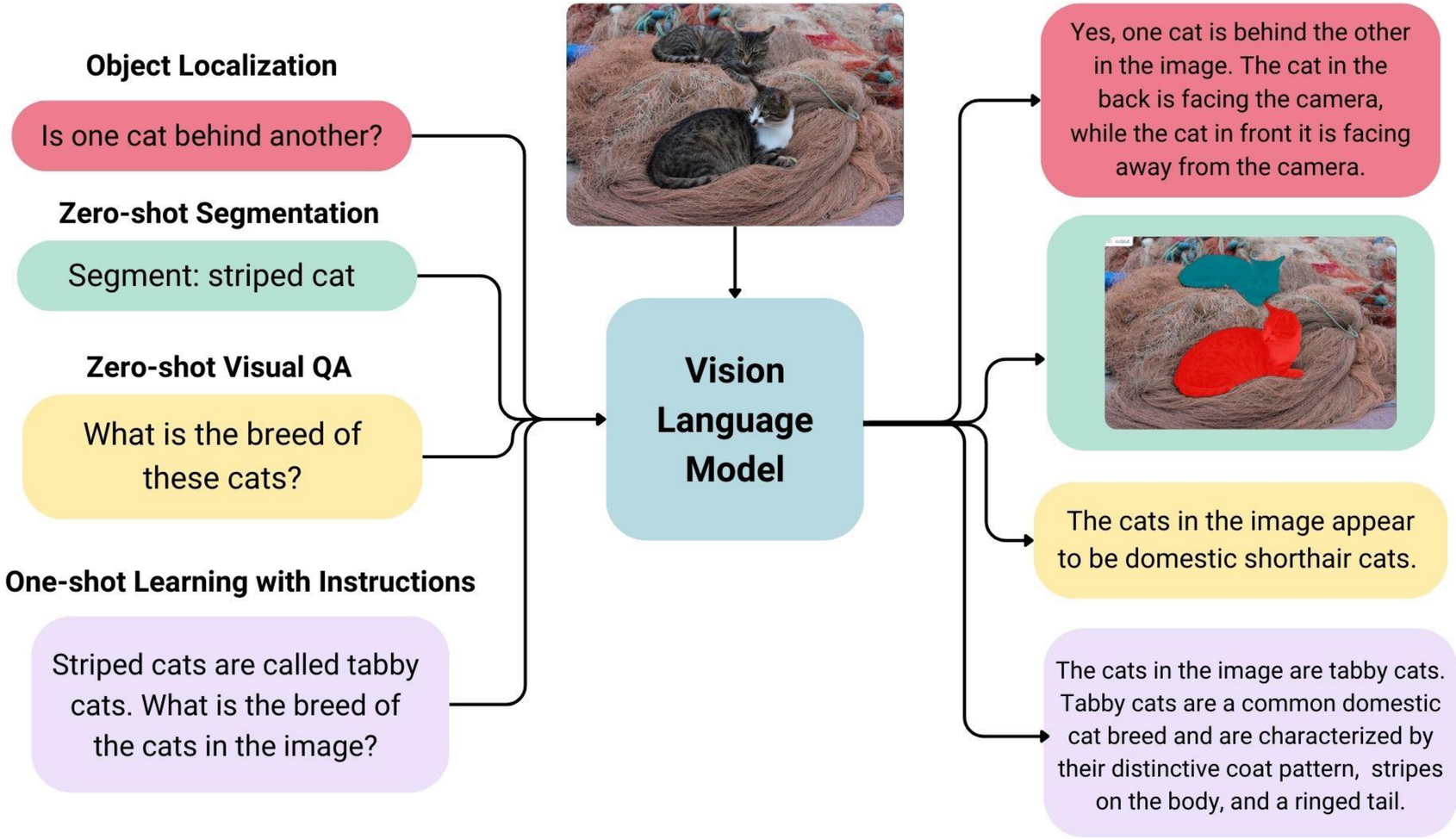
Google Developer Experts

Vision-Language Models

Ayyuce Demirbas
Google Developer Expert in ML

Vision Language Models



Source: https://huggingface.co/blog/vlms (Retrieved February 1, 2025.)

Tasks

Image Captioning

Visual Question Answering (VQA)

Text-to-Image Generation

Image Retrieval

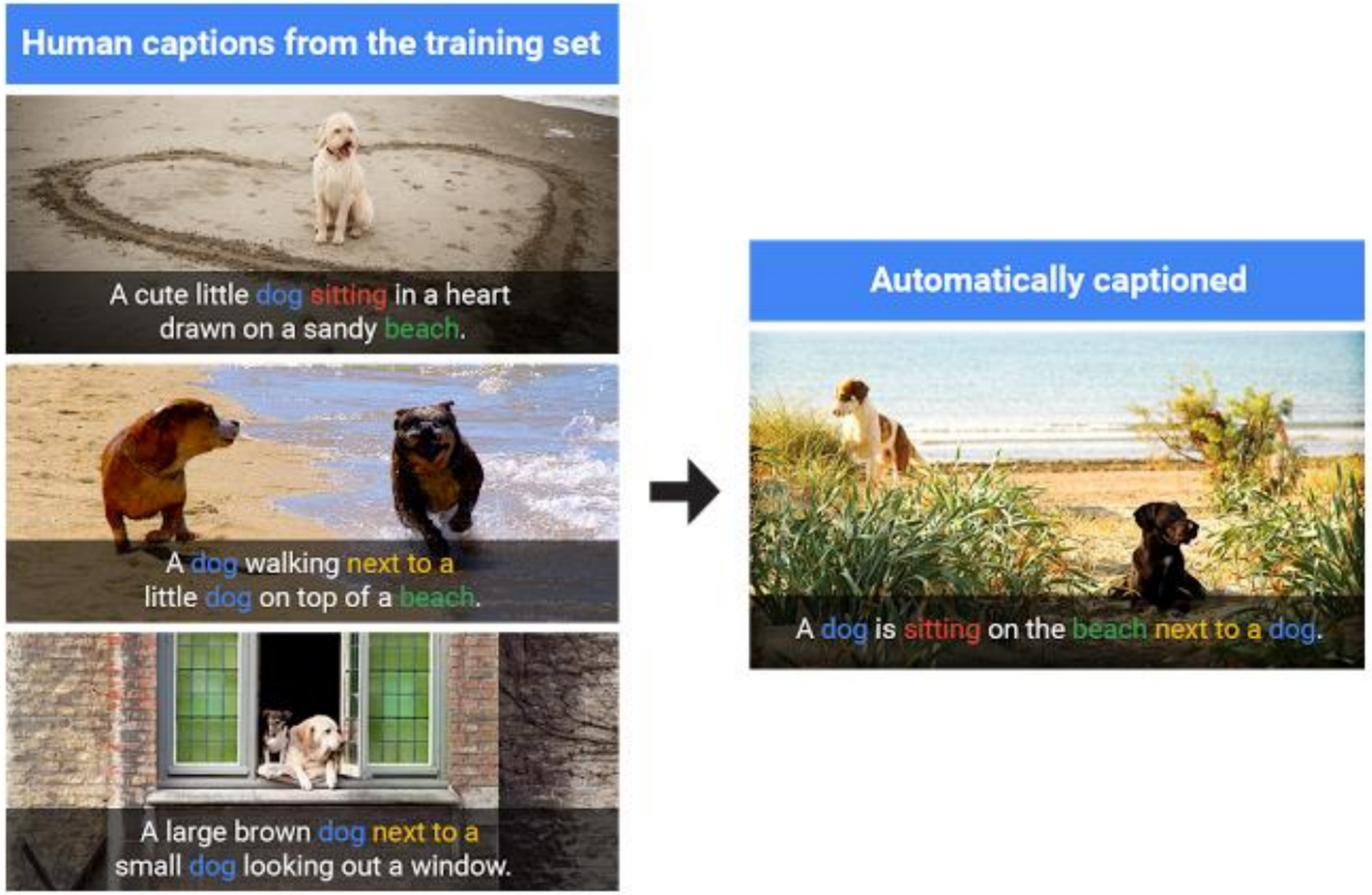
Image Editing with Textual Guidance

Document Understanding

Vision-Language Image Segmentation

Vision-Language Object Detection

Image Captioning



Source: https://www.popularmechanics.com/technology/robots/a23019/google-ai-captioning/ (Retrieved February 1, 2025.)

Image Captioning

What is Image Captioning?

Image captioning is the task of automatically generating a textual description of the content of an image. This involves understanding the objects, attributes, and relationships within the image and then expressing this understanding in natural language.

How Does Image Captioning Work?

Image captioning typically involves a combination of Convolutional Neural Networks (CNNs)or Vision Transformers (ViTs) for image understanding and Recurrent Neural Networks (RNNs) or Transformers for text generation.

Image Captioning - Example Models

ReCap: This model uses a linear alignment approach to map image and text embeddings efficiently. It's lightweight and can be trained up to 1000 times faster than existing methods. ReCap performs well on established metrics and new metrics better aligned with human ratings.

FUSECAP: This model leverages large language models (LLMs) and vision experts (like object detectors and OCR) to produce enriched, detailed captions. It outperforms current state-of-the-art approaches by generating more precise descriptions.

Image Captioning - Example Models

CLIP: A well-known model that has advanced the state of the art in multi-modal tasks, including image captioning. It uses a contrastive learning approach to align image and text embeddings.

BLIP: This model combines vision-language pre-training techniques with a large language model to generate captions. It has shown significant progress in producing detailed and accurate image descriptions.

CapText: This model leverages large language models to generate captions from textual descriptions and context alone, without directly processing the image, achieving performance that surpasses some existing image-text alignment models.

Image Captioning - Example Models

LEMON (LargE-scale iMage captiONer): LEMON scales up vision-language pre-training by utilizing up to 200 million image-text pairs. This extensive dataset enables the model to generate detailed and accurate captions, achieving state-of-the-art results on benchmarks like COCO Caption and nocaps.

VinVL (Visual Representations for Vision-Language): Developed by Microsoft Research, VinVL enhances image captioning by improving object detection and attribute recognition. This model has set new performance standards on several vision-language tasks, including image captioning.

Image Captioning - Example Datasets

COCO Captions: It is a large-scale dataset used for training and evaluating image captioning models. It was introduced by Microsoft and contains over 1.5 million captions describing more than 330,000 image.

Flickr30k: Comprising 31,000 images sourced from Flickr, this dataset includes five captions per image. It emphasizes everyday activities, events, and scenes, providing a rich resource for training and evaluating captioning models.

Image Captioning - Example Datasets

Conceptual Captions: This large-scale dataset consists of approximately 3.3 million images paired with captions derived from web alt-text. It offers a wide variety of visual concepts and linguistic expressions, enhancing the diversity of training data for captioning models.

VizWiz-Captions: Consisting of over 39,000 images originating from people who are blind, each paired with five captions, this dataset provides a unique perspective, emphasizing accessibility and real-world applicability in image captioning.

NoCaps: Designed to evaluate models on their ability to caption images containing novel objects not seen during training, NoCaps includes 15,100 images with 166,100 human-generated captions, challenging models to generalize beyond their training data.

Image Captioning - Example Datasets

TextCaps: Focusing on images that contain textual information, TextCaps comprises 28,000 images with 145,000 captions. It challenges models to read and comprehend text within images to generate accurate descriptions.

SCICAP: A large-scale dataset containing real-world scientific figures and captions, constructed using more than two million images from over 290,000 papers collected from arXiv. It is valuable for models aiming to generate captions for scientific imagery.

Indiana University Chest X-ray Collection (IU X-ray dataset): A publicly available dataset of chest radiographs and their corresponding reports. This dataset is widely used in the medical imaging community for tasks such as image captioning and report generation.

Image Captioning - Metrics

BLEU (Bilingual Evaluation Understudy): Originally developed for machine translation, BLEU measures the precision of n-grams in the generated caption against reference captions. It evaluates how many n-grams in the generated text appear in the reference texts.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation): ROUGE focuses on recall, assessing the overlap of n-grams between the generated and reference captions. It is particularly useful for evaluating the coverage of content in the generated captions.

CIDEr (Consensus-based Image Description Evaluation): CIDEr evaluates the consensus between the generated and reference captions by computing the cosine similarity between their TF-IDF vectors. It emphasizes the importance of content that is commonly agreed upon in the reference captions.

Image Captioning - Metrics

SPICE (Semantic Propositional Image Caption Evaluation): SPICE assesses the semantic content of the generated captions by comparing scene graphs, which represent objects and their relationships, extracted from both the generated and reference captions.

CLIPScore: Leveraging the CLIP model, which aligns visual and textual representations, CLIPScore evaluates the compatibility between the generated caption and the image without relying on reference captions. It has been shown to correlate well with human judgments.

FLEUR: An explainable reference-free metric that utilizes a large multimodal model to assess the quality of generated captions. FLEUR provides explanations for its scores, enhancing interpretability in evaluation.

Google Developer Experts

Image Captioning Example Notebook

https://github.com/ayyucedemirbas/VLM_from_scratch/blob/main/a_vision_language_model_from_scratch.ipynb

Visual Question Answering (VQA)

Q: How old do you have to be in Canada to do this?

A: 18

Q: What is the hairstyle of the person on the left called?

A: pony tail, ponytail

Q: When was this piece of sporting equipment invented?

A: 1926

Q: What toy store used a mascot similar to these animals for many years?

A: Toys R Us

Source: https://medium.com/data-science-at-microsoft/visual-question-answering-with-multimodal-transformers-d4f57950c867 (Retrieved February 1, 2025.)

Visual Question Answering (VQA)

Q: In what country would you find this bus?

A: United Kingdom, England

Q: What creates those type of clouds?

A: water vapor

Q: What movie is the elephant from?

A: Horton Hears a Who

Source: https://medium.com/data-science-at-microsoft/visual-question-answering-with-multimodal-transformers-d4f57950c867 (Retrieved February 1, 2025.)

Visual Question Answering (VQA)

Q: What century is this?

A: 20th

Q: When is best to use this toy?

A: when its windy, windy days

Q: What is the process called that produces the red area on the chair?

A: rust, oxidization

Q: Which of these streets is famous for theater?

A: Broadway

Q: If wearing proper glasses what might this picture do?

A: pop out, 3d

Q: Who is the owner of this building?

A: Pope, Catholic Church

Q: Where is the monument located?

A: washington dc

Q: Can you guess the celebration where the people are enjoying?

A: fourth of July, 4th of July

Source: https://blog.roboflow.com/what-is-vqa/ (Retrieved February 1, 2025.)

Visual Question Answering (VQA) - Example Models

“PaliGemma: Developed by Google, PaliGemma is a vision-language model with multimodal capabilities, enabling it to process and generate both visual and textual information.

GPT-4 with Vision: An extension of OpenAI's GPT-4, this model integrates vision capabilities, allowing it to interpret images and answer related questions.

LLaVA-1.5: An open-source multimodal language model designed for visual question answering, with limited support for object detection. “

Source: https://blog.roboflow.com/what-is-vqa/ (Retrieved February 1, 2025.)

Visual Question Answering (VQA) - Example Models

“BLIP-2: BLIP-2 (Bootstrapping Language-Image Pre-training) is a VQA model that emphasizes efficiency, achieving performance comparable to state-of-the-art models with a more streamlined approach.

Pix2Struct: A deep learning model that tackles VQA by leveraging image-to-text translation, utilizing an encoder-decoder transformer architecture to analyze visual components and comprehend textual content.

CogVLM: This model demonstrates strong performance in VQA and other vision tasks, showcasing its versatility in processing and understanding visual information. “

Source: https://blog.roboflow.com/what-is-vqa/ (Retrieved February 1, 2025.)

Querying Transformer (Q-Former)

The Querying Transformer (Q-Former) is a pivotal component in BLIP-2, designed to bridge the gap between frozen image encoders and large language models (LLMs). Its primary function is to extract and transform visual features into a format that frozen LLMs can interpret, enabling effective vision-language integration.

Querying Transformer (Q-Former) - Architecture

The Q-Former comprises two interconnected transformer submodules that share the same self-attention layers:

Image Transformer: This submodule interacts with a frozen image encoder to extract visual features. It utilizes a set of learnable query embeddings as input, which interact with each other through self-attention layers and with the frozen image features via cross-attention layers.

Text Transformer: Functioning as both a text encoder and decoder, this submodule processes textual data and can interact with the learnable queries through shared self-attention layers.

Querying Transformer (Q-Former)

The Q-Former is initialized with pre-trained BERTbase weights for its self-attention layers, while the cross-attention layers are randomly initialized. In total, the Q-Former contains approximately 188 million parameters, including the learnable queries. Typically, 32 queries are used, each with a dimension of 768, matching the hidden dimension of the Q-Former. This design ensures that the Q-Former extracts the most relevant visual features for the corresponding text.

Querying Transformer (Q-Former)

Pre-Training Strategy

The Q-Former undergoes a two-stage pre-training process to effectively align visual and textual modalities:

Vision-Language Representation Learning: In this initial stage, the Q-Former is connected to a frozen image encoder and trained using image-text pairs. The objective is to enable the Q-Former to extract visual features that are most pertinent to the associated text. This is achieved by jointly optimizing three pre-training objectives:

Image-Text Contrastive Learning (ITC): Aligns image and text representations by maximizing mutual information between them.

Querying Transformer (Q-Former)

Image-Grounded Text Generation (ITG): Trains the Q-Former to generate text conditioned on the visual features extracted from the image.

Image-Text Matching (ITM): A binary classification task where the model predicts whether an image-text pair is matched or unmatched.

Each objective employs different self-attention masking strategies to control the interaction between queries and text, ensuring effective learning.

Querying Transformer (Q-Former)

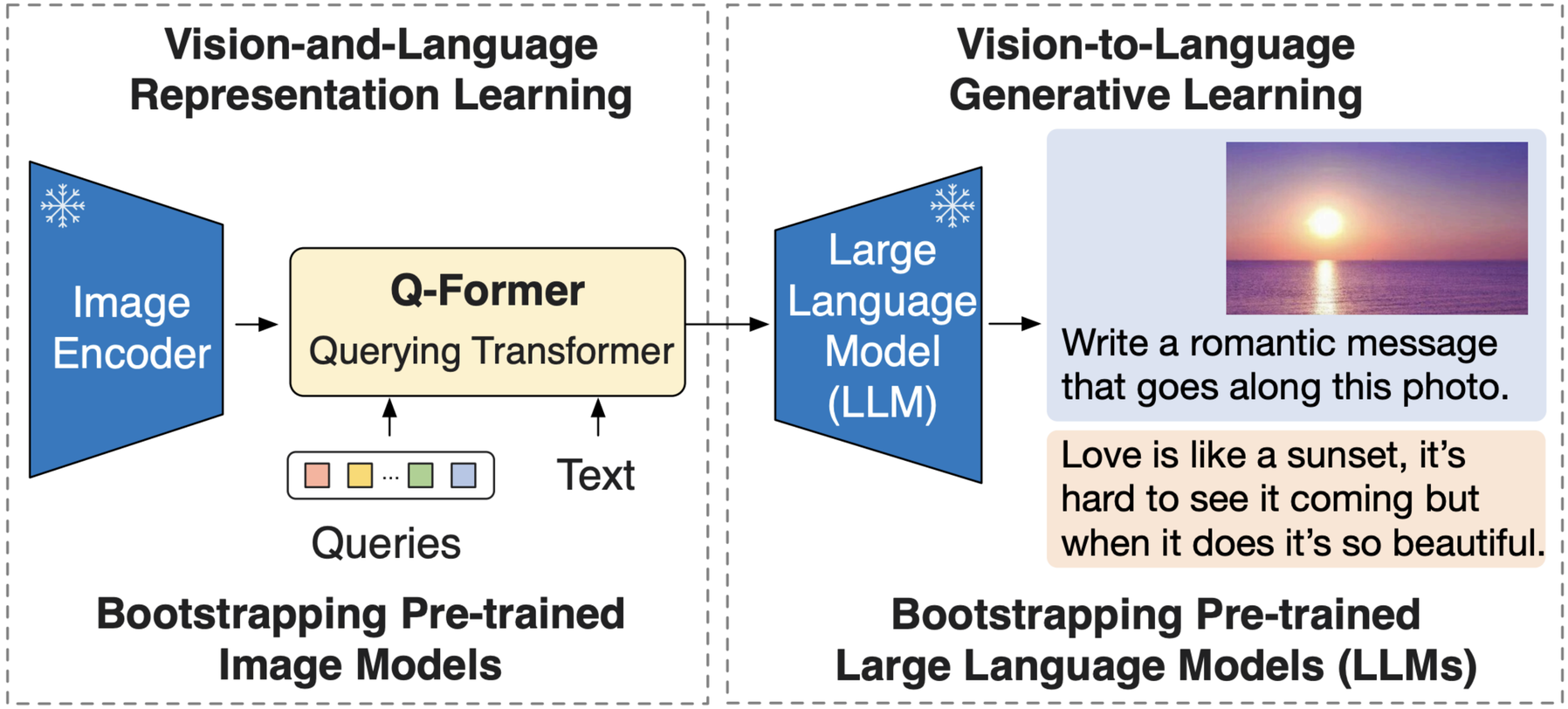
Vision-to-Language Generative Learning: In the second stage, the Q-Former's output is connected to a frozen LLM. The Q-Former is trained to produce output features that the LLM can interpret to generate the corresponding text. This involves using the Q-Former’s output as a visual prefix to the LLM, effectively conditioning the LLM on the visual information extracted by the Q-Former. This approach allows the LLM to generate text that accurately describes the visual content.

Querying Transformer (Q-Former)

Inference Process

During inference, the Q-Former extracts visual features from the input image using the learnable queries. These features are then transformed into a format compatible with the frozen LLM. By appending the text instruction after the Q-Former's output, the LLM can generate natural language responses based on the visual input and the provided instruction. This process enables BLIP-2 to perform tasks such as zero-shot image-to-text generation, where the model generates descriptive text for an image without prior exposure to similar image-text pairs during training.

Querying Transformer (Q-Former)



Li, J., Li, D., Savarese, S., & Hoi, S. (2023, July). Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In International conference on machine learning (pp. 19730-19742). PMLR.

Visual Question Answering (VQA) - Example Models

“**Qwen-VL:** Developed by Alibaba Cloud, Qwen-VL accepts images, text, and bounding boxes as inputs, naturally supporting English, Chinese, and multilingual conversations.“

Source: https://blog.roboflow.com/what-is-vqa/ (Retrieved February 1, 2025.)

Visual Question Answering (VQA) - Example Datasets

VQA Dataset: This foundational dataset contains over 250,000 images and 760,000 questions, each annotated with 10 open-ended answers. It serves as a benchmark for evaluating VQA models.
https://visualqa.org/vqa_v1_challenge.html

GQA (Graph Question Answering): Designed to address limitations in earlier datasets, GQA offers 22 million diverse reasoning questions derived from scene graphs, emphasizing real-world visual reasoning and compositional question answering.
https://arxiv.org/abs/1902.09506

Visual Question Answering (VQA) - Example Datasets

A-OKVQA: This dataset comprises approximately 25,000 questions that require commonsense and world knowledge beyond the image content, challenging models to incorporate external information for accurate answers.
https://arxiv.org/abs/2206.01718

ChiQA: Containing over 40,000 real-world queries paired with more than 200,000 question-image pairs, ChiQA emphasizes fine-grained vision and language reasoning, requiring models to determine the answerability of images concerning user queries.
https://arxiv.org/abs/2208.03030

Visual Question Answering (VQA) - Example Datasets

AQUA (Art Question Answering): Focusing on artworks, AQUA includes question-answer pairs generated from paintings and associated comments, challenging models to understand both visual content and contextual art knowledge. https://arxiv.org/abs/2008.12520

VizWiz: Developed to assist visually impaired individuals, VizWiz features visual questions originating from blind users, providing a unique perspective on real-world VQA applications. https://vizwiz.org/tasks-and-datasets/vqa/

Visual Question Answering (VQA) - Metrics

1. Accuracy-Based Metrics:

- Overall Accuracy:

- Measures the percentage of correctly answered questions.

- For VQA datasets like VQA v2, answers are typically compared against ground truth responses, with a soft voting mechanism applied (e.g., accepting an answer as correct if it matches responses from the majority of annotators).

- **Yes/No, Number, and Other Accuracy**:

- Sub-category evaluation to analyze performance on specific question types, such as binary (Yes/No), numerical answers, or descriptive responses.

Visual Question Answering (VQA) - Metrics

2. Normalized Metrics:

- Consensus Agreement:

- Evaluates how well the model's answers align with human annotators' responses.
- For datasets like VQA v2, answers are scored using:

$$\text{Score} = \min\left(\frac{\text{number of annotators providing this answer}}{3}, 1\right)$$

Visual Question Answering (VQA) - Metrics

3. Retrieval Metrics (For Open-Domain VQA):

- Precision, Recall, and F1-Score:

- Commonly used when VQA involves selecting or generating answers in retrieval-based - tasks.
- Measures the balance between correctly predicted answers and all possible relevant or retrieved answers.

Visual Question Answering (VQA) - Metrics

4. Specific Metrics for Generated Answers:

- BLEU: Evaluates the n-gram overlap between the generated answer and reference answers. Focuses on syntactic matching.

- **METEOR**: Considers synonymy and word order to provide more nuanced evaluations.

- **ROUGE**: Measures the recall of n-grams, primarily used when multiple answers are possible.

- **CIDEr**: Captures consensus by calculating TF-IDF-weighted cosine similarity, particularly for longer, descriptive answers.

Visual Question Answering (VQA) - Metrics

5. Robustness Metrics:

- Human Consistency:

- Evaluates how consistently the model's predictions align with human-annotated answers.

- **Model Robustness (Sensitivity Analysis)**:

- Tests robustness by introducing perturbations in questions or images (e.g., paraphrased questions or occluded images).

Visual Question Answering (VQA) - Metrics

6. Newer Multimodal Metrics:

- CLIPScore: Evaluates the alignment between image and generated answer embeddings using pre-trained CLIP models.

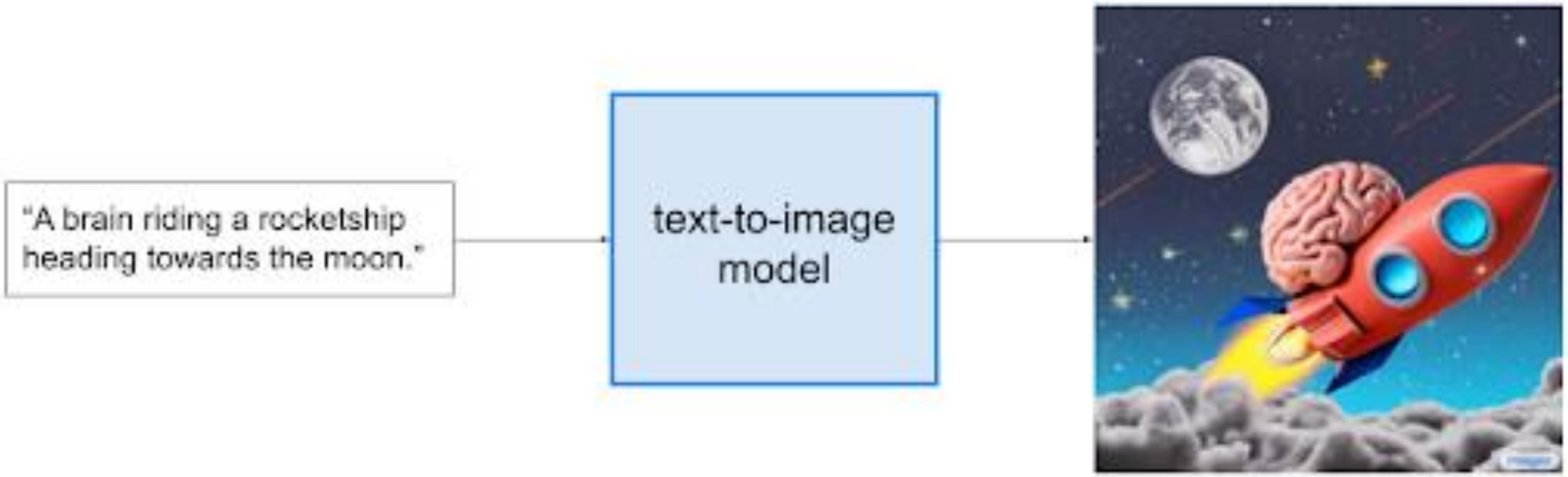
- **VQAR-Score**: A metric specifically designed for VQA tasks that incorporates both visual and textual relevance.

- **Visual Grounding Metrics**: Measures the model's ability to identify image regions related to the question and its answer.

Text-to-Image Generation

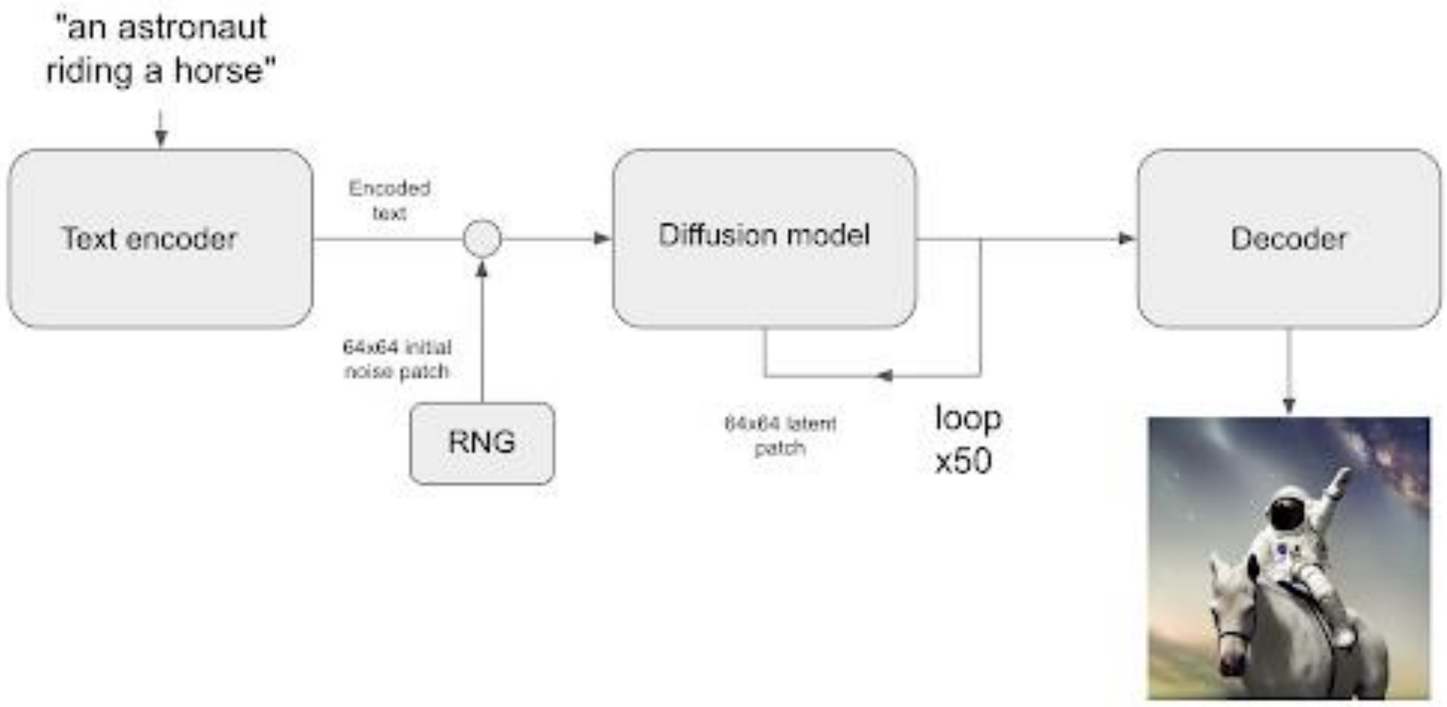
Text-to-image generation is a task within vision-language models (VLMs) that involves creating visual content from textual descriptions.

Text-to-Image Generation



Source: https://abdulkaderhelwan.medium.com/text-to-image-generation-model-with-cnn-ca904427d1e7 (Retrieved February 1, 2025.)

Text-to-Image Generation



Source: https://abdulkaderhelwan.medium.com/text-to-image-generation-model-with-cnn-ca904427d1e7 (Retrieved February 1, 2025.)

Text-to-Image Generation- Diffusion Models

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Text-to-Image Generation – Example Models

- OpenAI's DALL·E 3

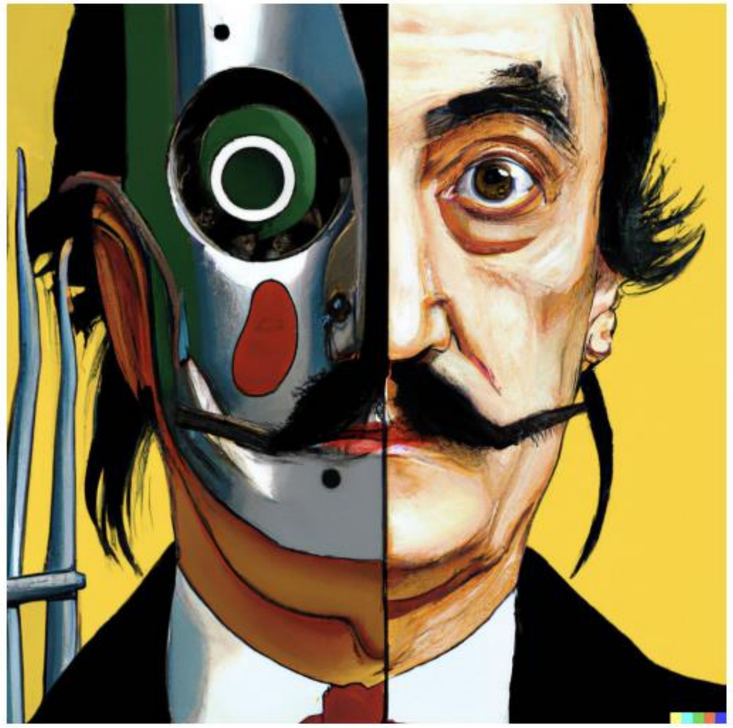
- Stable Diffusion 3

- DeepSeek's Janus Pro

- Midjourney

- Adobe Firefly

Text-to-Image Generation- unCLIP



vibrant portrait painting of Salvador Dalí with a robotic half face

a shiba inu wearing a beret and black turtleneck

a close up of a handpalm with leaves growing from it

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022). Hierarchical text-conditional image generation with CLIP latents. Proceedings of the International Conference on Machine Learning (ICML), 159, 12345-12356. https://doi.org/10.48550/arXiv.2204.06125

Text-to-Image Generation- unCLIP

an espresso machine that makes coffee from human souls, artstation

panda mad scientist mixing sparkling chemicals, artstation

a corgi's head depicted as an explosion of a nebula

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022). Hierarchical text-conditional image generation with CLIP latents. Proceedings of the International Conference on Machine Learning (ICML), 159, 12345-12356. https://doi.org/10.48550/arXiv.2204.06125

Text-to-Image Generation- unCLIP

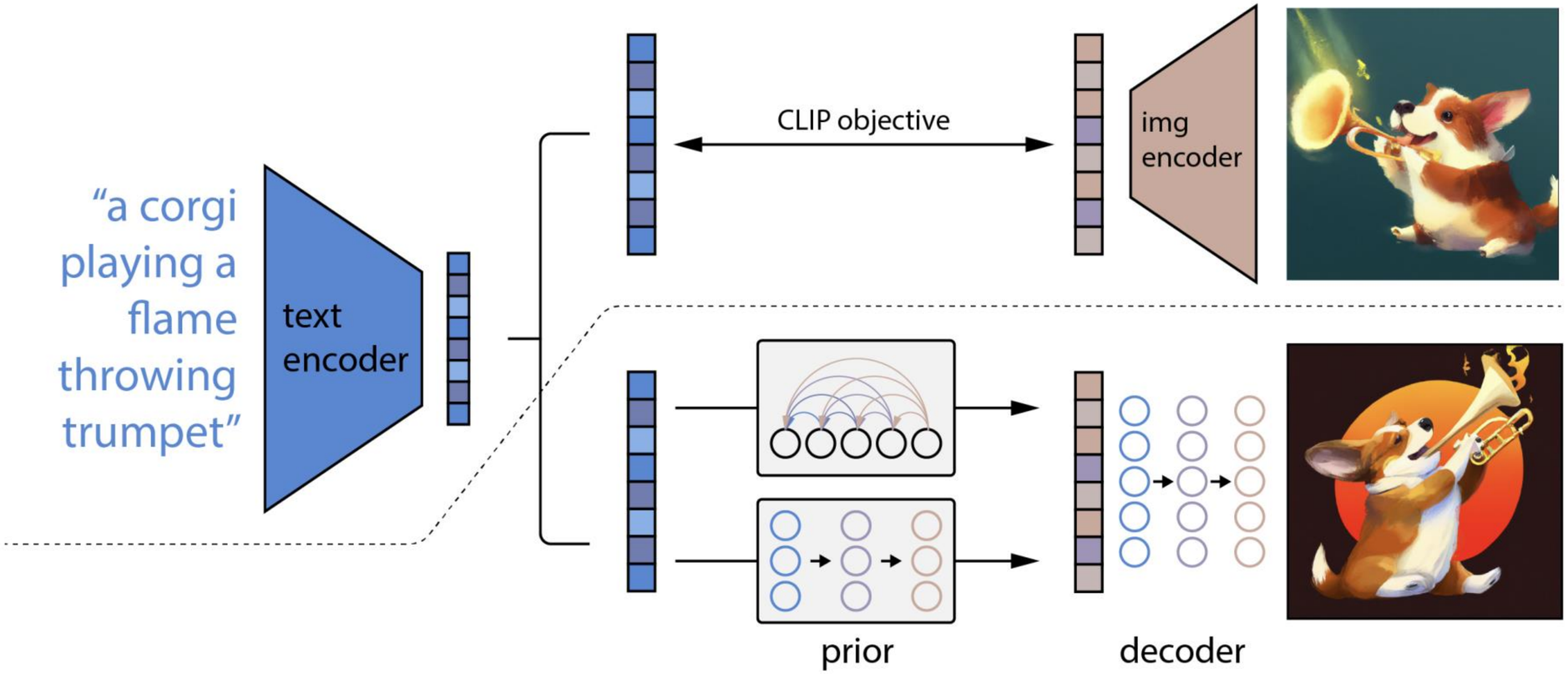
a dolphin in an astronaut suit on saturn, artstation

a propaganda poster depicting a cat dressed as french emperor napoleon holding a piece of cheese

a teddy bear on a skateboard in times square

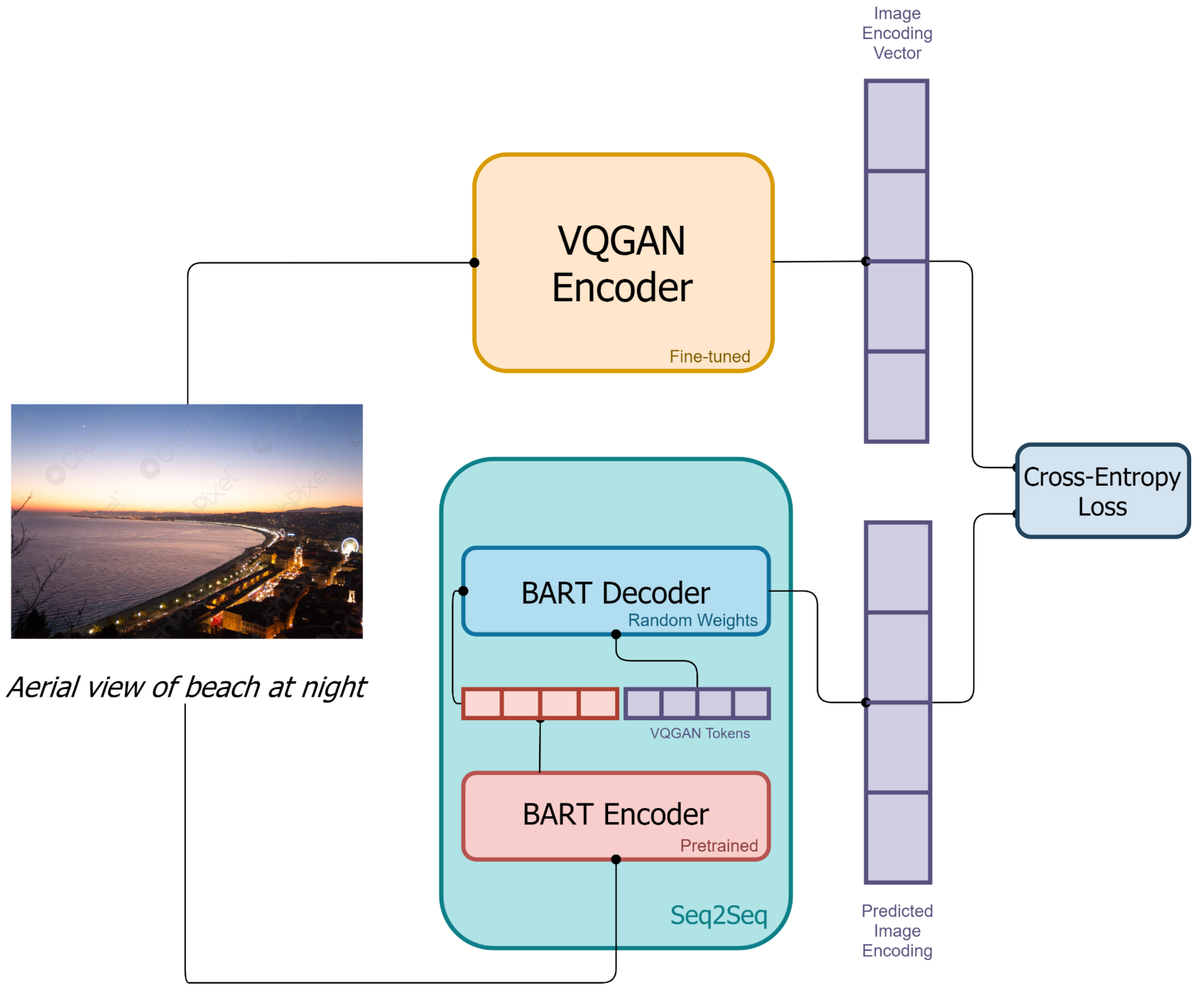
Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022). Hierarchical text-conditional image generation with CLIP latents. Proceedings of the International Conference on Machine Learning (ICML), 159, 12345-12356. https://doi.org/10.48550/arXiv.2204.06125

Text-to-Image Generation- unCLIP



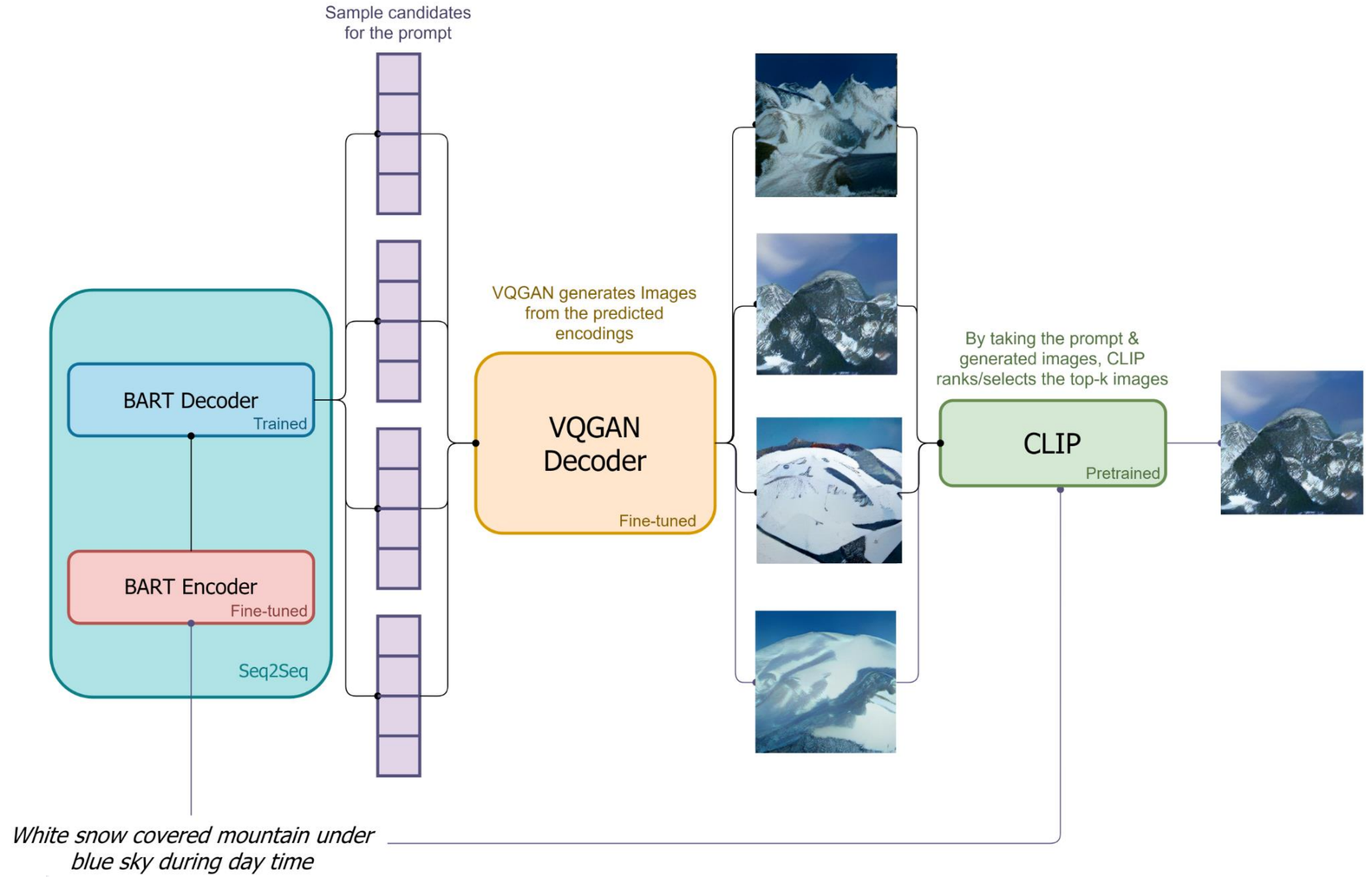
Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022). Hierarchical text-conditional image generation with CLIP latents. Proceedings of the International Conference on Machine Learning (ICML), 159, 12345-12356. https://doi.org/10.48550/arXiv.2204.06125

Text-to-Image Generation- DALL·E mini



https://wandb.ai/dalle-mini/dalle-mini/reports/OpenAI-s-DALL-E-Mini-Explained-with-Demo--Vmlldzo4NjIxODA#what-we'll-be-covering

Text-to-Image Generation- DALL·E mini



https://wandb.ai/dalle-mini/dalle-mini/reports/OpenAI-s-DALL-E-Mini-Explained-with-Demo--Vmlldzo4NjIxODA#what-we'll-be-covering

Text-to-Image Generation – Example Datasets

While image captioning datasets provide a good foundation, text-to-image generation may require datasets with highly detailed and varied captions to fully capture the intricacies of visual scenes.

Text-to-Image Generation – Example Datasets

MS COCO (Microsoft Common Objects in Context): A large-scale dataset containing over 328,000 images, each annotated with multiple captions. It is widely used for tasks like object detection, segmentation, and captioning.

CUB-200-2011 (Caltech-UCSD Birds-200-2011): This dataset includes 11,788 images of birds from 200 subcategories, each with detailed annotations. It's commonly used for fine-grained visual categorization tasks.

Oxford 102 Flower: Contains 8,189 images of flowers from 102 categories, each with 10 images. It's used for fine-grained image classification and captioning tasks.

Text-to-Image Generation – Example Datasets

Conceptual Captions: A large-scale dataset with over 3 million images, each paired with a caption. It's designed for training models on diverse and complex image-caption pairs.

Fashion-Gen: A dataset containing 293,000 images of fashion items, each annotated with detailed attributes and textual descriptions. It's used for fashion-related image generation tasks.

PixelProse: A comprehensive dataset of over 16 million synthetically generated captions, leveraging cutting-edge vision-language models for detailed and accurate descriptions. It aims to provide high-quality image-text pairs for training large vision-language models.

Text-to-Image Generation – Example Datasets

TextCaps: A dataset designed for image captioning with reading comprehension, containing 145,000 captions for 28,000 images. It challenges models to comprehend text in the context of an image.

IIITD-20K: A dataset comprising 20,000 unique identities captured in the wild, each with a minimum of 26 words for a description. It provides dense captions for text-to-image retrieval tasks.

Text-to-Image Generation – Metrics

Inception Score (IS): Measures the quality and diversity of generated images by evaluating the output of an Inception model. Higher scores indicate better quality and diversity.

Fréchet Inception Distance (FID): Assesses the distance between feature distributions of real and generated images, with lower scores indicating closer alignment.

Text-to-Image Faithfulness Evaluation with Question Answering (TIFA): Evaluates how well generated images align with textual descriptions by using visual question answering. It provides a reference-free, fine-grained assessment of image faithfulness.

Text-to-Image Generation – Metrics

Holistic Evaluation of Text-To-Image Models (HEIM): A benchmark that identifies 12 different aspects important in real-world model evaluation, offering a comprehensive assessment of text-to-image generation models.

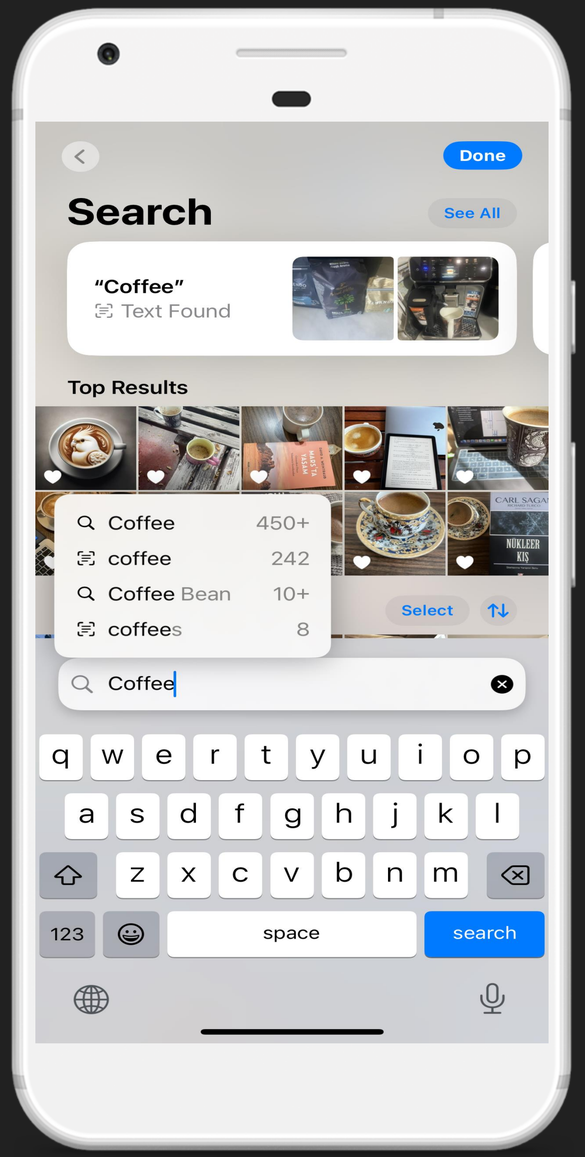
TISE (Text-to-Image Synthesis Evaluation): A framework that evaluates image realism, text relevance, and object accuracy, among other factors, to provide a comprehensive assessment of text-to-image synthesis models.

Text-to-Image Generation – Metrics

Learned Perceptual Image Patch Similarity (LPIPS): Unlike traditional pixel-wise metrics, LPIPS assesses the perceptual similarity between images by comparing high-level features extracted from deep neural networks. This approach aligns more closely with human visual perception, making it particularly useful for evaluating the quality of generated images.

Image Retrieval

Image retrieval in the context of vision-language models (VLMs) refers to the task where a model is trained to find relevant images based on a given textual query. In simpler terms, given a text description or question, the model identifies and retrieves the most relevant images from a database or set of images.



Experts

Image Retrieval – Example Models

DELG (DEep Local and Global features): DELG unifies deep local and global features into a single model, enabling accurate retrieval with efficient feature extraction. It combines generalized mean pooling for global features and attentive selection for local features, achieving state-of-the-art performance on various benchmarks.

STIR (Siamese Transformer for Image Retrieval Postprocessing): STIR introduces a novel approach for image retrieval postprocessing by directly comparing a query image and a retrieved candidate on a pixel level using an attention mechanism. This method defines a new state of the art on standard image retrieval datasets.

Image Retrieval – Example Models

SuperGlobal: SuperGlobal employs global features exclusively for both retrieval and reranking stages, improving efficiency without sacrificing accuracy. It introduces enhancements to the retrieval system, focusing on global feature extraction and reranking processes.

CFCD (Coarse-to-Fine Compact Discriminative representation): CFCD is a framework that learns compact discriminative representations for single-stage image retrieval. It employs an adaptive softmax-based loss and a mechanism to attentively select prominent local descriptors, achieving state-of-the-art performance on benchmarks like Revisited Oxford and Revisited Paris.

Image Retrieval – Example Models

AMES (Adaptive Metric Learning for Efficient Search): AMES is designed to enhance image retrieval efficiency by learning adaptive metrics that improve search performance. It has demonstrated strong results on benchmarks like ROxford and RParis.

SuperGlobal: SuperGlobal focuses on global feature representations to improve image retrieval accuracy. It excels particularly on challenging datasets, achieving notable performance on ROxford Hard and RParis Hard benchmarks.

ViT-L-14 (LAION400M): This model utilizes the Vision Transformer (ViT) architecture, specifically the large variant with a 14x14 patch size. Trained on the LAION400M dataset, it is effective in compositional representation evaluations.

Image Retrieval – Example Models

SPN4CIR: SPN4CIR is tailored for compositional image retrieval tasks, focusing on understanding and retrieving images based on complex queries. It has shown strong results on benchmarks like Fashion IQ and CIRR.

X-VLM (base): X-VLM is a vision-language model that integrates visual and textual data for various tasks, including image retrieval. The base variant has been evaluated on the Flickr30K 1K test set, demonstrating effective cross-modal retrieval capabilities.

Unicom+ViT-L@336px: This model combines the Unicom framework with a Vision Transformer at a resolution of 336x336 pixels. It has been effective in image retrieval tasks on datasets like SOP and iNaturalist.

Image Retrieval – Example Models

CN-CLIP (ViT-H/14): CN-CLIP is a variant of the CLIP model, utilizing a Vision Transformer with a 14x14 patch size. It has been applied to benchmarks such as MUGE Retrieval and COCO-CN, showcasing strong cross-modal retrieval performance.

CGD (MG/SG): CGD focuses on learning compact and discriminative representations for image retrieval. It has achieved notable results on datasets like CUB-200-2011 and CARS196.

BLIP-2 ViT-G: BLIP-2 integrates vision and language processing, utilizing a Vision Transformer for image understanding. It has been evaluated on datasets such as Flickr30k and MS COCO, demonstrating strong performance in image-text retrieval tasks.

Image Retrieval – Example Models

DINOv2 distilled (ViT-L/14 frozen): DINOv2 employs self-supervised learning with a Vision Transformer architecture, where the model is distilled and certain layers are frozen during training. It has been effective on the AmsterTime benchmark, showcasing its capability in temporal image retrieval tasks.

Image Retrieval – Example Datasets

Oxford Buildings Dataset (Oxford5k): Contains 5,062 images collected from Flickr, representing 11 different Oxford landmarks. Designed for evaluating building retrieval systems.

Paris Buildings Dataset (Paris6k): Comprises 6,412 images of Paris landmarks, also sourced from Flickr. Similar to Oxford5k, it is used for benchmarking building retrieval performance.

Revisited Oxford and Paris Datasets (ROxford and RParis): These are updated versions of the original Oxford and Paris datasets, featuring corrected annotations and additional challenging sets. Designed to provide a more rigorous evaluation framework for image retrieval algorithms.

Image Retrieval – Example Datasets

Google Landmarks Dataset v2 (GLDv2): A large-scale dataset with over 5 million images and 200,000 distinct landmarks. Serves as a benchmark for large-scale, fine-grained instance recognition and image retrieval.

Flickr30k: Contains 31,783 images sourced from Flickr, each paired with five textual descriptions. Used for evaluating image-text retrieval and captioning models.

MS COCO (Microsoft Common Objects in Context): Features over 330,000 images, with more than 200,000 labeled and 80 object categories. Widely used for object detection, segmentation, and image captioning tasks.

Image Retrieval – Example Datasets

Stanford Online Products (SOP): Contains 120,053 product images across 22,634 classes. Designed for evaluating metric learning algorithms in image retrieval.

iNaturalist: A large-scale dataset with over 859,000 images spanning more than 5,000 species. Used for fine-grained image classification and retrieval in the context of species recognition.

FashionIQ: Comprises images of fashion products along with textual descriptions. A benchmark for evaluating image retrieval models that incorporate both visual and textual queries.

Image Retrieval – Example Datasets

CIRR (Compose Image Retrieval on Real-life images): Contains over 36,000 pairs of images with associated textual descriptions. Designed to evaluate models on composed image retrieval tasks, where both image and text are used as queries.

CUB-200-2011: A dataset with 11,788 images of 200 bird species, used for fine-grained image retrieval.

CARS196: Contains 16,185 images of 196 car models, used for fine-grained image retrieval tasks.

Image Retrieval – Metrics

Precision@K (P@K): The proportion of relevant images among the top K retrieved images. Measures the accuracy of the retrieval system in returning relevant images within the top K results.

Recall@K (R@K): The proportion of relevant images retrieved out of the total number of relevant images available in the dataset. Assesses the system's ability to retrieve all relevant images, considering the top K results.

Mean Average Precision (mAP): The mean of the average precision scores for all queries. Average precision calculates the precision at each position in the ranked list of results where a relevant image is found, then averages these values. Provides a single-figure measure of quality across recall levels, balancing precision and recall.

Image Retrieval – Metrics

Normalized Discounted Cumulative Gain (NDCG): A metric that accounts for the position of relevant images in the retrieval list, applying a logarithmic discount factor to lower-ranked results. Evaluates the usefulness, or gain, of a document based on its position in the result list, emphasizing the importance of retrieving relevant images early.

F1-Score: The harmonic mean of precision and recall. Balances the trade-off between precision and recall, providing a single metric that accounts for both false positives and false negatives.

Mean Reciprocal Rank (MRR): The average of the reciprocal ranks of the first relevant image retrieved for each query. Focuses on the rank position of the first relevant result, rewarding systems that retrieve relevant images at higher positions.

Image Retrieval – Metrics

Area Under the Precision-Recall Curve (AUC-PR): The area under the curve plotted with precision on the y-axis and recall on the x-axis. Summarizes the trade-off between precision and recall across different threshold settings, providing insight into the system's performance across various recall levels.

Google Developer Experts

Image Editing with Textual Guidance

Source: https://www.youtube.com/watch?v=JL46D_CbxGs&ab_channel=StealTheseThoughts%21 (Retrieved February 1, 2025.)

Source: Zhang, Z., Han, L., Ghosh, A., Metaxas, D. N., & Ren, J. (2023). Sine: Single image editing with text-to-image diffusion models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 6027-6037).

Image Editing with Textual Guidance – Example Models

Forgedit: Built upon the Stable Diffusion framework, Forgedit introduces a vision-language joint optimization approach. It employs a novel vector projection mechanism in the text embedding space of diffusion models, allowing for separate control of identity similarity and editing strength. Additionally, it utilizes a "forgetting" mechanism to address overfitting issues during fine-tuning. Forgedit achieves state-of-the-art results on the TEdBench benchmark, surpassing previous methods like Imagic with Imagen in both CLIP and LPIPS scores.

Image Editing with Textual Guidance – Example Models

InstructEdit: This framework focuses on fine-grained editing based on user instructions. It comprises three components: a language processor that parses user instructions using large language models, a segmenter that generates high-quality masks based on segmentation prompts, and an image editor that utilizes these masks and captions to perform the edits. InstructEdit outperforms previous editing methods in applications involving complex or multiple objects, improving mask quality and, consequently, the quality of edited images.

Image Editing with Textual Guidance – Example Models

Locate and Forget (LaF): LaF introduces a method that effectively locates potential target concepts in an image for modification by comparing the syntactic structures of the target prompt and scene descriptions in the input image. The approach aims to "forget" the existence clues of these concepts in the generated image. Compared to baseline methods, LaF demonstrates superior performance in text-guided image editing tasks, both qualitatively and quantitatively.

Image Editing with Textual Guidance – Example Models

SDEdit: SDEdit is an image synthesis and editing framework based on stochastic differential equations (SDEs) or diffusion models. It allows for stroke-based image synthesis, stroke-based image editing, and image compositing without task-specific optimization.SDEdit can be directly integrated with pre-trained score-based or diffusion models and has been applied to text-guided image editing with large-scale text-to-image models.

Image Editing with Textual Guidance – Example Models

Blended Diffusion: This method offers a highly intuitive interface for image editing using natural language. It enables users to specify desired modifications through text prompts, facilitating text-driven editing of natural images. Blended Diffusion has been recognized for its effectiveness in applying text-driven edits to natural images.

Image Editing with Textual Guidance – Example Models

Grounding DINO + Stable Diffusion

Grounding DINO + GLIGEN

https://github.com/IDEA-Research/GroundingDINO

Image Editing with Textual Guidance – Example Datasets

MagicBrush: MagicBrush is a large-scale, manually annotated dataset designed for instruction-guided real image editing. It comprises over 10,000 triplets, each consisting of a source image, an editing instruction, and the corresponding target image. The dataset covers diverse scenarios, including single-turn and multi-turn edits, as well as cases with and without provided masks. Aims to support the training of large-scale text-guided image editing models by providing high-quality, real-world editing examples.

Image Editing with Textual Guidance – Example Datasets

iEdit: iEdit is a dataset designed for localized text-guided image editing with weak supervision. It focuses on enabling models to perform fine-grained, localized edits based on textual instructions. The dataset includes images paired with edit prompts and target images, facilitating the training of models capable of understanding and applying localized edits. Supports the development of image editing models that can interpret and execute detailed, localized editing instructions.

Image Editing with Textual Guidance – Example Datasets

EditVal: EditVal is a benchmark dataset developed for evaluating diffusion-based text-guided image editing methods. It provides a standardized evaluation protocol to compare different editing approaches. The dataset consists of curated images along with a set of editable attributes for each image, drawn from 13 possible edit types. It also includes an automated evaluation pipeline that uses pre-trained vision-language models to assess the fidelity of generated images for each edit type. Aims to provide a comprehensive benchmark for quantitatively evaluating text-guided image editing methods across various fine-grained edits.

Image Editing with Textual Guidance – Metrics

CLIP Similarity (Text-to-Image): Measures the alignment between the text instruction and the edited image using the CLIP model's vision-language embeddings. High similarity indicates better adherence to the textual guidance.

CLIP Similarity (Image-to-Image): Assesses how similar the edited image is to the input image, ensuring changes are localized and minimal if required.

Fréchet Inception Distance (FID): Measures the distribution difference between the generated images and real images. Lower FID scores indicate more realistic and high-quality outputs.

Image Editing with Textual Guidance – Metrics

Perceptual Image Quality Evaluator (PIQE): Focuses on perceptual quality without requiring reference images.

Intersection over Union (IoU): Measures the overlap between the edited region and the region indicated by the textual instruction (if a mask is provided).

Edit Localization Accuracy: Evaluates whether the edits are correctly applied to the specified areas.

Structural Similarity Index (SSIM): Quantifies the structural similarity between the original and edited images. Ensures that changes are minimal and localized when required.

Image Editing with Textual Guidance – Metrics

LPIPS (Learned Perceptual Image Patch Similarity): A learned metric that evaluates the perceptual similarity between the input and edited images.

Instruction Compliance Score: Measures the model's ability to execute the specified textual edits correctly.

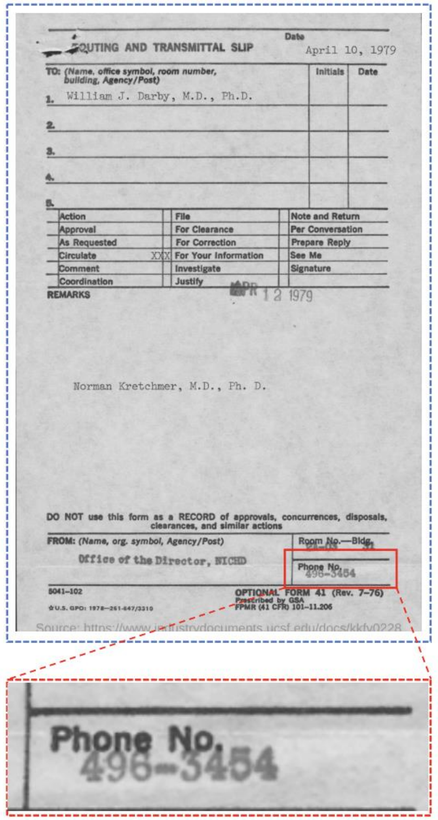
Mask Accuracy: Evaluates the quality of generated masks when text guidance specifies regions for editing.

Image Editing with Textual Guidance – Metrics

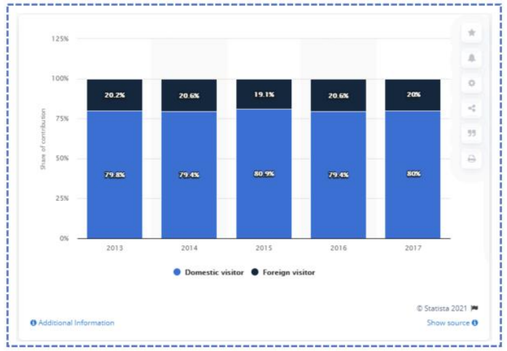
Human Preference Score: Collected through surveys, where participants rate how well the edits match the instructions and their realism.

Human Adherence Rating: Users rank the fidelity of edits concerning the textual guidance.

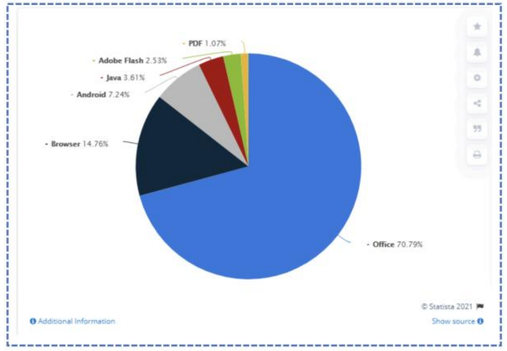
Document Understanding



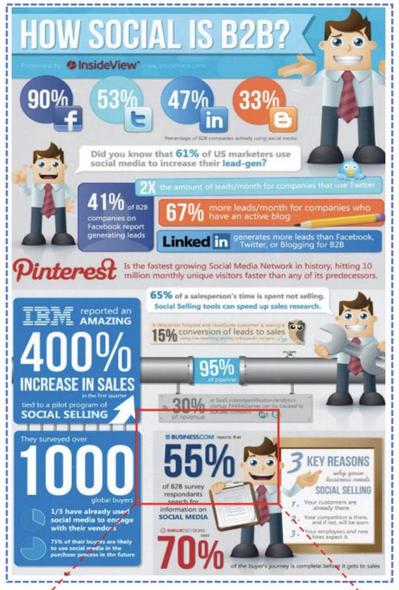
Question: What is the "phone number" given on the slip?
Qwen-VL-Chat†: 5041-102
Qwen-VL-Chat††: 496-3454



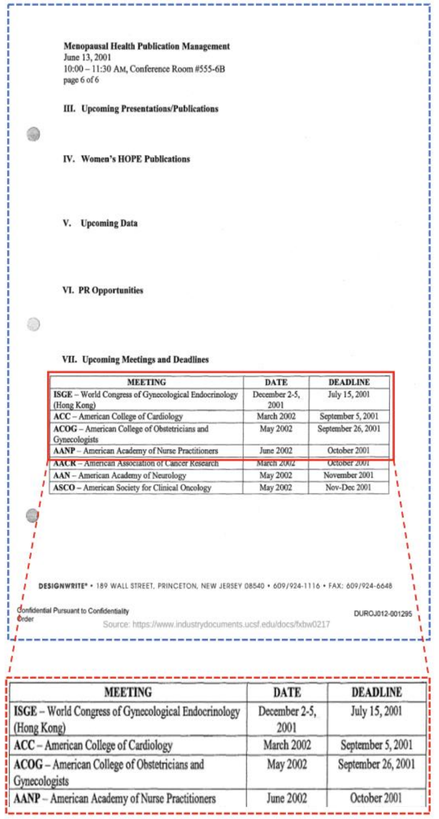
Question : Can you identify what is the percentage value of foreign visitor in 2017?
Qwen-VL-Chat†: 80
Qwen-VL-Chat††: 20



Question : Find which category is shown in yellow color?
Qwen-VL-Chat†: Adobe Flash 2.53
Qwen-VL-Chat††: PDF



Question : What percentage of B2B survey respondents search for information on social media?
Qwen-VL-Chat†: 41%
Qwen-VL-Chat††: 55%



MEETING	DATE	DEADLINE
ISGE – World Congress of Gynecological Endocrinology (Hong Kong)	December 2-5, 2001	July 15, 2001
ACC – American College of Cardiology	March 2002	September 5, 2001
ACOG – American College of Obstetricians and Gynecologists	May 2002	September 26, 2001
AANP – American Academy of Nurse Practitioners	June 2002	October 2001

Question : What is the deadline given for AANP?
Qwen-VL-Chat†: june 2002
Qwen-VL-Chat††: October 2001

Source: Li, X., Wu, Y., Jiang, X., Guo, Z., Gong, M., Cao, H., ... & Sun, X. (2024). Enhancing visual document understanding with contrastive learning in large visual-language models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 15546-15555).

Document Understanding – Example Models

Bi-VLDoc (Bidirectional Vision-Language Modeling for Visually-Rich Document Understanding): Bi-VLDoc employs a bidirectional vision-language supervision strategy and a vision-language hybrid-attention mechanism to fully explore and utilize the interactions between visual and textual modalities. Enhances cross-modal document representations with richer semantics. Demonstrates significant improvements in tasks like form understanding, receipt information extraction, and document classification.

DUBLIN (Document Understanding By Language-Image Network): DUBLIN is pretrained on web pages using novel objectives that leverage both spatial and semantic information in document images. Excels in tasks such as Web-Based Structural Reading Comprehension, Document Visual Question Answering, Key Information Extraction, Diagram Understanding, and Table Question Answering. Achieves state-of-the-art performance on multiple benchmarks, including WebSRC, DocVQA, and InfographicsVQA.

Document Understanding – Example Models

UDOP (Unifying Vision, Text, and Layout for Universal Document Processing): UDOP is a foundation Document AI model that unifies text, image, and layout modalities with varied task formats, including document understanding and generation. Leverages the spatial correlation between textual content and document images to model multiple modalities with a uniform representation. Sets state-of-the-art performance on various Document AI tasks across diverse data domains like finance reports, academic papers, and websites.

Document Understanding – Example Datasets

InstructDoc: A large-scale collection comprising 30 publicly available Visual Document Understanding (VDU) datasets, unified under diverse instructions. Covers 12 distinct tasks, including question answering and information extraction. Encompasses a wide range of document types and formats. Facilitates zero-shot generalization in VDU tasks by providing a diverse set of instructions and document types.

BigDocs-7.5M: A large-scale, open multimodal dataset curated to advance visual document understanding across diverse document types and tasks. Contains 7.5 million document images with corresponding annotations. Designed to support a variety of VLDU tasks, including document classification, information extraction, and visual question answering. Aims to provide a comprehensive dataset to train and evaluate models in understanding complex document structures and content.

Document Understanding – Example Datasets

VisualMRC: A dataset designed for machine reading comprehension on document images. Contains over 30,000 question-answer pairs related to more than 10,000 document images from various domains.Focuses on developing natural language understanding and generation abilities in the context of document images. Facilitates research aimed at connecting vision and language understanding by providing a benchmark for evaluating models' comprehension of textual and visual information in documents.

VRDU Benchmark: A benchmark designed for evaluating models on visually-rich document understanding tasks. Includes datasets that represent challenges such as diverse data types, hierarchical entities, complex templates, and varied layouts. Provides both few-shot and conventional experiment settings. Aims to reflect the complexity of real-world documents and assess models' abilities to extract structured data from them.

Document Understanding – Metrics

MMDocBench: MMDocBench is a benchmark designed to evaluate large vision-language models on a range of document understanding tasks. It encompasses 15 main tasks with 4,338 question-answer pairs and 11,353 supporting regions, covering diverse document types such as research papers, receipts, financial reports, Wikipedia tables, charts, and infographics. This benchmark allows for a comprehensive assessment of a model's capabilities across different tasks and document image types.

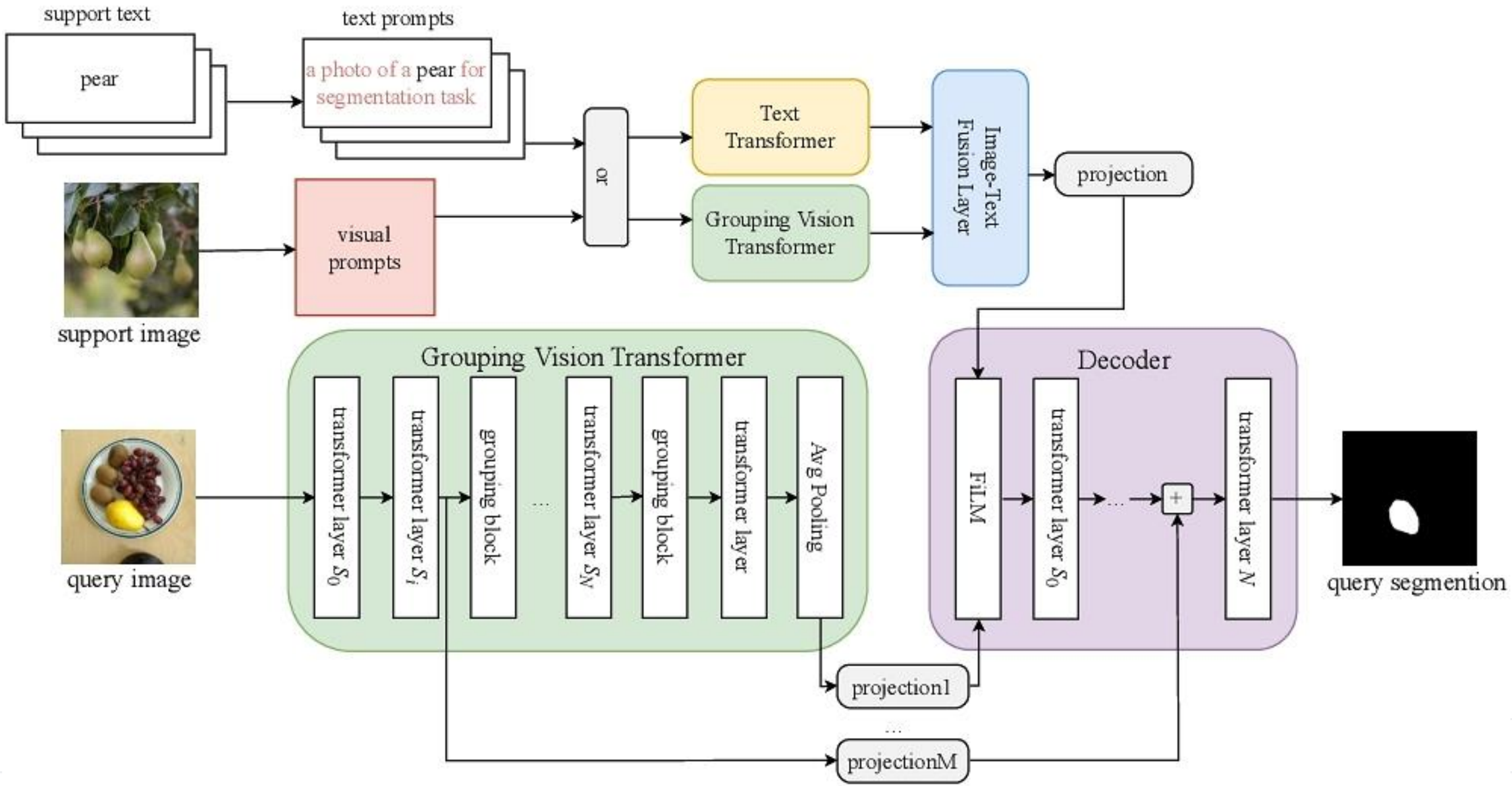
VQAScore: VQAScore is a metric introduced to evaluate vision-language generative models. It aligns closely with human judgments and significantly outperforms previous metrics like CLIPScore on challenging compositional prompts. VQAScore provides a more accurate assessment of a model's ability to generate coherent and contextually relevant outputs in response to visual inputs.

Document Understanding – Metrics

TouchStone: TouchStone is an evaluation method that uses strong language models as judges to comprehensively assess various abilities of large vision-language models. It involves constructing a comprehensive visual dialogue dataset consisting of open-world images and questions, covering five major categories of abilities and 27 subtasks. This method enables a detailed evaluation of a model's performance across a wide range of tasks.

Google Developer Experts

Vision-Language Image Segmentation



Source: Lihu Pan, Yunting Yang, Zhengkui Wang, Rui Zhang, Wen Shan, and Jiashu Li. 2024. Image Segmentation with Vision-Language Models. In Proceedings of the 2023 7th International Conference on Computer Science and Artificial Intelligence (CSAI '23). Association for Computing Machinery, New York, NY, USA, 233–238. https://doi.org/10.1145/3638584.3638624

Vision-Language Image Segmentation – Example Models

ViLaSeg: A SoTA model that generates binary segmentation maps for query images based on free-text descriptions. It leverages vision-language models to understand and segment images according to textual prompts.

LLM-Seg: This model bridges image segmentation and large language models by enabling segmentation based on natural language queries. It utilizes the reasoning capabilities of large language models to interpret and segment images as per textual instructions.

EAVL (Explicitly Align Vision and Language for Referring Image Segmentation): EAVL focuses on aligning vision and language features explicitly during the segmentation process. It generates convolution kernels based on input language queries to guide the segmentation, enhancing localization accuracy.

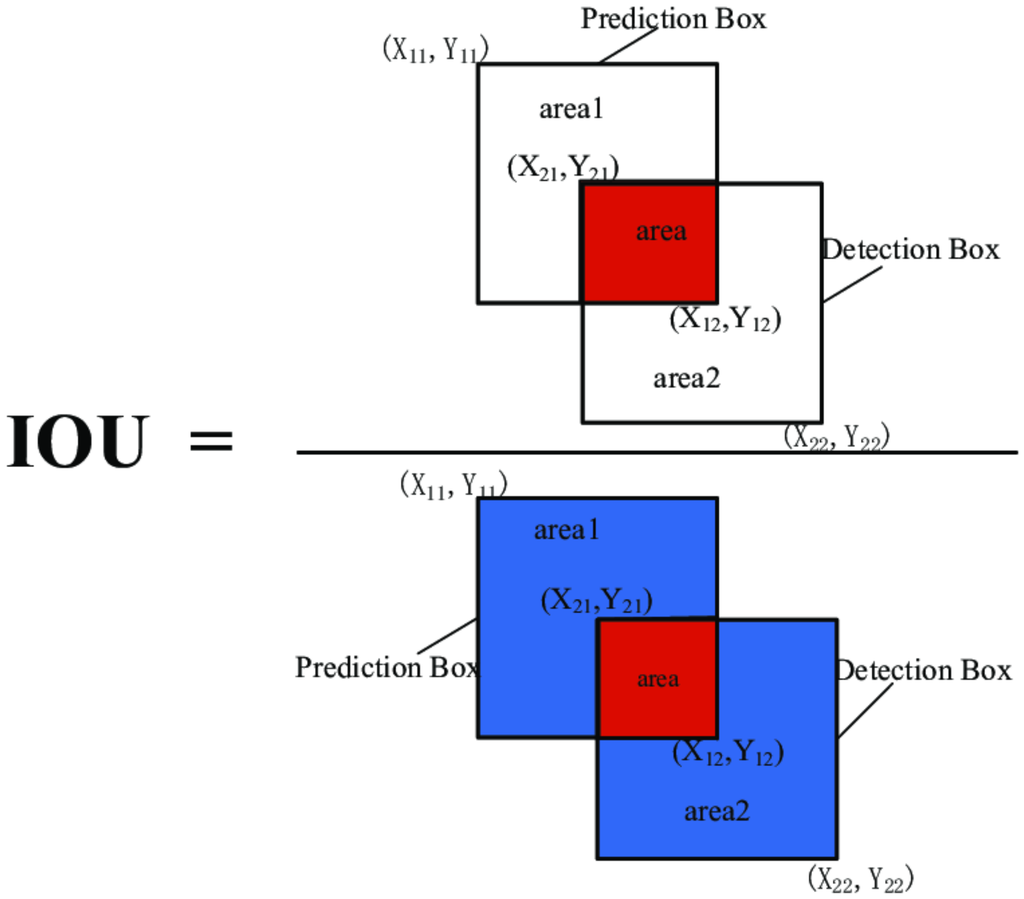
Vision-Language Image Segmentation – Example Models

LAVT (Language-Aware Vision Transformer for Referring Image Segmentation): LAVT integrates language features early in the vision transformer encoder to improve cross-modal alignment. This approach enhances the model's ability to segment objects referred to by natural language expressions.

SemiVL (Semi-Supervised Semantic Segmentation with Vision-Language Guidance): SemiVL combines vision-language models with semi-supervised learning to improve segmentation performance with limited labeled data. It incorporates language guidance to refine semantic decision boundaries.

Vision-Language Image Segmentation – Metrics

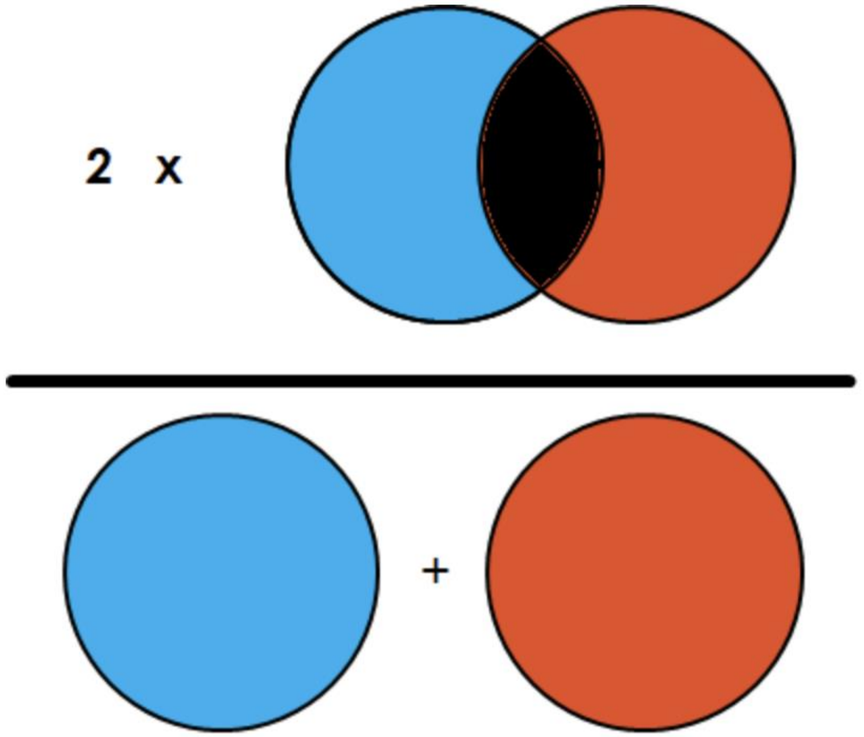
$$IoU = \frac{A \cap B}{A \cup B}$$



Source: https://github.com/Made-Jaya/IoU_Calculator_YOLOv5 (Retrieved February 4, 2025.)

Vision-Language Image Segmentation – Metrics

Dice coefficient



The formula is given by:

$$\frac{2*|X \cap Y|}{|X|+|Y|}$$

Source: https://www.kaggle.com/code/yerramvarun/understanding-dice-coefficient (Retrieved February 4, 2025.)

Google Developer Experts

Vision Language Object Detection

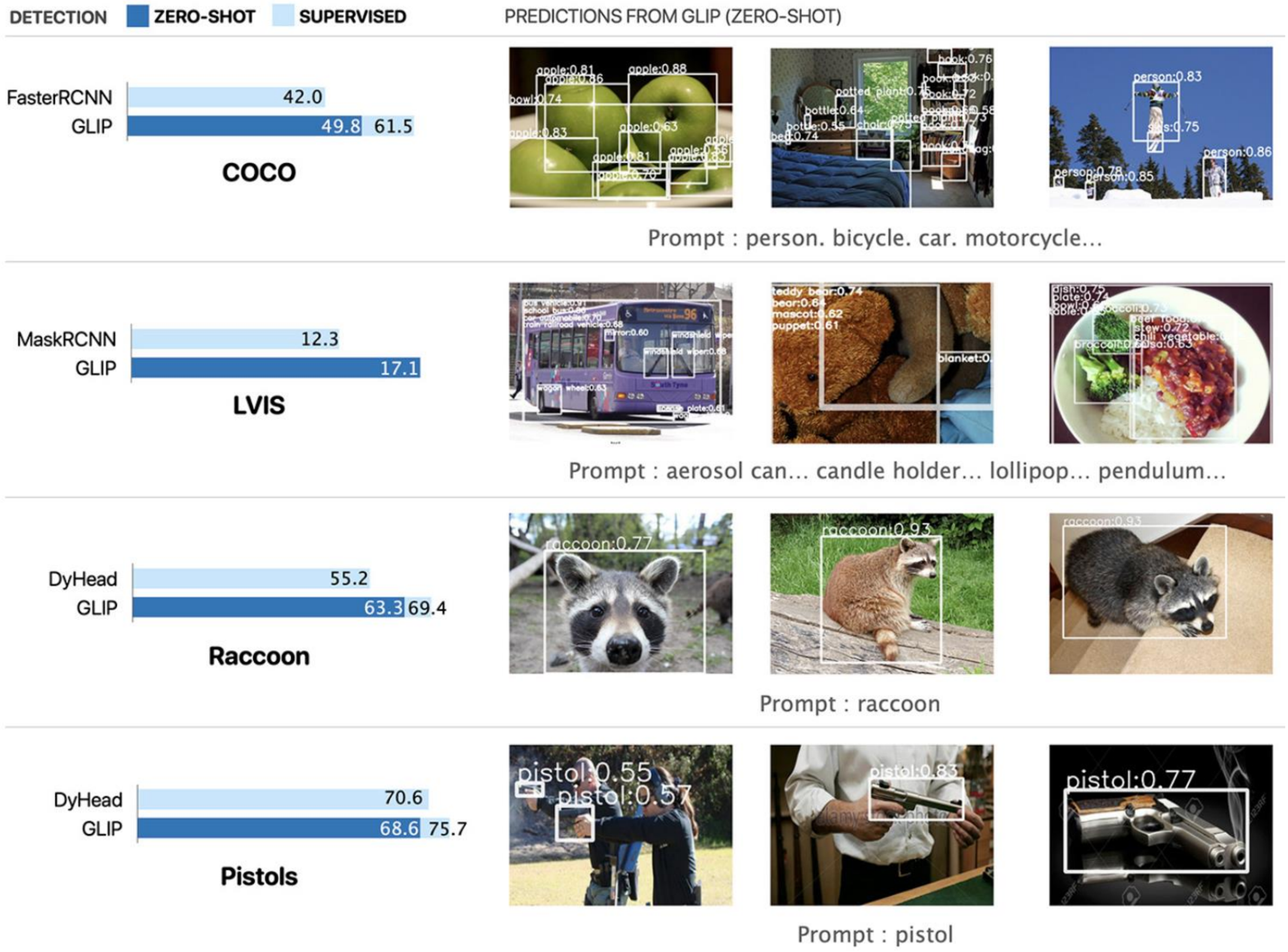
Google Developer Experts

Object Detection Notes

Important Note: These notes are from Andrew Ng's Coursera course.

https://github.com/ayyucedemirbas/VLM_presentation/tree/main/object_detection_notes

Vision Language Object Detection



Source: https://www.microsoft.com/en-us/research/articles/object-detection-in-the-wild-via-grounded-language-image-pre-training/ (Retrieved February 4, 2025.)

Vision Language Object Detection– Example Models

GLIP (Grounded Language-Image Pre-training): GLIP enhances object detection by grounding language in images, enabling zero-shot detection of novel objects. It leverages large-scale vision-language pre-training to understand and detect objects based on textual descriptions.

YOLO-World: YOLO-World extends the YOLO architecture to support open-vocabulary object detection. By integrating vision-language modeling, it can detect a wide range of objects based on textual prompts, achieving high efficiency and accuracy.

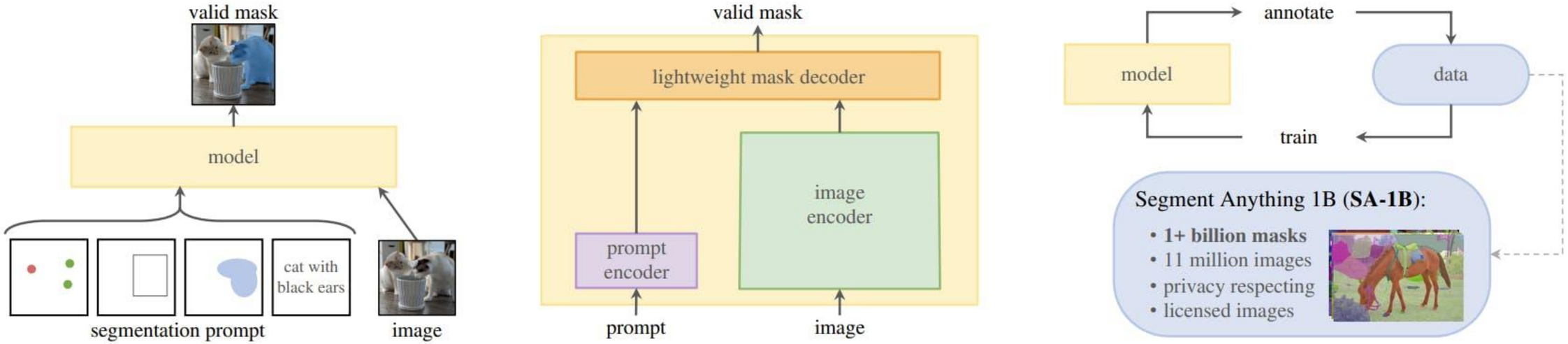
VinVL (Visual features in Vision-Language): VinVL focuses on improving image encoding for vision-language tasks. It enhances object-attribute detection by providing rich visual features, contributing to better performance in object detection tasks.

Vision Language Object Detection– Example Models

OmDet: OmDet is a language-aware object detection architecture that utilizes multi-dataset vision-language pre-training. It unifies object detection as a language-conditioned task, enabling detection across various datasets without manual label taxonomy merging.

DesCo (Description-Conditioned Object Recognition): DesCo learns object recognition using rich language descriptions. It employs large language models to generate detailed descriptions of objects, enhancing the model's ability to detect and recognize objects based on complex textual inputs.

Segment Anything Model (SAM)

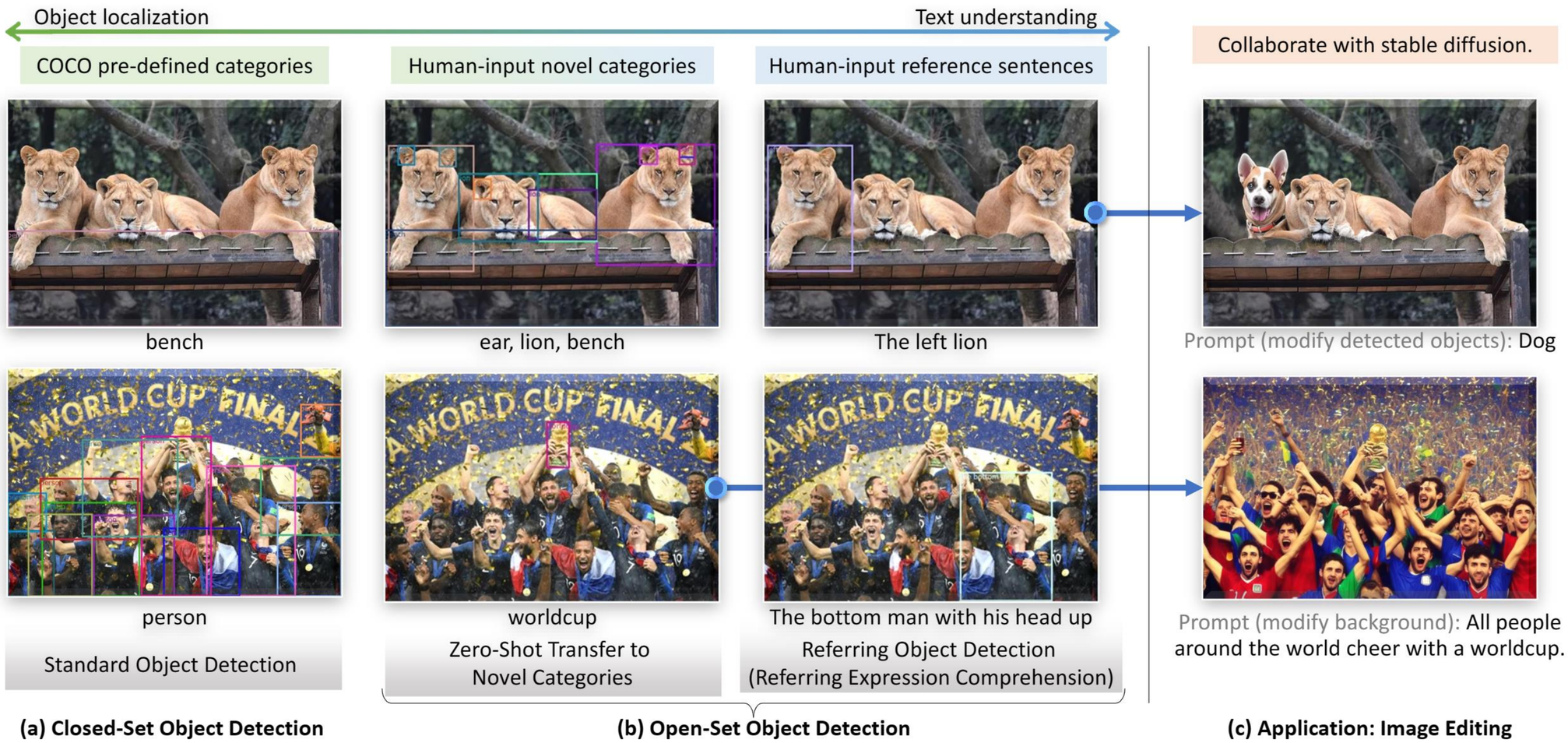


(a) **Task**: promptable segmentation (b) **Model**: Segment Anything Model (**SAM**) (c) **Data**: data engine (top) & dataset (bottom)

Figure 1: We aim to build a foundation model for segmentation by introducing three interconnected components: a prompt-able segmentation *task*, a segmentation *model* (SAM) that powers data annotation and enables zero-shot transfer to a range of tasks via prompt engineering, and a *data* engine for collecting SA-1B, our dataset of over 1 billion masks.

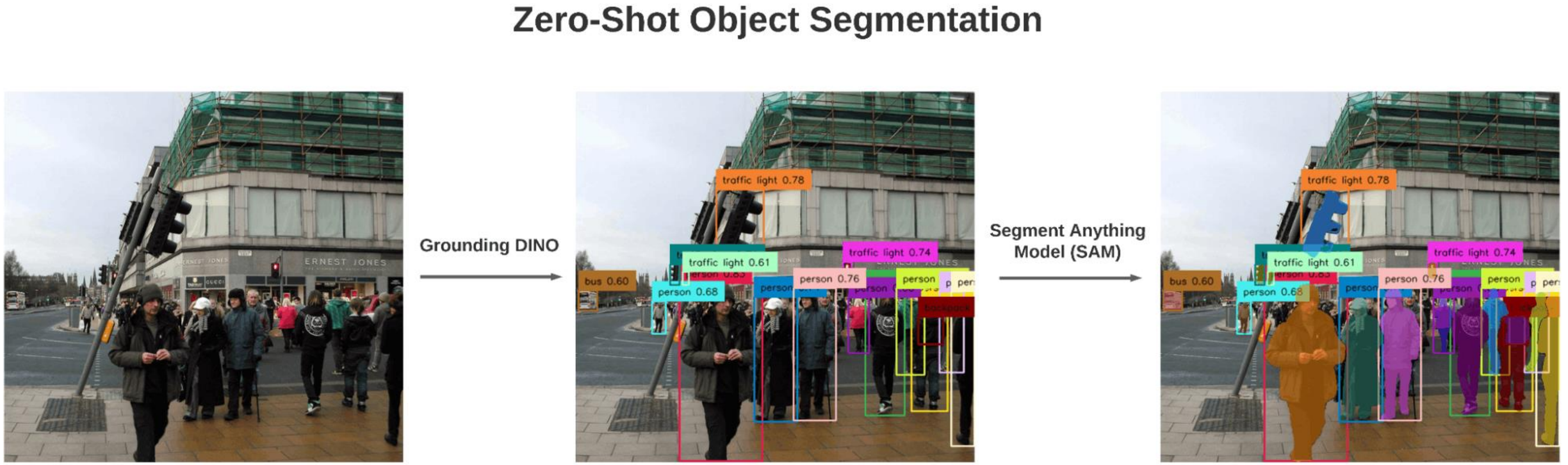
Source: Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., ... & Girshick, R. (2023). Segment anything. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 4015-4026).

Grounding DINO



Source: https://github.com/IDEA-Research/GroundingDINO (Retrieved February 4, 2025.)

Grounding DINO + SAM

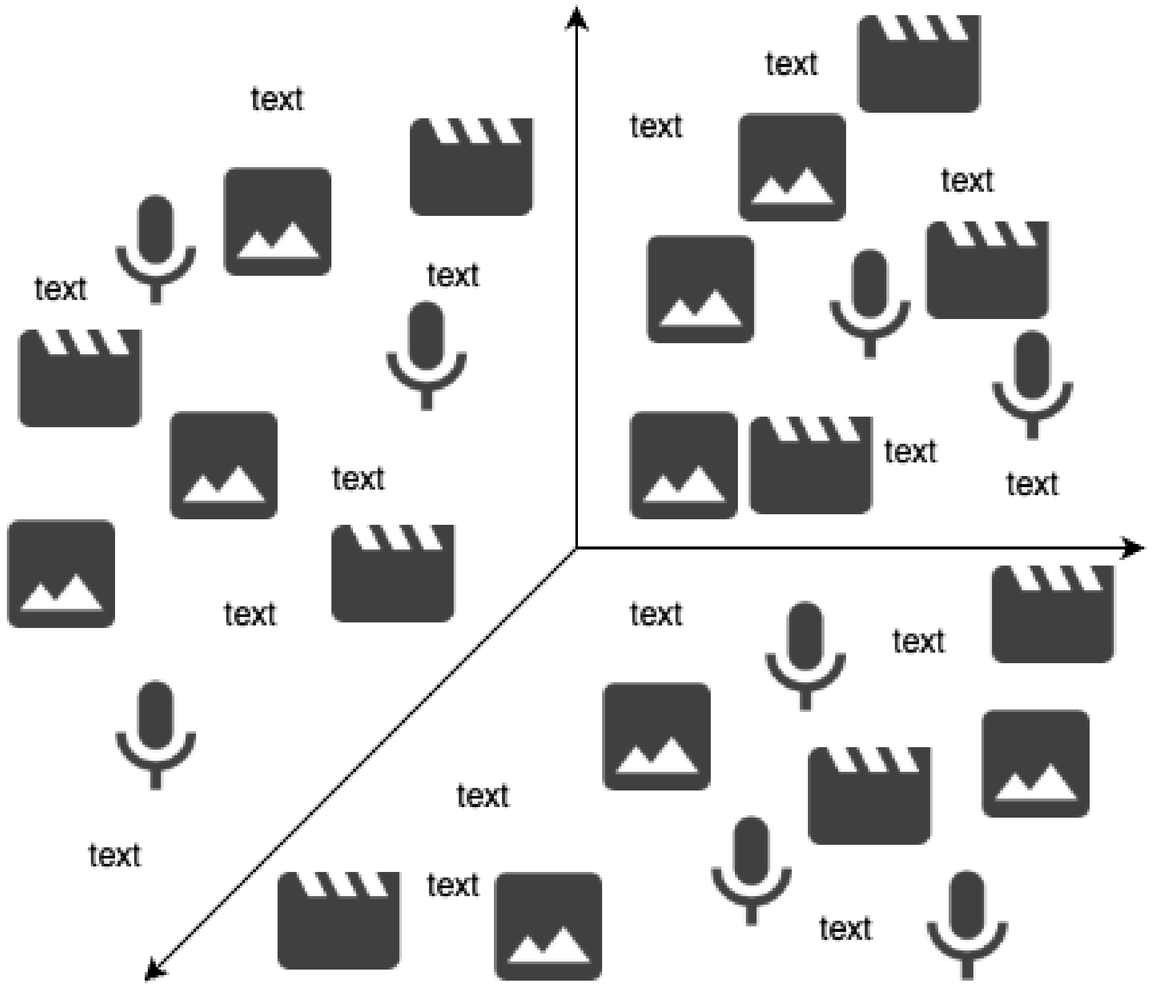


Source: https://encord.com/blog/grounding-dino-sam-vs-mask-rcnn-comparison (Retrieved February 4, 2025.)

Google Developer Experts

Techniques

Multimodal Embedding Spaces



Contrastive Language-Image Pre-training (CLIP)

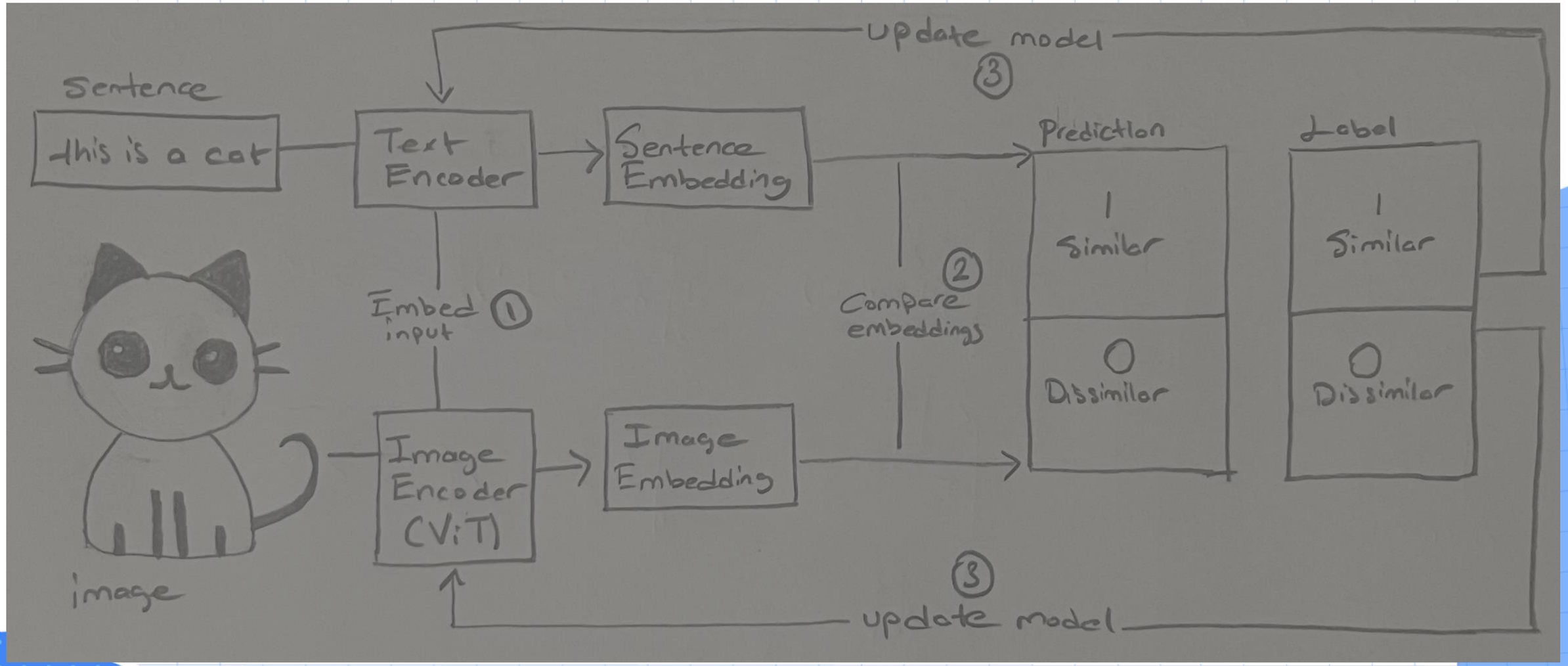
CLIP is a joint image and text embedding model trained using 400m image and text pairs in a self supervised way. It maps both text and images to the same embedding space. The embeddings of images can be compared with the embeddings of text. This comparison capability makes CLIP and similar models usable for tasks such as:

Zero-shot classification: We can compare the embedding of an image with that of the description of its possible classes to find which class is most similar.

Clustering: Cluster both images and a collection of keywords to find which keywords belong to which sets of images.

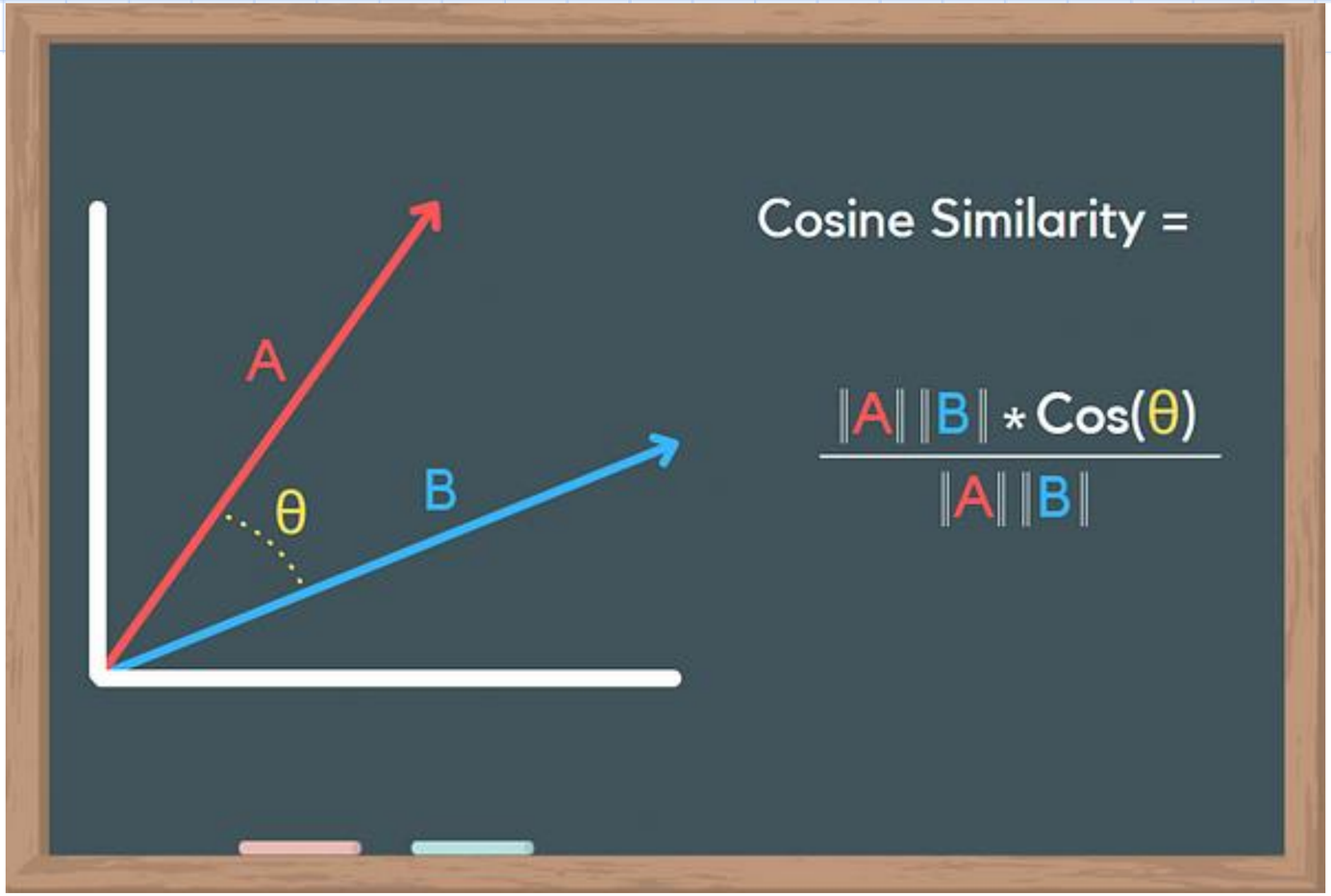
Search: Across billions of texts or images, we can quickly find what relates to an input text or image.

Generation: Use multimodal embeddings to drive the generation of images.

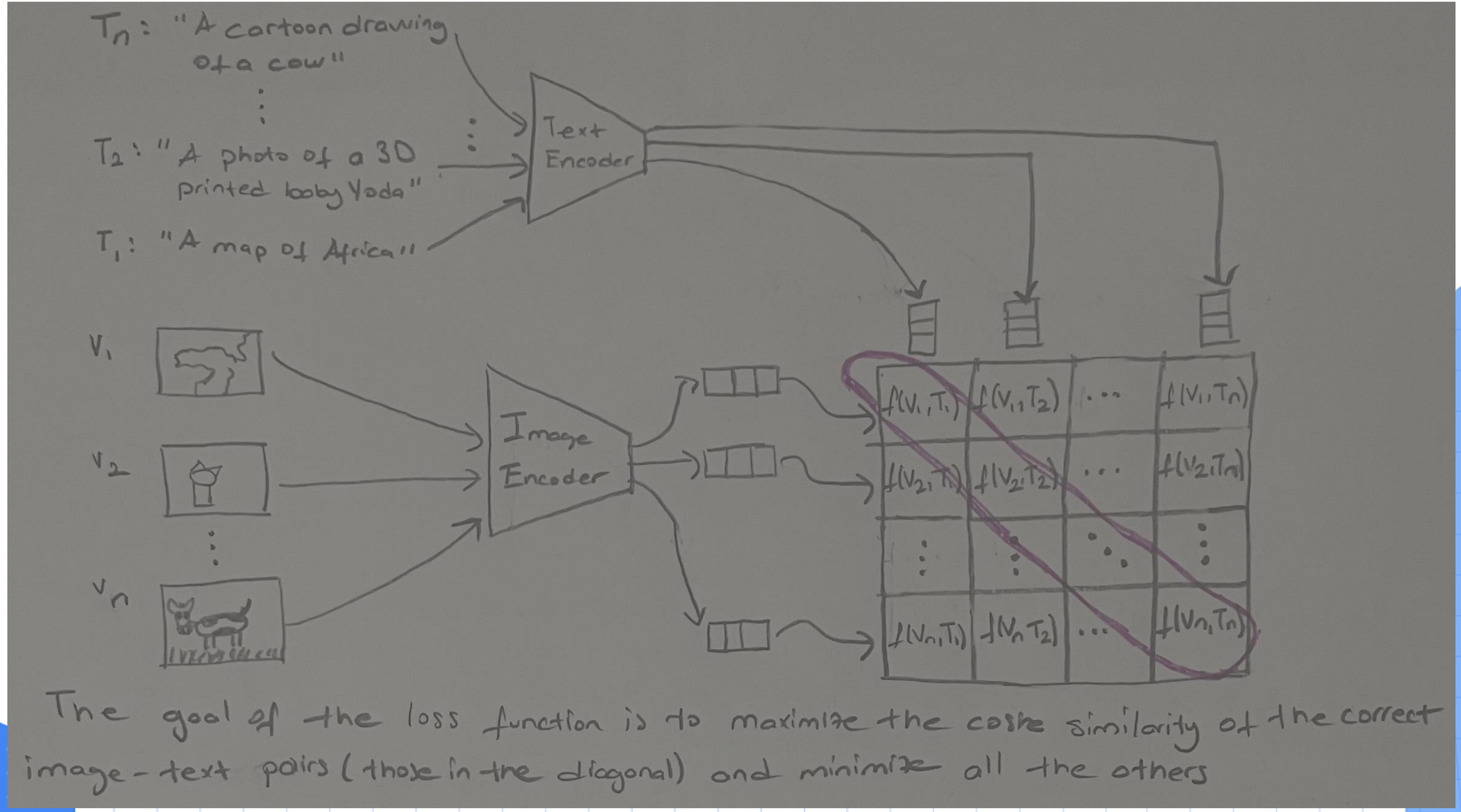

update model
3
Sentence
this is a cat
Text Encoder
Sentence Embedding
Prediction
Label
1
Similar
1
Similar
Embed input
1
Compare embeddings
2
0
Dissimilar
0
Dissimilar
Image Encoder (ViT)
Image Embedding
image
3
update model

Cosine similarity is calculated through the dot product of the embeddings and divided by the product of their lengths.

When we start training, the similarity between the image embedding and text embedding will be low as they are not yet optimized to be within the same vector space. During training, we optimize for the similarity between the embeddings and want to maximize them for similar image/caption pairs and minimize them for dissimilar image/caption pairs.

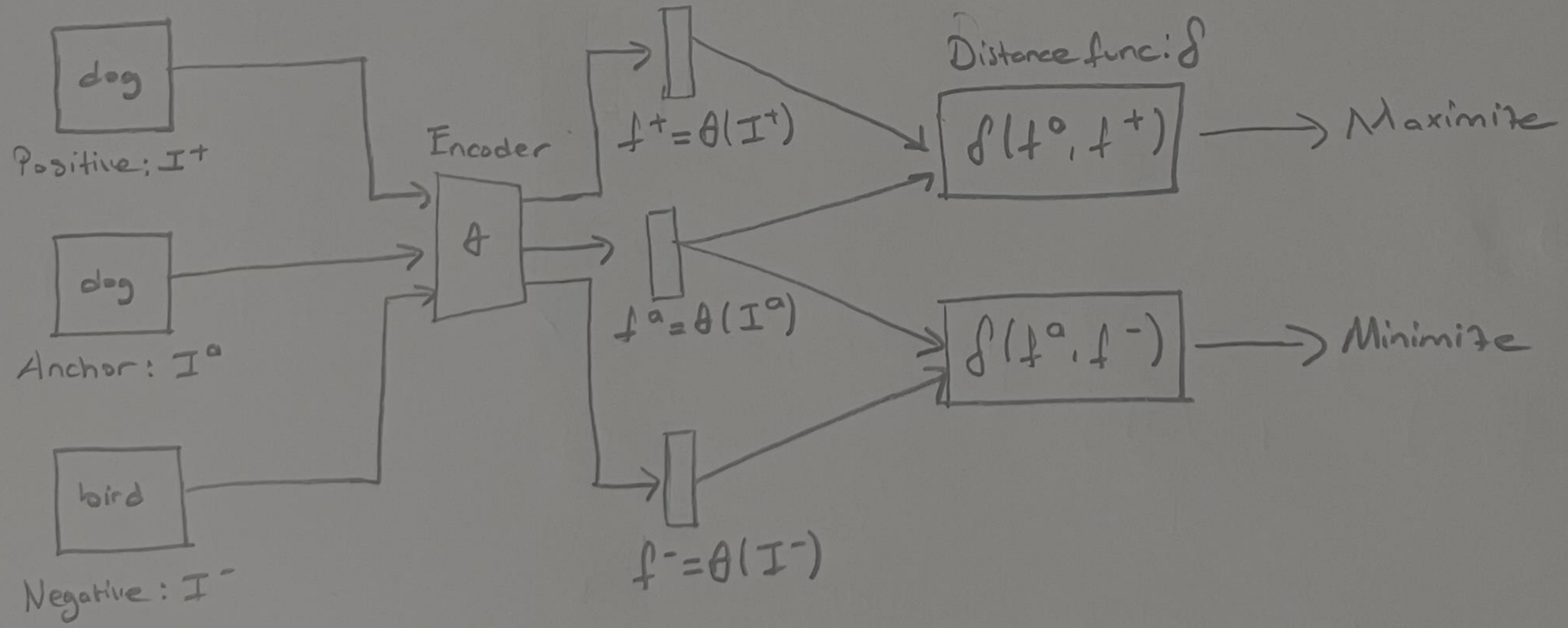


Source: https://medium.com/@rohan10dalvi/cosine-similarity-in-graph-similarity-3c260e7efd3f (Retrieved February 4, 2025.)


T_n: "A cartoon drawing of a cow"
T_2: "A photo of a 3D printed baby Yoda"
T_1: "A map of Africa"
Text Encoder
V_1
V_2
V_n
Image Encoder
$f(V_1, T_1)$ $f(V_1, T_2)$ ... $f(V_1, T_n)$
$f(V_2, T_1)$ $f(V_2, T_2)$ ... $f(V_2, T_n)$
$f(V_n, T_1)$ $f(V_n, T_2)$ ... $f(V_n, T_n)$
The goal of the loss function is to maximize the cosine similarity of the correct image-text pairs (those in the diagonal) and minimize all the others

The diagonal elements of the similarity matrix represent the cosine similarities between each image and its corresponding textual description within a batch. These diagonal elements indicate the alignment between paired image-text data, which is the primary focus during training.

Contrastive Learning Approach: Give the model examples of the form (anchor, positive, negative), where anchor is an image from one class, say a dog, positive is an alternative image from the same class, a dog, and negative is an image of another class, say bird.



Anchor: An image of a dog

Positive: "an image of a dog"

Negative: "an image of a bird"

```
function filterStudies({ studies, filterByOrg = false, filte
  studies.filter(study => {
```

https://lilianweng.github.io/posts/2021-05-31-contrastive/

2.2 Contrastive-based VLMs

Contrastive-based training is often better explained through an Energy-Based Models (EBM) point of view [LeCun et al., 2006] in which a model E_θ, parameterized by θ, is trained to assign low energy to observed variables and high energy to unobserved ones. Data from a target distribution should have low energy while *any other data points* should have higher energy. To train these models, we consider input data x with an energy function $E_\theta(x)$ of parameters θ. The corresponding Boltzman distribution density function to learn can be written as:

$$p_\theta(x) = \frac{e^{-E_\theta(x)}}{Z_\theta}$$

with normalization factor $Z_\theta = \sum_x e^{-E_\theta(x)}$. To estimate the target distribution P_D from which input data are drawn, we can in principle use the traditional maximum likelihood objective:

$$\arg\min_\theta \mathbb{E}_{x \sim P_D(x)}[-\log P_\theta(x)]$$

whose gradient is:

$$\frac{\partial \mathbb{E}_{x \sim P_D(x)}[-\log P_\theta(x)]}{\partial \theta} = \mathbb{E}_{x^+ \sim P_D(x)} \frac{\partial E_\theta(x^+)}{\partial \theta} - \mathbb{E}_{x^- \sim P_\theta(x)} \frac{\partial E_\theta(x^-)}{\partial \theta}$$

Source: Bordes, F., Pang, R. Y., Ajay, A., Li, A. C., Bardes, A., Petryk, S., ... & Chandra, V. (2024). An introduction to vision-language modeling. arXiv preprint arXiv:2405.17247.

Contrastive Language-Image Pre-training (CLIP)

CLIP employs a contrastive learning approach, a self-supervised method where the model learns to distinguish between related and unrelated image-text pairs. During training, images are processed through a vision encoder, such as a Convolutional Neural Network (CNN) or Vision Transformer (ViT), while text is processed through a language encoder, typically a Transformer model. The model aligns the embeddings from both modalities in a shared latent space, maximizing the similarity of correct image-text pairs and minimizing it for incorrect ones.

Contrastive Language-Image Pre-training (CLIP)

One of CLIP's most remarkable features is its zero-shot learning ability. This means that CLIP can generalize to new tasks without additional training. For instance, it can classify images into categories it has never explicitly seen during training by simply providing textual prompts.

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)},$$

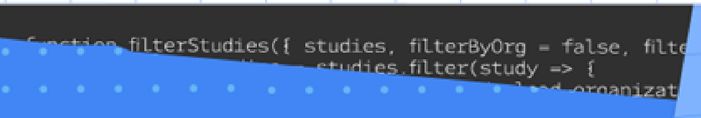
Source: https://medium.com/towards-data-science/contrastive-loss-explaned-159f2d4a87ec (Retrieved February 4, 2025.)

Contrastive Language-Image Pre-training (CLIP)

CLIP consists of two primary components:

Image Encoder: This component processes images to generate fixed-size embeddings. CLIP is architecture-agnostic, meaning it can utilize various models for the image encoder, such as ResNet-50, ResNet-101, or Vision Transformers (ViT). These models extract features from images and map them into a continuous vector space.

Text Encoder: This component processes textual data to produce corresponding embeddings. CLIP typically employs a Transformer-based architecture for the text encoder. The text encoder maps textual descriptions into the same embedding space as the image encoder, allowing for direct comparison between image and text representations.



Contrastive Language-Image Pre-training (CLIP)

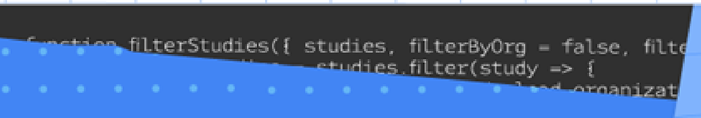
CLIP is trained using a contrastive learning approach on a large dataset of image-text pairs. The training process involves the following steps:

Data Preparation: A batch of image-text pairs is selected from the dataset.

Encoding: Each image and its corresponding text are processed through their respective encoders to generate embeddings.

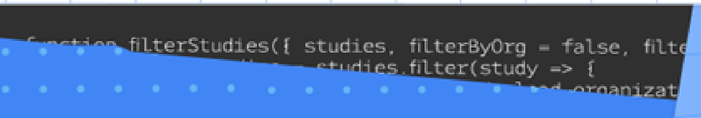
Similarity Computation: The model computes the cosine similarity between all possible image and text embedding pairs within the batch, resulting in a similarity matrix.

Contrastive Loss Calculation: The model applies a contrastive loss function to maximize the similarity of correct image-text pairs (diagonal elements of the similarity matrix) and minimize the similarity of incorrect pairs (off-diagonal elements).



Contrastive Language-Image Pre-training (CLIP)

This training strategy enables the model to learn a shared embedding space where related images and texts are positioned close together, while unrelated pairs are spaced apart.



Google Developer Experts

CLIP-like Model Notebook

https://github.com/ayyucedemirbas/RadiologyCLIP/blob/main/RadiologyCLIP.ipynb

Multimodal Data Fusion Techniques

Multimodal data fusion integrates heterogeneous data from multiple sources (modalities) such as text, images, audio, sensors, and video to enhance machine learning models. By leveraging complementary information, fusion techniques aim to improve robustness, accuracy, and generalization compared to unimodal approaches.

Early Fusion

Early Fusion combines data from different modalities at an early stage, typically at the feature or input level, so that a single model can learn a joint representation of the various sources of information.

Features from different modalities (e.g., text, images, audio, or sensor data) are combined into a unified representation before being passed into subsequent layers of a model. Rather than processing each modality separately, the model receives an integrated set of features that represent all modalities simultaneously. This early integration can be as simple as concatenating feature vectors or may involve more sophisticated joint embedding techniques.

Early Fusion

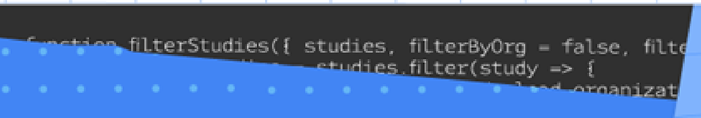
Preprocessing and Feature Extraction:

Modality-Specific Encoders:

Each data type is processed using specialized encoders. For example, textual data might be processed by a transformer-based encoder, while visual data might be processed through a CNN. The key here is to obtain feature representations that are semantically rich.

Dimensionality Alignment:

Before fusion, the extracted features may require scaling or dimensionality reduction to ensure that no single modality overwhelms the joint representation. Techniques like normalization or even additional transformation layers (e.g., fully connected layers) are used to balance the contributions of each modality.



Early Fusion

Fusion Operation:

Concatenation:

The simplest and most common method is to concatenate the feature vectors from each modality into one long vector. This method preserves all the original information but may lead to high-dimensional inputs.

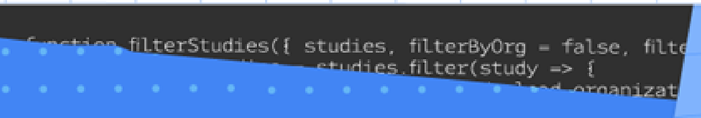
Joint Embedding:

More advanced approaches involve projecting features into a common latent space where similarities across modalities can be better captured. This could involve learning a mapping function during training that optimally combines the features into a compact, unified representation.

Early Fusion

End-to-End Training:

With early fusion, the entire network—from the initial feature extraction layers through the fusion layer to the final prediction layer—is often trained jointly. This allows the network to learn the optimal way to combine modalities from scratch, potentially capturing complex interdependencies.



Late Fusion

Each data modality (such as images, text, audio, or sensor signals) is handled by its own specialized model.

Fusion at the decision level. Once each model has produced its own prediction (or a high-level feature representation), a separate fusion mechanism combines these outputs into a single, final decision.

Works like model ensembling.

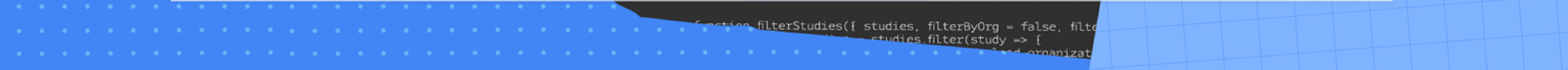
Late Fusion

Once each model has produced its own prediction (or a high-level feature representation), a separate fusion mechanism combines these outputs into a single, final decision. This could be implemented through:

Averaging or Weighted Averaging: The final output is a mean of the individual predictions, potentially with different weights assigned based on each modality's reliability.

Majority Voting: In classification tasks, the final decision is made based on the most common prediction among the modalities.

Ensemble Methods: More sophisticated techniques, such as stacking or meta-learning, where a secondary model is trained to interpret the collection of unimodal outputs.



Late Fusion

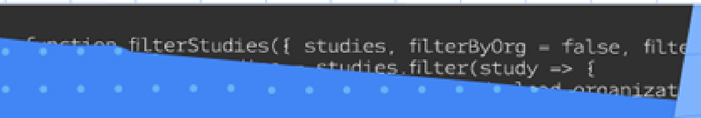
Mathematically, if you denote the output (e.g., probability scores for classification) from each unimodal model as p_1, p_2, ..., p_n, a late fusion system can be described by a fusion function f such that: $p = f(p_1, p_2, ..., p_n)$

For example, with a simple weighted average:

$$p = \sum_{i=1}^{n} w_i \, pi \text{ with } \sum_{i=1}^{n} w_i = 1$$

Intermediate Fusion

Intermediate fusion is a strategy in multimodal learning that blends the benefits of both early and late fusion methods by integrating modality-specific features not right at the input but not only at the final decision stage either. Instead, it fuses the representations produced by separate “unimodal” networks at one or more intermediate layers in the model. This allows the network to first extract rich, specialized features from each modality and then combine these features to learn cross-modal interactions at a level where the features are abstract yet still retain modality-specific nuances.



Intermediate Fusion

Step 1. Unimodal Feature Extraction:

Each modality is processed by its own specialized network (often called a unimodal module) that is designed to extract salient features. For instance, a deep CNN might extract high-level visual features from an image, while a transformer or RNN extracts contextual embeddings from text. These features are typically higher-level representations—more abstract than raw pixels or word tokens.

Intermediate Fusion

Step 2. Fusion at an Intermediate Level:

After the unimodal modules have produced their feature maps or vectors, these representations are fused together. The fusion can be performed using several techniques:

Concatenation: Simply joining the feature vectors along the feature dimension.

Attention-Based Methods: Weighting the contributions of each modality by learning attention scores so that more relevant features are emphasized.

Tensor Operations: Combining the modalities using element-wise multiplication, outer products, or other tensor-based arithmetic that can capture interactions between modalities.

Dynamic Fusion Modules: Some models introduce additional layers (or modules) that refine the fused representation dynamically before it is passed on to later layers.

Intermediate Fusion

Step 3. Joint Processing:

The fused representation is then fed into subsequent layers (often a multimodal module) that learn to map the combined features to the final output (e.g., a classification decision or a regression value). This joint processing layer can further capture interdependencies that were not apparent within the individual unimodal features.

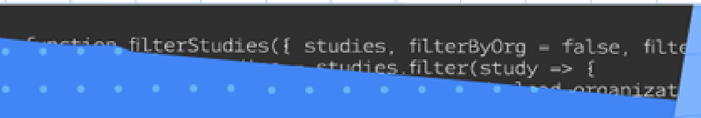
Intermediate Fusion

"*For example*, one recent systematic review on intermediate fusion in biomedical applications describes the process formally as taking modality-specific features x_1, x_2, ..., x_N (each already processed through several layers) and fusing them into a joint representation h using a fusion function f:

$h = f(x_1, x_2, ..., x_N)$

h is then used by the remainder of the network to produce the task-specific output."

Guarrasi, V., Aksu, F., Caruso, C. M., Di Feola, F., Rofena, A., Ruffini, F., & Soda, P. (2024). A systematic review of intermediate fusion in multimodal deep learning for biomedical applications. *arXiv preprint arXiv:2408.02686*.

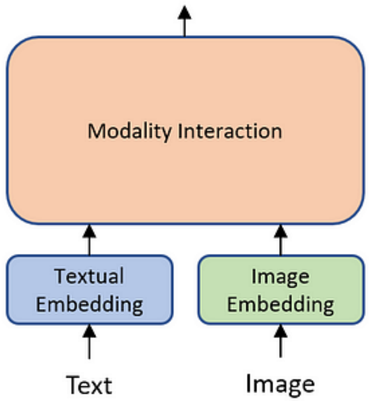
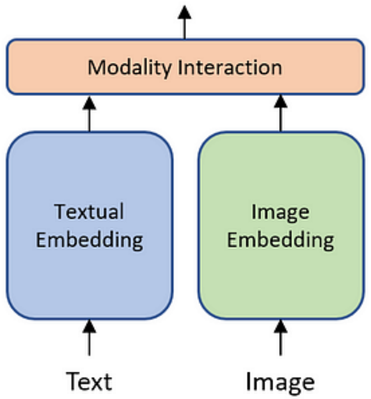


“Intermediate fusion is the most widely used fusion strategy. It involves processing each modality into a latent representation, fusing them, and then doing some final processing to produce the output scores.”

https://medium.com/@raj.pulapakura/multimodal-models-and-fusion-a-complete-guide-225ca91f6861

Aspect	Early Fusion (Feature-level Fusion)	Intermediate Fusion (Mid-level Fusion)	Late Fusion (Decision-level Fusion)
Definition	Combines raw inputs or low-level features from different modalities at the very beginning of the model.	Fuses features extracted independently by modality-specific networks at one or more intermediate layers, before the final decision stage.	Processes each modality independently (often using separate specialized networks) and fuses their outputs or decisions only at the final stage.
Processing Stage	- Fusion occurs at the input or feature extraction stage. - Data from different modalities is merged prior to significant independent processing.	- Fusion occurs after each modality has undergone initial processing (feature extraction) but before final decision layers. - Typically, unimodal networks produce abstract representations that are then fused.	- Fusion occurs after all unimodal processing is completed. - Each modality produces an independent output (e.g., classification scores) which are then combined.
Key Fusion Operation	- Simple concatenation of raw data or early features. - Sometimes accompanied by normalization or dimensionality reduction to handle heterogeneous data.	- Techniques include concatenation of intermediate features, attention mechanisms, tensor operations (e.g., element-wise multiplication, outer products), or dynamic modules.	- Fusion methods include averaging, weighted averaging, majority voting, ensemble methods (stacking/meta-learning), or probabilistic combination of predictions.
Advantages	- Captures cross-modal correlations from the outset. - Allows the model to learn joint representations directly. - Simplifies architecture by using a single unified model.	- Balances the strengths of unimodal feature extraction and joint learning. - Preserves modality-specific abstractions while enabling cross-modal interactions. - Offers flexibility in fusion operations.	- Modular: each modality can be processed with its best-suited network. - Enables use of state-of-the-art, modality-specific pre-trained models. - Robust if one modality fails.
Limitations/Challenges	- Directly fusing raw or low-level features may lead to high-dimensional input spaces. - Difficult to handle heterogeneity among modalities (e.g., image vs. text).	- Requires careful design to align and match features from different modalities. - More complex training as gradients must flow through both unimodal and fusion modules.	- May lose fine-grained inter-modal correlations because modalities are fused only at the end. - Fusion performance heavily depends on the chosen decision-level mechanism.
Training Complexity	- Lower training complexity initially (due to a single network), but may require heavy preprocessing.	- Moderate complexity: each modality is trained separately first, then fused, requiring end-to-end fine-tuning of both unimodal and fusion layers.	- Typically simpler training for unimodal networks (they can be trained independently), but the fusion of decisions may require additional tuning to optimally weight the modalities.
Flexibility & Modularity	- Less modular since all modalities are fused at the input level. - Difficult to add or remove a modality without retraining the entire network.	- Moderately modular: unimodal modules can be developed independently, then integrated via the fusion module. - Easier to adjust the fusion mechanism without re-designing unimodal networks.	- Highly modular: unimodal systems can be updated or replaced independently. - Fusion stage can be adapted without changing unimodal networks.
Information Interaction	- High potential for early cross-modal interactions but may mix noise with signal.	- Provides a balance by allowing initial independent processing then enabling richer cross-modal interaction at an abstract level.	- Minimal inter-modal interaction during feature extraction; interactions occur only via combination of final outputs.
Computational Cost	- Can be computationally efficient if raw data dimensions are managed; however, may incur heavy preprocessing costs.	- Moderate cost: additional fusion layers add computational overhead, but unimodal networks reduce complexity by extracting lower-dimensional representations before fusion.	- Often lower fusion cost since unimodal models operate independently; however, overall cost depends on the complexity of each unimodal network and the final fusion strategy.
Typical Applications	- Suitable when modalities are homogeneous or have a similar nature (e.g., multiple image channels). - Useful when joint learning from raw data is feasible (e.g., sensor data fusion).	- Widely used in applications where different modalities (such as text, images, audio, and sensor data) provide complementary information and need cross-modal interactions, e.g., biomedical diagnostics, sentiment analysis, and autonomous driving.	- Common in scenarios where high-performance unimodal models exist independently, such as combining clinical data with medical images in diagnostics, or integrating audio with video in multimedia analysis.
Robustness to Missing Data	- Lower robustness; if one modality is missing, the model may struggle since it expects all inputs to be fused from the start.	- Can be made more robust if unimodal networks are strong; however, mismatched or missing modalities can still impact the quality of the fused representation if not properly handled.	- Often more robust since each modality's output is computed independently; a missing or degraded modality may be compensated by the others in the final decision fusion process.

Types of multimodal data fusion

Model Type	Architecture	Single Modality Features	Modality Interaction	Training
Early Fusion		Very simple Example: GLoVe, CNNs	Complex Examples: Cross-Modal Transformers	Slow & difficult to train
Late Fusion		Heavy Example: Pretrained Transformers	Very simple Example: Concatenation + Single hidden layer	Simple to train

Source: https://medium.com/data-science-at-microsoft/visual-question-answering-with-multimodal-transformers-d4f57950c867 (Retrieved February 1, 2025.)

BERT-like Architectures

BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based architecture that has been highly influential in natural language processing (NLP).

VisualBERT, ViLBERT, Pixel-BERT, ImageBERT, VL-BERT, VD-BERT, LXMERT, UNITER, CXR-BERT

Two-stream Models

Two-stream models are a type of architecture used in vision-language models to process and integrate information from two distinct modalities: visual data (images or videos) and textual data (language).

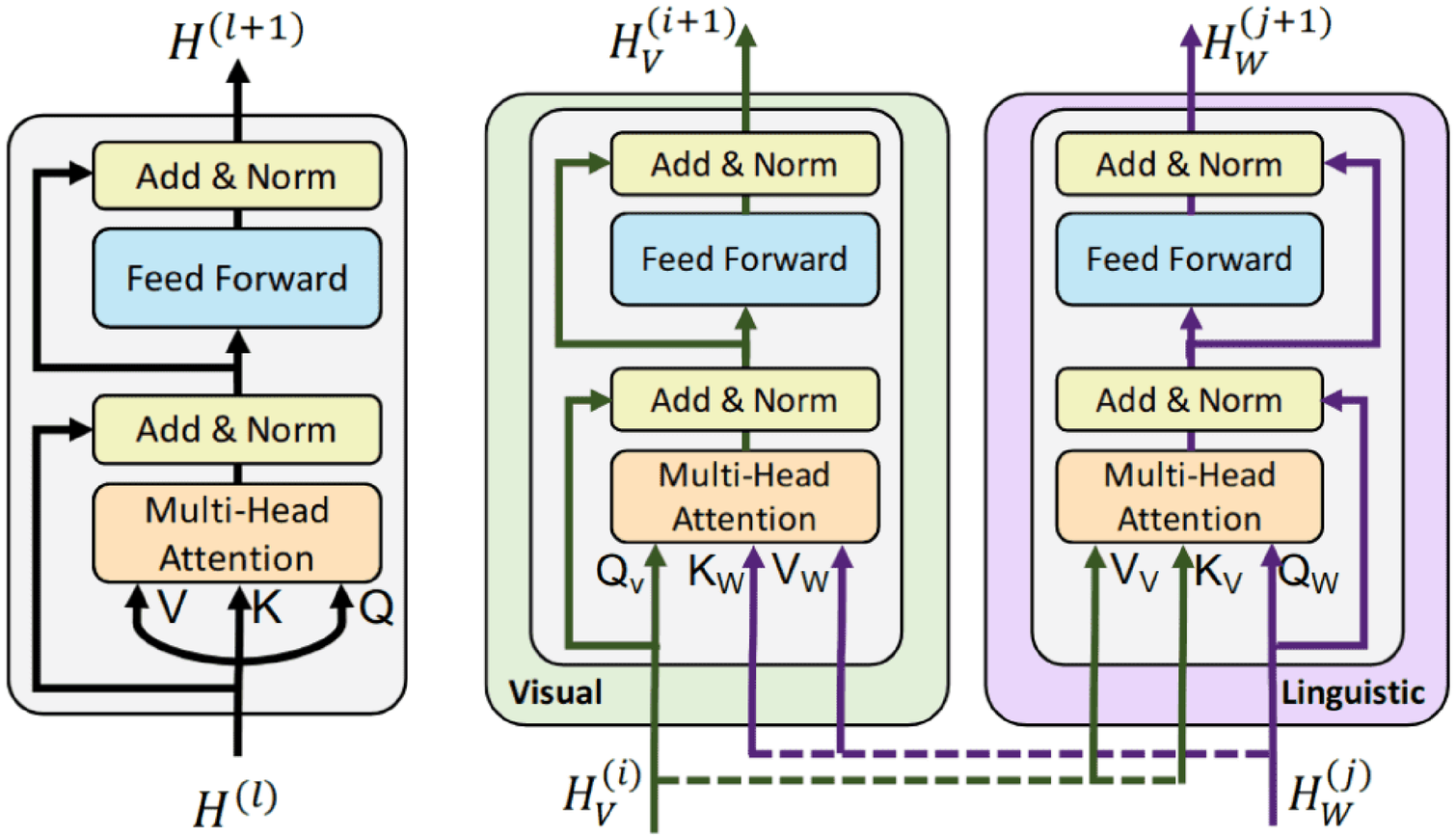
“ViLBERT is trained on image-text pairs. The text is encoded with the standard transformer process using tokenization and positional embeddings. It is then processed by the self-attention modules of the transformer. Images are decomposed into non-overlapping patches projected in a vector, as in vision transformer’s patch embeddings.”

https://theaisummer.com/vision-language-models/

ViLBERT

“To learn a joint representation of images and text, a “co-attention” module is used. The “co-attention” module calculates importance scores based on both images and text embeddings.
In a way, the model is learning the alignment between words and image regions. Another transformer module is added on top for refinement. This “co-attention” / transformer block can, of course, be repeated many times.” https://theaisummer.com/vision-language-models/

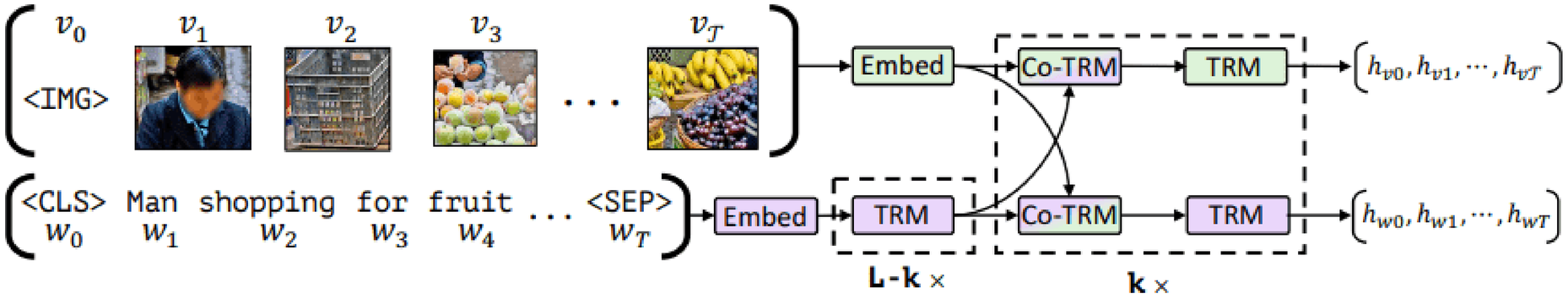
ViLBERT co-attention transformer layer



Standard encoder transformer block VS **co-attention** transformer layer

Source: https://theaisummer.com/vision-language-models (Retrieved February 1, 2025.)

ViLBERT



Source: https://theaisummer.com/vision-language-models (Retrieved February 1, 2025.)

VilBERT processes images and text in two parallel streams that interact through co-attention

Single-Stream Models

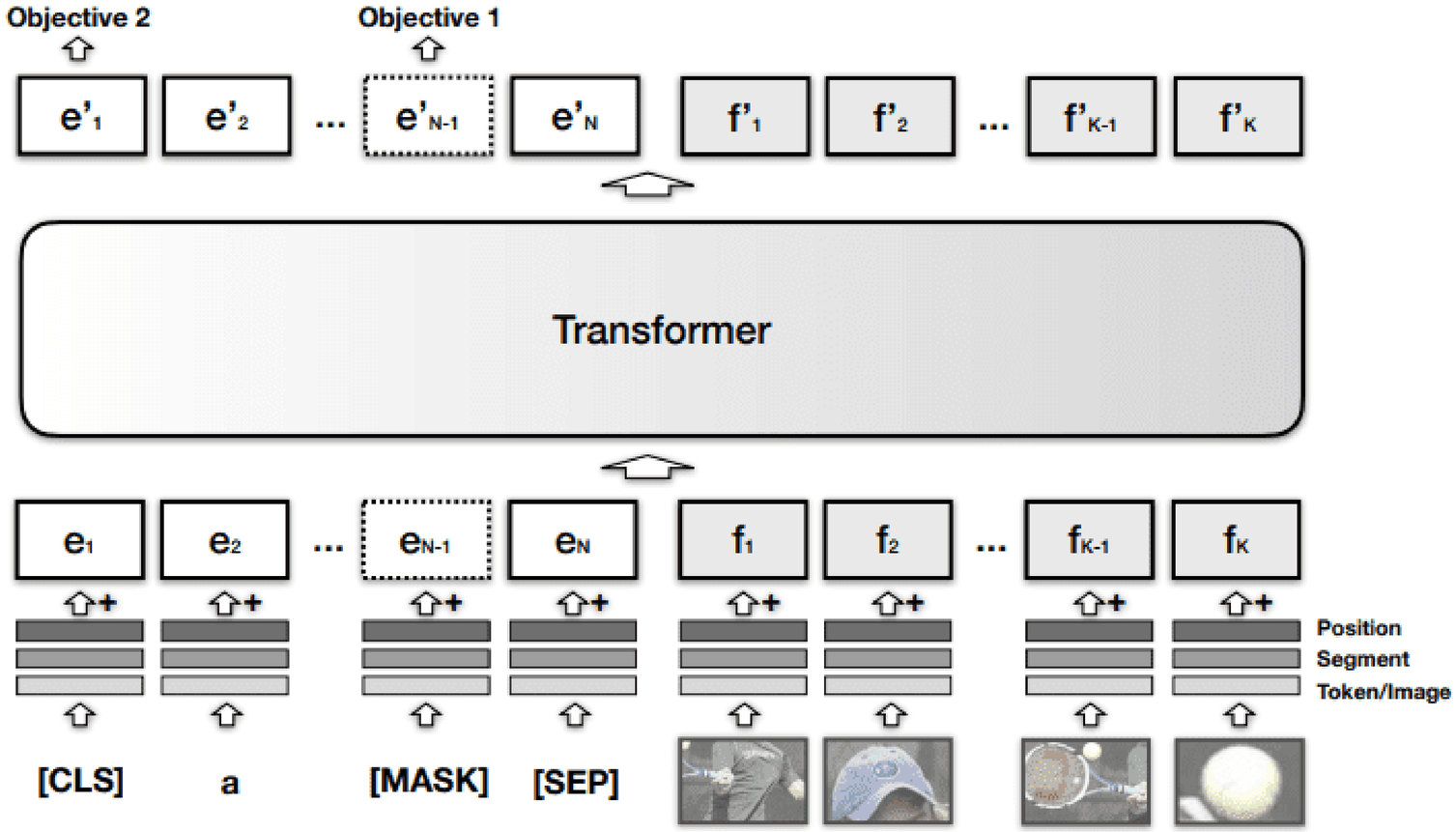
Single-stream models treat vision and language as a unified input, processing them together in a shared architecture.

“VisualBERT combines image regions and language with a transformer in order for self-attention to discover alignments between them. In essence, they added a visual embedding to the standard BERT architecture. The visual embedding consists of :

- A visual feature representation of the region produced by a CNN
- A segment embedding that distinguishes image from text embeddings
- A positional embedding to align regions with words if provided in the input”

https://theaisummer.com/vision-language-models/

VisualBERT



Source: https://theaisummer.com/vision-language-models (Retrieved February 1, 2025.)

VisualBERT combines image regions and text with a transformer module

	Single-Stream Models	Two-Stream Models
Architecture	Unified processing of modalities in one network	Separate streams for each modality, fused later
Interaction Level	Early and deep interaction between modalities	Modality-specific processing with later interaction
Complexity	Simpler, single architecture	More modular, with task-specific fusion
Computational Cost	Higher (processing combined inputs in a single model)	Lower during inference (modality-specific encoders)

VLM Pretraining Objectives

Contrastive Learning

Goal: Align image-text pairs in a shared embedding space.

Loss (InfoNCE):

$$\mathcal{L}_{\text{cont}} = -\log \frac{\exp(s(I,T)/\tau)}{\sum_{j=1}^{N} \exp(s(I,T_j)/\tau)}$$

s(I,T): Cosine similarity; τ: Temperature.

Models: CLIP, ALBEF.

Hard Negatives: Mine difficult negatives to improve alignment (ALBEF).

VLM Pretraining Objectives

Masked Language Modeling (MLM)

Goal: Predict masked tokens using visual context.

Loss (Cross-Entropy):

$$\mathcal{L}_{\text{MLM}} = -\sum_{i \in \text{masked}} \log P(w_i | I, T_{\setminus i})$$

Models: VisualBERT, LXMERT.

Variants: Masked region modeling (e.g., mask image patches in BEiT).

VLM Pretraining Objectives

Image-Text Matching (ITM)

Goal: Classify if image-text pairs match (binary task).

Loss (Binary Cross-Entropy):

$$\mathcal{L}_{\text{ITM}} = -y \log p - (1 - y) \log(1 - p)$$

Models: UNITER, OSCAR.

VLM Pretraining Objectives

Generative Objectives

Prefix Language Modeling: Generate text given image and text prefix.

$$\mathcal{L}_{\text{LM}} = -\sum_t \log P(w_t | I, w_{<t})$$

Models: SimVLM, GPT-4V (https://openai.com/index/gpt-4v-system-card/).

Image Generation: Reconstruct masked patches (e.g., MAE).

VLM Pretraining Objectives

“Masked Language Modeling is often used when the transformer is trained only on text. Certain tokens of the input are being masked at random. The model is trained to simply predict the masked tokens (words). In the case of BERT, bidirectional training enables the model to use both previous and following tokens as context for prediction.

Next Sequence Prediction works again only with text as input and evaluates if a sentence is an appropriate continuation of the input sentence. By using both false and correct sentences as training data, the model is able to capture long-term dependencies.” https://theaisummer.com/vision-language-models/

VLM Pretraining Objectives

“Masked Region Modeling masks image regions in a similar way to masked language modeling. The model is then trained to predict the features of the masked region.

Image-Text Matching forces the model to predict if a sentence is appropriate for a specific image.

Word-Region Alignment finds correlations between image region and words.

Masked Region Classification predicts the object class for each masked region.

Masked Region Feature Regression learns to regress the masked image region to its visual features.

” https://theaisummer.com/vision-language-models/

VLM Pretraining Objectives

Vision-Language Instruction Tuning (VLIT) is an advanced training methodology designed to enhance the capabilities of Large Vision-Language Models (LVLMs). By aligning these models with specific instructions that integrate both visual and textual data, VLIT enables LVLMs to perform a wide array of multimodal tasks, such as image captioning, visual question answering, and complex visual reasoning.

VLM Pretraining Objectives

Foundations of Instruction Tuning

Instruction tuning is a supervised training phase in Large Language Models (LLMs) that aims to improve the model's ability to generalize instruction execution and adapt to user preferences. This process involves training models on datasets that pair instructions with corresponding outputs, thereby enabling them to follow human-like directives more effectively. When extended to the multimodal domain, this approach becomes Vision-Language Instruction Tuning, where models are trained to interpret and generate responses based on combined visual and textual inputs.

VLM Pretraining Objectives

Challenges

Data Complexity: Integrating visual and textual data requires models to process and understand diverse data types simultaneously.

Task Diversity: VLIT encompasses a broad range of tasks, from simple image descriptions to complex reasoning that involves both visual and textual elements.

Quality of Instruction Data: High-quality VLIT data should possess characteristics such as diversity, clarity, and relevance to ensure effective model training.

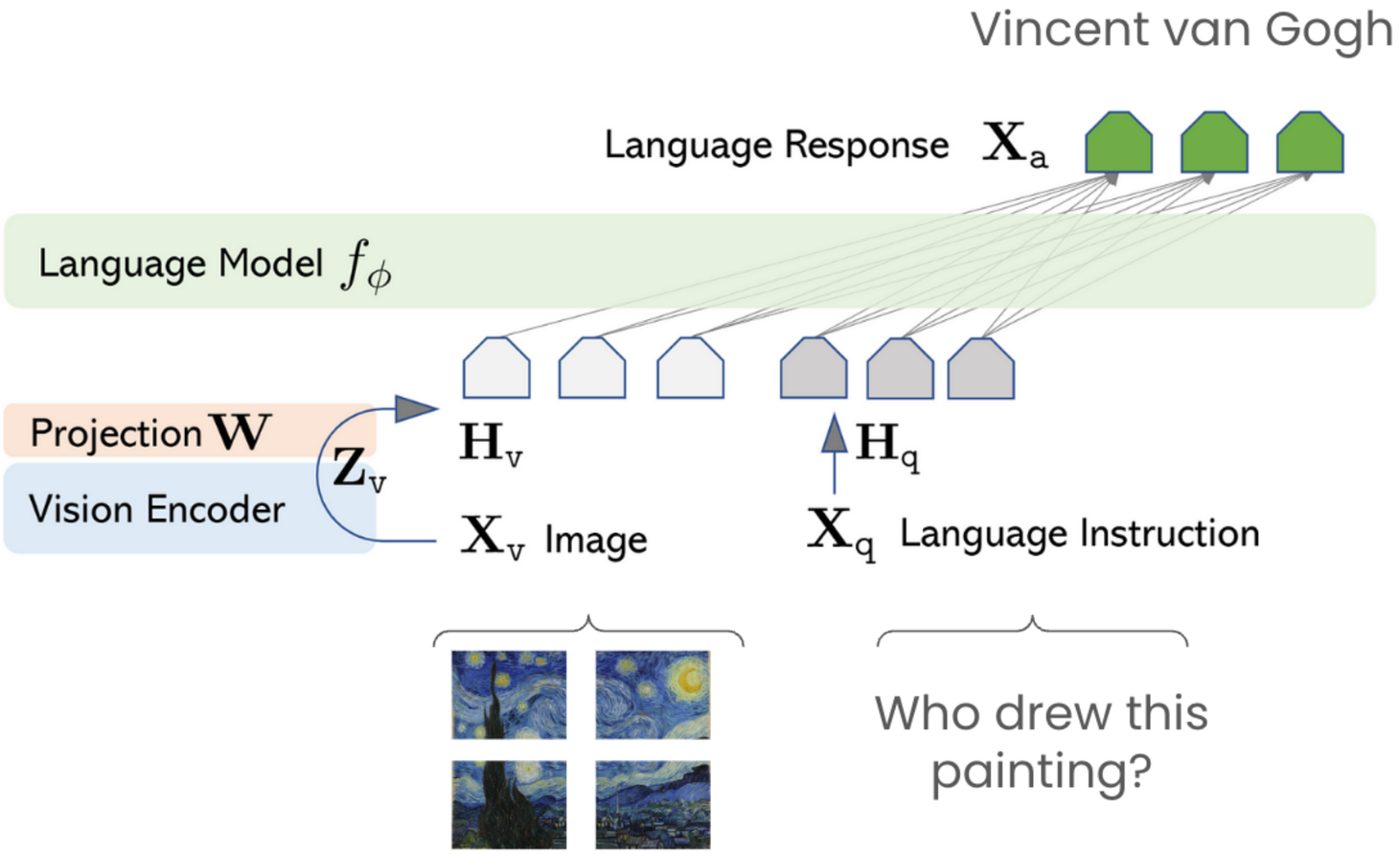
VLM Pretraining Objectives

Approaches to VLIT Data Generation

Annotation Adaptation: This approach involves adjusting and rewriting existing annotation data to fit the VLIT data template, ensuring that the instructions are aligned with both visual and textual content.

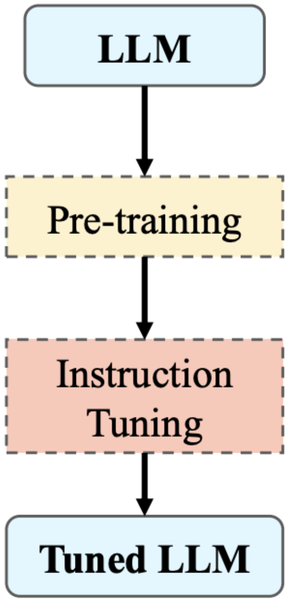
Self-Instruct: Leveraging Large Language Models (LLMs) to synthesize annotation data from multiple sources, this method creates VLIT data with greater diversity and complexity. However, it may also introduce noise and hallucinations.

Visual Instruction Tuning

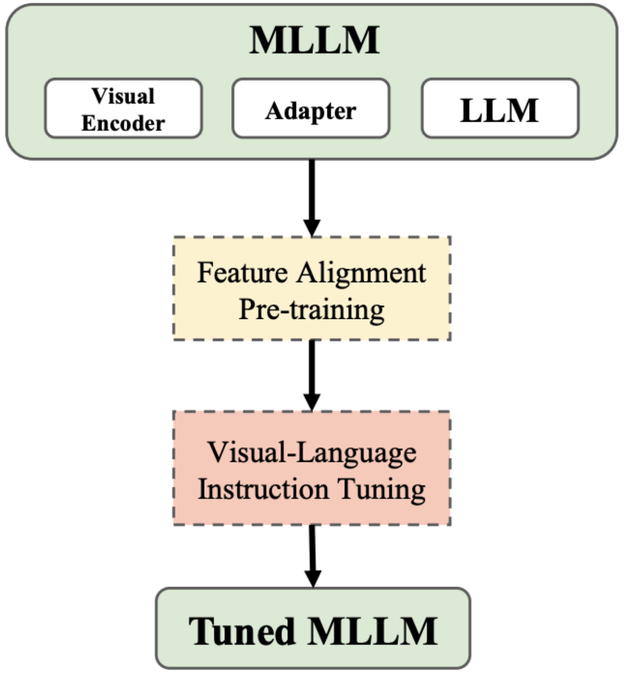


Source: https://weaviate.io/papers/vit (Retrieved February 1, 2025.)

LLM training paradigm



MLLM training paradigm



Pure Text Instruction

```
Instruction: <instruction>
Input: <text>
Response: <output>
```

Visual-Language Instruction

```
<BOS> <context>
### Instruction: <instruction>
### Input: {<images>, <text>}
### Response: <output> <EOS>
```

Source: Li, C., Ge, Y., Li, D., & Shan, Y. (2023). Vision-language instruction tuning: A review and analysis. arXiv preprint arXiv:2311.08172.

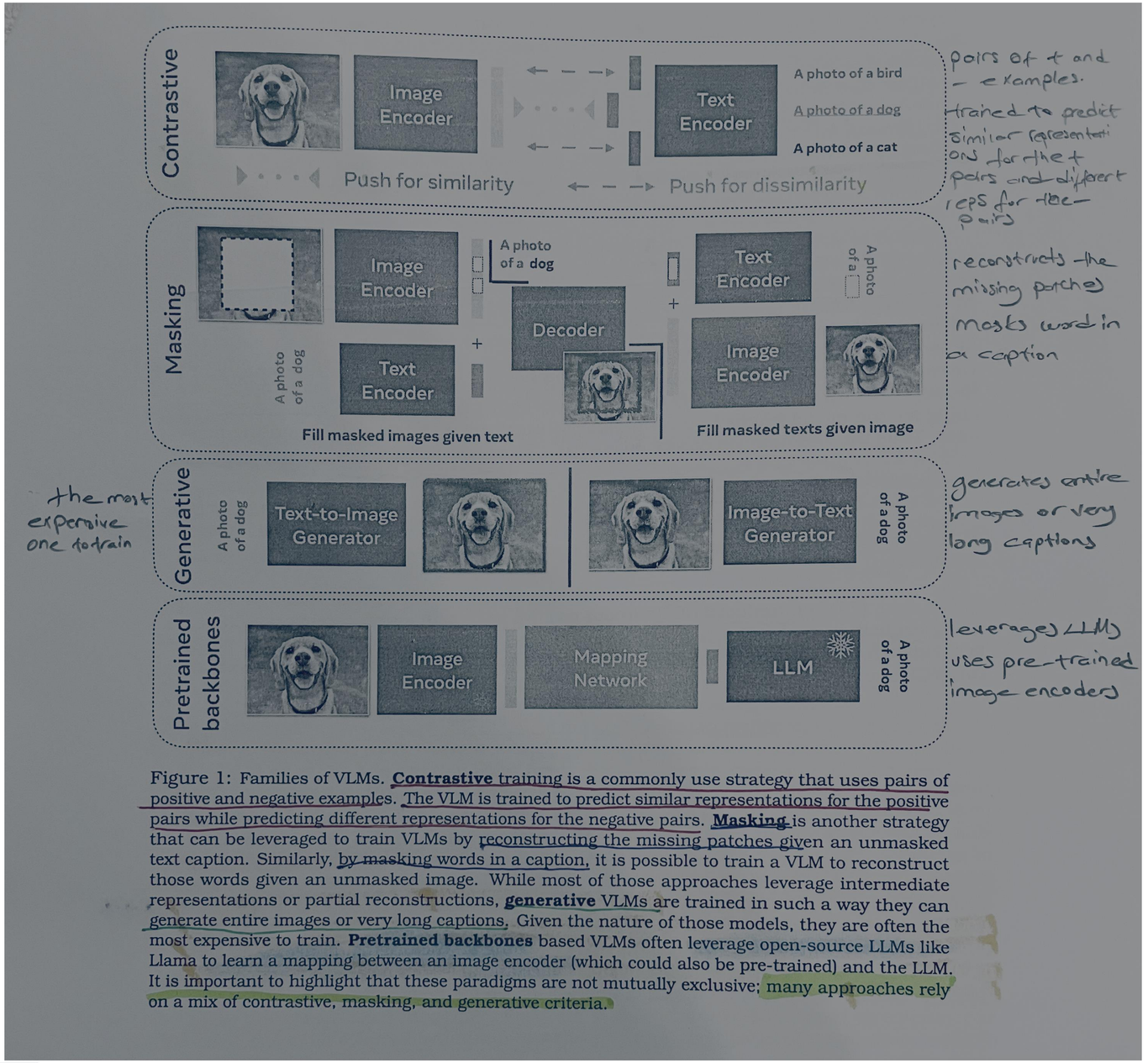


Figure 1: Families of VLMs. **Contrastive** training is a commonly use strategy that uses pairs of positive and negative examples. The VLM is trained to predict similar representations for the positive pairs while predicting different representations for the negative pairs. **Masking** is another strategy that can be leveraged to train VLMs by reconstructing the missing patches given an unmasked text caption. Similarly, by masking words in a caption, it is possible to train a VLM to reconstruct those words given an unmasked image. While most of those approaches leverage intermediate representations or partial reconstructions, **generative** VLMs are trained in such a way they can generate entire images or very long captions. Given the nature of those models, they are often the most expensive to train. **Pretrained backbones** based VLMs often leverage open-source LLMs like Llama to learn a mapping between an image encoder (which could also be pre-trained) and the LLM. It is important to highlight that these paradigms are not mutually exclusive; many approaches rely on a mix of contrastive, masking, and generative criteria.

pairs of + and − examples. trained to predict similar representations for the + pairs and different reps for the − pairs

reconstructs the missing patches) masks words in a caption

generates entire images or very long captions

the most expensive one to train

leverages LLMs uses pre-trained image encoders

Source: Bordes, F., Pang, R. Y., Ajay, A., Li, A. C., Bardes, A., Petryk, S., ... & Chandra, V. (2024). An introduction to vision-language modeling. arXiv preprint arXiv:2405.17247.

When to use masking?

“Masking is an alternative strategy to train VLMs. By learning to reconstruct data from both masked images and text, it is possible to jointly model their distributions. In contrast to contrastive models which operate in a representation space, models based on masking might need to leverage a decoder to map back the representation to the input space (and thus to apply a reconstruction loss). Training an additional decoder might add an additional bottleneck which might make these methods less efficient than a purely contrastive one. However, the advantage is that there is no batch dependency anymore since each example can be considered separately (because we do not need negative examples). Removing negative examples can enable the use of smaller mini-batches without the need to finetune additional hyper-parameters such as the softmax temperature. Many VLM methods leverage a mix of masking strategies along with some contrastive loss.”

Source: Bordes, F., Pang, R. Y., Ajay, A., Li, A. C., Bardes, A., Petryk, S., ... & Chandra, V. (2024). An introduction to vision-language modeling. arXiv preprint arXiv:2405.17247.

How to train a VLM?

1- Selecting a Training Paradigm

VLMs can be trained using different approaches:

Contrastive Learning (e.g., CLIP): Aligns images and text in a shared embedding space by maximizing similarities between positive pairs and minimizing similarities with negative pairs.

Masking (e.g., FLAVA, MaskVLM): Uses masked image patches or text tokens to predict missing parts, improving multimodal understanding.

Generative Modeling (e.g., CoCa, CM3Leon): Trains the model to generate images or text, often using transformer-based architectures.

Pretrained Backbone Approaches (e.g., MiniGPT, Frozen): Maps image representations to a frozen LLM's input space, enabling visual question answering and image description.

Source: Bordes, F., Pang, R. Y., Ajay, A., Li, A. C., Bardes, A., Petryk, S., ... & Chandra, V. (2024). An introduction to vision-language modeling. arXiv preprint arXiv:2405.17247.

How to train a VLM?

2. Preparing Training Data

Data Sources: Use large-scale image-text datasets like LAION, COCO, or Conceptual Captions.

Data Filtering:

Remove low-quality text and images.

Use contrastive-based filtering like CLIP-Score to rank image-text alignment.

Ensure diversity by balancing different concepts.

Synthetic Data: Generate additional training data using text-to-image models or auto-captioning techniques.

Data Augmentation: Apply transformations like cropping, color jittering, or text paraphrasing to improve model robustness.

Source: Bordes, F., Pang, R. Y., Ajay, A., Li, A. C., Bardes, A., Petryk, S., ... & Chandra, V. (2024). An introduction to vision-language modeling. arXiv preprint arXiv:2405.17247.

How to train a VLM?

3. Model Architecture

Encoders: Use separate vision and text encoders (e.g., ViT for images, Transformer for text) or a unified multimodal encoder.

Modality Fusion: Implement cross-attention or multimodal transformers for deeper integration of vision and language.

Loss Functions:

Contrastive loss (InfoNCE) for aligning embeddings.

Masked modeling loss for reconstruction tasks.

Cross-entropy for generative models.

Source: Bordes, F., Pang, R. Y., Ajay, A., Li, A. C., Bardes, A., Petryk, S., ... & Chandra, V. (2024). An introduction to vision-language modeling. arXiv preprint arXiv:2405.17247.

How to train a VLM?

4. Training Strategies

Compute Resources: Large-scale VLMs typically require 64+ GPUs for efficient training.

Batch Size: Large mini-batches improve contrastive learning but require high memory.

Optimization: Use AdamW optimizer with learning rate warm-up and cosine decay.

Speed Optimization:

Use torch.compile and efficient attention mechanisms.

Optimize data loading (e.g., using FFCV to store uncompressed images).

Source: Bordes, F., Pang, R. Y., Ajay, A., Li, A. C., Bardes, A., Petryk, S., ... & Chandra, V. (2024). An introduction to vision-language modeling. arXiv preprint arXiv:2405.17247.

How to train a VLM?

5. Evaluation and Fine-Tuning

Benchmarking: Assess performance on tasks like image captioning, VQA, and zero-shot classification.

Bias & Hallucination Checks: Analyze biases in model predictions and minimize unrealistic outputs.

Parameter-Efficient Fine-Tuning (PEFT): Use LoRA or adapters to fine-tune specific tasks with minimal compute.

Source: Bordes, F., Pang, R. Y., Ajay, A., Li, A. C., Bardes, A., Petryk, S., ... & Chandra, V. (2024). An introduction to vision-language modeling. arXiv preprint arXiv:2405.17247.

Parameter-Efficient Fine-Tuning

Parameter-Efficient Fine-Tuning (PEFT) methods are designed to address the high computational cost associated with fine-tuning large-scale Vision-Language Models (VLMs) by training only a subset of parameters rather than the entire model.

Source: Bordes, F., Pang, R. Y., Ajay, A., Li, A. C., Bardes, A., Petryk, S., ... & Chandra, V. (2024). An introduction to vision-language modeling. arXiv preprint arXiv:2405.17247.

Parameter-Efficient Fine-Tuning

- Low Rank Adapters (LoRA) Based Methods:

LoRA: Adds trainable low-rank matrices to the pre-trained model, allowing efficient adaptation. Variants include:

QLoRA: Integrates LoRA with a quantized backbone, enabling gradient back-propagation through a frozen, quantized language model.

VeRA: Reduces the number of trainable parameters by sharing a single pair of low-rank matrices across all layers, maintaining performance while reducing complexity.

DoRA: Decomposes pre-trained weights into magnitude and direction, extending low-rank adaptation from language models to VLM benchmarks

Source: Bordes, F., Pang, R. Y., Ajay, A., Li, A. C., Bardes, A., Petryk, S., ... & Chandra, V. (2024). An introduction to vision-language modeling. arXiv preprint arXiv:2405.17247.

Low-Rank Adaptation (LoRA)

Freezes the pre-trained model weights and injects trainable rank decomposition matrices into each layer of the transformer architecture, greatly reducing the number of trainable parameters for downstream tasks.

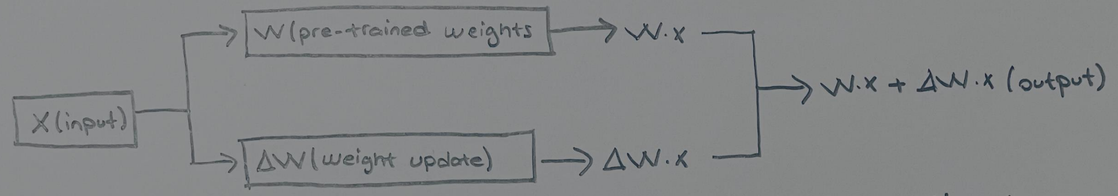
Decomposition of ΔW → reduces the computational overhead

In traditional fine-tuning, we modify a pre-trained NN's weights (the original weight matrix) to adapt to a new task.

ΔW: The changes made to W during fine-tuning collectively

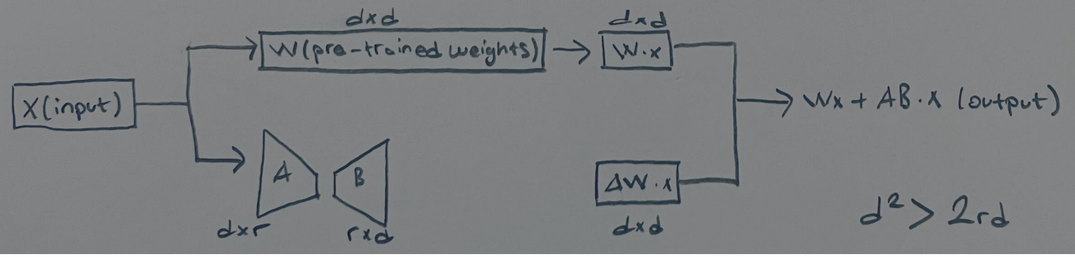
$W + \Delta W$: Updated weights

LoRA doesn't modify the weight matrix W directly, it seeks to decompose ΔW.



The intrinsic rank hypothesis suggests that significant changes to the NN can be capturet using a lower-dimension representation. Essentially, it posits that not all elements of ΔW are equally important; instead, a smaller subset of these changes can effectively encapsulate the necessary adjustments.

LoRA proposes representing ΔW as the product of two smaller matrices, A and B, with a lower rank. The updated weight matrix W' becomes: $W' = W + BA$. In this equation, W remains frozen (not updated during training), the matrices B and A are of lower dimensionality, with their product BA representing a low rank approximation of ΔW.



$d^2 > 2rd$

Source: https://medium.com/towards-data-science/understanding-lora-low-rank-adaptation-for-finetuning-large-models-936bce1a07c6

Parameter-Efficient Fine-Tuning

Prompt-Based Methods:

Context Optimization (CoOp): Optimizes context words in prompts as learnable vectors for downstream tasks, eliminating manual prompt engineering.

Visual Prompt Tuning (VPT): Introduces minimal trainable parameters in the input space while keeping the backbone frozen, achieving comparable accuracy to full fine-tuning

Source: Bordes, F., Pang, R. Y., Ajay, A., Li, A. C., Bardes, A., Petryk, S., ... & Chandra, V. (2024). An introduction to vision-language modeling. arXiv preprint arXiv:2405.17247.

Parameter-Efficient Fine-Tuning

Adapter-Based Methods:

CLIP-Adapter: Adds feature adapters to visual or language branches, using a bottleneck layer for new feature learning.

VL-adapter: Evaluates multi-task framework adapters, demonstrating competitive performance with minimal parameter updates.

Mapping-Based Methods:

Trains a mapping between modalities (e.g., text and image) without altering the main model architecture. This approach requires minimal parameter training

Source: Bordes, F., Pang, R. Y., Ajay, A., Li, A. C., Bardes, A., Petryk, S., ... & Chandra, V. (2024). An introduction to vision-language modeling. arXiv preprint arXiv:2405.17247.

Mixture of Experts (MoE)

Mixture of Experts (MoE) are designed to improve model performance and efficiency by combining specialized submodels ("experts") and a dynamic routing mechanism. Unlike traditional models that use the same parameters for all inputs, MoE allows different parts of the model to handle different inputs, enabling specialization and scalability.

An MoE model consists of two key parts:

Experts: Multiple neural networks (e.g., feed-forward layers) each specializing in specific input patterns.

Gating Network: A mechanism that determines which experts to use for a given input. It outputs weights indicating each expert's relevance.

Example: Imagine a team of doctors (experts) where a coordinator (gating network) assigns patients (inputs) to the best-suited specialists.

Mixture of Experts (MoE) - How MoE Works?

Step 1: Input Processing

The input (e.g., a sentence or image) is fed into both the gating network and all experts.

Step 2: Gating Decision

The gating network computes scores for each expert, often selecting the top-k experts (e.g., top 2).

These scores are converted into probabilities via softmax.

Step 3: Expert Processing

Selected experts process the input independently.

Step 4: Output Combination

The final output is a weighted sum of the experts' outputs, using the gating network's probabilities as weights.

Mixture of Experts (MoE) - How MoE Works?

$$\text{Output} = \sum_{i=1}^{n} G_i(x) \cdot E_i(x)$$

Where $G_i(x)$ is the gating weight for expert i, and $E_i(x)$ is the expert's output.

Mixture of Experts (MoE)

Sparse vs. Dense MoE

Sparse MoE: Activates only a subset of experts per input (e.g., top-2), reducing computation.

Dense MoE: Uses all experts for every input, which is computationally expensive and rarely used.

Training MoE Models

Joint Training: Experts and the gating network are trained simultaneously.

Loss Function: Combines task-specific loss (e.g., cross-entropy) with auxiliary losses for load balancing.

Mixture of Experts (MoE)

Challenges:

Expert Imbalance: Some experts may be underused. Solutions include adding a diversity loss or capping expert capacity.

Gradient Flow: The gating network uses soft weights to ensure differentiability during backpropagation.

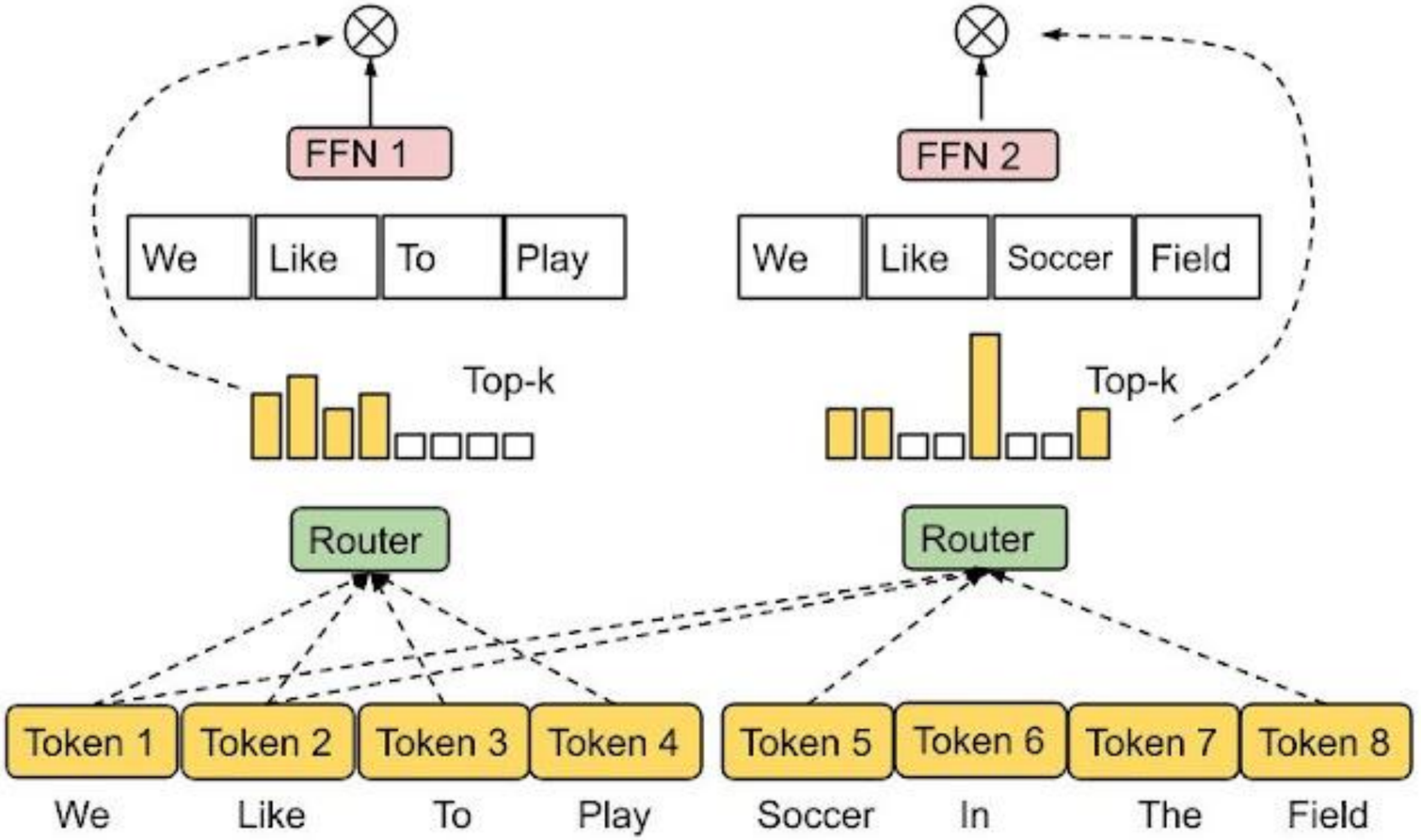
Mixture of Experts (MoE)

Scalability: MoE scales model capacity without proportional computational increase. For example, Google's Switch Transformer uses MoE in language models.

Memory vs. Computation: MoE models require more memory (storing all experts) but use less computation per input.

Dynamic Routing: The gating network learns to route inputs dynamically during training, unlike static rules.

Mixture of Experts (MoE)



Source: https://research.google/blog/mixture-of-experts-with-expert-choice-routing/

Google Developer Experts

NLP Fundamentals

Important Note: These notes are from Andrew Ng's Coursera course, François Chollet's "Deep Learning with Python" book and various online resources.

https://github.com/ayyucedemirbas/VLM_presentation/tree/main/NLP_notes

Google Developer Experts

PaliGemma FineTuning Notebook

https://colab.research.google.com/drive/1aG0Lk_eBDTzELN8s8iVKdHeI05j2m_SM#scrollTo=zGLIp1Cx3_CX

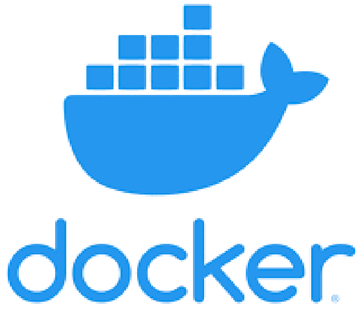
Google Developer Experts

Vision-Language Model Deployment

Flask

Streamlit

gradio



kubernetes

Google Cloud

Amazon SageMaker

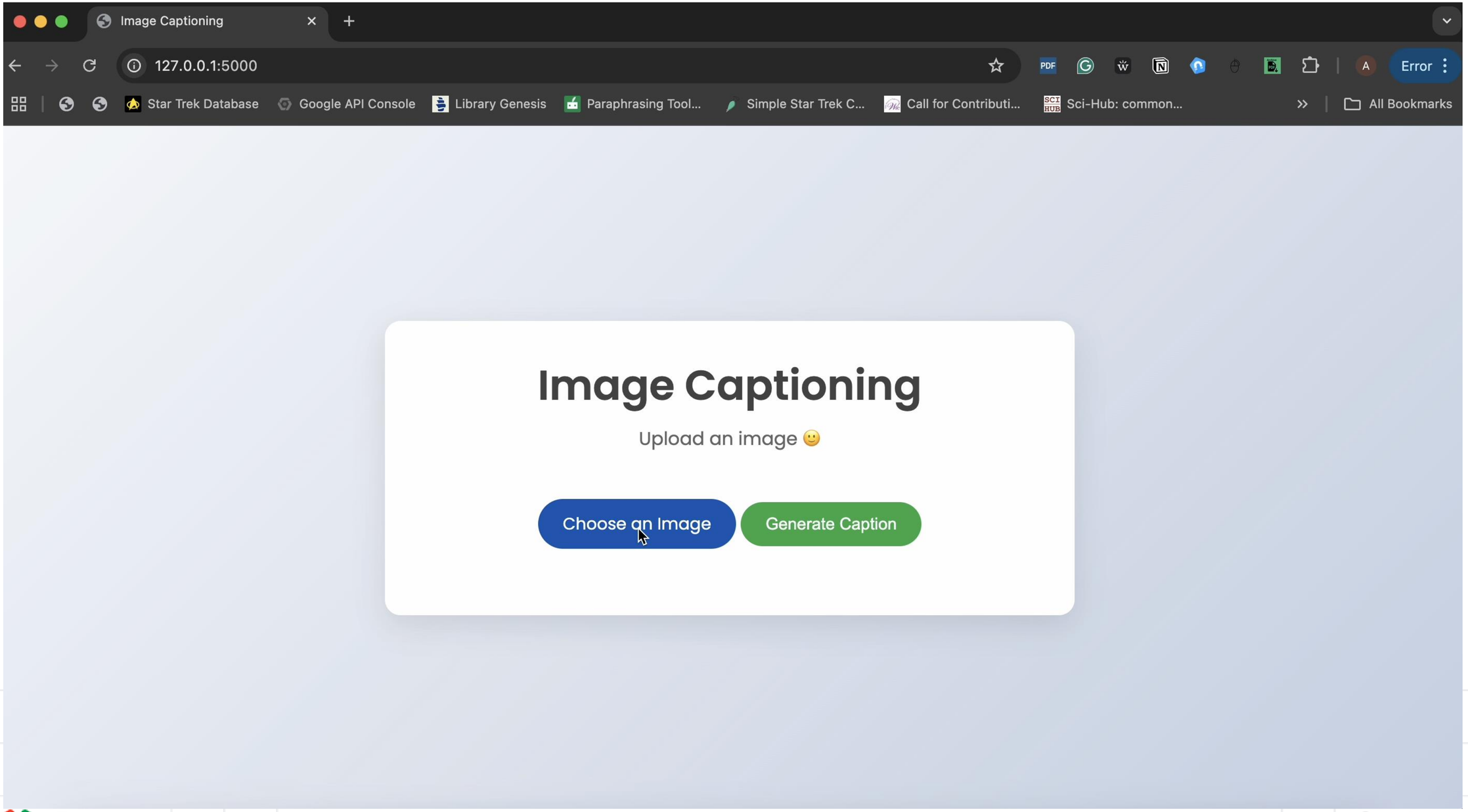


Image Captioning
127.0.0.1:5000
Error
Star Trek Database
Google API Console
Library Genesis
Paraphrasing Tool...
Simple Star Trek C...
Call for Contributi...
Sci-Hub: common...
All Bookmarks
Image Captioning
Upload an image 🙂
Choose an Image
Generate Caption

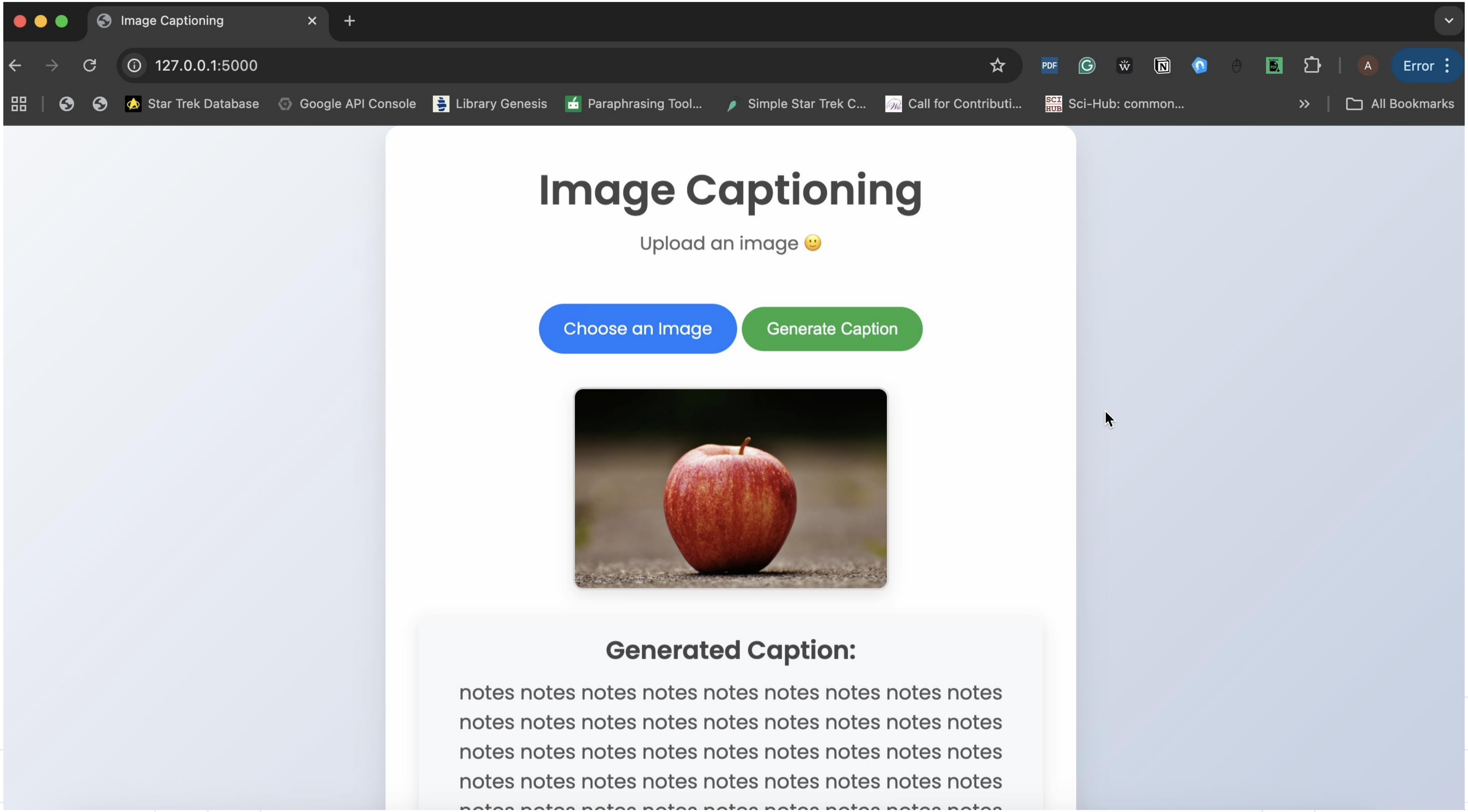
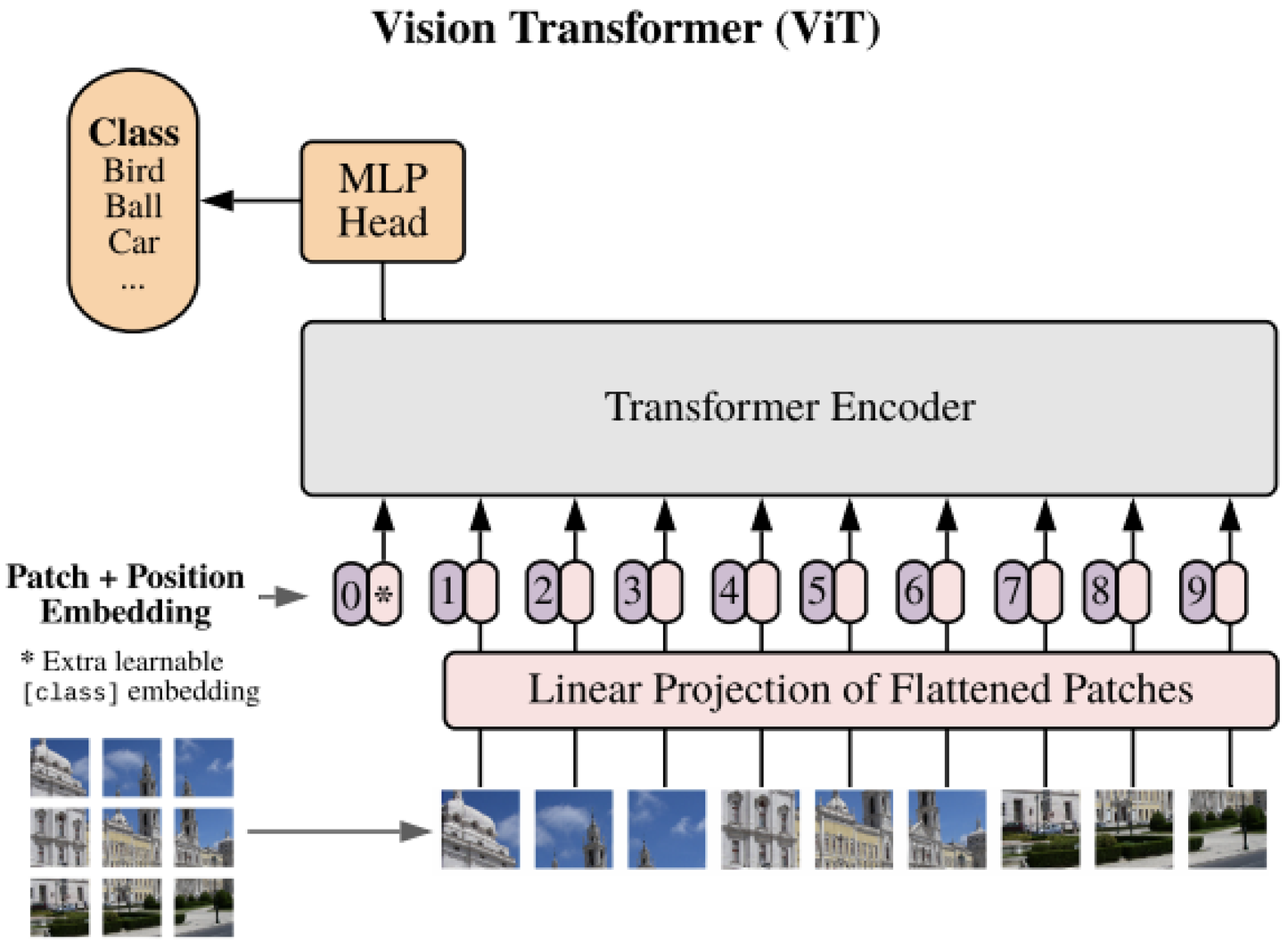


Image Captioning
127.0.0.1:5000
Star Trek Database
Google API Console
Library Genesis
Paraphrasing Tool...
Simple Star Trek C...
Call for Contributi...
Sci-Hub: common...
All Bookmarks
Error
Image Captioning
Upload an image 🙂
Choose an Image
Generate Caption
Generated Caption:
notes notes notes notes notes notes notes notes notes
notes notes notes notes notes notes notes notes notes
notes notes notes notes notes notes notes notes notes
notes notes notes notes notes notes notes notes notes



Source: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" by Alexey Dosovitskiy et. al.

Vision Transformers (ViT)

Background:

The transformer architecture was introduced in the paper "Attention Is All You Need" by Vaswani et al. in 2017. It revolutionized natural language processing by leveraging the self-attention mechanism to capture long-range dependencies in sequences effectively.

Inspired by the success of transformers in language tasks, researchers explored the idea of applying transformers to image recognition tasks, leading to the development of Vision Transformers (ViT).

Vision Transformers have demonstrated state-of-the-art performance on various computer vision benchmarks, often outperforming traditional convolutional neural networks (CNNs) in large-scale image recognition tasks.

Vision Transformers (ViT)

Architecture:

The core building blocks of Vision Transformers are multi-layer transformers. A typical ViT consists of multiple transformer encoder layers stacked on top of each other.

In the ViT architecture, the input image is divided into fixed-size non-overlapping patches, and each patch is linearly embedded into a vector.

A learnable positional encoding is added to the patch embeddings, allowing the model to incorporate spatial information from the original image.

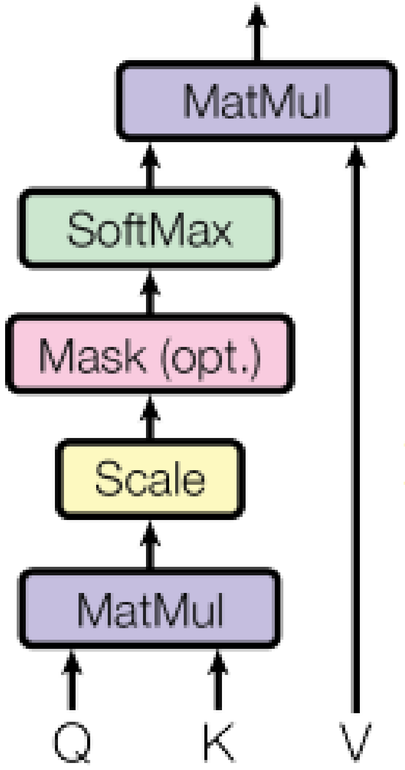
Vision Transformers (ViT)

Self-Attention Mechanism:

The self-attention mechanism is the key to the success of transformers. It allows the model to attend to all positions in a sequence (or image) to capture relationships between different elements effectively.

In Vision Transformers, self-attention is applied to both the patch embeddings and the positional embeddings, enabling the model to model dependencies between image patches and their spatial positions.

Vision Transformers (ViT)



$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$

$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Source: https://ai.stackexchange.com/questions/39151/attention-is-all-you-need-paper-how-are-the-q-k-v-values-calculated

Vision Transformers (ViT)

Patch-wise Processing:

By processing the image in patches, Vision Transformers can handle images of arbitrary sizes, providing a more flexible approach than traditional CNNs with fixed-size inputs.
The patch-wise processing also allows the model to capture global information in the image, avoiding the local receptive field limitations of CNNs.

Vision Transformers (ViT)

Training:

Vision Transformers are typically trained using supervised learning with large-scale image datasets, such as ImageNet or COCO.

The models are optimized using standard backpropagation techniques and a suitable loss function (e.g., cross-entropy loss for classification tasks).

Pretraining on a large dataset and fine-tuning on the target task are common practices to achieve state-of-the-art performance.

Vision Transformers (ViT)

Attention vs. Convolution:

Vision Transformers differ from traditional CNNs, which rely on convolutional layers for feature extraction. Attention mechanisms in transformers allow for more direct interactions between all image patches, capturing long-range dependencies without stacking multiple convolutional layers.

Vision Transformers (ViT)

Efficiency and Scalability:

Initially, transformers were computationally expensive, requiring large amounts of memory and computation due to their quadratic complexity.

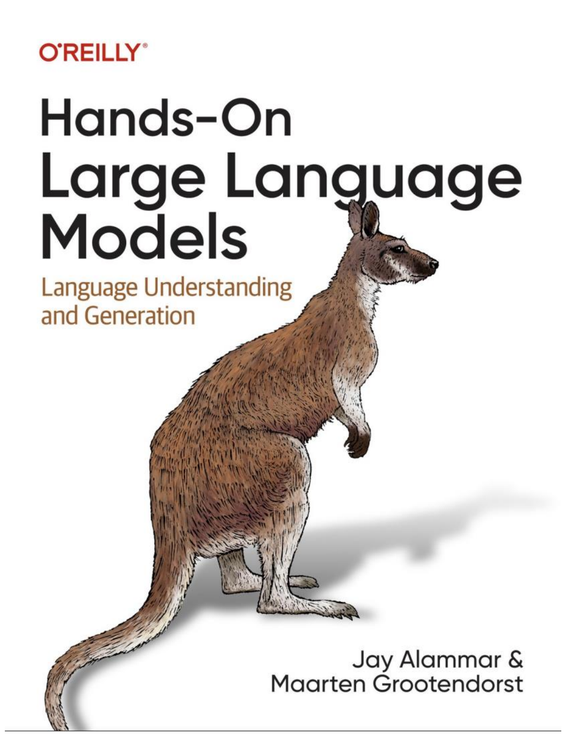
Various optimizations, such as sparse attention mechanisms and hybrid models (combining transformers with CNNs), have been proposed to make Vision Transformers more efficient and scalable for real-world applications.

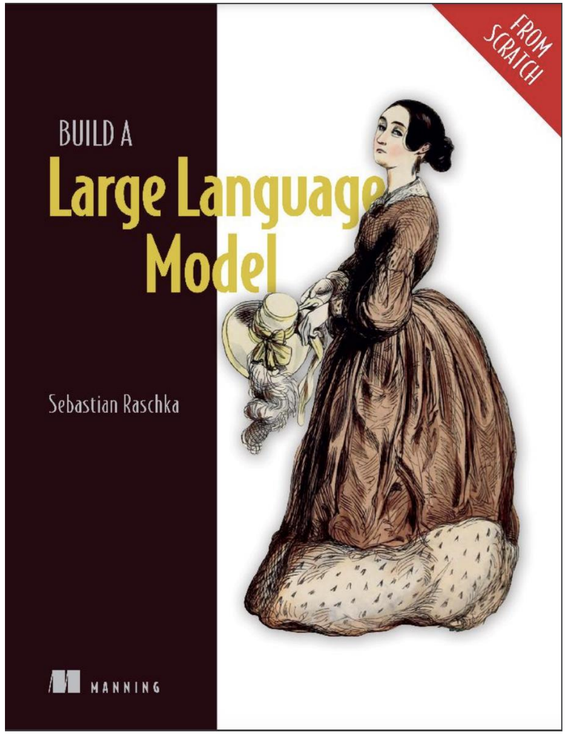
Vision Transformers (ViT)

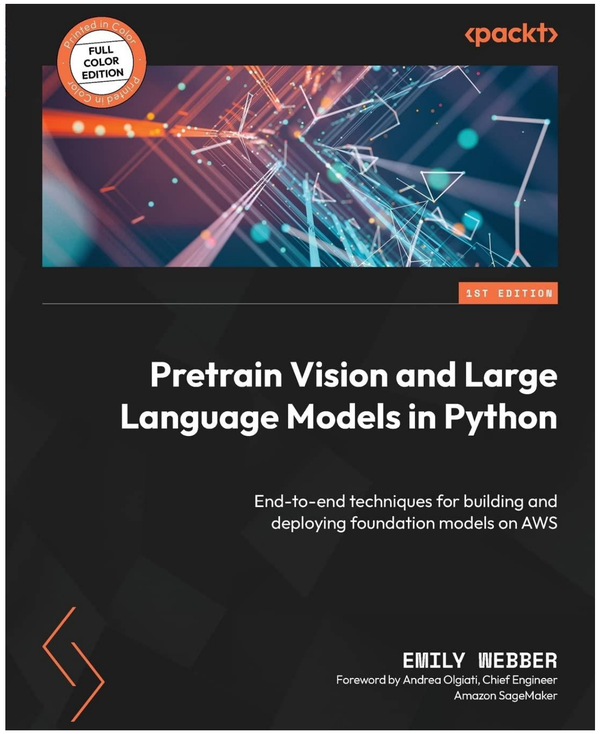
Applications:

Vision Transformers have demonstrated outstanding performance in image classification, object detection, image segmentation, and other computer vision tasks.

They are particularly effective for large-scale recognition tasks and are extensively used in pretraining models for transfer learning in computer vision.


O'REILLY
Hands-On
Large Language
Models
Language Understanding
and Generation
Jay Alammar &
Maarten Grootendorst


FROM SCRATCH
BUILD A
Large Language
Model
Sebastian Raschka
MANNING


Printed in Color
FULL COLOR EDITION
Printed in Black & White
<packt>
1ST EDITION
Pretrain Vision and Large Language Models in Python
End-to-end techniques for building and deploying foundation models on AWS
EMILY WEBBER
Foreword by Andrea Olgiati, Chief Engineer
Amazon SageMaker

Foundations of Large Language Models

Tong Xiao and Jingbo Zhu

January 17, 2025

https://arxiv.org/pdf/2501.09223

An Introduction to Vision-Language Modeling

Florian Bordes[*], Richard Yuanzhe Pang[*^], Anurag Ajay[**♣], Alexander C. Li[**♠], Adrien Bardes[*], Suzanne Petryk[△], Oscar Mañas[*‡], Zhiqiu Lin[♠], Anas Mahmoud[†], Bargav Jayaraman[*], Mark Ibrahim[*], Melissa Hall[*], Yunyang Xiong[*], Jonathan Lebensold[*♡], Candace Ross[*], Srihari Jayakumar[*], Chuan Guo[*], Diane Bouchacourt[*], Haider Al-Tahan[*], Karthik Padthe[*], Vasu Sharma[*], Hu Xu[*], Xiaoqing Ellen Tan[*], Megan Richards[*], Samuel Lavoie[*‡], Pietro Astolfi[*], Reyhane Askari Hemmat[*], Jun Chen[**◇], Kushal Tirumala[*], Rim Assouel[*‡], Mazda Moayeri[▽], Arjang Talattof[*], Kamalika Chaudhuri[*], Zechun Liu[*], Xilun Chen[*], Quentin Garrido[*], Karen Ullrich[*], Aishwarya Agrawal[‡•], Kate Saenko[*], Asli Celikyilmaz[*] and Vikas Chandra[*]

[*]Meta
[**]Work done while at Meta
[‡]Université de Montréal, Mila
[♡]McGill University, Mila
[†]University of Toronto
[♠]Carnegie Mellon University
[♣]Massachusetts Institute of Technology
[^]New York University
[△]University of California, Berkeley
[▽]University of Maryland
[◇]King Abdullah University of Science and Technology
[•]Canada CIFAR AI Chair
Core contributors, random ordering
Additional contributors, random ordering
Senior contributors, random ordering

https://arxiv.org/pdf/2405.17247


Google Scholar


arXiv

Experts

Thank You!

Ayyuce Demirbas
Google Developer Expert in ML