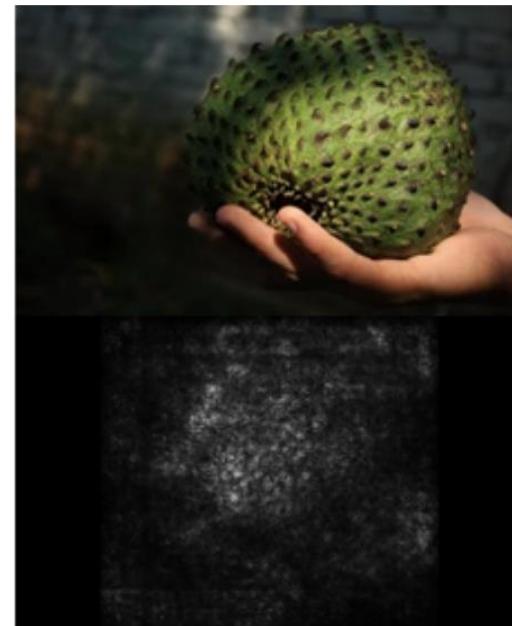
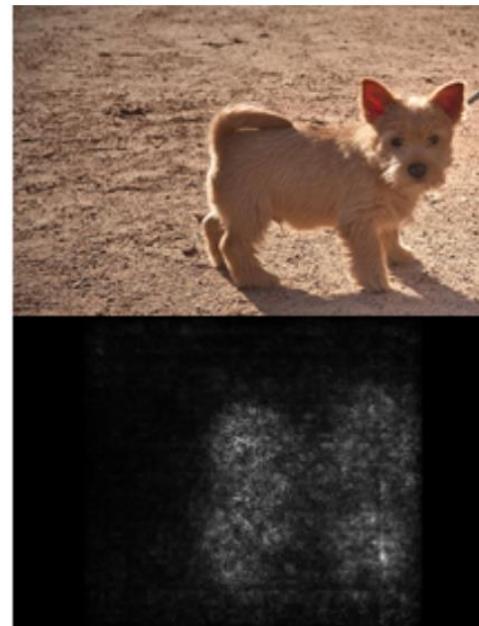


Explainability

How did the model come to its predictions?

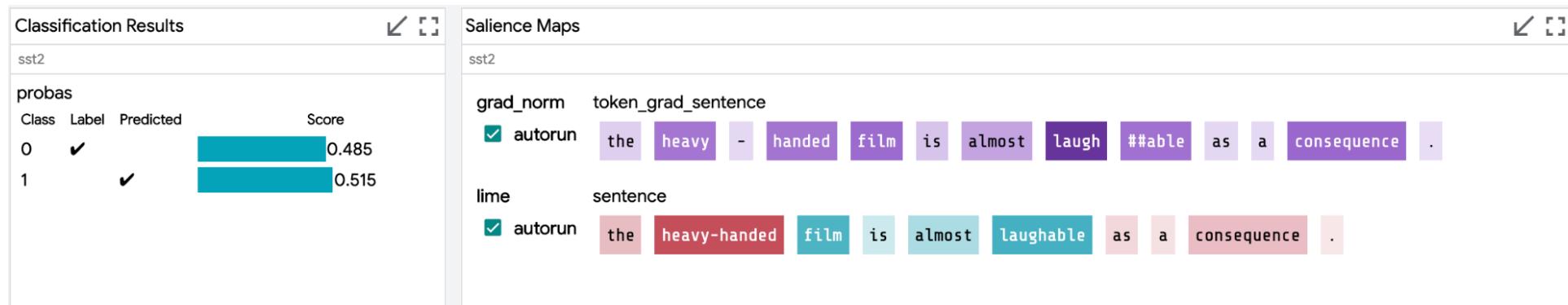
Saliency Maps (Vision)

- Determine each input pixel's importance for prediction
- Take derivative of class score wrt. input image



Saliency Maps (NLP)

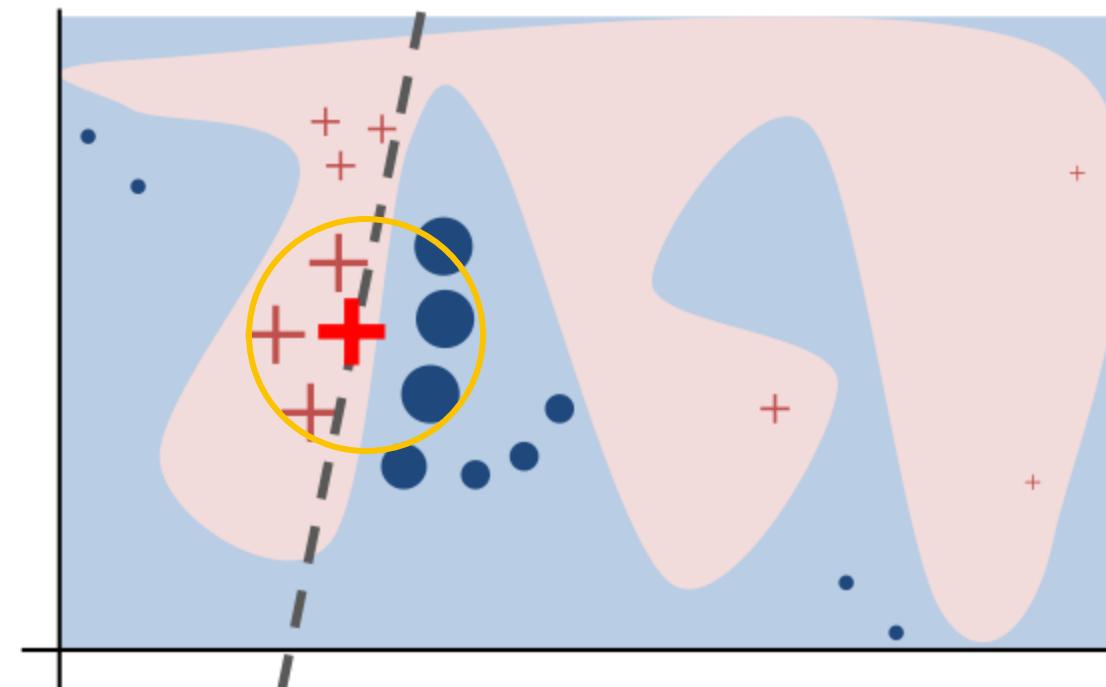
- Determine each input token's importance for prediction
- Wrongly predicted a positive sentiment:



- “laughable” incorrectly raises positive score
 - So does “film”

LIME

- Local Interpretable Model-agnostic Explanations
- Local explanations are easier to give than global ones:



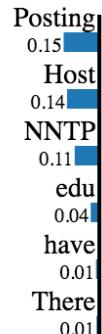
LIME

- Ribeiro et al., 2016
- Works for vision and text
- Which inputs are most important for a predicted class

Prediction probabilities



atheism christian



Text with highlighted words

From: johnchad@triton.unm.edu (jchadwic)

Subject: Another request for Darwin Fish

Organization: University of New Mexico, Albuquerque

Lines: 11

NNTP-Posting-Host: triton.unm.edu

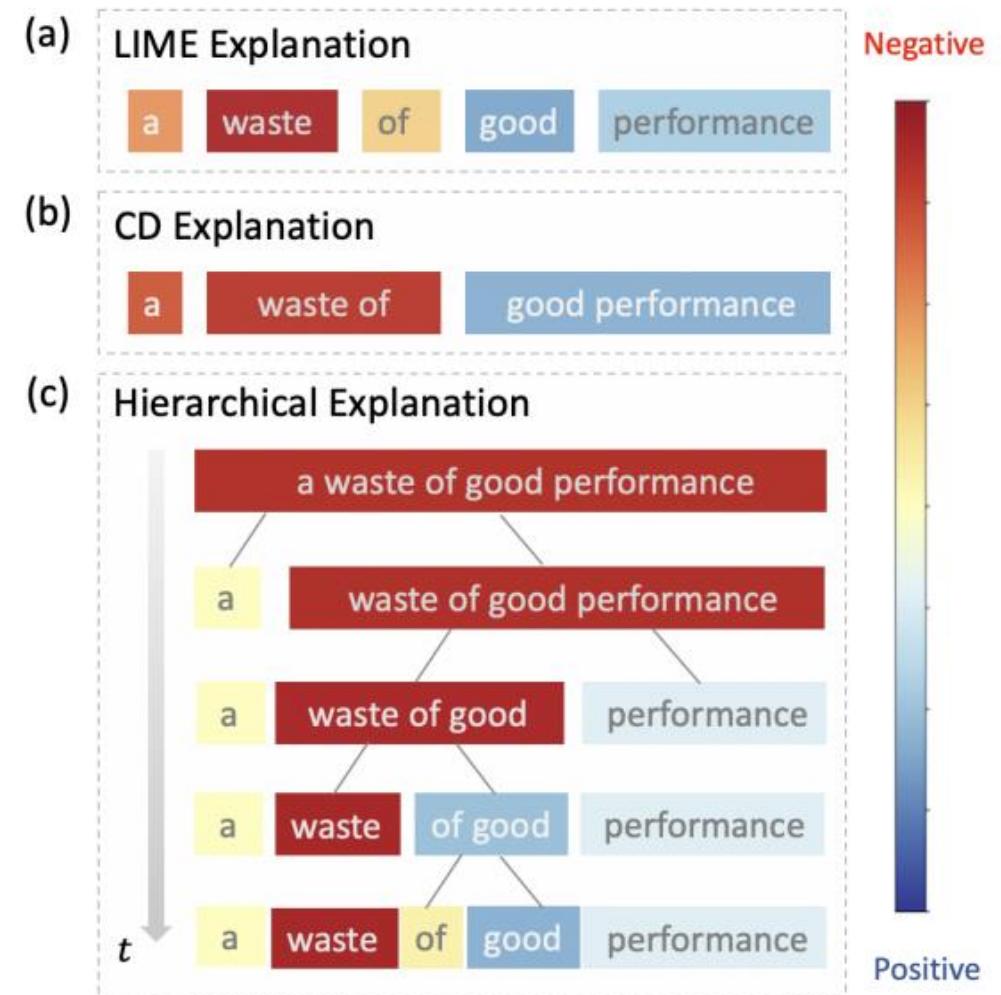
Hello Gang,

There have been some notes recently asking where to obtain the DARWIN fish.

This is the same question I have and I have not seen an answer on the net. If anyone has a contact please post on the net or email me.

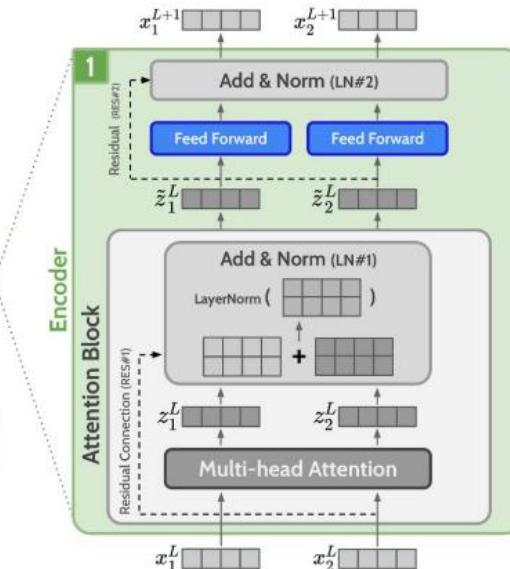
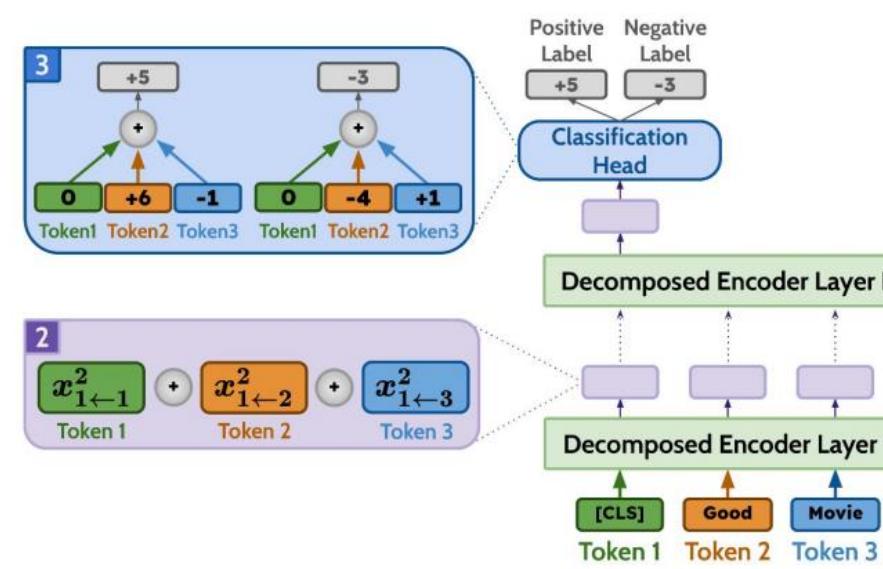
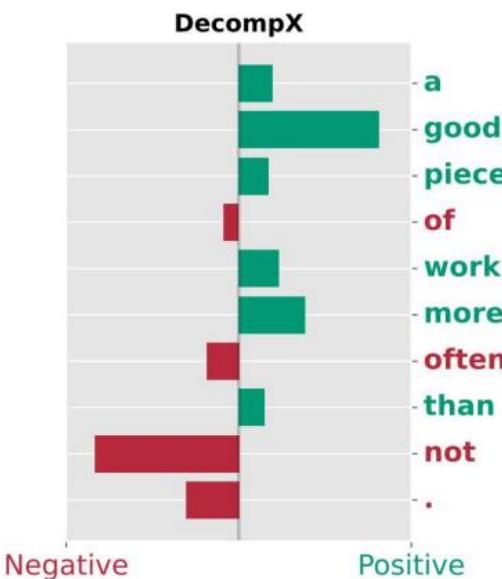
Hierarchical Explanation

- Visualizes interactions between input features
- Hierarchical view can give more intuitive explanation



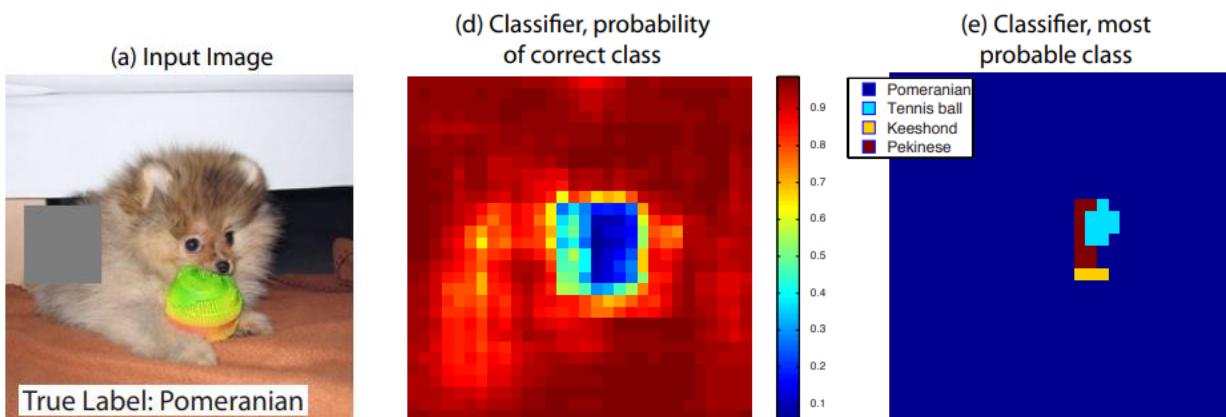
DecompX

- Modarressi et al., 2023
- Keep token contribution to labels separate during forward pass (specific to Transformers)
- More accurate approximation than previous methods



Erasures Techniques

- Remove parts of the input
- See if label (or how much the model confidence) changes



Example for vision: Occlude part of the image and see if predicted label changes.

Example for text: Remove the least important word until the label flips.

SQuAD

Context: The Panthers used the San Jose State practice facility and stayed at the San Jose Marriott. The Broncos practiced at Stanford University and stayed at the Santa Clara Marriott.

Question:

- (0.90, 0.89) Where did the Broncos practice for the Super Bowl ?
- (0.92, 0.88) Where did the practice for the Super Bowl ?
- (0.91, 0.88) Where did practice for the Super Bowl ?
- (0.92, 0.89) Where did practice the Super Bowl ?
- (0.94, 0.90) Where did practice the Super ?
- (0.93, 0.90) Where did practice Super ?
- (0.40, 0.50) did practice Super ?

Linear RNNs

Motivation

- RNNs can't be parallelized
 - Computation of next time step requires all information from previous time step
- Can some formulations be relaxed to parallelize more/all computation?

Linear RNNs

- Main idea: Use a linear hidden state update

$$h_t = \Lambda h_{t-1} + Bx_t$$

- Λ, B : learnable weights

Linear RNNs

- Main idea: Use a linear hidden state update

$$h_t = \Lambda h_{t-1} + Bx_t$$

- Λ, B : learnable weights

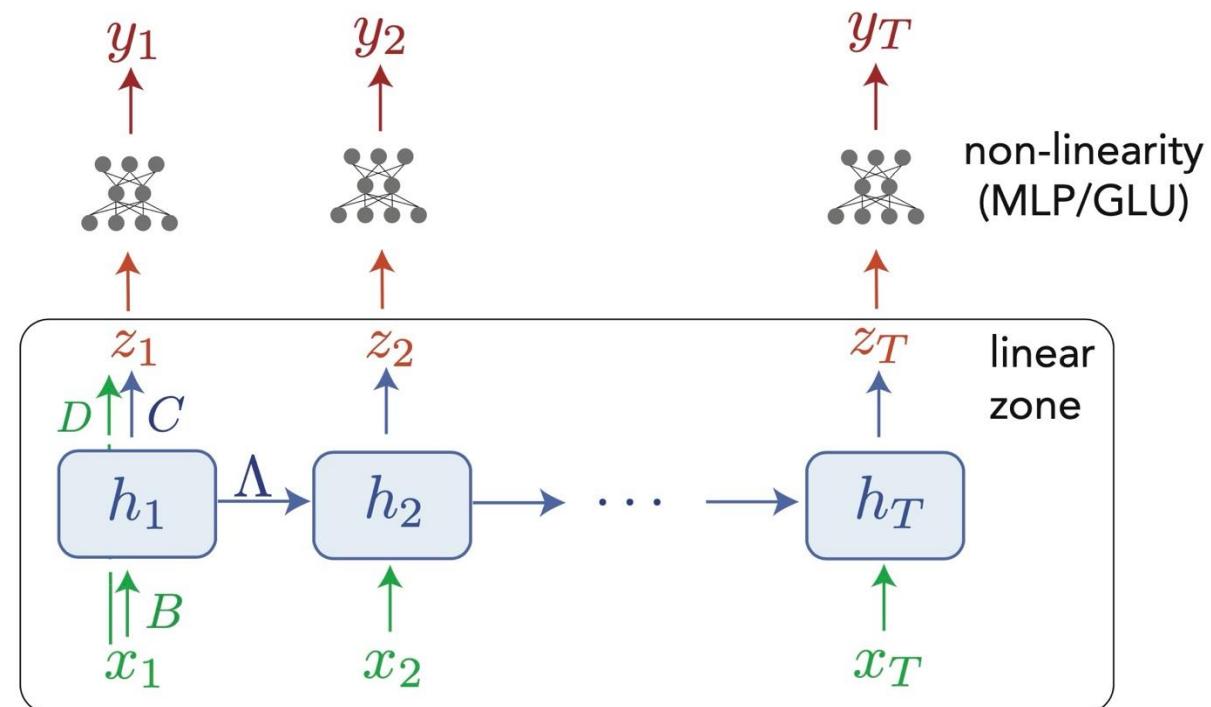
Careful:

- $\Lambda > 1$: exploding gradients
- $\Lambda < 1$: vanishing gradients

Linear RNNs

- Non-linearities should not occur for internal states
 - Can be used for computing outputs
- Now we can compute every h_t without having to wait for previous time steps to finish:

$$h_t = \sum_{k=0}^t \Lambda^k B x_{t-k} + \Lambda^t h_0$$



Aside: Why use non-linearities?

Universal approximation theorem ([Wikipedia](#)):

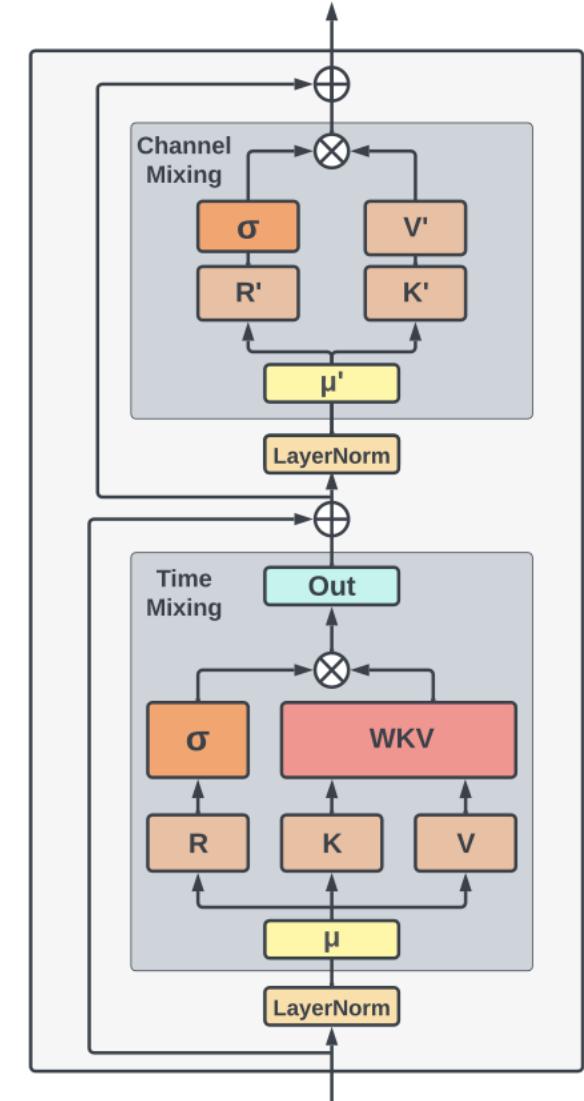
Feed-forward networks with 2 hidden layers and a non-linearity can approximate any possible function
([Maiorov and Pinkus, 1999](#); [Guliyev and Isamilov, 2018](#))

Retentive Network (RetNet)

- [Sun et al., 2023](#)
- Dual formulation for fast training (Transformer-like) and fast inference (RNN-like)
- Replaces non-linearity in Transformers (softmax on key-value attention) with a linear Hadamard product, and adds a group norm
- Adds a decay to avoid exploding gradients and allows the network to forget

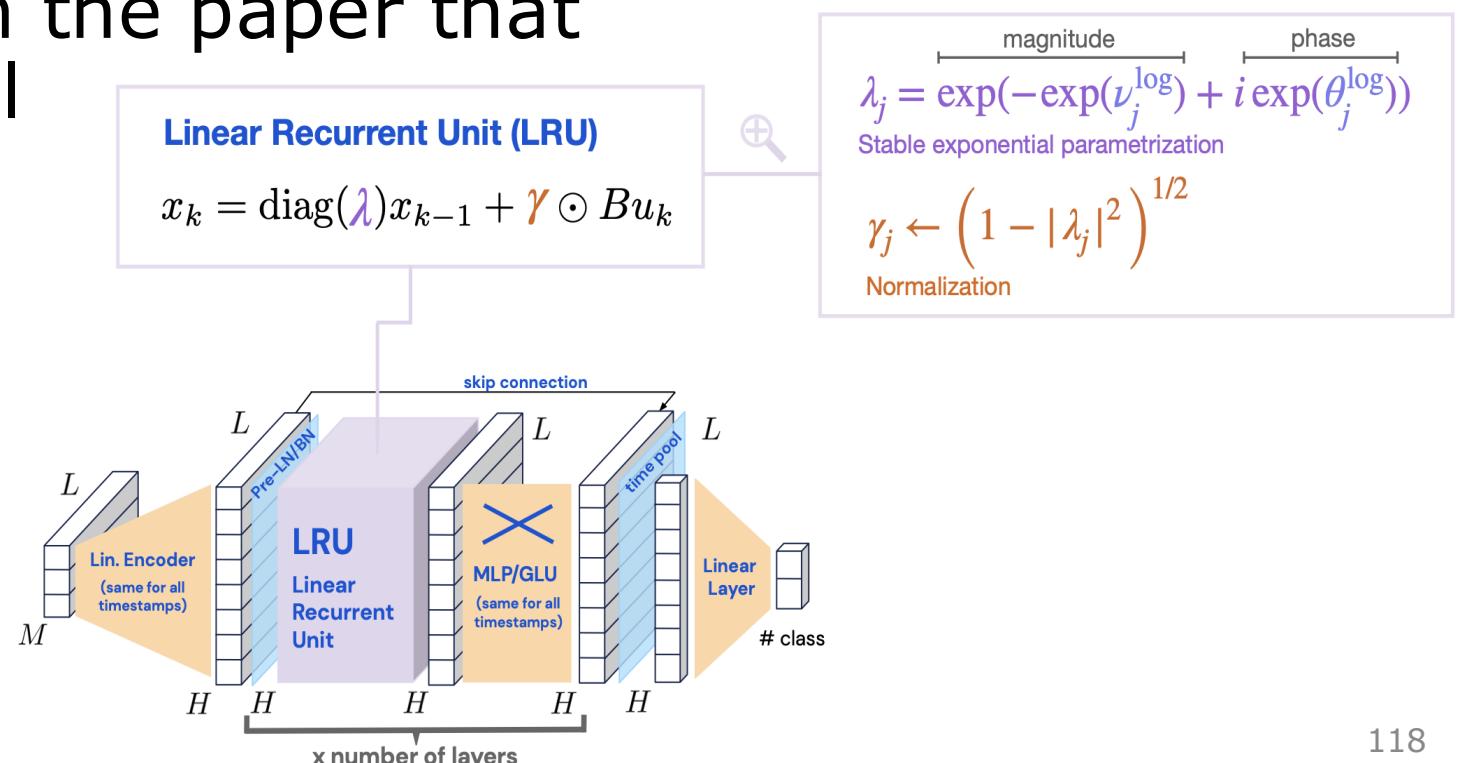
RWKV

- [Peng et al., 2023](#)
- Combines ideas from Transformer and RNN architectures
 - Same structure as a Transformer layer
- Receptance vector R to carry information from the past
- Positional weight decay W , forget information over time
- Updated with more recent innovations
(now at version 6; <https://www.rwkv.com>)



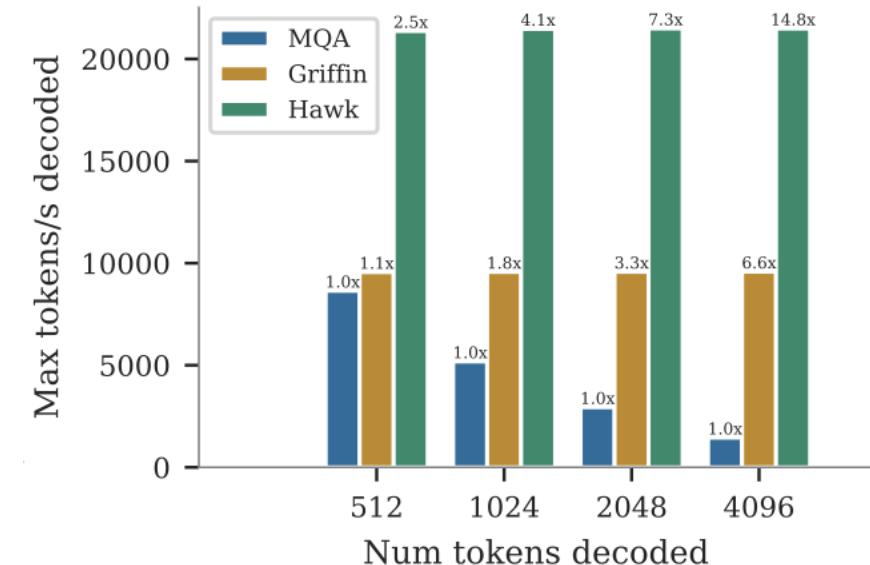
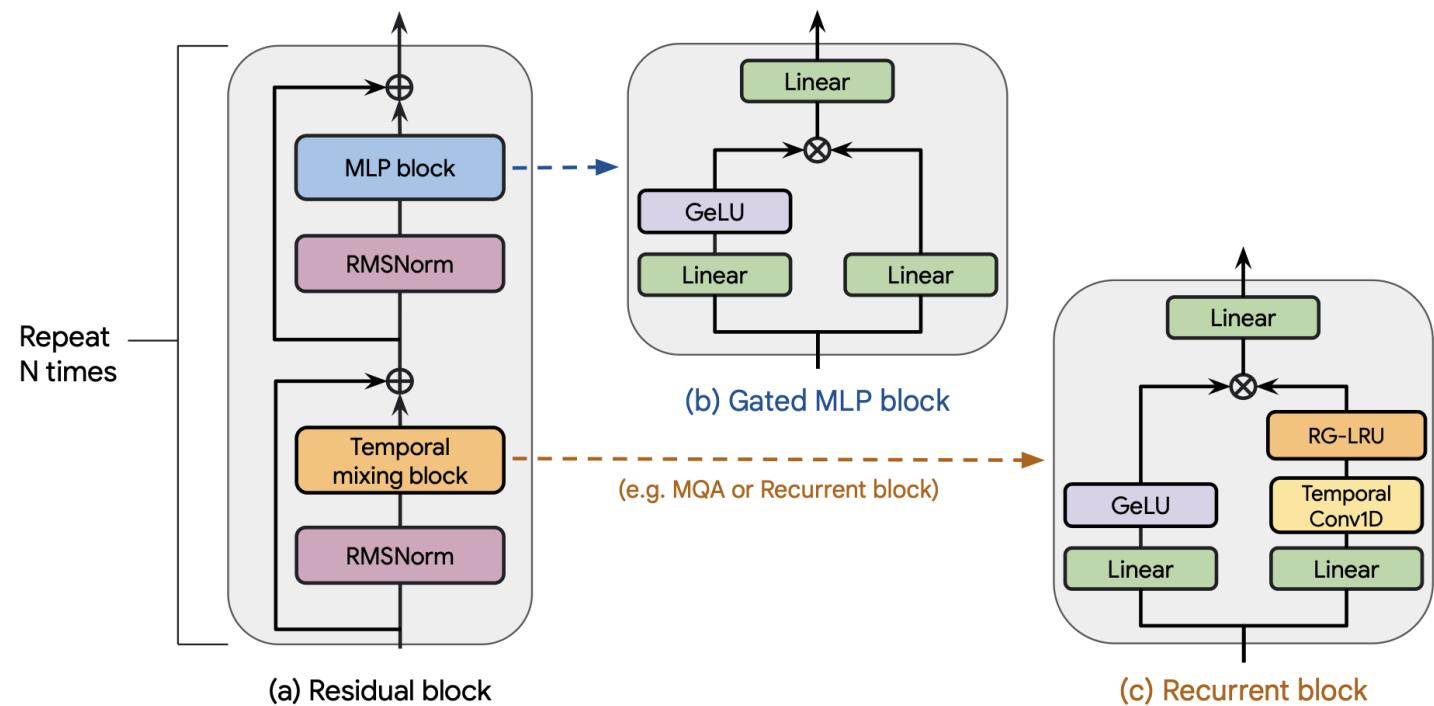
Linear Recurrent Unit

- Orvieto et al., 2023
- Stabilize the recurrence in Λ with complex numbers
- Already mention in the paper that a solution with real numbers would be almost as efficient



Hawk & Griffin

- [De et al., 2024](#)
- Layer again like Transformer (a)
- Uses an updated (only real numbers) version of the LRU in its recurrent block (c)



Transformer Variants

Transformer Variants – Overview

- Architecture components
 - Activation functions
 - Normalization layers
 - Position embeddings
- Architectures for long inputs
 - Transformer-XL
 - Longformer
 - Perceiver

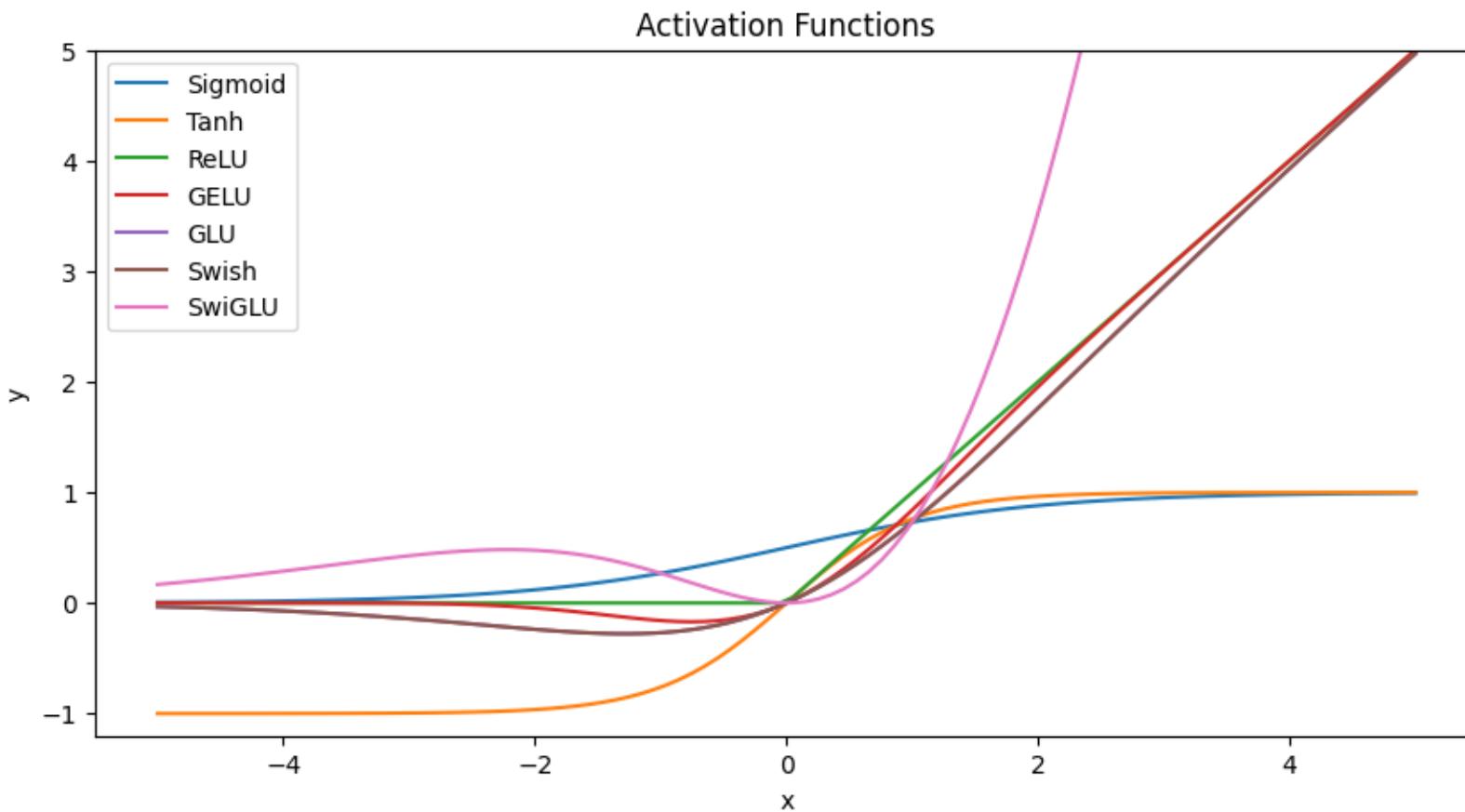
Architecture Components

- Transformer architecture was frequently modified
- Only few changes are better on multiple datasets & tasks
- Architecture becomes part of hyperparameter optimization
- Architecture and hyperparameter choices of many LLMs

Activation Functions

- Sigmoid
- Tanh
- Rectified Linear Unit (ReLU; [apparently 1941](#))
- Gaussian Error Linear Unit (GELU; [Hendrycks and Gimpel, 2016](#))
- Swish ([Ramachandran, 2017](#))
- With learnable parameters:
 - Gated Linear Unit (GLU; [Dauphin et al., 2016](#))
 - SwiGLU ([Shazeer, 2020](#))

Activation Functions



RMSNorm

- Root Mean Square Layer Normalization
([Zhang and Sennrich, 2019](#))
- PyTorch [nn.RMSNorm](#)
- LayerNorm re-centers (0 mean) and re-scales (“unit” variance = variance of 1) inputs
- RMSNorm only does the rescaling

$$y = \frac{x}{\sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}} * \gamma$$

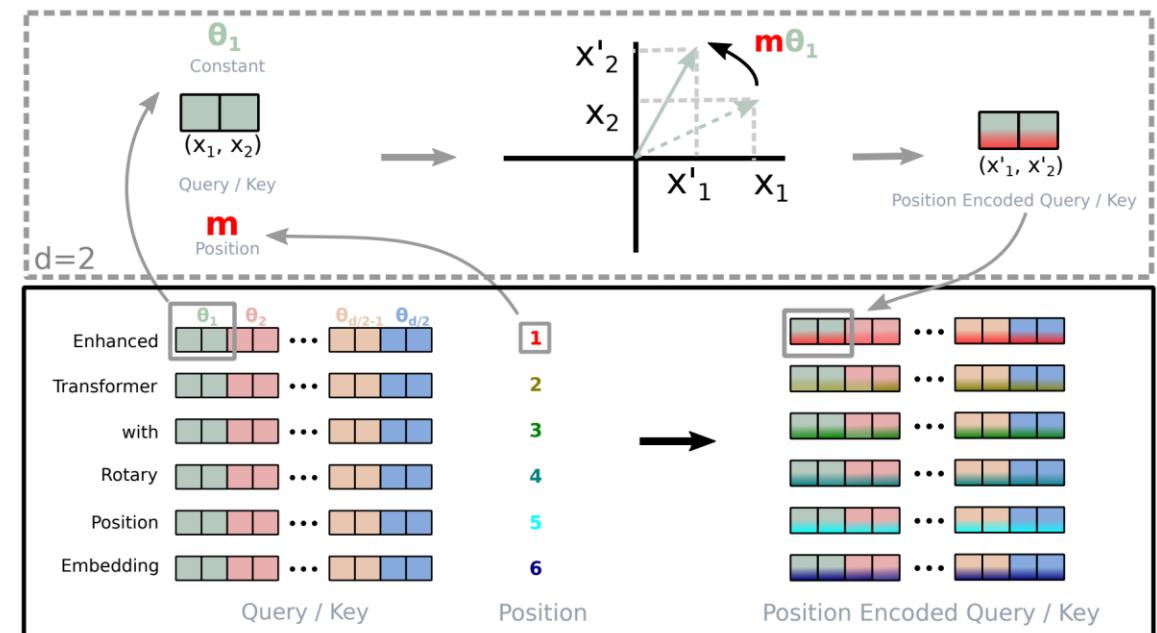
ALiBi Position Embeddings

- Attention with Linear Biases ([Press et al., 2022](#))
- Relative position embedding (added during attention computation, replaces absolute position embedding)
- m : attention head-specific scalar hyperparameter (not learned)
- m can be set as $2^{-\frac{8}{h}}$ and doesn't have to be tuned

$$\begin{matrix} q_1 \cdot k_1 \\ q_2 \cdot k_1 & q_2 \cdot k_2 \\ q_3 \cdot k_1 & q_3 \cdot k_2 & q_3 \cdot k_3 \\ q_4 \cdot k_1 & q_4 \cdot k_2 & q_4 \cdot k_3 & q_4 \cdot k_4 \\ q_5 \cdot k_1 & q_5 \cdot k_2 & q_5 \cdot k_3 & q_5 \cdot k_4 & q_5 \cdot k_5 \end{matrix} + \begin{matrix} 0 \\ -1 & 0 \\ -2 & -1 & 0 \\ -3 & -2 & -1 & 0 \\ -4 & -3 & -2 & -1 & 0 \end{matrix} \cdot m$$

Rotary Position Embedding (RoPE)

- [Su et al., 2024](#) (initial version from 2021)
- Relative position embedding used in modern LLMs
- Multiplies queries and keys with rotation matrix
 - Different rotation matrix for every position
 - Multiplying two rotation matrices will give the rotation matrix of the relative position difference
 - Works with linear attention variants
- Inner product decreases with larger distances

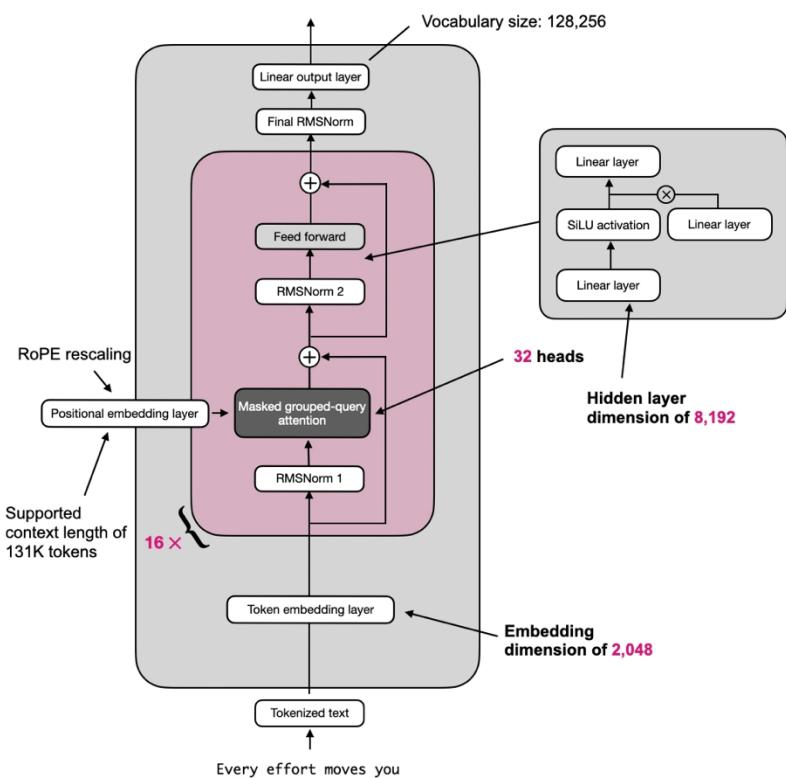


ALiBi's Analysis of RoPE

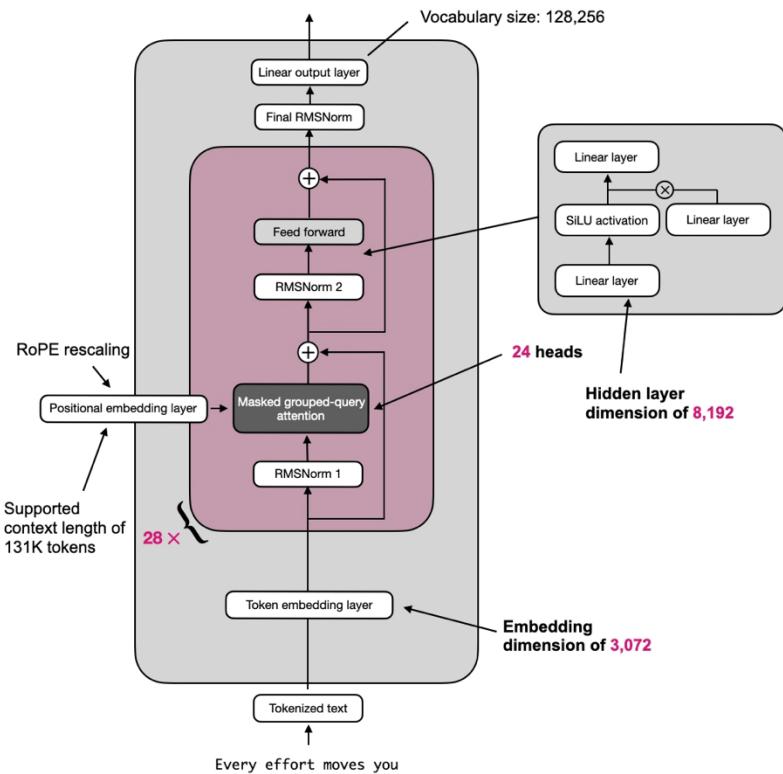
- [Press et al., 2022](#), section 2.2
- RoPE (and ALiBi) doesn't include position embeddings in values (and therefore in final hidden states)
- RoPE's rotation matrices are just the sinusoidal position embeddings from the original Transformer's position encoding
 - RoPE multiplies them with the queries and keys in self-attention instead of adding them as absolute embeddings to the word embeddings

Llama 3.2 Architecture

Llama 3.2 1B



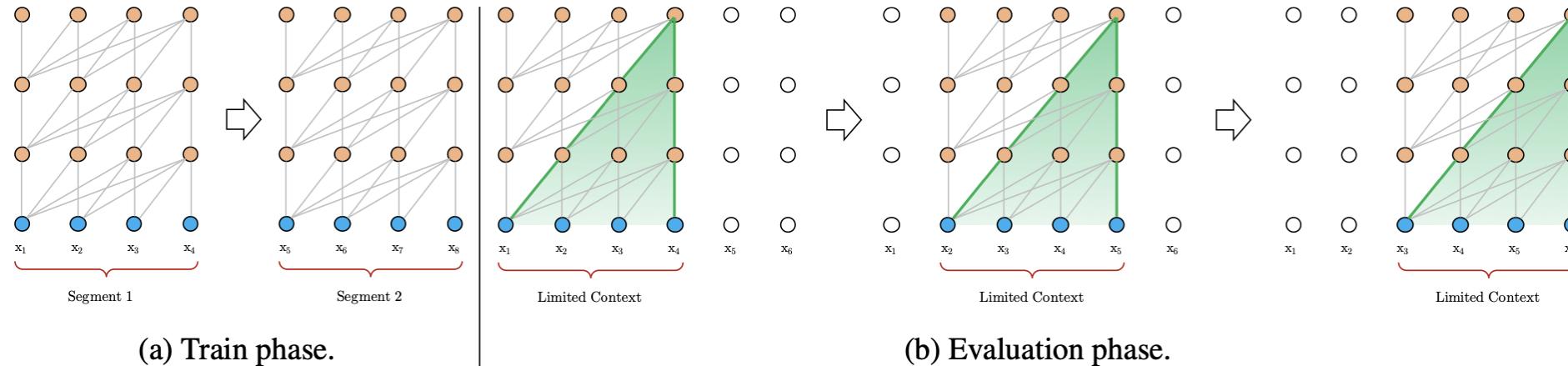
Llama 3.2 3B



Long-context Transformers

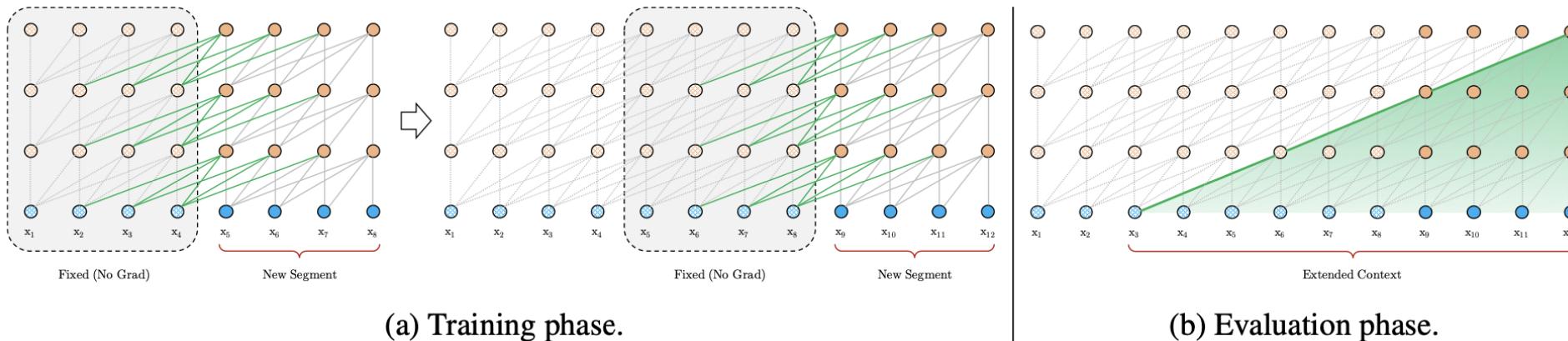
Transformer-XL

- [Dai et al., 2019](#)
- The original Transformer either uses:
 - Attention on the whole sequence (very expensive)
 - Restricted attention to a fixed segment during training and a sliding window during inference ([Al-Rfou et al., 2018](#))



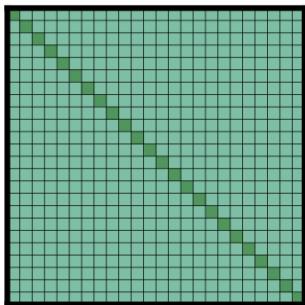
Transformer-XL

- Transformer-XL increases the visible segment by attention to previous segments
- Speeds up computation
 - Cache hidden states of the previous segment
 - Gradient is only computed in the current segment

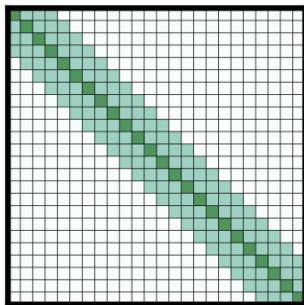


Longformer

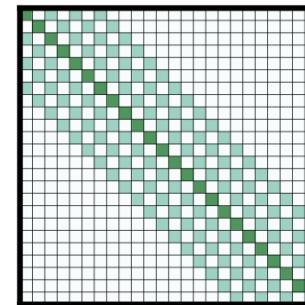
- [Beltagy et al., 2020](#)
- Can change the attention pattern to avoid n^2 attention
- Combine local and global information
 - Local: sliding window around current position
 - Global: certain tokens attend to all other tokens and are attended to by all other tokens
 - Can be task specific: BOS token for classification, all question tokens for QA



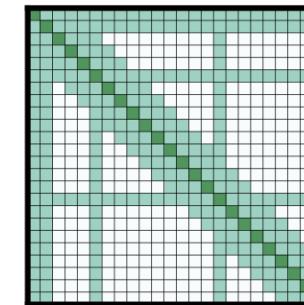
(a) Full n^2 attention



(b) Sliding window attention



(c) Dilated sliding window



(d) Global+sliding window

Perceiver

- [Jaegle et al., 2021](#)
- Developed for high-dimensional inputs (images, videos, audio, point clouds, multimodal combinations)
- Problem: High-dimensional inputs (e.g. images: height x width x channels) results in long sequences
- Idea
 - Reduce frequency of expensive computation on the full input
 - Perform most computation on cheaper (smaller) internal representation

Perceiver

- Cross-attention layers integrate information from the high-dimensional input into a low-dimensional latent representation
 - Use few cross-attention layers ([8 for ImageNet](#))
 - Interleaved with a Transformer on the latent representation ([6 layers each](#))
- M (width \times height $= 224 \times 224 = 50'176$) $\gg N$ ($= 512$)

