Introduction to Multimodal Deep Learning

Mustafa Hajij

What are multimodal models?

- A multimodal model is a model capable of processing and integrating multiple types of data (such as text, images, and audio) simultaneously to perform tasks more effectively.

Why Multimodal models?

- Enhanced Understanding:
  - By combining different types of data, multimodal models can achieve a deeper and more comprehensive understanding of complex information, similar to how humans process various sensory inputs.

- Improved Performance:
  - These models leverage the strengths of each data modality, leading to improved accuracy and performance in tasks like natural language processing, image recognition, and multimedia analysis.

- Versatile Applications:
  - Multimodal models are used in a wide range of applications, from virtual assistants that understand voice commands and visual cues to medical diagnostics that analyze patient data from various sources.

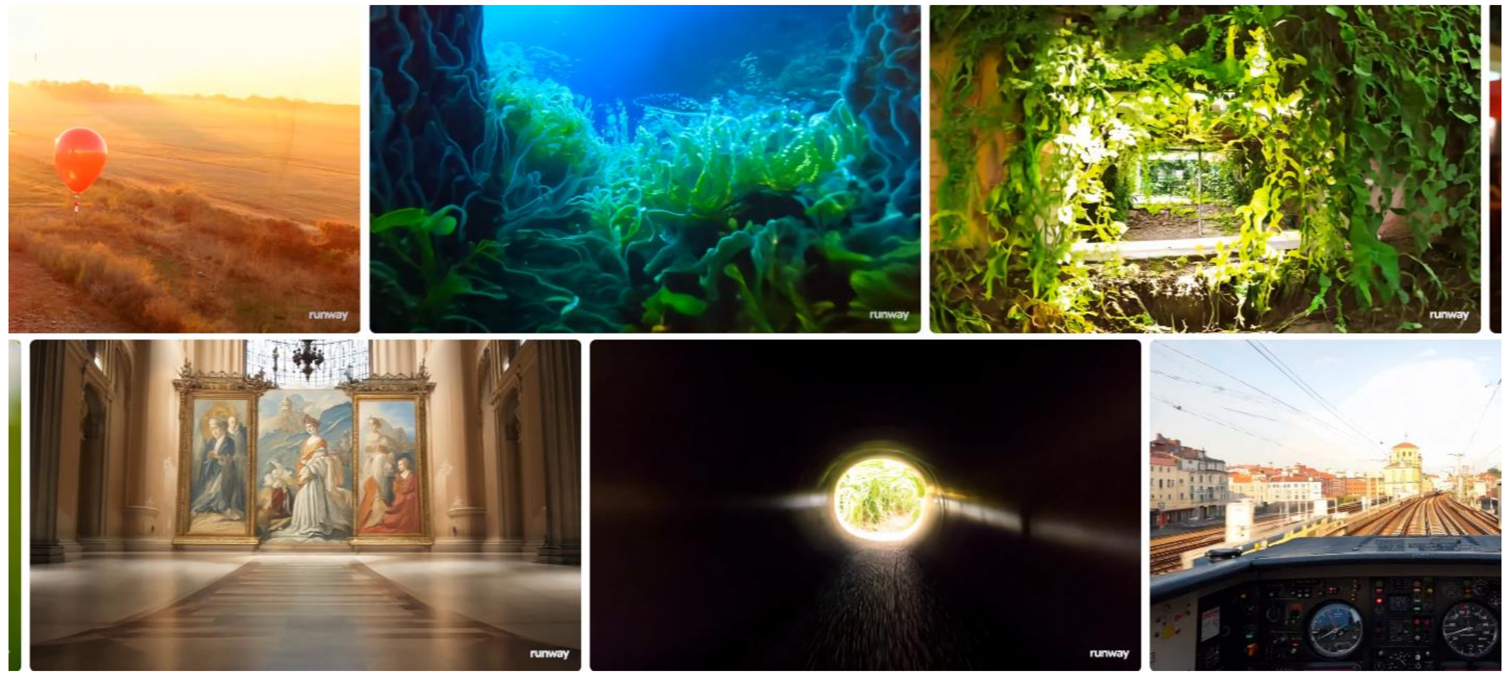
- **Data Availability :**
  - We are running out of high quality text data, and generally speaking more data is better.

Multimodal models is one of the main frontier of foundation models

Video Generation

[Introducing Gen-3 Alpha: A New Frontier for Video Generation (runwayml.com)](runwayml.com)

Prompt: FPV flying through a colorful coral lined streets of an underwater suburban neighborhood.



Gen-3 Alpha examples

Image Generation

GenType (labs.google)

Combining different modality

The process of combining different modalities to enable a model to learn from them is called multimodal fusion. Models that utilize multimodal fusion are referred to as multimodal models.

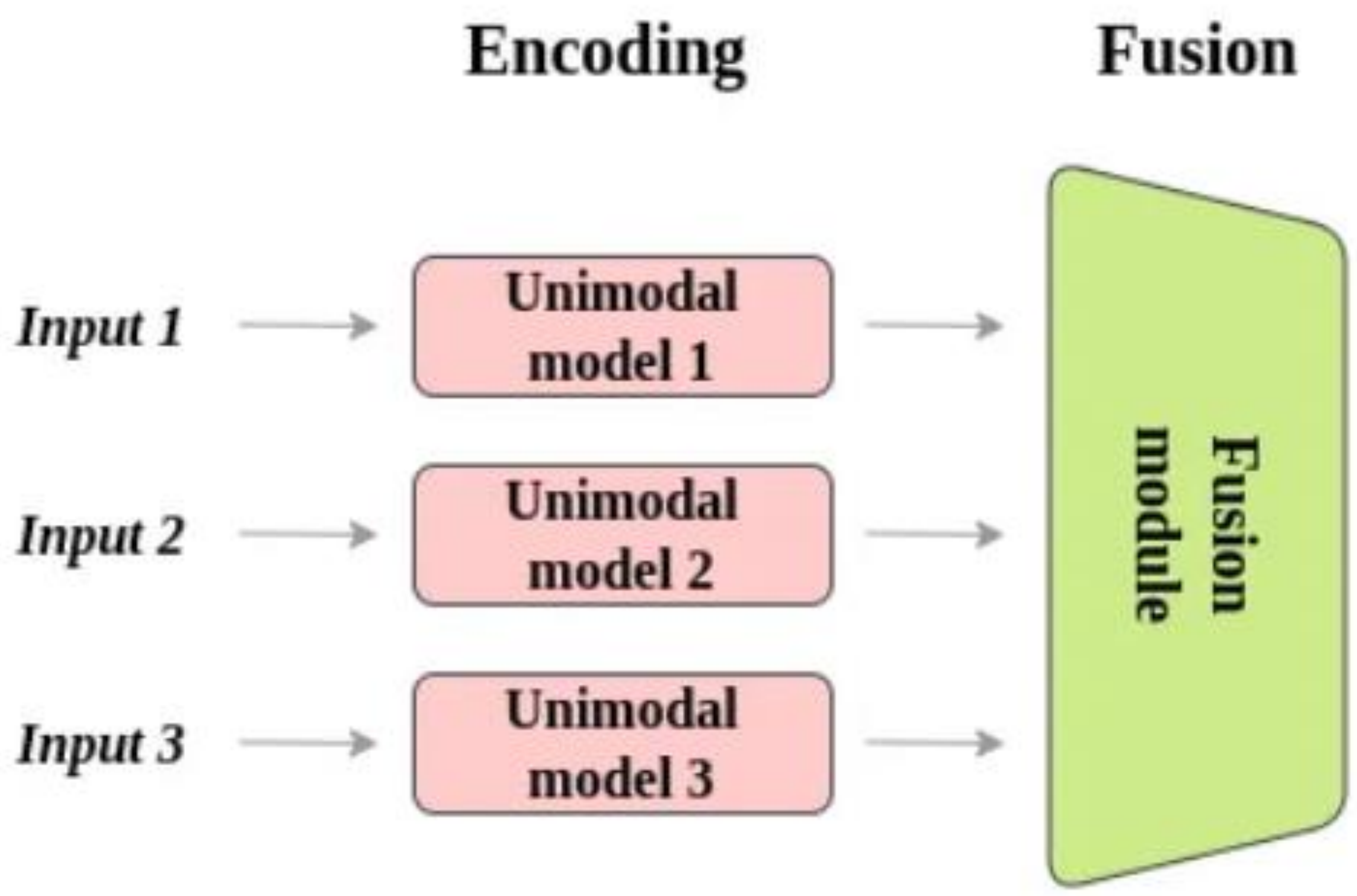


Image source

Multimodal Fusion

Similarity

- Inner product: $\mathbf{uv}$

Linear / sum

- Concat: $W[\mathbf{u},\mathbf{v}]$
- Sum: $W\mathbf{u}+V\mathbf{v}$
- Max: $\max(W\mathbf{u}, V\mathbf{v})$

Multiplicative

- Multiplicative: $W\mathbf{u}\odot V\mathbf{v}$
- Gating: $\sigma(W\mathbf{u})\odot V\mathbf{v}$
- LSTM-style: $\tanh(W\mathbf{u})\odot V\mathbf{v}$

Attention

- Attention: $\alpha W\mathbf{u}+\beta V\mathbf{v}$
- Modulation: $[\alpha\mathbf{u},(1-\alpha)\mathbf{v}]$

Bilinear

- Bilinear: $\mathbf{u}W\mathbf{v}$
- Bilinear gated: $\mathbf{u}W\boldsymbol{\sigma}(\mathbf{v})$
- Low-rank bilinear: $\mathbf{u}U^{T}V\mathbf{v}=P(U\mathbf{u}\odot V\mathbf{v})$
- Compact bilinear: $FFT^{-1}(FFT(\Psi(\mathbf{x},\mathbf{h}_1,\mathbf{s}_1))\odot FFT(\Psi(\mathbf{x},\mathbf{h}_2,\mathbf{s}_2)))$

Image source

How Does Multimodal Learning Work?

In general, multimodal architectures consist of:
- Unimodal encoders for individual modalities.
- A fusion network to combine features from each modality.
- A decision network for the final task.

1. Unimodal Encoding:
- Separate neural networks (unimodal encoders) process each input modality independently.
- Example: An audiovisual model has one network for audio and another for visual data.

2. Fusion:
- The information extracted from each modality during encoding is integrated.
- Various fusion techniques are used, such as simple concatenation or attention mechanisms.
- Effective fusion is critical for the model's success.

3. Decision Network:
- A decision network processes the fused encoded information.
- It is trained to perform the specific task at hand.

CLIP: Connecting text and images

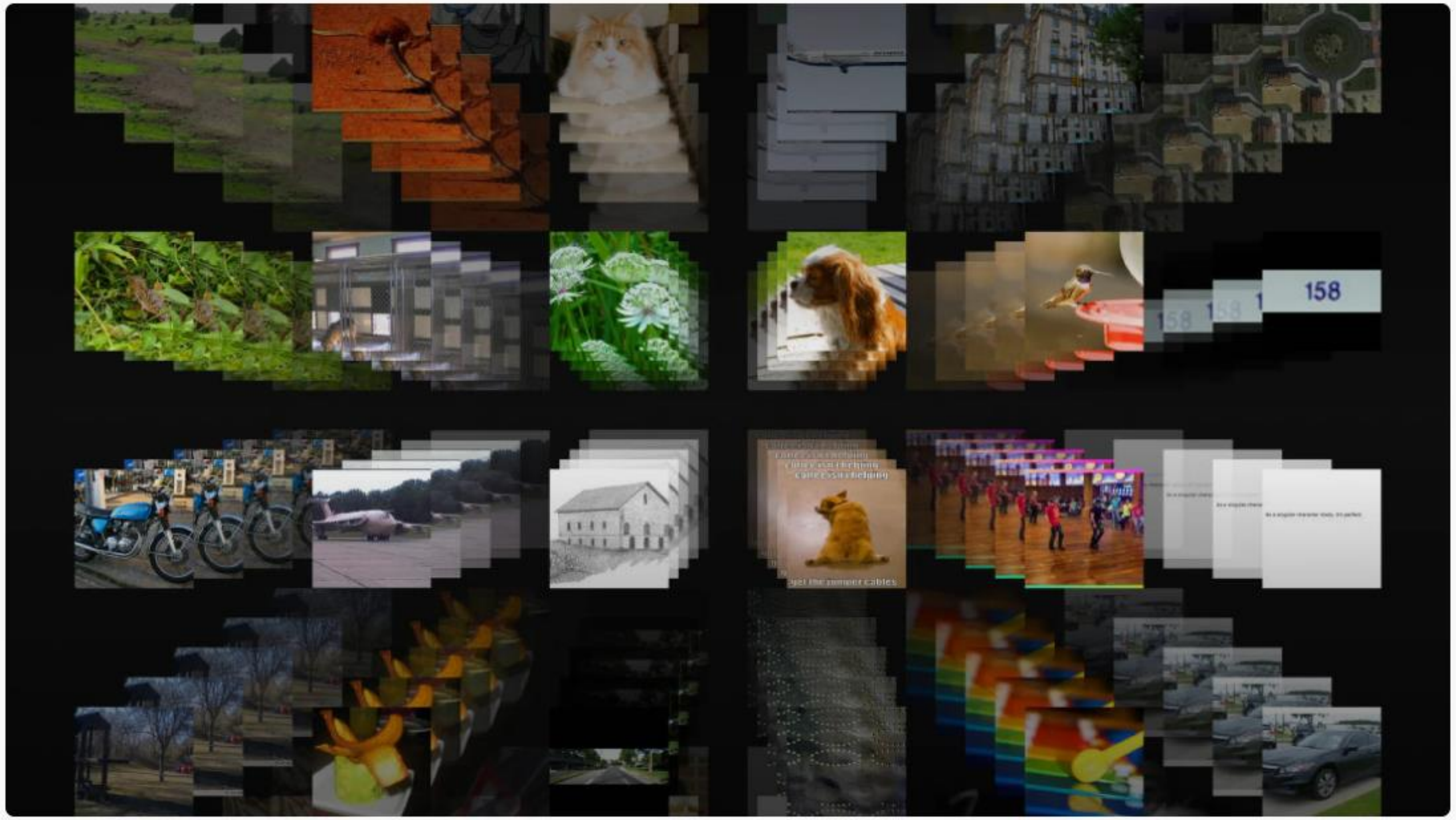


Illustration: Justin Jay Wang

Github openAI open source : openai/CLIP: CLIP (Contrastive Language-Image Pretraining), Predict the most relevant text snippet given an image (github.com)

mlfoundations/open_clip: An open source implementation of CLIP. (github.com)

Image source

CLIP: Connecting text and images

- CLIP is a neural network model designed to learn visual representations from textual descriptions.
- It achieves this by jointly training on a large dataset of images paired with corresponding textual descriptions.
- The core idea is to use contrastive learning, where the model maximizes the similarity between the correct image-text pairs while minimizing it for incorrect pairs.
- This training strategy enables CLIP to perform a variety of tasks such as image classification, zero-shot learning, and image-text retrieval without requiring task-specific fine-tuning.

Image source

CLIP: Connecting text and images

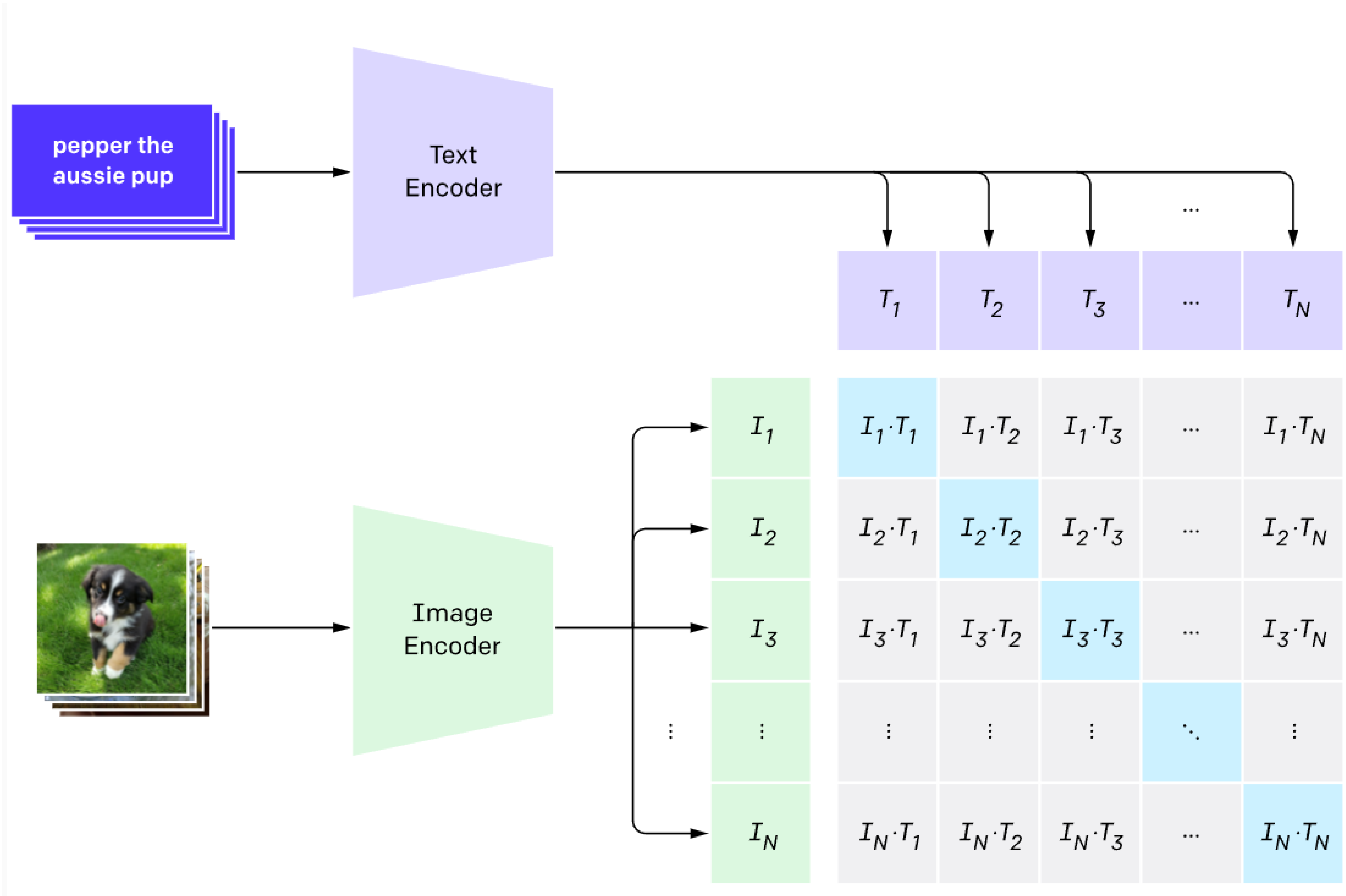


Image source

CLIP: Connecting text and images

Key Components:
1. **Contrastive Learning:** CLIP uses contrastive learning to align visual and textual representations in a shared embedding space. This involves training the model to bring the embeddings of matching image-text pairs closer together while pushing apart the embeddings of non-matching pairs.

2. **Joint Training:** The model is trained on a diverse dataset containing images and their corresponding textual descriptions, which allows it to learn from the rich contextual information provided by natural language.

3. **Zero-Shot Learning:** One of CLIP's remarkable capabilities is zero-shot learning, where it can generalize to new tasks and datasets without additional training. By leveraging its broad understanding of visual and textual concepts, CLIP can classify images based on textual prompts it has never seen before.

4. **Versatility:** CLIP can be applied to various applications, including image classification, object detection, and image-to-text or text-to-image retrieval, making it a versatile tool in the field of AI.

Advantages:
- **No Task-Specific Training** CLIP can perform well on a variety of tasks without needing task-specific training data.
- Broad Understanding: By training on diverse image-text pairs from the internet, CLIP develops a broad understanding of visual and textual concepts.
- **Flexible Deployment:** CLIP's ability to handle different types of inputs and outputs makes it flexible for numerous applications in AI and machine learning.

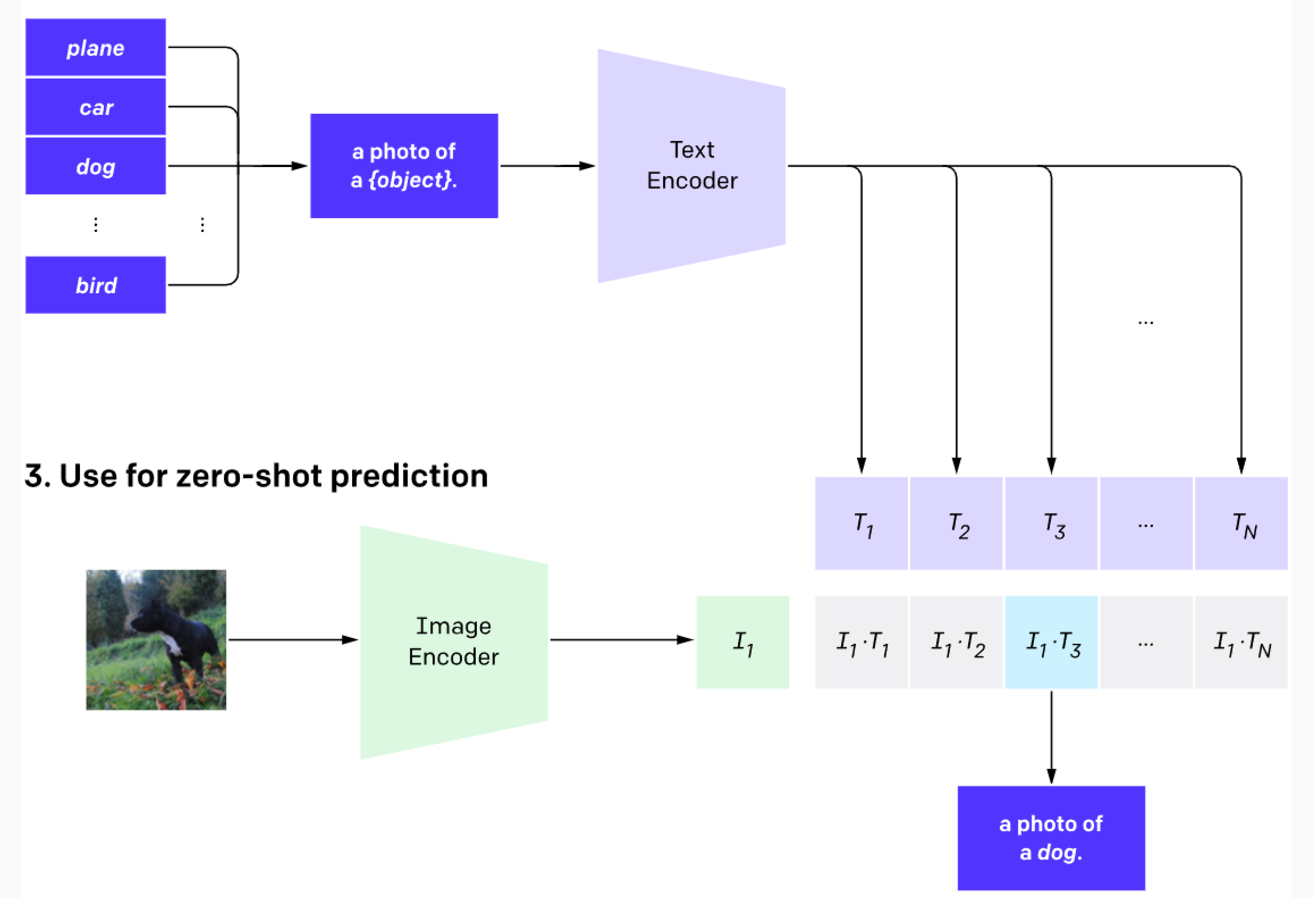
CLIP: Training Clip

```
import torch
from x_clip import CLIP

clip = CLIP(
    dim_text = 512,
    dim_image = 512,
    dim_latent = 512,
    num_text_tokens = 10000,
    text_enc_depth = 6,
    text_seq_len = 256,
    text_heads = 8,
    visual_enc_depth = 6,
    visual_image_size = 256,
    visual_patch_size = 32,
    visual_heads = 8,
    visual_patch_dropout = 0.5,             # patch dropout probability, used in Kaiming He's FLIP to s
    use_all_token_embeds = False,           # whether to use fine-grained contrastive learning (FILIP)
    decoupled_contrastive_learning = True,  # use decoupled contrastive learning (DCL) objective functi
    extra_latent_projection = True,         # whether to use separate projections for text-to-image vs
    use_visual_ssl = True,                  # whether to do self supervised learning on iages
    use_mlm = False,                        # use masked language learning (MLM) on text (DeCLIP)
    text_ssl_loss_weight = 0.05,            # weight for text MLM loss
    image_ssl_loss_weight = 0.05            # weight for image self-supervised learning loss
)

# mock data

text = torch.randint(0, 10000, (4, 256))
images = torch.randn(4, 3, 256, 256)

# train

loss = clip(
    text,
    images,
    freeze_image_encoder = False,   # whether to freeze image encoder if using a pretrained image net,
    return_loss = True              # needs to be set to True to return contrastive loss
)

loss.backward()
```

[Vision transformer : lucidrains/vit-pytorch: Implementation of Vision Transformer, a simple way to achieve SOTA in vision classification with only a single transformer encoder, in Pytorch (github.com)]

[Main ref: lucidrains/x-clip: A concise but complete implementation of CLIP with various experimental improvements from recent papers (github.com)]

Using Clip for zero-shot learning



Using Clip for zero-shot learning