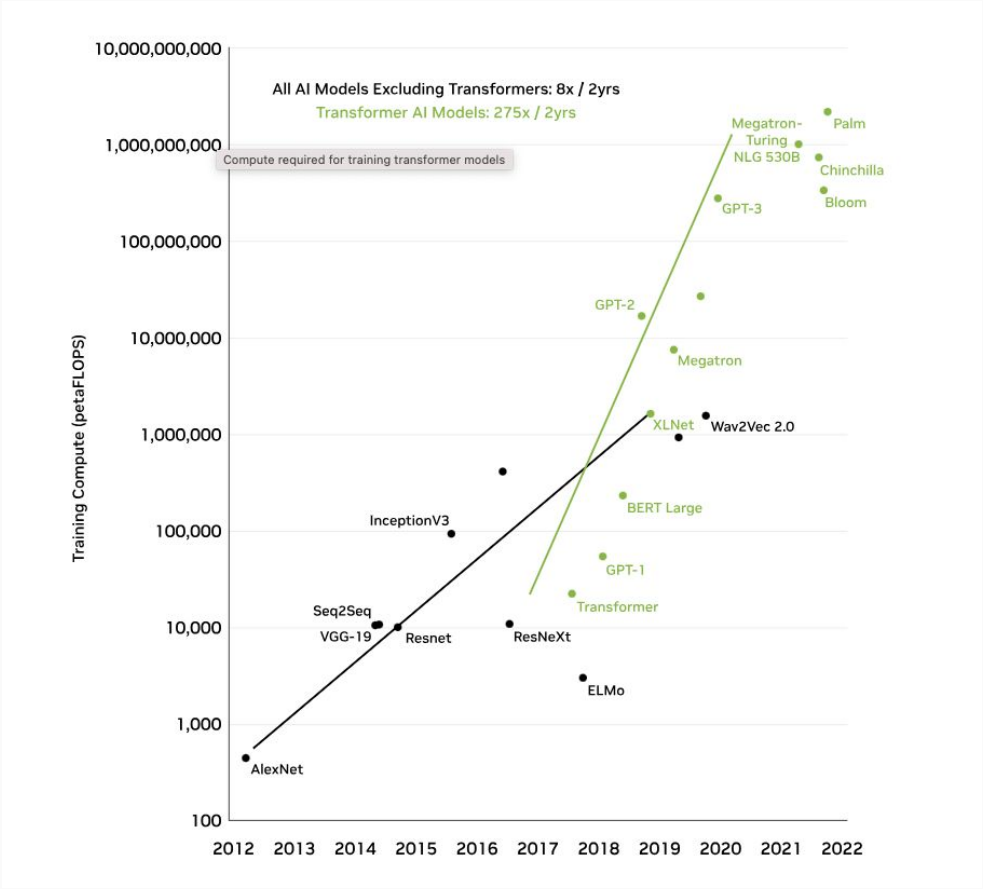
Multimodal Large Language Models

Agenda

1. LLM Evolution
2. Transformers for Vision
    a. Vision Transformer
    b. QFormer
3. Contrastive Language Image Pre-training
4. Making Text Generation Models Multimodal
    a. DALL-E
    b. BLIP2
    c. Pic2Word
    d. Screen AI
    e. Flamingo

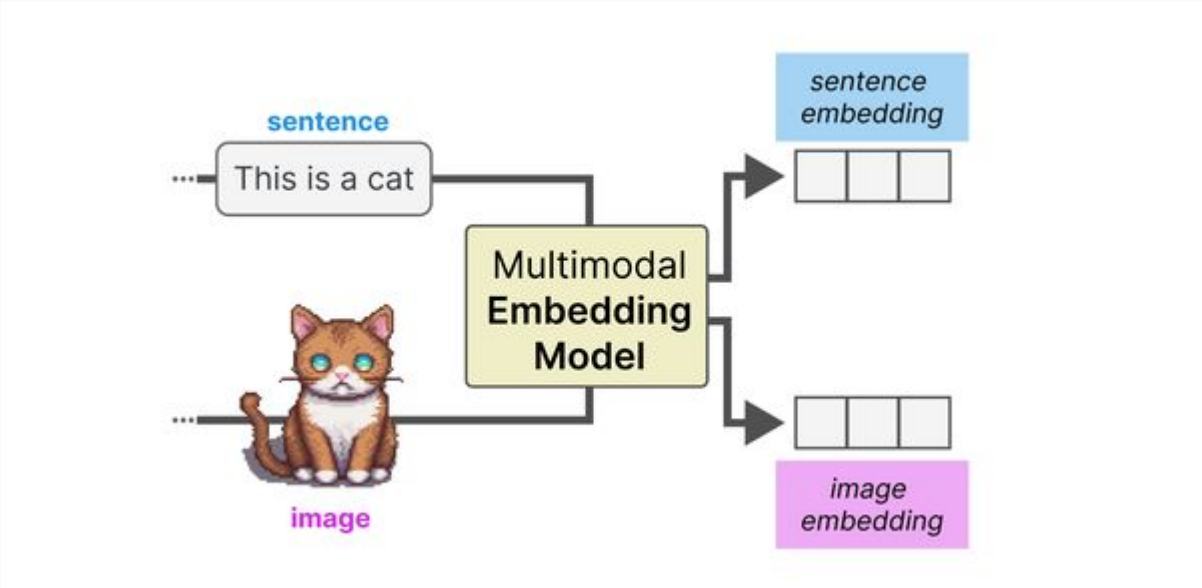
Compute required for training transformer models



Factors to consider

1. **Compute, cost, and time intensive workload**: Training an LLM requires thousands of GPUs and weeks to months of dedicated training time. Some estimates indicate that a single training run for a GPT3 model with 175 billion parameters, trained on 300 billion tokens, may cost over $12 million dollars in just compute.
2. **Scale of data required**: Training a large model requires a significant amount of data. Many companies struggle to get access to large enough datasets to train their large language models. This issue is compounded for use cases that require private—such as financial or health—data. In fact, it's possible that the data required to train the model doesn't even exist.
3. **Technical expertise**: Due to their scale, training and deploying large language models are very difficult and require a strong understanding of deep learning workflows, transformers, and distributed software and hardware, as well as the ability to manage thousands of GPUs simultaneously.

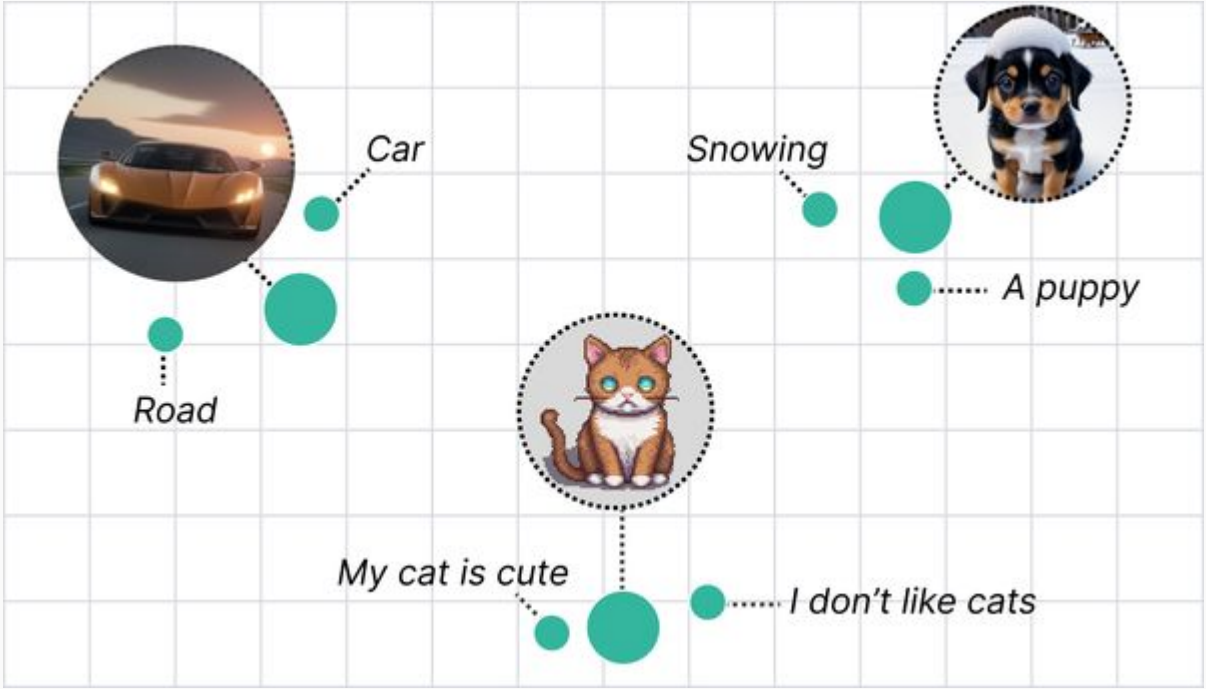
Multimodal embedding models



Hands-On Large Language Models Jay Alammar, Maarten Grootendorst

Embedding Space

Multimodal embedding models can create embeddings for multiple modalities in the same vector space.



LLM Evolution

Generative pretraining Byte Pair Encoding

- It starts by encoding each character as a byte pair, so the vocabulary just contains single characters at the start.
- It then iteratively merges the most frequent pair of bytes in the text into a new single byte.
- For example, if "th" is the most frequent pair, each occurrence of "t" followed by "h" will now be encoded as a single byte "th" rather than two separate "t" and "h" bytes.
- This merge process repeats thousands of times on the entire corpus, each time creating new byte pairs from the most common pair of bytes.
- Eventually frequent subwords like "pre" or "ing" will emerge in the encoding.
- The final output vocabulary contains these common bytes pairs, starting from single characters up through merged multi character sequences.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pretraining.

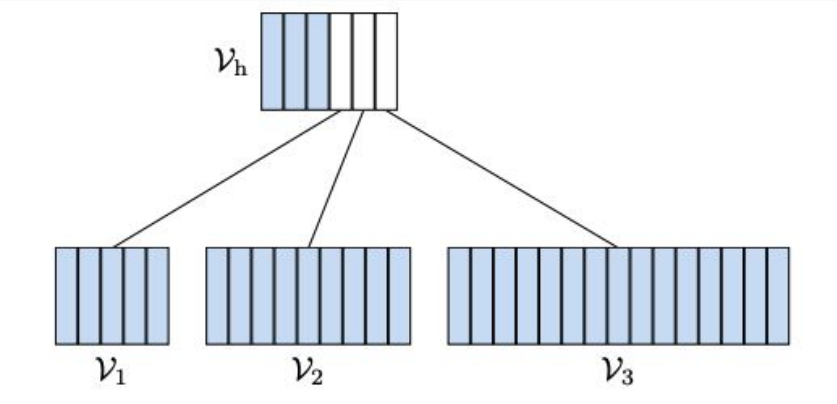
GPT2 Adaptive input

- Adaptive input representations assign vector embeddings of different sizes to words based on their frequency, rather than fixing the size for all words.
- More frequent words get larger vector representations with more parameters, while less frequent words get smaller vectors.
- This allows the model to allocate capacity more effectively, focusing more representational power on the common words that matter most rather than wasting parameters on rare words.
- The result is a more parameter efficient model overall compared to traditional fixed size word embeddings.
- This helps use model capacity judiciously for better computational and statistical efficiency.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multi task learners. *OpenAI blog*, *1*(8), 9.

GPT2 Adaptive Softmax

- The softmax converts logits to probabilities by taking the exponential and normalizing. This can be expensive with a large vocabulary.
- The adaptive softmax clusters words by frequency common words in smaller clusters, rare words in larger clusters.
- Common word clusters are computed more frequently, rare words less frequently.
- This reduces the softmax computation cost for large vocabularies significantly.
- The approximation works well without much performance loss.



- First level includes both the most frequent words and vectors representing clusters.
- Clusters on the second level are associated with rare words, the largest ones being associated with the less frequent words.
- The sizes are determined so as to minimize computational model on GPU.

Joulin, A., Cissé, M., Grangier, D., & Jégou, H. (2017, July). Efficient softmax approximation for GPUs. In *International conference on machine learning* (pp. 13021310). PMLR.

GPT2 Adaptive Softmax Example

Suppose we have a vocabulary with 10,000 words. Rather than applying the expensive full softmax across all 10,000 outputs, we can cluster words as follows:

Cluster 1 (frequent words): The most common 100 words

Cluster 2 (less common words): The next 900 common words

Cluster 3 (rare words): The remaining 9,000 rare words

When computing the softmax probabilities:

- Cluster 1 words have a softmax applied over just 100 words
- Cluster 2 words have a softmax over 900 words
- Cluster 3 words have a softmax over 9,000 words

So common words get a faster softmax over a smaller set, while rare words have the softmax over a larger set.

Additionally, during training we can update Cluster 1 more frequently than Cluster 3, since getting common word predictions accurate matters more.

This allows directing most computation to the words that matter most, greatly reducing overall softmax computation cost. The approximations for rare words have minimal impact since those words are intrinsically less important.

Improvements for GPT3

In June 2020, OpenAI released GPT3, a 175 billion parameter model that generated text and code with short written prompts.

- Sparse Attention:
  - Unlike traditional attention mechanisms that consider the entire input sequence, sparse attention allows the model to focus computation on a subset of the input tokens.
  - This improves the model's efficiency, especially for longer sequences, and enables it to capture long range dependencies more effectively.
- MultiHead Attention Refinements:
  - Optimizations in how attention heads are weighted and combined, improving the model's ability to extract and synthesize information from the input.
- Efficiency and Scalability:
  - Efficient softmax computation in the attention scores, better memory management
  - and optimizations in the attention computation pipeline help in handling the model's increased complexity and size.

Vision Transformer (ViT)

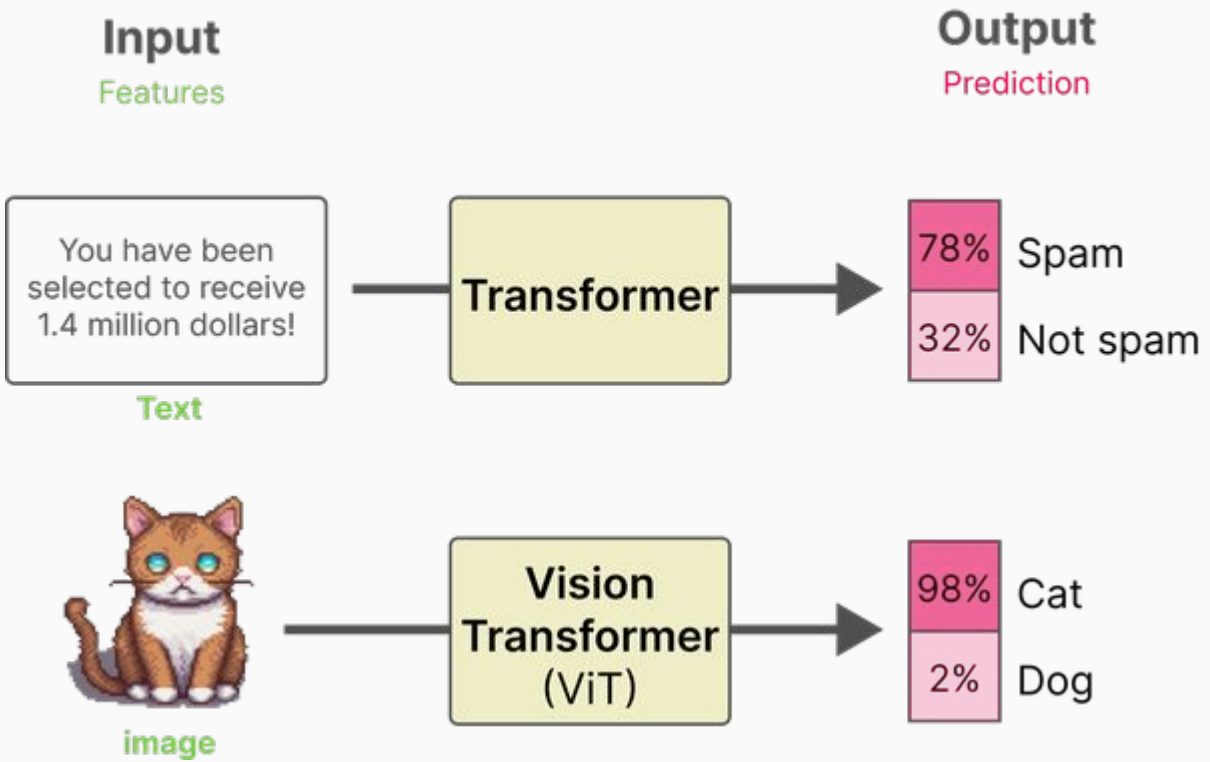
Vision Transformer (ViT) - Introduction

- CNNs have been the dominant architecture for computer vision tasks
  - CNNs utilize inductive biases specific to images, such as translation equivariance and locality
  - State-of-the-art CNNs (e.g., ResNet, EfficientNet) rely on hand-crafted architectures and extensive engineering
- Challenges in applying Transformers to vision tasks:
  - High computational cost of self-attention on high-resolution images
  - Lack of inductive biases that are inherent to CNNs
- Previous attempts to combine CNNs and self-attention:
  - Self-attention applied to feature maps extracted by CNNs
  - Local self-attention to reduce computational cost
  - Hybrid architectures that replace some CNN components with self-attention

Kolesnikov, A., Dosovitskiy, A., Weissenborn, D., Heigold, G., Beyer, L., Minderer, M., ... & Zhai, X. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.

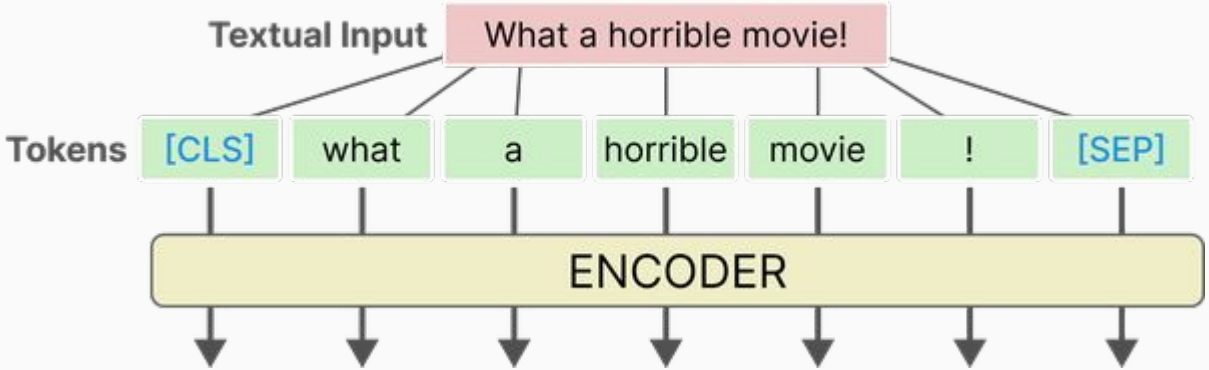
Transformers for Vision

- Both the original transformer as well as the vision transformer take unstructured data, convert it to numerical representations
- and use that for tasks like classification.



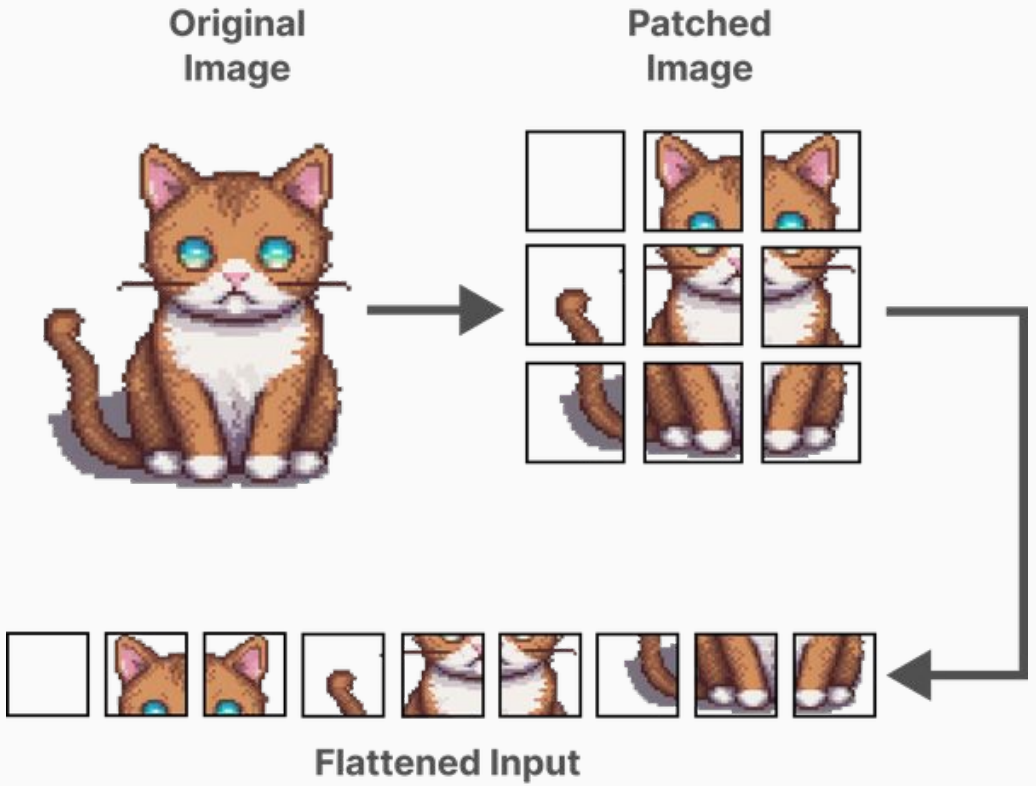
Input to a typical Encoder

Text is passed to one or multiple encoders by first tokenizing it using a tokenizer.



Input to a ViT

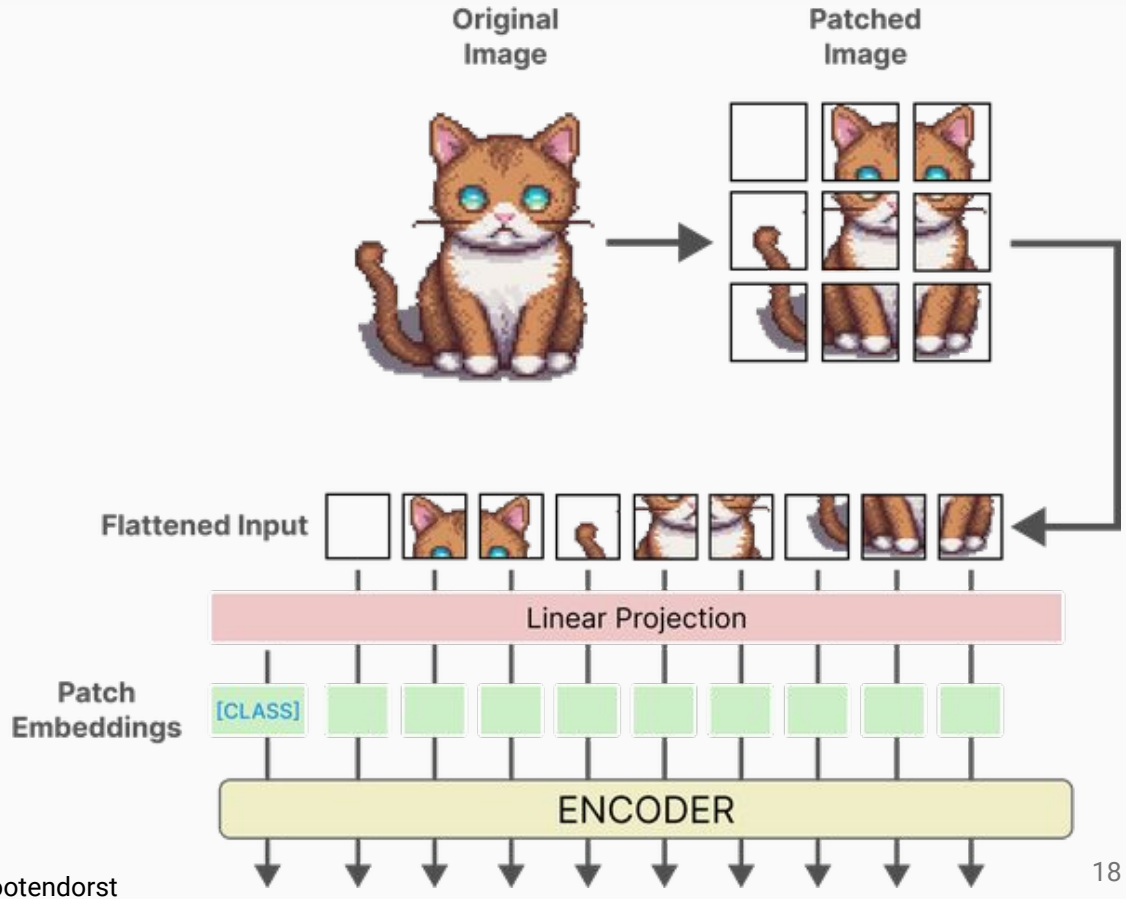
- The “tokenization” process for image input.
- It converts an image into patches of sub-images.



Hands-On Large Language Models Jay Alammar, Maarten Grootendorst

The main algorithm behind ViT

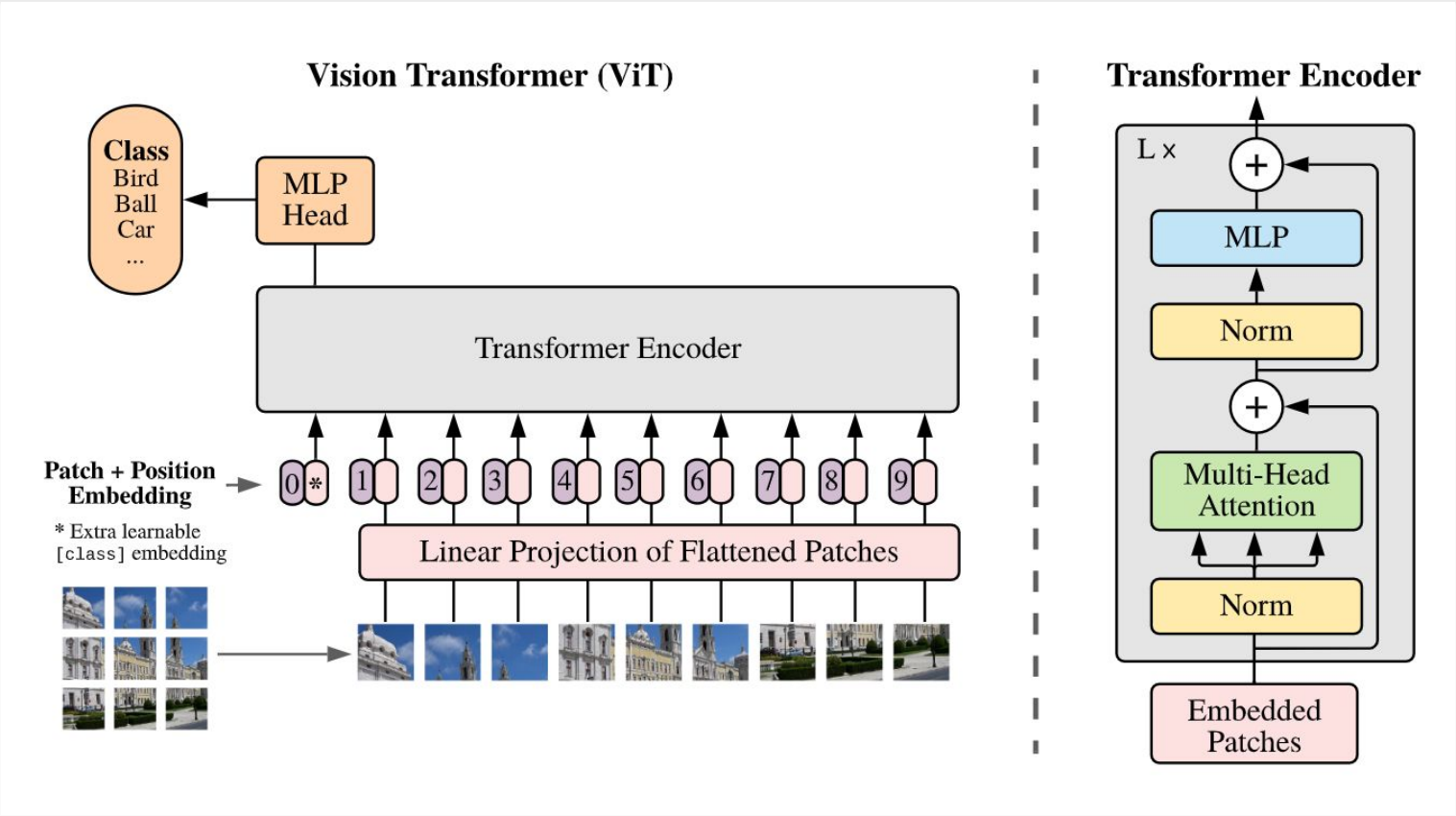
- After patching the images and linearly projecting them
- the patch embeddings are passed to the encoder
- and treated as if they were textual tokens.



Vision Transformer (ViT) Architecture

- Split image into patches of size 16x16 or 32x32
- Linearly project flattened patches into a lower-dimensional space (e.g., 768-d)
- Add learnable 1D position embeddings to the patch embeddings
- Pass the sequence through a standard Transformer encoder
    - Alternating layers of Multi-head Self-Attention (MSA) and MLP blocks
    - Layer normalization (LN) applied before each block, residual connections after each block
    - MLP contains two layers with GELU activation
- Prepend a learnable [class] token to the sequence for classification

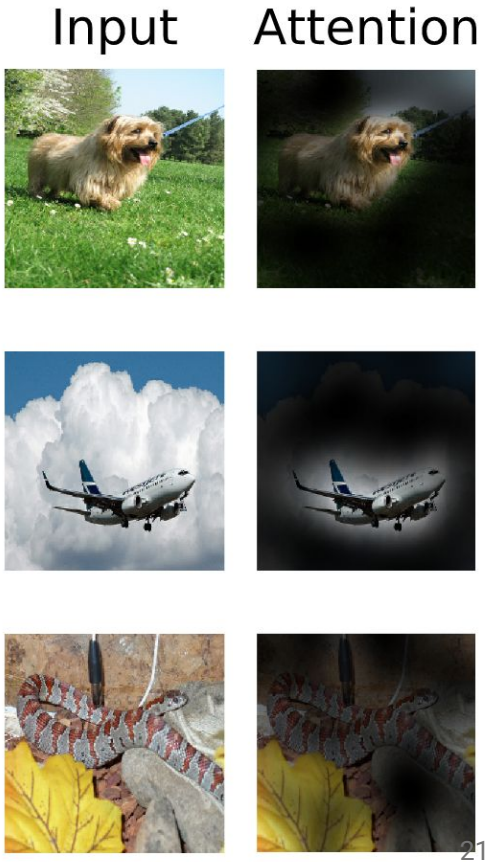
Model Overview



Kolesnikov, A., Dosovitskiy, A., Weissenborn, D., Heigold, G., Beyer, L., Minderer, M., ... & Zhai, X. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.

Observations

- Patch embeddings learn frequency-selective filters similar to CNN filters
- Position embeddings learn 2D structure despite being 1D
- Attention distance varies across heads and layers
- Some heads attend broadly even in lower layers
- Attention distance increases in higher layers
- Attention maps highlight semantically relevant regions for classification



Kolesnikov, A., Dosovitskiy, A., Weissenborn, D., Heigold, G., Beyer, L., Minderer, M., ... & Zhai, X. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.

Positional Embeddings in Vision Transformers

- 1D positional embeddings (default):
  - Learnable 1D embeddings, as in the original Transformer
  - Each position in the patch sequence is assigned a unique embedding vector
  - Embeddings are learned during training
- 2D positional embeddings:
  - Learnable 2D embeddings that capture the spatial structure of the image
  - Separate embedding vectors for each row and column of the patch grid
  - Each patch position is assigned a combination of the corresponding row and column embeddings
- Relative positional embeddings:
  - Embeddings that capture the relative distance between patches
  - Computed based on the relative position of query and key patches in the self-attention mechanism
  - Can be implemented using 1D or 2D relative position encodings

Kolesnikov, A., Dosovitskiy, A., Weissenborn, D., Heigold, G., Beyer, L., Minderer, M., ... & Zhai, X. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.

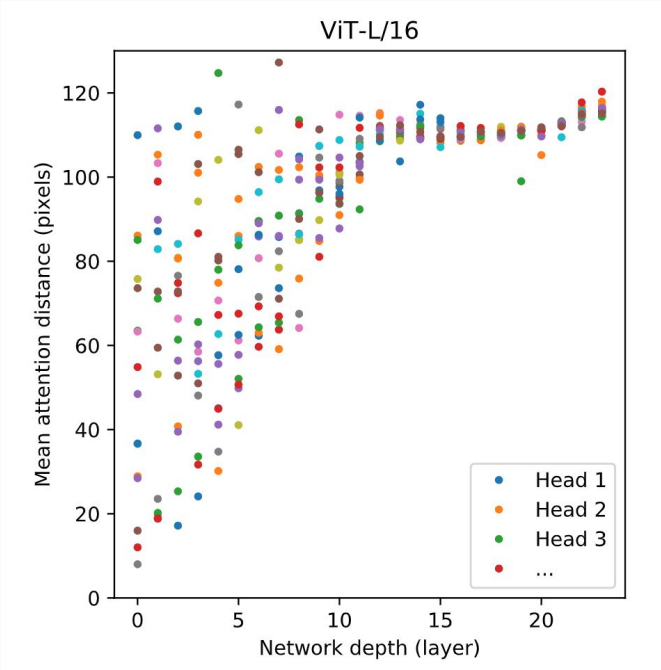
Performance comparison: 1D, 2D, and relative positional embeddings

- All types of embeddings performed similarly well
- Choice of positional embedding did not significantly impact the model's performance
- Learned 1D positional embeddings:
  - Visualization of learned 1D embeddings shows they capture the 2D spatial structure of the image
  - Similar embeddings assigned to patches in the same row or column
  - 1D positional embeddings are computationally more efficient than 2D or relative embeddings

Kolesnikov, A., Dosovitskiy, A., Weissenborn, D., Heigold, G., Beyer, L., Minderer, M., ... & Zhai, X. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.

Mean Attention Distance: the average distance between the query patch and the patches

- Higher mean attention distance indicates that the model is attending to patches farther away from the query patch
- Lower mean attention distance suggests that the model is focusing on nearby patches
- In the early layers, some attention heads have low mean attention distances, focusing on local regions around the query patch
- In the later layers, most attention heads have high mean attention distances, indicating global integration of information from the entire image
- The pattern of increasing attention distances across layers suggests that the model gradually incorporates information from larger regions of the image
- The presence of both local and global attention in the early layers allows the model to capture both fine-grained details and high-level context
- The global attention in the later layers enables the model to reason about the entire image and make predictions based on the integrated information



Kolesnikov, A., Dosovitskiy, A., Weissenborn, D., Heigold, G., Beyer, L., Minderer, M., ... & Zhai, X. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.

QFormer: Transforming Vision with Quadrangle Attention

The need

- Vision Transformers (ViTs) have shown great promise in computer vision tasks
- ViTs process images as a sequence of patches and use self-attention to capture global context
- Most ViTs use fixed square windows for local attention, which limits their ability to adapt to objects of different shapes and sizes

Zhang, Q., Zhang, J., Xu, Y., & Tao, D. (2024). Vision transformer with quadrangle attention. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Quadrangle Attention (QA)

- QA is a new attention mechanism that allows ViTs to attend to adaptive quadrangle-shaped regions
- Quadrangles are generated by learning a transformation matrix that deforms default square windows
- This allows attention windows to better fit the sizes, shapes, and orientations of objects in the image

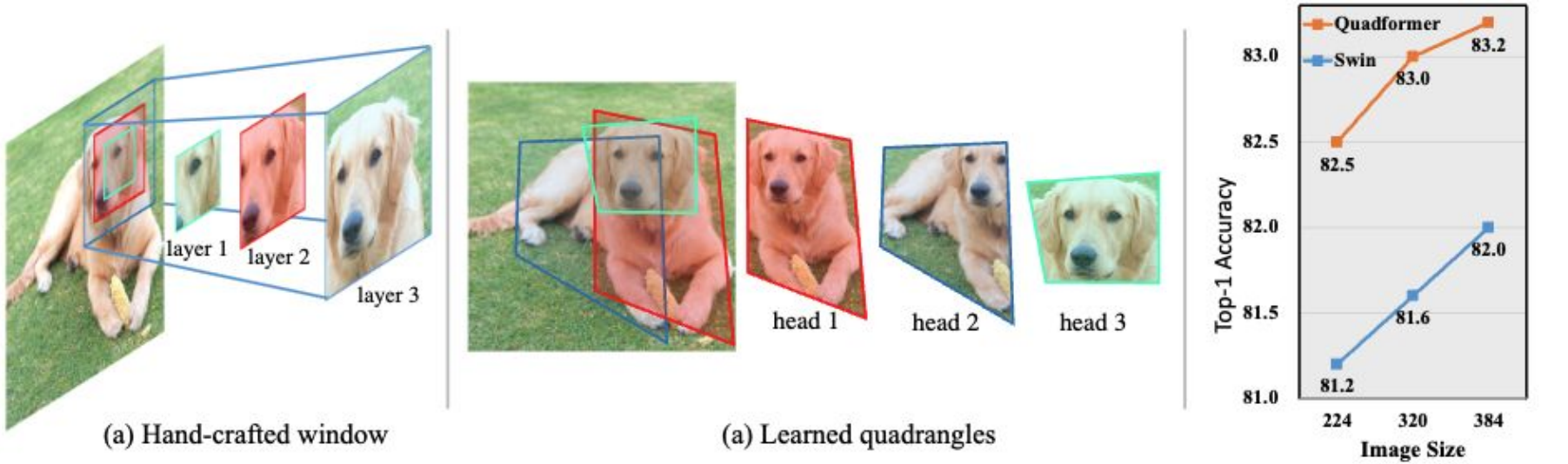
Zhang, Q., Zhang, J., Xu, Y., & Tao, D. (2024). Vision transformer with quadrangle attention. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

How QA Works

- The learned transformation matrix is decomposed into basic transformations: scaling, shearing, rotation, translation, and projection
- This decomposition makes the optimization of the quadrangle shapes easier
- The model learns to transform default square windows into adaptive quadrangles that cover relevant object regions

Zhang, Q., Zhang, J., Xu, Y., & Tao, D. (2024). Vision transformer with quadrangle attention. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Quadrangles



(a) Hand-crafted window

(a) Learned quadrangles

Fig. 1: Comparison of window configurations in previous hand-crafted designs [2] (a) and the proposed learnable quadrangle design (b), as well as their image classification performance under different settings of input size on the ImageNet [13] validation set (c).

Plain QFormer

- Similar to ViT a the input image is divided into a sequence of fixed-size patches
- The patches are linearly embedded and added to position embeddings to form a sequence of tokens
- Standard Multi-Head Self-Attention (MHSA) is replaced with Quadrangle Attention (QA)
- QA learns to transform default square windows into adaptive quadrangles using a projective transformation matrix
- The projective transformation matrix is decomposed into basic transformations (scaling, shearing, rotation, translation, and projection) to facilitate optimization
- The adaptive quadrangles allow the model to better capture relevant regions of objects and promote cross-window information exchange

Zhang, Q., Zhang, J., Xu, Y., & Tao, D. (2024). Vision transformer with quadrangle attention. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Hierarchical QFormer

- The input image is processed at multiple scales
- The input image is first divided into patches, which are then embedded and passed through a series of transformer blocks
- At each stage, the number of tokens is reduced by merging patches, while the channel dimension is increased
- The standard window-based MHSA in each transformer block is replaced with QA
- QA learns to transform default square windows into adaptive quadrangles using a projective transformation matrix
- The hierarchical structure allows the QFormer to process information at different scales and capture both local and global context

Zhang, Q., Zhang, J., Xu, Y., & Tao, D. (2024). Vision transformer with quadrangle attention. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Key benefits of the QFormer architecture

1. Adaptive attention regions: The learned quadrangle windows can better fit the sizes, shapes, and orientations of objects in the image, leading to improved representation power.
2. Cross-window information exchange: The adaptive shapes of the quadrangles promote information exchange between different regions of the image, enhancing the model's ability to capture context.
3. Computational efficiency: Despite the added flexibility of quadrangle windows, QFormer maintains computational efficiency comparable to standard vision transformers with fixed window attention.

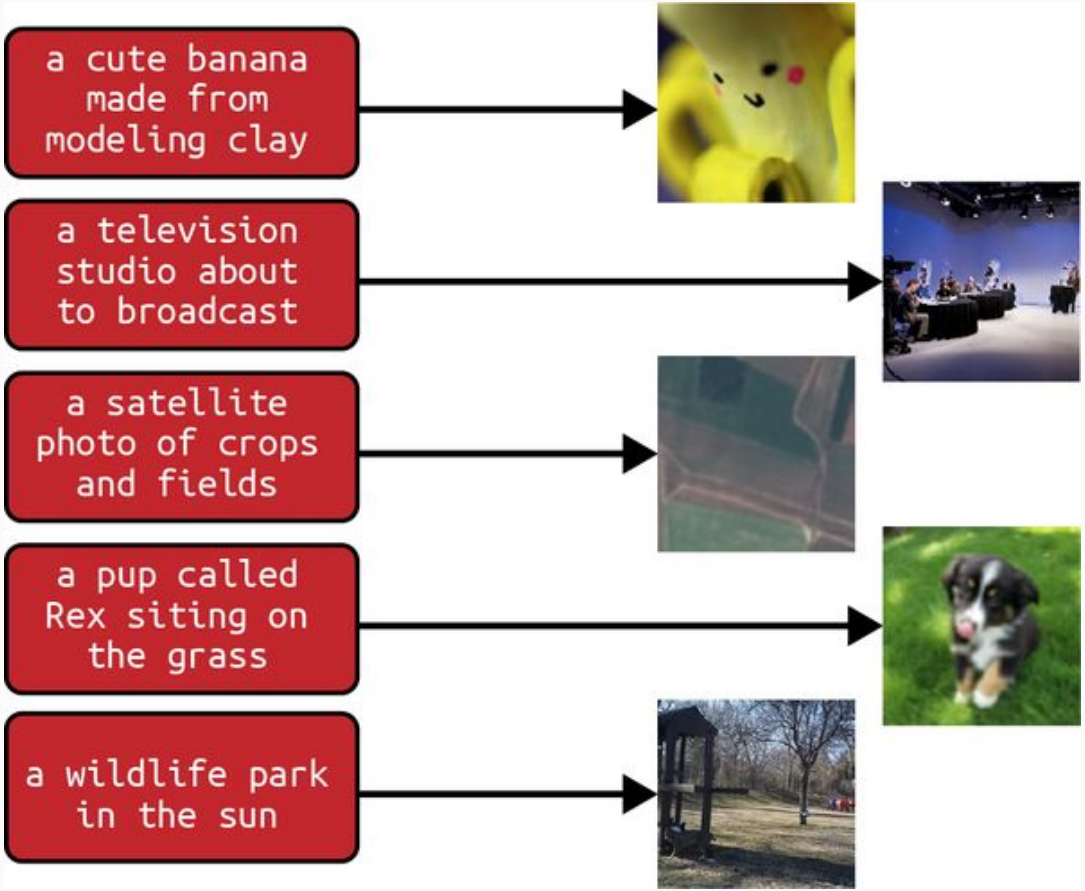
Zhang, Q., Zhang, J., Xu, Y., & Tao, D. (2024). Vision transformer with quadrangle attention. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Contrastive Language–Image Pre-training (CLIP)

Contrastive Language–Image Pre-training (CLIP)

The type of data that is needed to train a multimodal embedding model.

Examples of text–image pairs



Motivation

- Task Agnostic pre training has revolutionized NLP, enabling models to learn general language representations that transfer well to downstream tasks.
- CLIP applies this approach to computer vision by learning visual concepts from natural language supervision using a contrastive objective.
- Advantages of CLIP include enabling zero shot transfer to downstream tasks and improving robustness to distribution shift.
- This approach has the potential to learn more general and flexible visual representations compared to traditional supervised learning on fixed datasets.

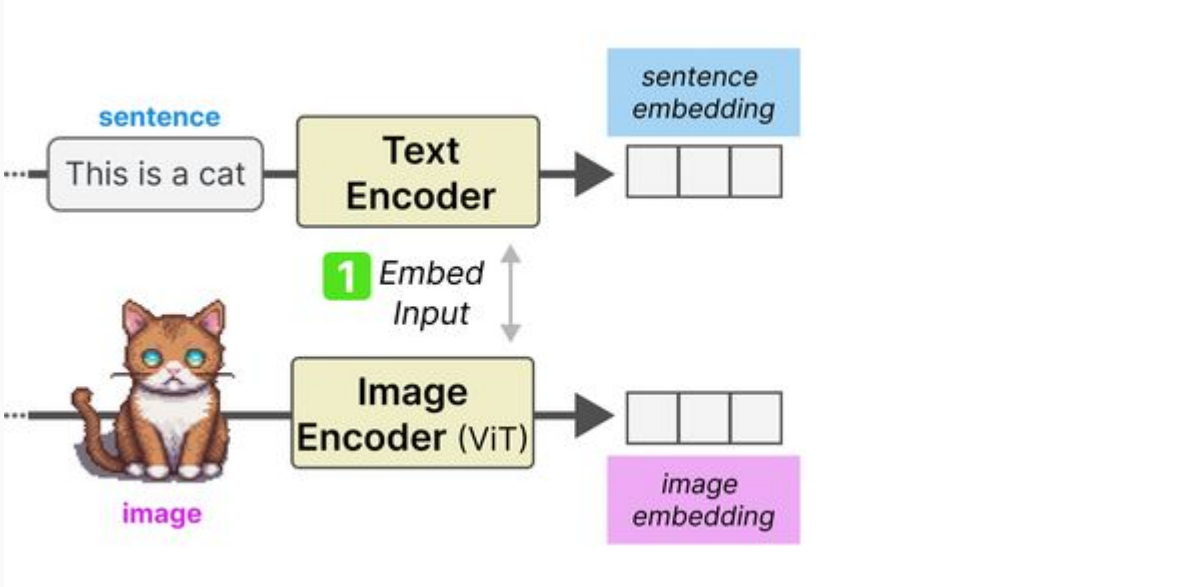
Pre Training Dataset (WIT)

- WIT (WebImageText) is a new dataset of 400 million (image, text) pairs collected from the internet.
- The dataset was constructed using a set of 500,000 search queries to cover a wide range of visual concepts.
- Class balance was maintained by allowing up to 20,000 (image, text) pairs per query.
- Compared to existing vision datasets, WIT is significantly larger and more diverse in terms of visual concepts and language descriptions.

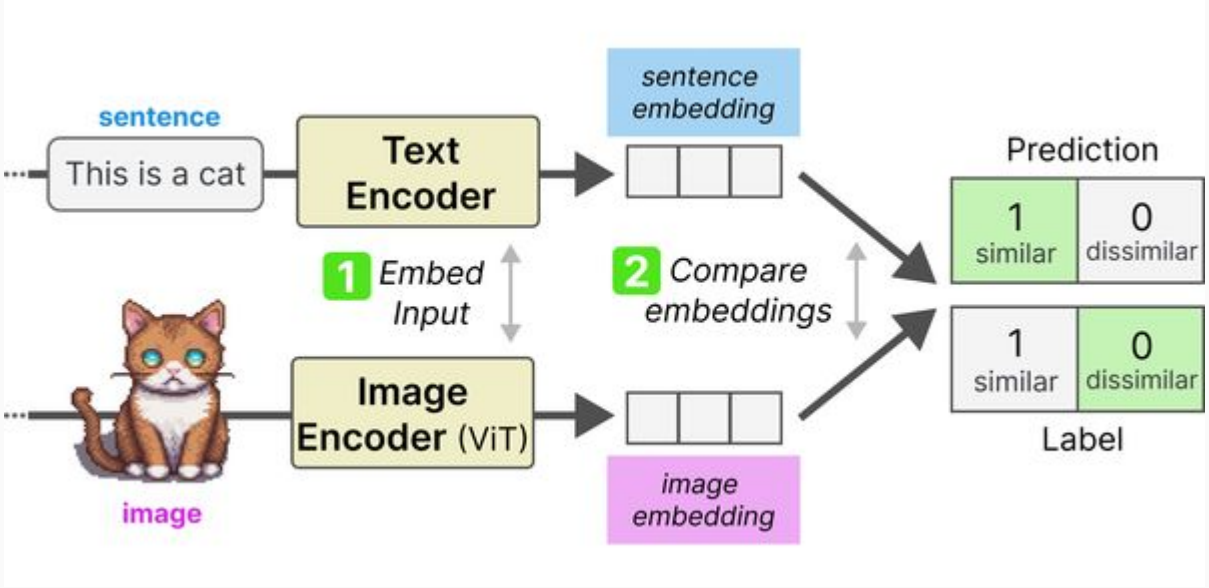
CLIP Training Methodology

- CLIP jointly trains an image encoder and text encoder to predict the correct pairings of images and text in a batch.
- The contrastive learning objective uses a symmetric cross entropy loss over the cosine similarity scores of the image and text embeddings.
- The image encoder uses a ResNet architecture, while the text encoder uses a Transformer architecture.
- Contrastive learning is computationally efficient and scalable compared to generative modeling or autoregressive prediction approaches.

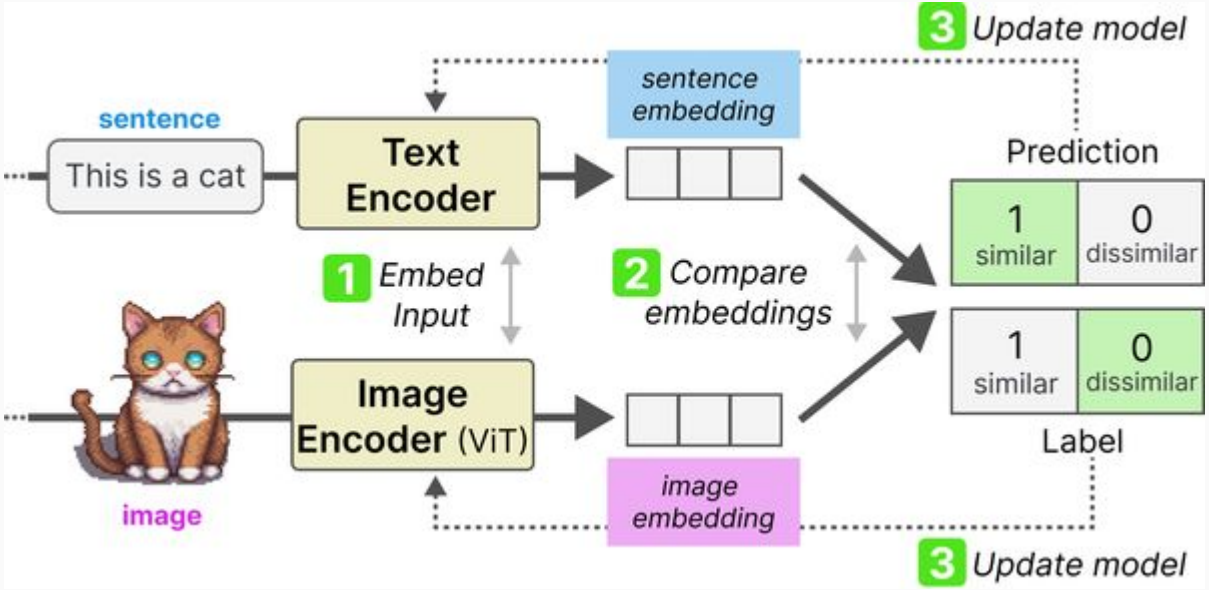
1. CLIP Training both images and text are embedded using an image and text encoder respectively



2. The similarity between the sentence and image embedding is calculated using cosine similarity



3. Text and image encoders are updated to match what the intended similarity should be



ZeroShot Transfer Capabilities

- CLIP's flexible text encoder enables "zeroshot" transfer to downstream datasets without using any of their training examples.
- The text encoder can be used to construct zeroshot classifiers by encoding the names or descriptions of the target classes.
- On average across 27 datasets, zeroshot CLIP outperforms a ResNet50 trained with labeled examples from each dataset.
- Zeroshot CLIP is competitive with fewshot logistic regression using 4 examples per class on CLIP features, demonstrating its sample efficiency.

Robustness to Natural Distribution Shift

- Distribution shift is a significant challenge for real world applications of computer vision models.
- CLIP was evaluated on a suite of 7 distribution shift datasets that measure natural variations in image data, such as different camera poses, backgrounds, and object renditions.
- Zeroshot CLIP significantly outperforms standard ImageNet models on these distribution shift datasets, reducing the accuracy gap between the ImageNet test set and the distribution shift datasets by up to 75%.
- Supervised CLIP models fine tuned on ImageNet lose most of this robustness benefit, suggesting that task specific training on a narrow distribution can reduce generalization.

Emergent Capabilities from Natural Language Supervision

- CLIP learns a wide range of visual concepts and skills through natural language supervision, including:
- Optical Character Recognition (OCR): CLIP can recognize and read handwritten digits (MNIST) and rendered text (Rendered SST2) with high accuracy.
- Facial emotion recognition: CLIP achieves performance comparable to human subjects on the Facial Emotion Recognition 2013 dataset.
- Action recognition: CLIP obtains strong results on video action recognition datasets (UCF101 and Kinetics 700) without using any video training data.
- Geolocalization: CLIP can estimate the geographic location of images using its learned representations.
- These emergent capabilities demonstrate the power of natural language supervision in enabling models to learn a broad range of visual concepts and skills.

Limitations, Biases, and Societal Impact

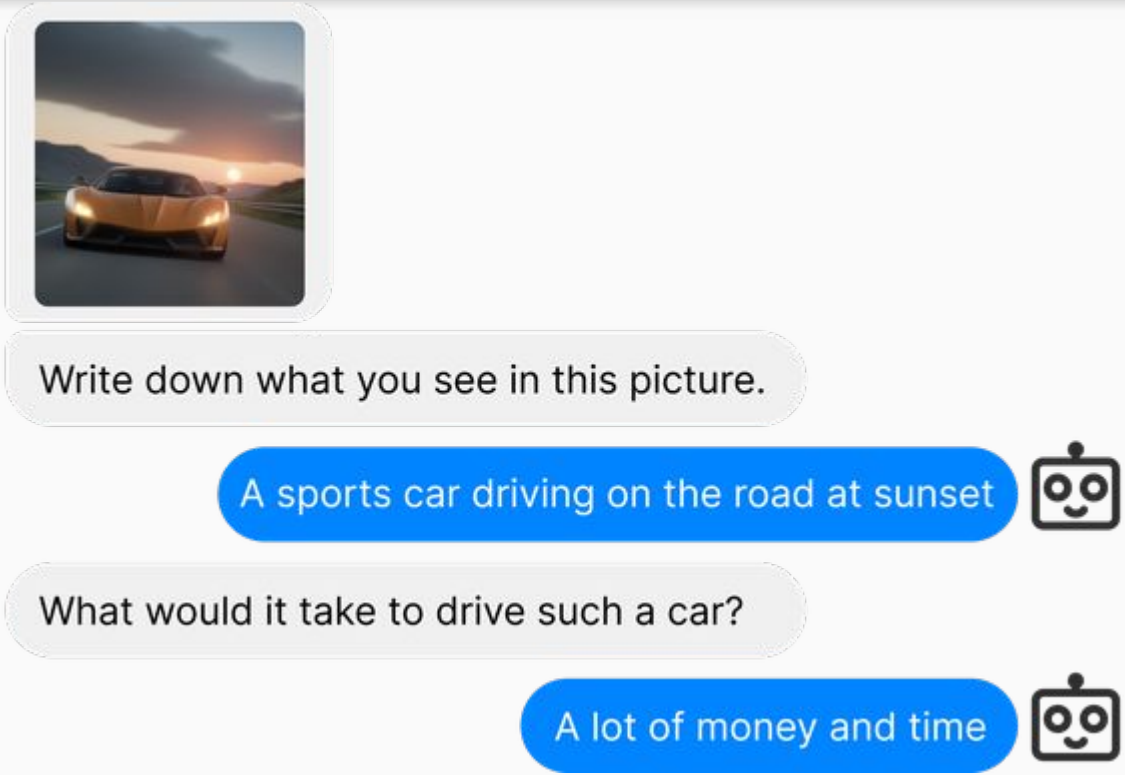
- CLIP still struggles with some abstract image classification tasks that require specialized domain knowledge, such as lymph node tumor detection (PatchCamelyon).
- The pretraining data and model may contain and amplify social biases present in web scraped data. More work is needed to fully characterize and mitigate these biases.
- The flexibility of CLIP's zeroshot classifiers could potentially be misused, such as for unauthorized surveillance or invasion of privacy.
- It is important to develop strategies for the responsible deployment of powerful visual learning systems like CLIP, such as implementing safeguards against misuse and promoting transparency.

Conclusion and Future Directions

- Contrastive pretraining with natural language supervision is a promising approach for learning transferable visual representations.
- CLIP achieves strong zero shot and few shot transfer performance on a wide range of downstream tasks, demonstrating the generality and flexibility of its learned representations.
- The diversity of CLIP's pretraining data contributes to its improved robustness to natural distribution shifts compared to models trained on a single dataset.
- Future directions include scaling up pretraining to larger datasets and models, improving data quality and diversity, and developing better evaluation benchmarks for measuring general visual understanding.
- As powerful visual learning systems like CLIP become more prevalent, it is crucial to consider both their potential benefits and risks to society and to engage in multidisciplinary research to ensure their responsible development and deployment.

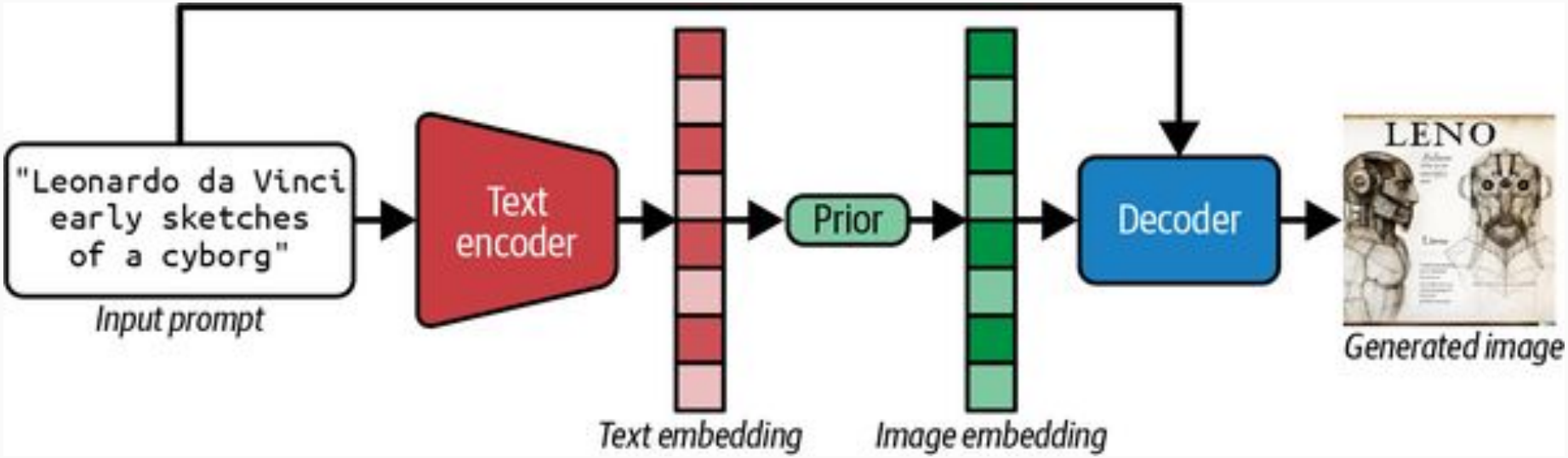
Making Text Generation Models Multimodal

A multimodal text generation model that can reason about input images



Decoder-Only Autoregressive Language and Image Synthesis (DALL-E)

The DALL.E 2 architecture



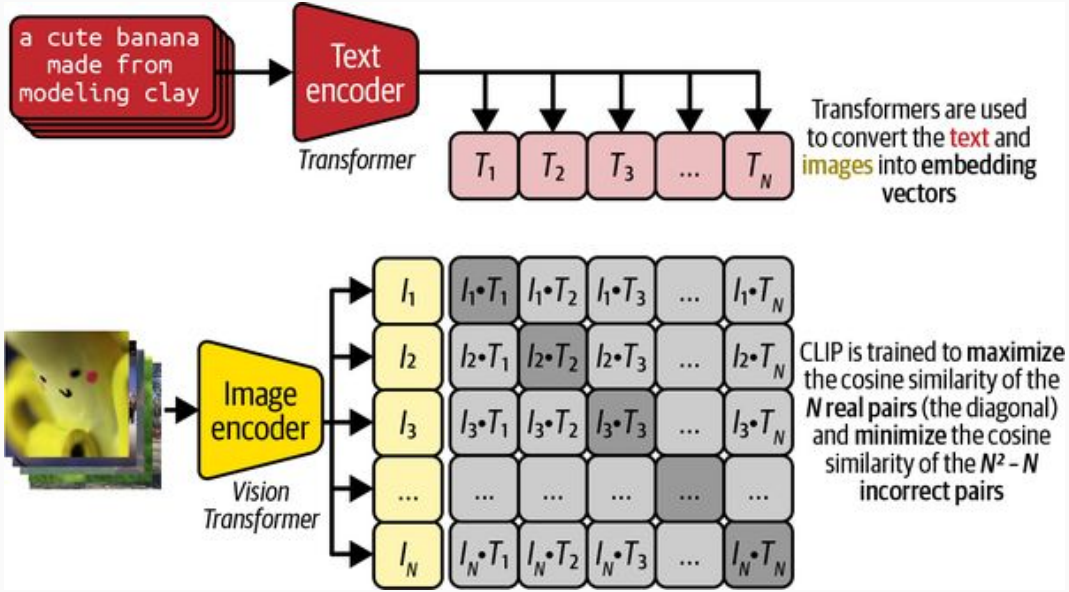
Workflow - Encoder, Prior, Decoder

There are three distinct parts to consider: the text encoder, the prior, and the decoder.

- Text is first passed through the text encoder to produce a text embedding vector.
- This vector is then transformed by the prior to produce an image embedding vector.
- Finally, this is passed through the decoder, along with the original text, to produce the generated image.

The Text Encoder

- The aim of the text encoder is to convert the text prompt into an embedding vector that represents the conceptual meaning of the text prompt within a latent space.
- In DALL.E 2, the authors do not train the text encoder from scratch, but instead make use of an existing model called Contrastive Language–Image Pre-training (CLIP).

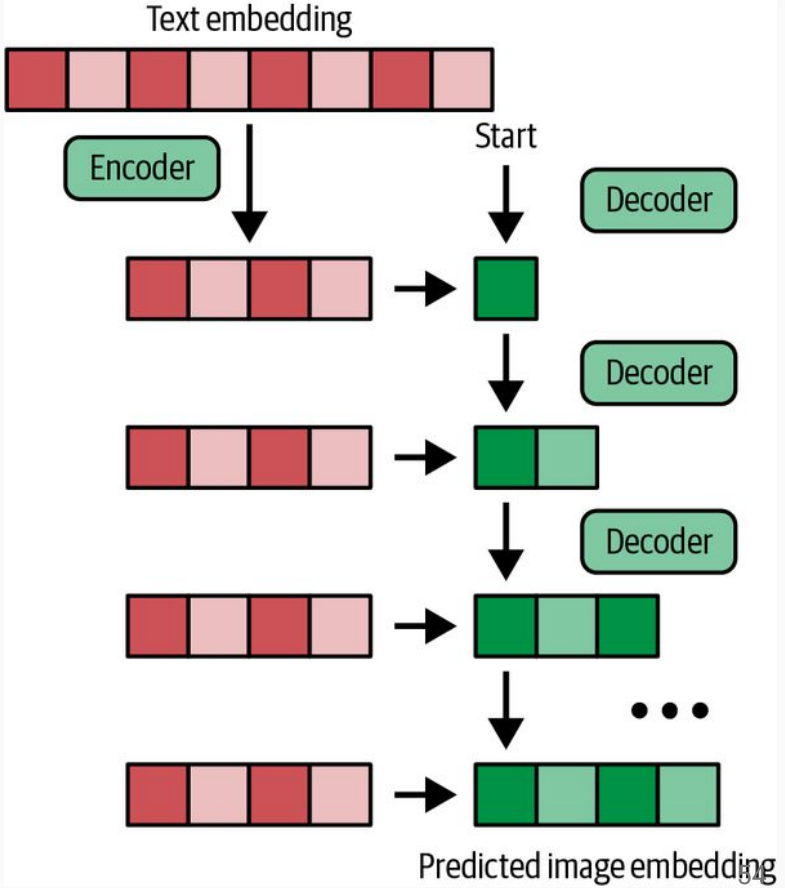


The Prior - converting the text embedding into a CLIP image embedding

- Autoregressive prior - An autoregressive model generates output sequentially, by placing an ordering on the output tokens (e.g., words, pixels) and conditioning the next token on previous tokens.
- The autoregressive prior of DALL.E 2 is an encoder-decoder Transformer. It is trained to reproduce the CLIP image embedding given a CLIP text embedding.

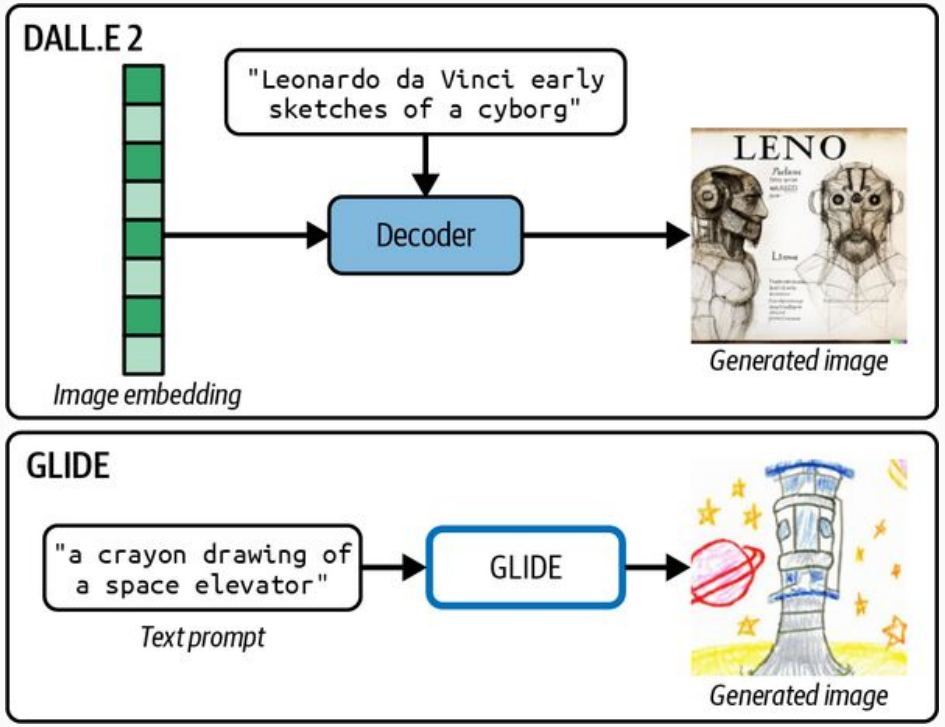
A simplified diagram of the autoregressive prior of DALL.E 2

- The model is trained on the CLIP text–image pair dataset.
- We are converting a vector from the text embedding latent space to the image embedding latent space.
- The input text embedding is processed by the encoder of the Transformer to produce another representation
- That is fed to the decoder, alongside the current generated output image embedding.

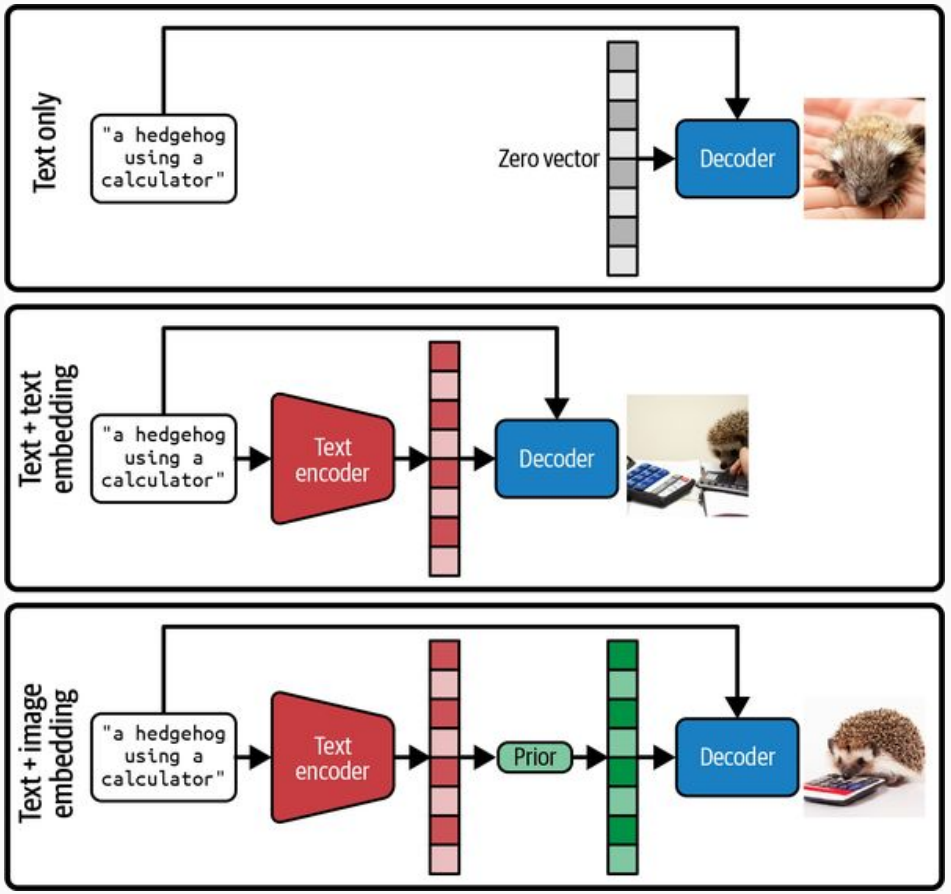


The Decoder

- This is the part of the model that generates the final image conditioned on the text prompt
- and the predicted image embedding output by the prior.



The prior - provides the model with additional context and helps the decoder to produce more accurate generations



BLIP2: Bootstrapping Language Image Pretraining with Frozen Image Encoders and Large Language Models

Introduction

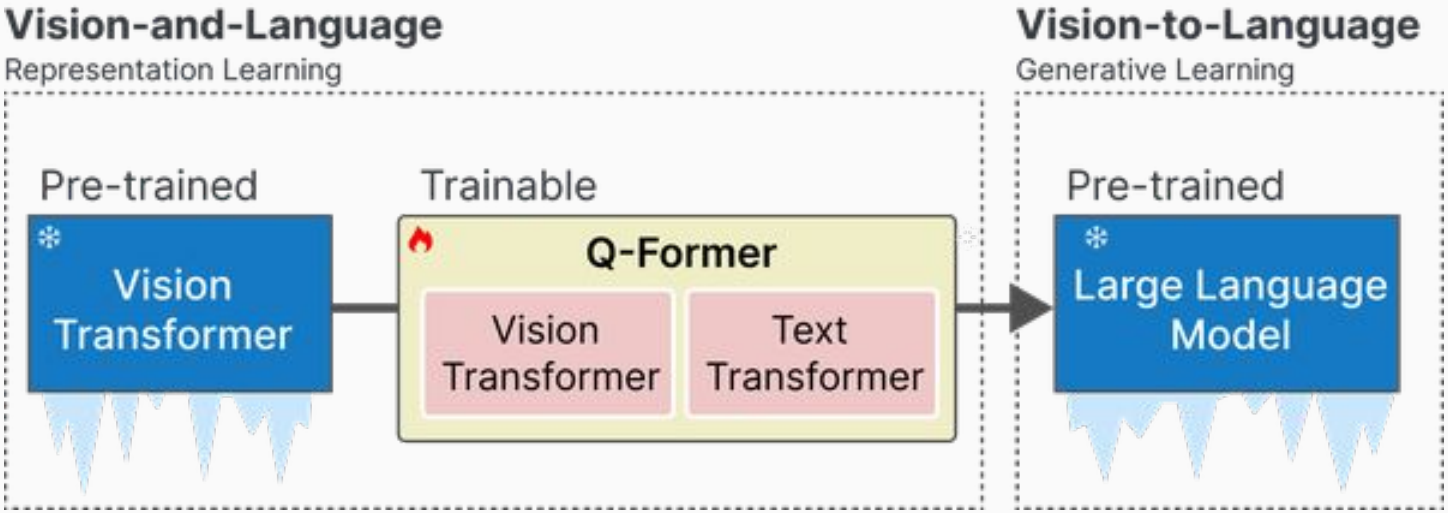
- BLIP (Bootstrapping Language Image Pretraining) is a new framework for vision language pretraining
- Enables a wider range of downstream tasks compared to existing methods
- Introduces Multimodal Mixture of Encoder Decoder (MED) architecture for multitask pretraining
- Proposes Captioning and Filtering (CapFilt) method to learn from noisy image text pairs by generating synthetic captions and filtering noisy ones
- Achieves stateoftheart results on imagetext retrieval, image captioning, VQA, visual reasoning, and visual dialog tasks

Motivation

- Most existing vision language models are limited to either understanding based tasks (retrieval) or generation based tasks (captioning)
- Performance gains have relied on scaling noisy web collected imagetext datasets which are suboptimal for learning
- BLIP aims to build a unified framework supporting both understanding and generation tasks
- BLIP utilizes web datasets more effectively by filtering noise and generating synthetic captions

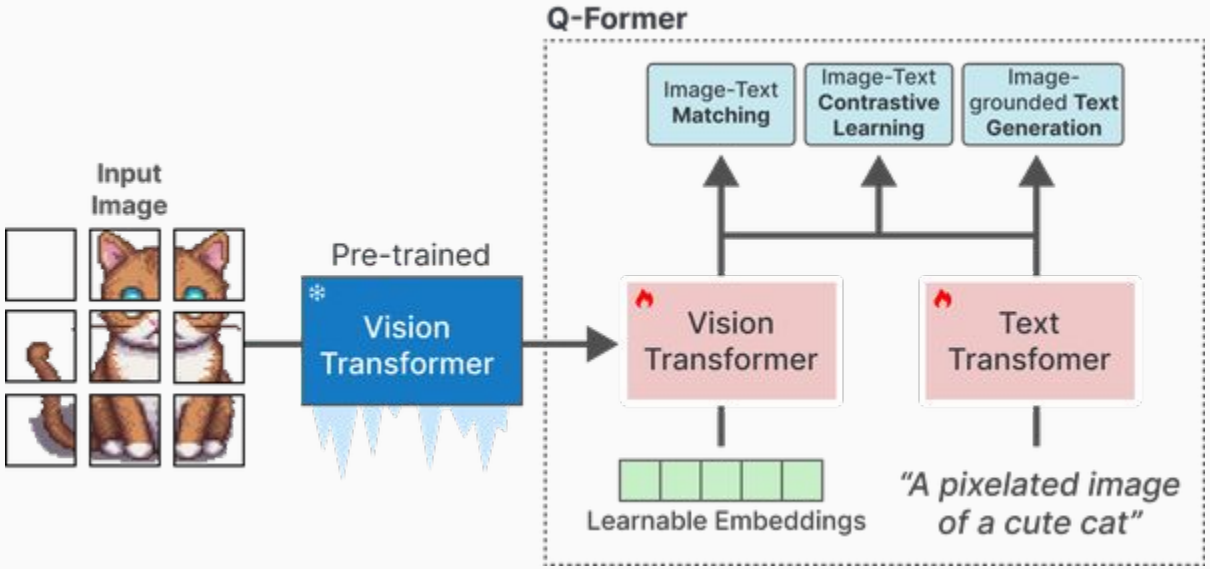
QFormer

- In step 1, representation learning is applied to learn representations for vision and language simultaneously.
- In step 2, these representations are converted to soft visual prompts to feed the LLM.



Step 1

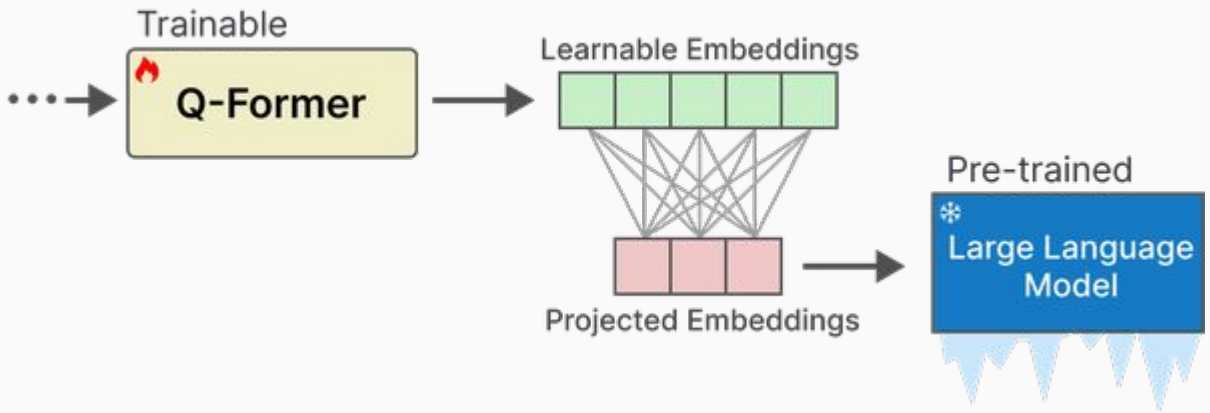
The output of the frozen vision transformer is used together with its caption and trained on three contrastive like tasks to learn visual text representations.



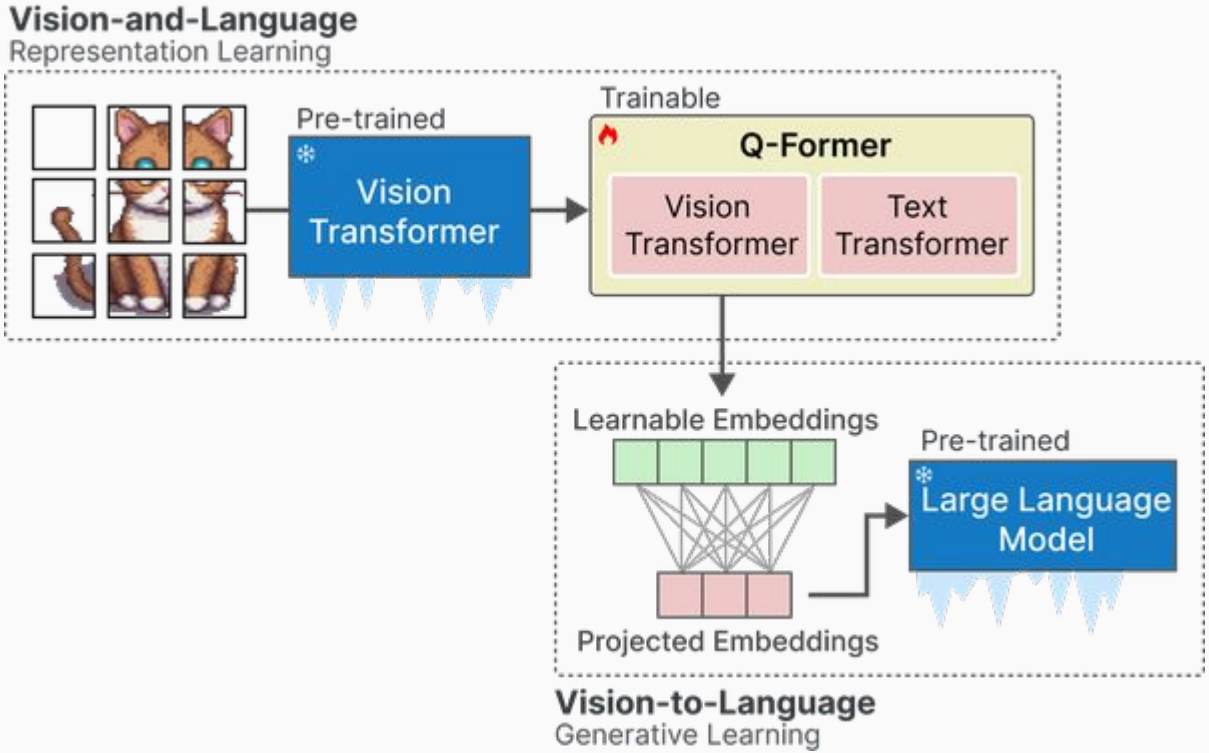
Step 2

The learned embeddings from the QFormer are passed to the LLM through a projection layer.

The projected embeddings serve as a soft visual prompt.



BLIP2


Vision-and-Language
Representation Learning
Pre-trained
Vision Transformer
Trainable
Q-Former
Vision Transformer
Text Transformer
Learnable Embeddings
Pre-trained
Large Language Model
Projected Embeddings
Vision-to-Language
Generative Learning

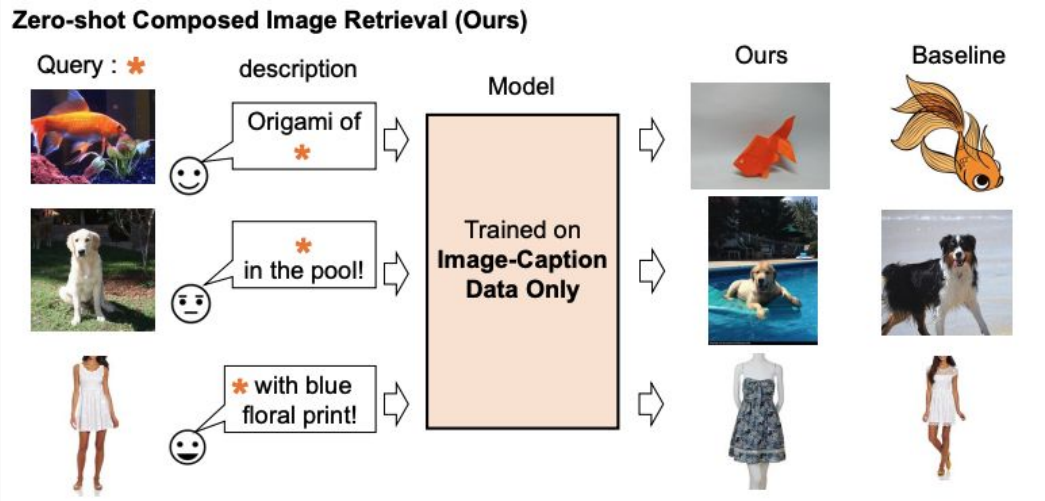
Pic2Word: Mapping Pictures to Words for Zero-shot Composed Image Retrieval

Composed Image Retrieval (CIR)

Definition: Retrieving images based on a query composed of an image and a textual description

Applications:

- Attribute editing (e.g., changing color, style, or texture)
- Object composition (e.g., adding or removing objects)
- Domain conversion (e.g., converting a real image to a sketch or cartoon)



Saito, Kuniaki, et al. "Pic2word: Mapping pictures to words for zero-shot composed image retrieval." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023.

Pic2Word Approach - Two-stage training process

1. Contrastive Language-Image Pretraining (CLIP) - Learns to align visual and textual representations using image-caption pairs
2. Mapping Network Training - Bridges the gap between visual and textual representations using unlabeled images

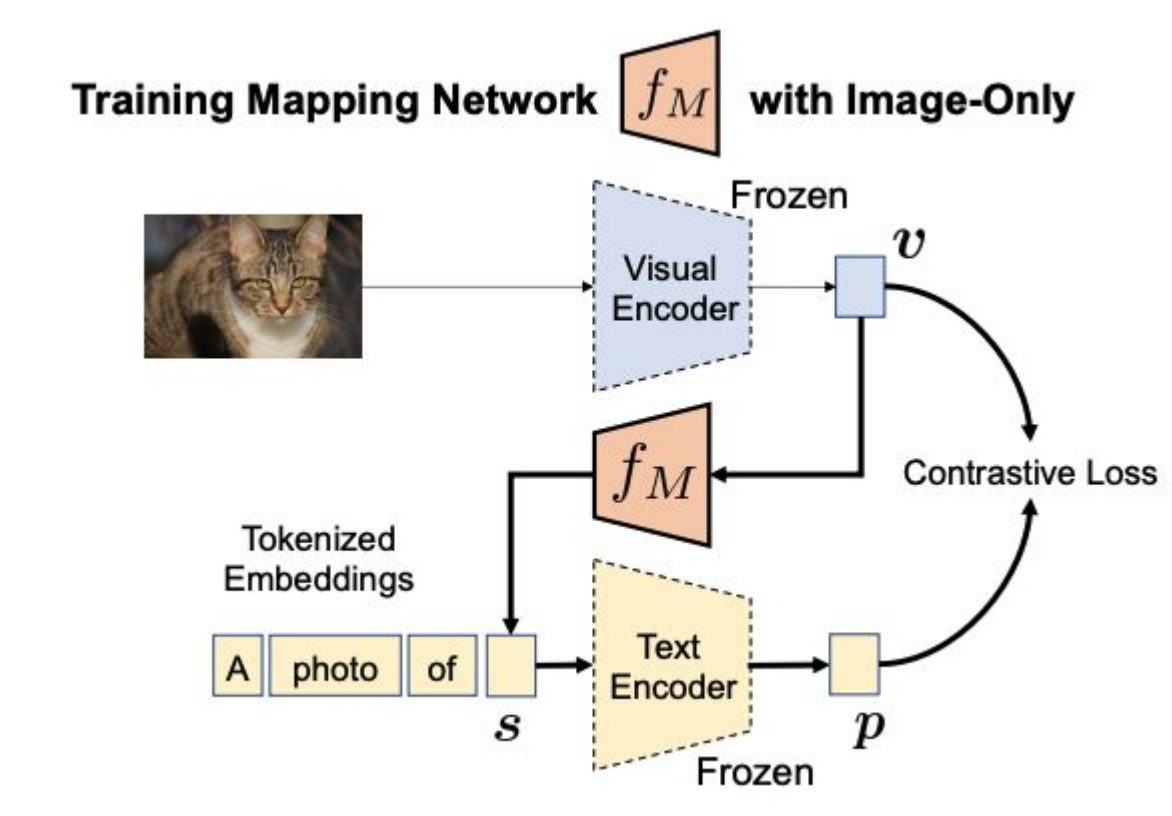
Inference: Composes query image and text modification to retrieve relevant images

Mapping Network Training

- Objective: Reconstruct original image embedding from the pseudo-token using contrastive loss
- Lightweight mapping network transforms image embeddings into pseudo-token embeddings
- Enables language encoder to flexibly compose transformed visual features with textual descriptions
- Scalable and cost-effective due to the use of unlabeled images

Pic2Word mapping network.

- Given a frozen visual and text encoder
- The mapping network, f_M, is optimized to minimize the contrastive loss between the image embedding and the language embedding
- That is generated by the pseudo word token s

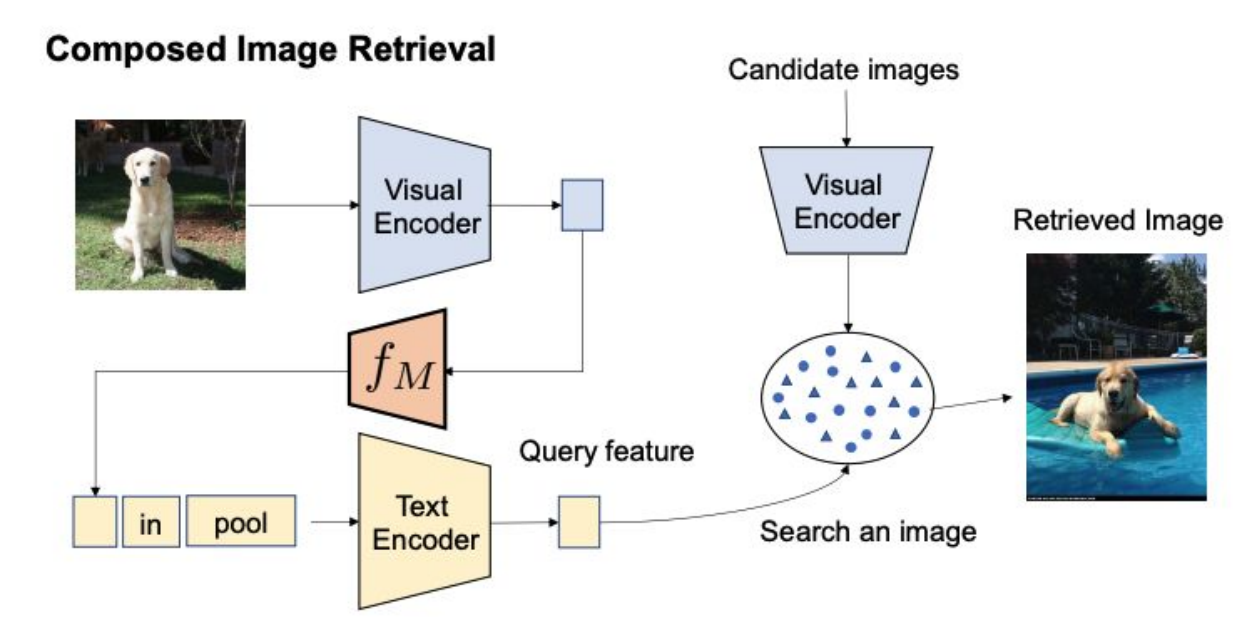


Inference Process

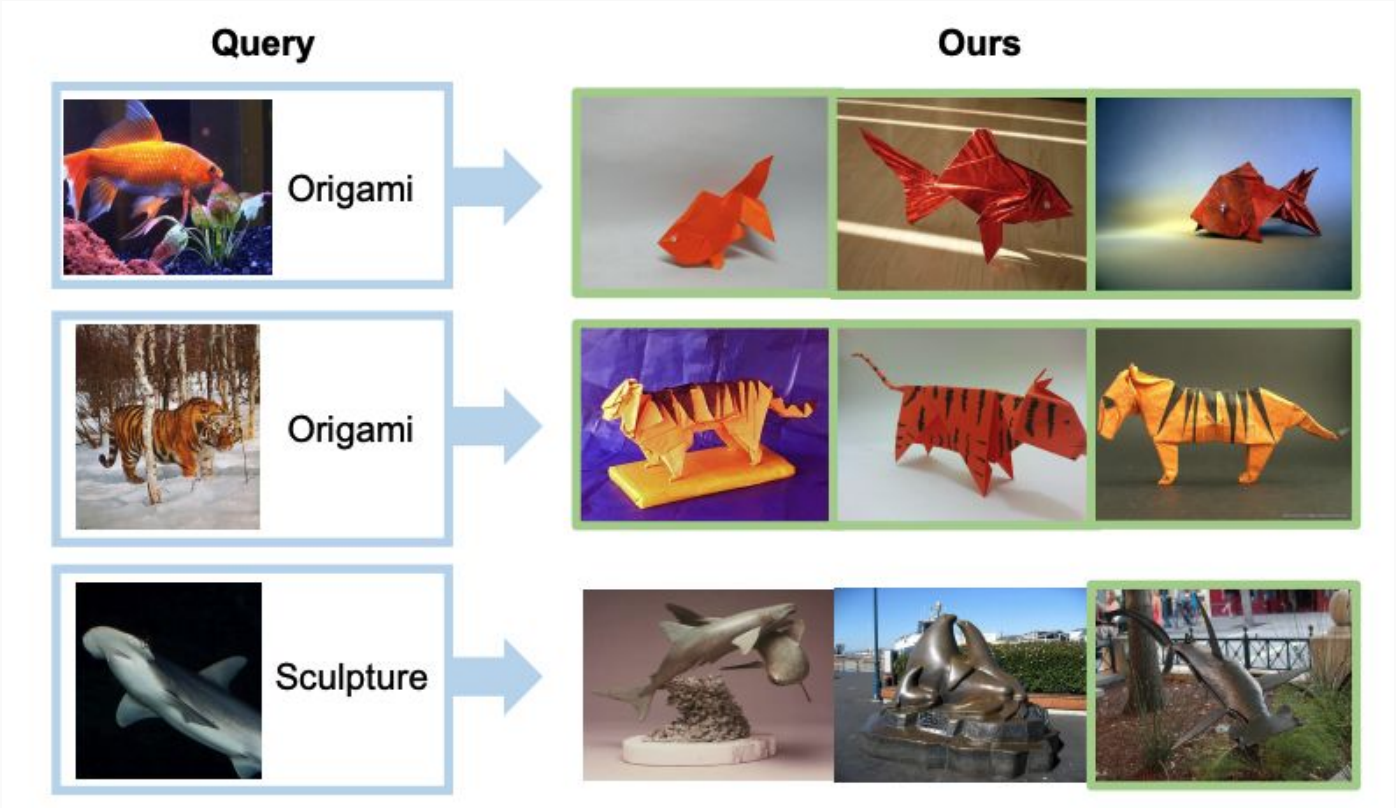
1. Pass query image through visual encoder to obtain visual embedding
2. Transform visual embedding into a pseudo image token using mapping network
3. Insert pseudo image token into a text prompt template along with query text modification
    a. Text prompt template is designed to capture the desired modification (e.g., "a [domain] of [pseudo image token]")
4. Retrieve relevant images by comparing composed query embedding with candidate image embeddings
    a. Similarity calculated using a metric such as cosine similarity

Overview of inference

The estimated token is used to fill in the given prompt sentence.



Top-3 retrievals in the domain-conversion experiment



Strengths of Pic2Word

- Zero-shot generalization: Performs well on unseen CIR tasks without task-specific training data
- Cost-effectiveness: Leverages readily available image-caption pairs and unlabeled images
- Flexibility and versatility: Handles a wide range of CIR tasks using a single model
- Scalability: Training process, especially the mapping network stage, scales well to large datasets
- Interpretability: Representing images as pseudo language tokens enhances explainability

ScreenAI: A Vision-Language Model for UI and Infographics Understanding

ScreenAI

- ScreenAI is a Vision-Language Model (VLM) for comprehensive UI and infographics understanding
- Combines PaLI architecture with flexible patching from Pix2struct
- Introduces a textual UI annotation schema used to automatically generate pre-training data
- Achieves state-of-the-art results on UI and infographic tasks at only 5B parameters

Baechler, G., Sunkara, S., Wang, M., Zubach, F., Mansoor, H., Etter, V., ... & Sharma, A. (2024). ScreenAI: A Vision-Language Model for UI and Infographics Understanding. *arXiv preprint arXiv:2402.04615*.

Methodology

- Encoder-decoder transformer architecture
- Vision encoder (ViT) processes image patches
- Language encoder (mT5/UL2) processes concatenated text and image embeddings
- Decoder generates text output
- Uses Pix2struct's adaptive patching to handle different image sizes and aspect ratios
- Trained models with 670M, 2B and 5B parameters

Baechler, G., Sunkara, S., Wang, M., Zubach, F., Mansoor, H., Etter, V., ... & Sharma, A. (2024). ScreenAI: A Vision-Language Model for UI and Infographics Understanding. *arXiv preprint arXiv:2402.04615*.

Automatic Data Generation

- Uses specialized models to automatically annotate UI screenshots at scale
- Annotation pipeline: object detection, icon classification, OCR, image captioning
- Generates a detailed textual "screen schema" describing the UI
- Uses the screen schema with LLMs to automatically generate QA, navigation, summarization tasks
- Allows creating large and diverse pretraining datasets with minimal manual labeling

Baechler, G., Sunkara, S., Wang, M., Zubach, F., Mansoor, H., Etter, V., ... & Sharma, A. (2024). ScreenAI: A Vision-Language Model for UI and Infographics Understanding. *arXiv preprint arXiv:2402.04615*.

Pretraining and Fine-tuning

- Pre-Training tasks: screen annotation, QA, navigation, summarization
- Pre-Training data sources: auto-generated tasks, web images, charts, VQA
- Fine-tuning tasks: screen analysis, QA, navigation, summarization, infographic/doc QA
- Fine-tuning data: human-labeled datasets and benchmarks
- Much more pretraining data compared to fine-tuning

Baechler, G., Sunkara, S., Wang, M., Zubach, F., Mansoor, H., Etter, V., ... & Sharma, A. (2024). ScreenAI: A Vision-Language Model for UI and Infographics Understanding. *arXiv preprint arXiv:2402.04615*.

Experiments and Results

- ScreenAI achieves state-of-the-art on MoTIF, Widget Captioning, MPDocVQA, WebSRC
- Best-in-class on ChartQA, DocVQA, InfographicVQA
- Maintains competitive performance on other tasks like Screen2Words and OCR-VQA
- Introduces new benchmarks: Screen Annotation, ScreenQA Short, Complex ScreenQA
- Ablations show benefits of Pix2struct patching and LLM-generated pretraining data
- Performance consistently improves with model scale from 670M to 5B parameters

Baechler, G., Sunkara, S., Wang, M., Zubach, F., Mansoor, H., Etter, V., ... & Sharma, A. (2024). ScreenAI: A Vision-Language Model for UI and Infographics Understanding. *arXiv preprint arXiv:2402.04615*.

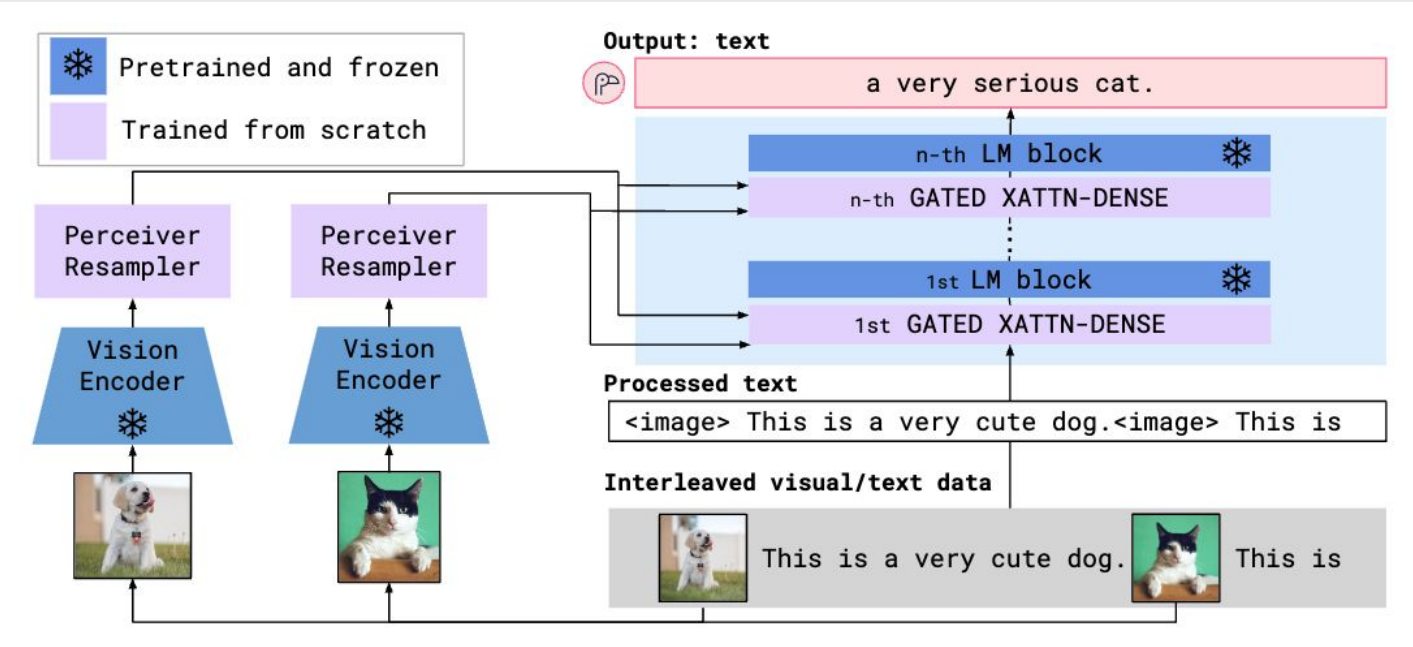
Flamingo

Flamingo - Visual Language Model (VLM)

- Flamingo can adapt to new tasks simply by being prompted with a few input/output examples, without the need for fine-tuning.
- Flamingo leverages two pre-trained and frozen models: a vision model for "perceiving" visual scenes and a large language model for basic reasoning.
- Flamingo can process high-resolution images or videos using a Perceiver-based architecture.
- The training data for Flamingo consists of carefully selected, large-scale multimodal web data, without using any data annotated for machine learning purposes.

Alayrac, Jean-Baptiste, et al. "Flamingo: a visual language model for few-shot learning." *Advances in neural information processing systems* 35 (2022): 23716-23736.

Flamingo architecture overview



- Input - visual data interleaved with text
- Output - free-form text

Approach

- The Perceiver Resampler receives spatio-temporal features from the Vision Encoder (obtained from either an image or a video) and outputs a fixed number of visual tokens.
- These visual tokens are used to condition the frozen language model (LM) using freshly initialized cross-attention layers that are interleaved between the pretrained LM layers.
- These new layers allow the LM to incorporate visual information for the next-token prediction task.
- Flamingo models the likelihood of text conditioned on interleaved images and videos using a product of conditional probabilities
- Each token's probability is conditioned on the preceding tokens and the images/videos preceding the token in the interleaved sequence.

Vision Encoder

- It uses a pretrained and frozen Normalizer-Free ResNet (NFNet) F6 model.
- The encoder is pre trained using a contrastive objective on datasets of image and text pairs, employing the two-term contrastive loss from Radford et al.
- The output is a 2D spatial grid of features, which is flattened to a 1D sequence.
- For video inputs, frames are sampled at 1 FPS and encoded independently, resulting in a 3D spatio-temporal grid of features with learned temporal embeddings added.
- The features are then flattened to 1D before being fed to the Perceiver Resampler.

The Vision Encoder

The Vision Encoder is responsible for processing visual data (images or videos) within the input and converting it into embedding vectors.

- Flamingo vs. text-to-image models:
  - Unlike pure text-to-image models like DALL.E 2 and Imagen, Flamingo can accept a combination of interleaved text and visual data, including both images and videos.
- Vision Encoder architecture:
  - Flamingo uses a pre-trained Normalizer-Free ResNet (NFNet) as the Vision Encoder, specifically the NFNet-F6 model.
  - This differs from the CLIP image encoder, which uses a Vision Transformer (ViT) architecture.
- Training:
  - The Vision Encoder is trained on image-text pairs using the same contrastive objective as introduced in the CLIP paper.
  - After training, the weights of the Vision Encoder are frozen, so any further training of the Flamingo model does not affect the Vision Encoder's weights.

The Vision Encoder - Output and Video

- Output
  - The Vision Encoder produces a 2D grid of features, which is then flattened to a 1D vector before being passed to the Perceiver Resampler.

- Handling video:
  - For video input, frames are sampled at 1 frame per second and passed through the Vision Encoder independently, producing several feature grids.
  - Learned temporal encodings are then added before flattening the features and concatenating the results into a single vector.

Perceiver Resampler

- This module connects the vision encoder to the frozen language model.
- It takes a variable number of image or video features from the vision encoder and produces a fixed number of visual outputs (64), reducing the computational complexity of the vision-text cross-attention.
- It learns a predefined number of latent input queries which are fed to a Transformer and cross-attend to the visual features.
- Studies show that using a vision-language resampler module outperforms a plain Transformer and an MLP.

Conditioning frozen language models on visual representations for text generation

- The text generation is performed by a Transformer decoder, which is conditioned on the visual representations produced by the Perceiver Resampler.
- Pretrained and frozen text-only language model (LM) blocks are interleaved with blocks trained from scratch that cross-attend to the visual output from the Perceiver Resampler.
- The interleaved blocks are called gated cross-attention dense blocks (GATED XATTN-DENSE). These blocks are inserted between the original LM layers and are trained from scratch.
- To ensure that the conditioned model yields the same results as the original language model at initialization, a tanh-gating mechanism is used.
- This mechanism multiplies the output of a newly added layer by tanh(α) before adding it to the input representation from the residual connection, where α is a layer-specific learnable scalar initialized to 0.

Handling multiple visual inputs (images or videos) in a sequence

- The image-causal modeling is achieved by masking the full text-to-image cross-attention matrix, which limits the visual tokens that the model sees at each text token.
- At a given text token, the model attends only to the visual tokens of the image that appeared just before it in the interleaved sequence, rather than attending to all previous images.
- Although the model directly attends to a single image at a time, the dependency on all previous images is maintained through self-attention in the language model (LM).
- This single-image cross-attention scheme allows the model to seamlessly generalize to any number of visual inputs, regardless of how many are used during training.
- During training, the model uses up to 5 images per sequence. However, during evaluation, the model can benefit from sequences of up to 32 pairs (or "shots") of images/videos and corresponding texts.
- This approach enables the Flamingo model to efficiently process multiple visual inputs while maintaining the ability to generalize to different numbers of visual inputs during evaluation.

Limitations

- Inherited weaknesses from pretrained language models (LMs):
  - Flamingo models build on pretrained LMs and inherit their weaknesses.
  - LM priors, while generally helpful, may contribute to occasional hallucinations and ungrounded guesses.
- Classification performance lags behind state-of-the-art contrastive models:
  - Flamingo's classification performance is not as good as state-of-the-art contrastive models, which directly optimize for text-image retrieval.
- Drawbacks of in-context learning:
  - In-context learning is highly sensitive to various aspects of the demonstrations and its inference compute cost and absolute performance scale poorly with the number of shots beyond the low-data regime.

References

- Foster, D. (2022). *Generative deep learning*. " O'Reilly Media, Inc. - Chapter 13
- Hands-On Large Language Models Jay Alammar, Maarten Grootendorst - Chapter 5

This Week

Homework

a. Run this notebook - Please write a summary (in a paragraph or two) of how the code in the notebook works.
b. Run this notebook - Please discuss the positives/negatives of the Hugging Face workflow