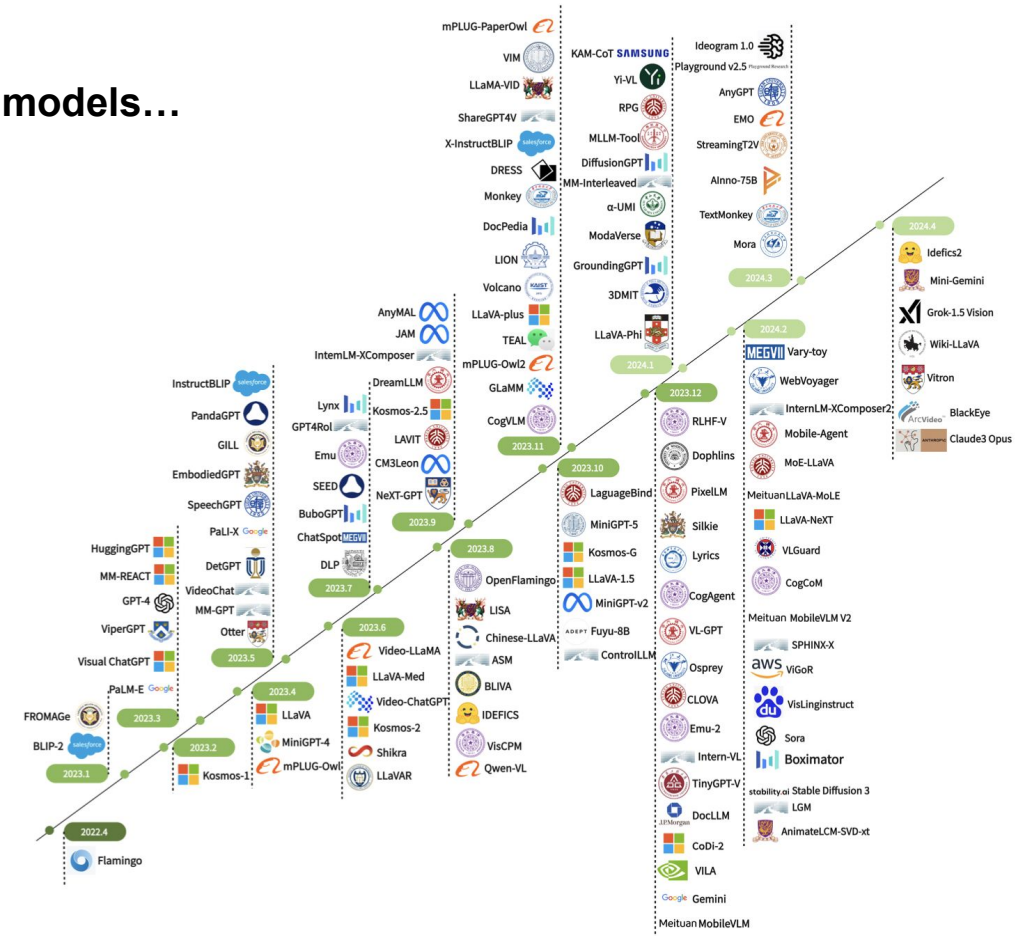
Multimodal Representation Learning: Foundations, Architectures, and Applications

Large number of models…



A Comprehensive Review of Multimodal Large Language Models: Performance and Challenges Across Different Tasks - https://arxiv.org/pdf/2408.01319

KNOWLEDGE CLUB

Agenda

1. **Welcome & Objectives**
   - ➔ Brief overview of goals and structure.
2. **Where we can use multimodality: typical tasks**
3. **Foundational Backbones**
   - ➔ **ViT, Swin Transformer** — transformer, but for the images (Vision Transformer).
   - ➔ **CLIP, SigLIP** — contrastive pre-training for image–text alignment.
4. **Fusion Architectures**
   - ➔ **BLIP-2 (Q-Former)** — Querying Transformer bridge.
   - ➔ **Flamingo** — Perceiver Resampler for flexible context windows.
   - ➔ **Fromage** — joint cross-modal embedding.
   - ➔ **GILL & LLAVA** — instruction-tuned VLMs.
   - ➔ **PaliGemma**
5. **How to train this models?**
6. **Benchmarks**

KNOWLEDGE CLUB

Multimodal tasks

Visual Question Answering and Visual Reasoning

Visual Question Answering
Q: What is the dog holding with its paws?
A: Frisbee.

Visual Reasoning
Q: Is the dog in the air **AND** is the frisbee in the air?
A: Yes

Image Captioning

Image Captioning (Paragraph)
Caption: There is a white dog lying on a grass field. There are a lot of leaves on the grass field. There is a chain-link fence next to the dog. There is a red frisbee under the dog's left-front paw.

Image Captioning (Single Sentence)
Caption: A dog tries to catch a yellow, flying frisbee.

Multimodal tasks

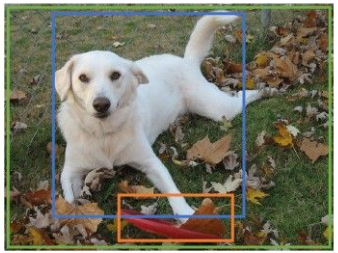
Image-Text Retrieval

Image-text Retrieval (Text-to-Image Retrieval)
Text Query: A dog lying on the grass next to a frisbee

Match

Not Match

Visual Grounding

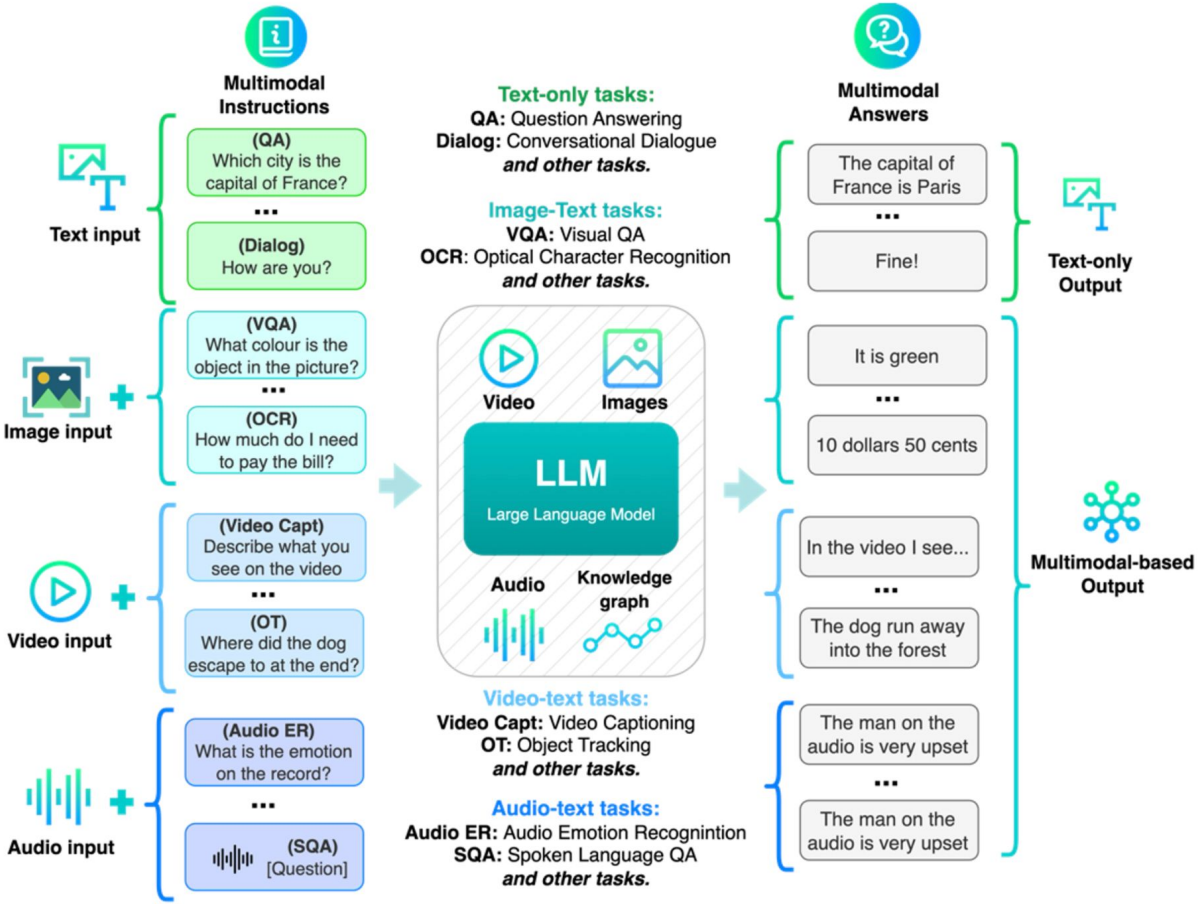


A dog is lying on the grass next to a frisbee.

(a) Phrase grounding.

The red frisbee next to the dog.

(b) Referring expression comprehension.



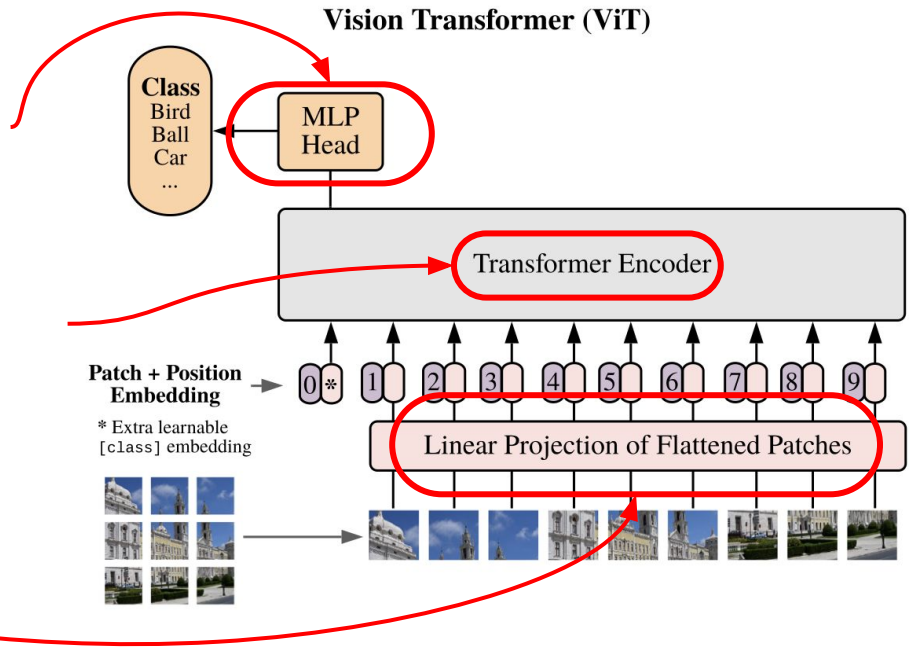
The image is taken from https://yatalks.yandex.ru/ru/hub/silnyj-ii--mif-ili-uzhe-realnost

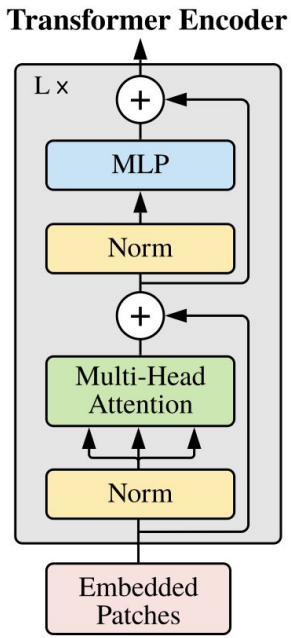
ViT: Vision Transformer

3) MLP Head — a fully connected classifier layer

2) Transformer Encoder — the transformer's encoder (only the encoder, no decoder!)

1) Linear Projection — convert the image into vectors for the transformer

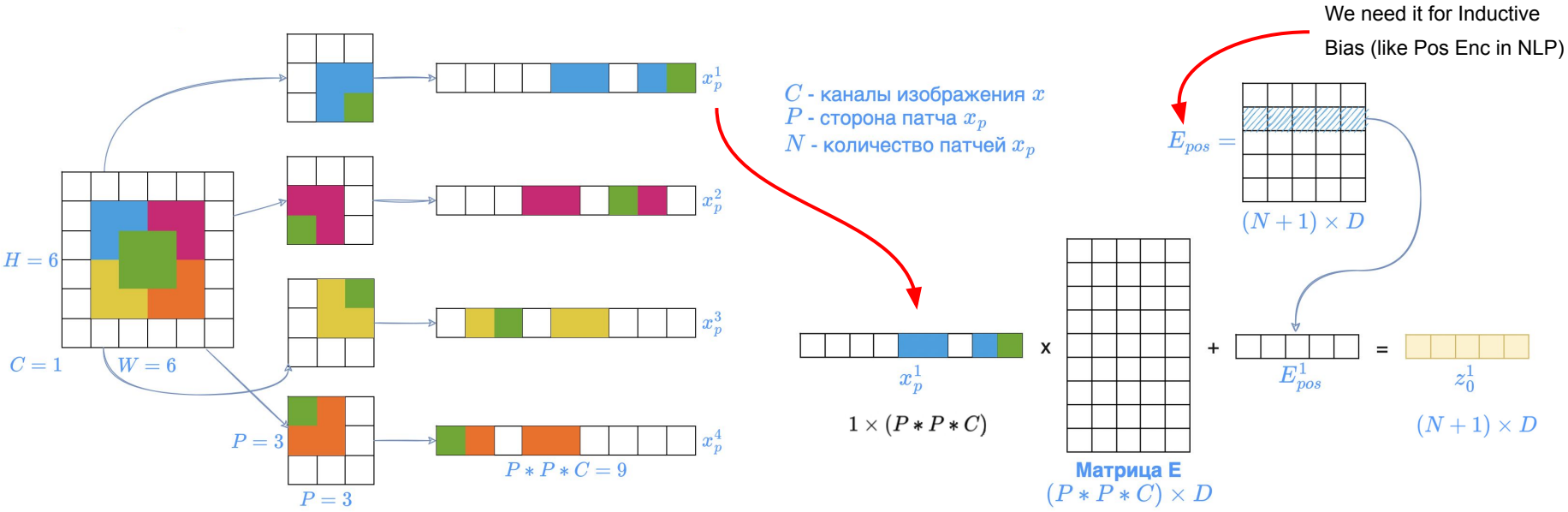




AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE, ICLR 2021 (https://arxiv.org/pdf/2010.11929)

ViT - Linear Projection

1. Cut the image of size (W × H × C) into patches of size (P × P × C).
2. Reshape each (P × P × C) patch into a vector of size (1 × (P · P · C)).
3. Project every (1 × (P · P · C)) vector to (1 × D) using a matrix E — one linear layer with no activation function.
4. Add the resulting (1 × D) vectors to the corresponding positional-embedding vectors from the matrix Epos.



The image is taken from https://blog.deepschool.ru/cv/vit-vision-transformer/ (An excellent explanation of the ViT model in Russian language — highly recommend)

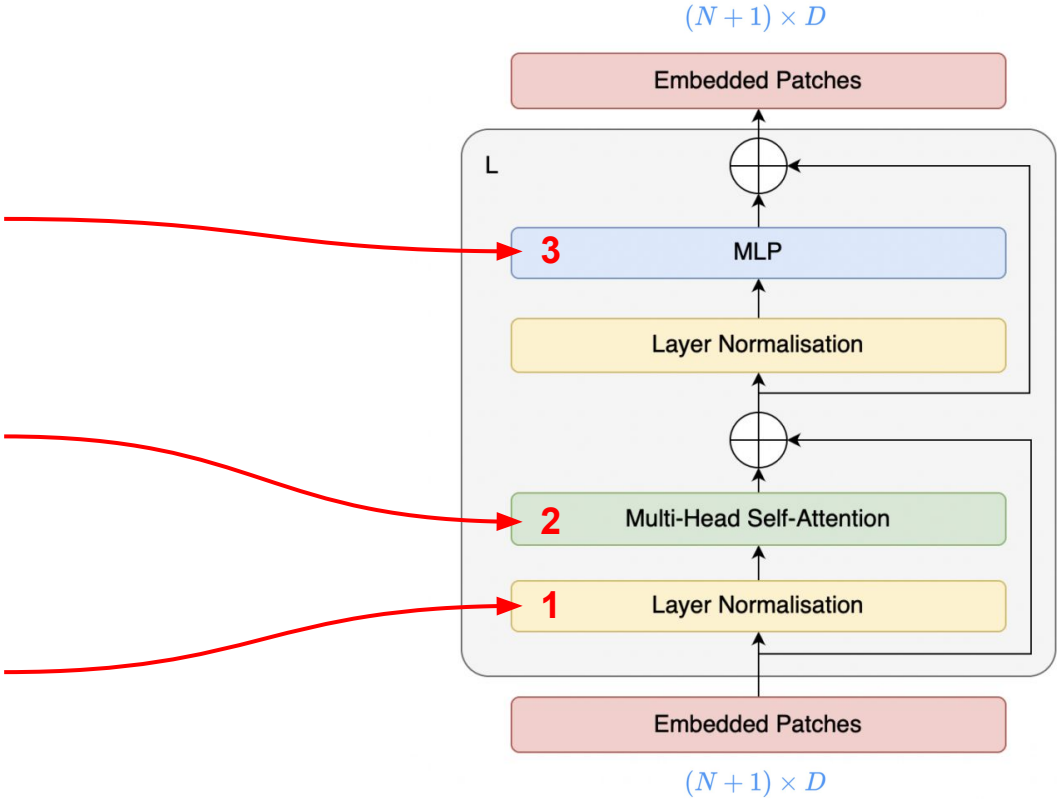
ViT - Transformer Encoder

The Input and Output is the same

Multi-Layer Perceptron (MLP) head
A small feed-forward network applied to each token independently—typically two linear layers with a non-linearity between

Multi-Head Self-Attention (MHSA)
For every token, computes attention scores to every other token

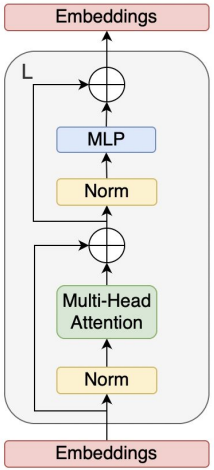
Layer Normalization (Layer Norm)
Normalises the activations along the feature dimension so that each token's vector has zero mean and unit variance (after scaling γ and shifting β, which are learned).



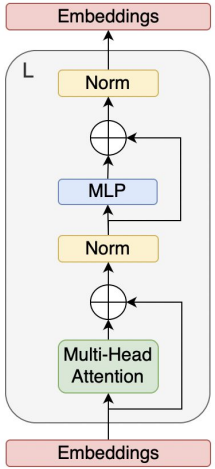
ViT - Transformer Encoder

Layer Normalization (Layer Norm)

This from ViT (Pre-LN)



This from "Attention is all you need" (Post-LN)



Mean

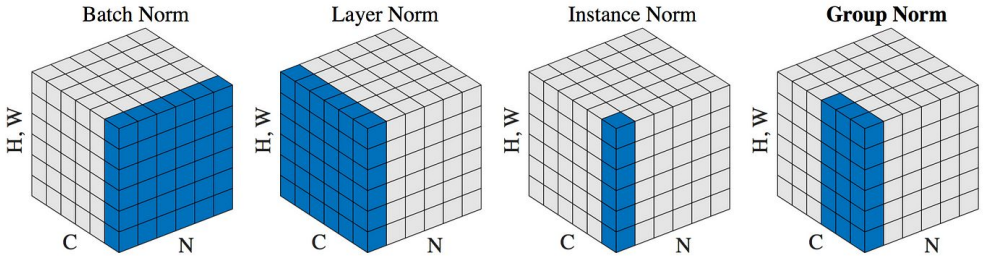
$$\mu = \frac{1}{d}\sum_{i=1}^{d} x_i$$

Variance

$$\sigma^2 = \frac{1}{d}\sum_{i=1}^{d}(x_i - \mu)^2$$

Normalization

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \qquad y_i = \gamma\hat{x}_i + \beta$$

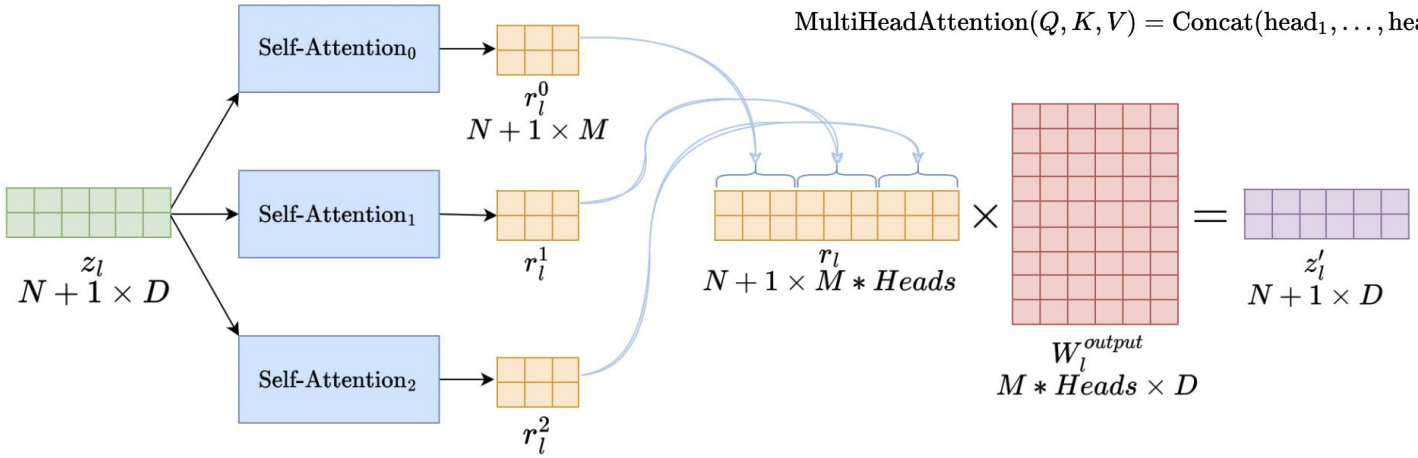


ViT - Transformer Encoder

Multi-Head Self-Attention (MHSA)

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^{\mathsf{T}}}{\sqrt{d_k}}\right)V,$$

$$\text{MultiHeadAttention}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O,$$



1) Multiple heads let the model focus on several parts of the input within the same layer.
2) Each head projects embeddings into its own sub-space, giving the attention block more variety. In other words, more heads ≈ more filters in a CNN: the more heads, the more diverse patterns the model can capture.

The image is taken from https://blog.deepschool.ru/cv/vit-vision-transformer/

ViT: Vision Transformer

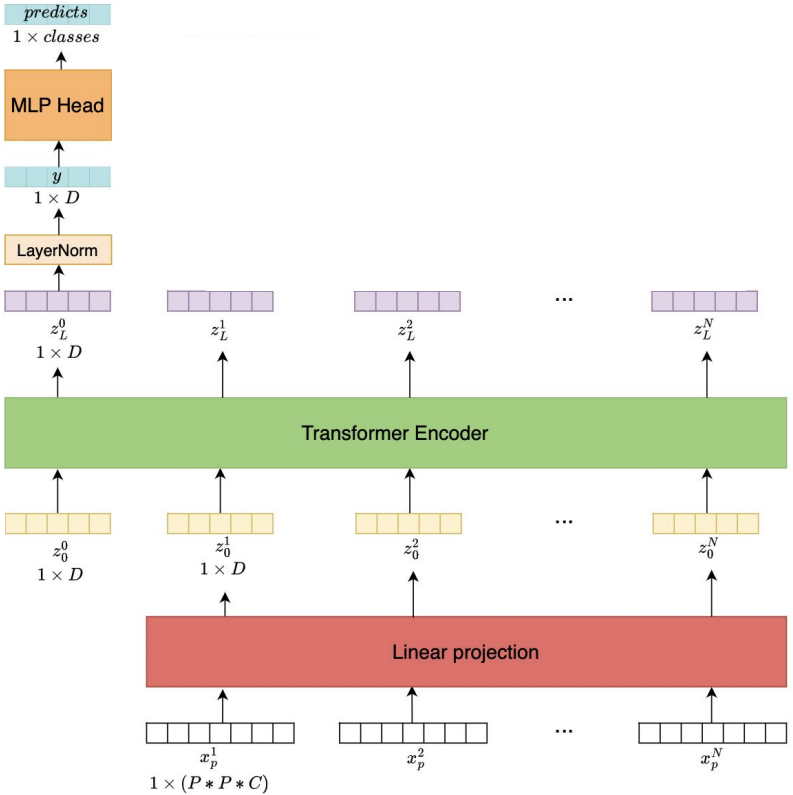
They trained every model using the Adam optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$) with a batch size of 4,096. L2 regularization was applied with a weight-decay coefficient of 0.1, and the learning rate was linearly warmed up and then decayed.

For fine-tuning, they switched to SGD with momentum and a batch size of 512.

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

Table 1: Details of Vision Transformer model variants.

https://huggingface.co/google/vit-base-patch16-224

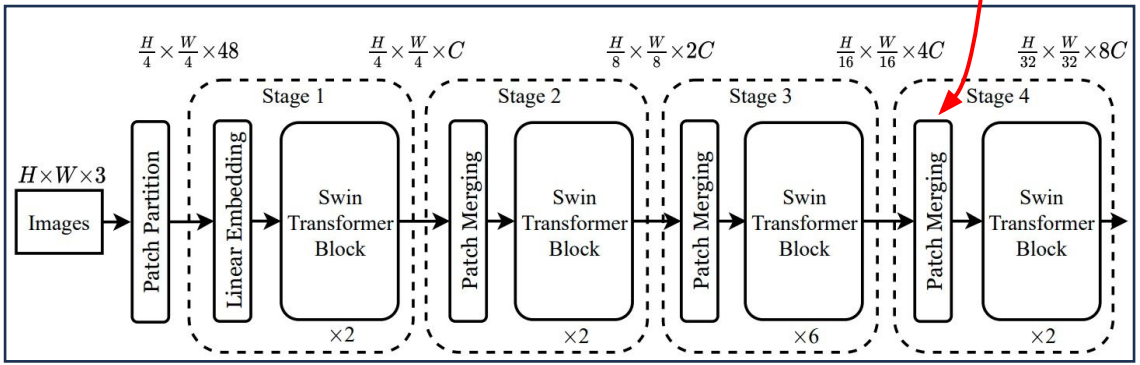


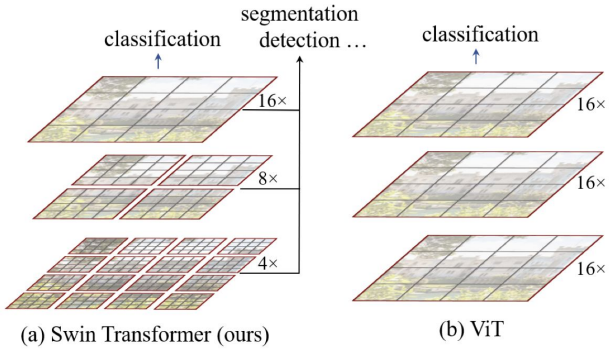
AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE (https://arxiv.org/pdf/2010.11929)

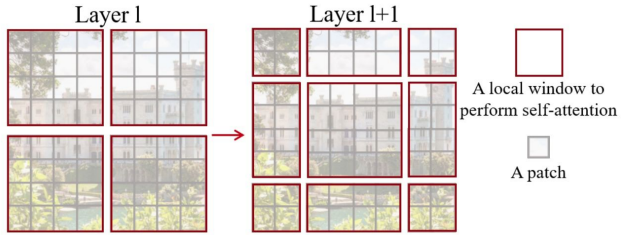
Swin Transformer

Patch Merging concatenates the features of adjacent tokens within a 2 × 2 window and then reduces the dimensionality, producing a higher-level representation.

As a result, after each stage we obtain feature maps that capture information at multiple spatial scales, giving the model a hierarchical view of the image.



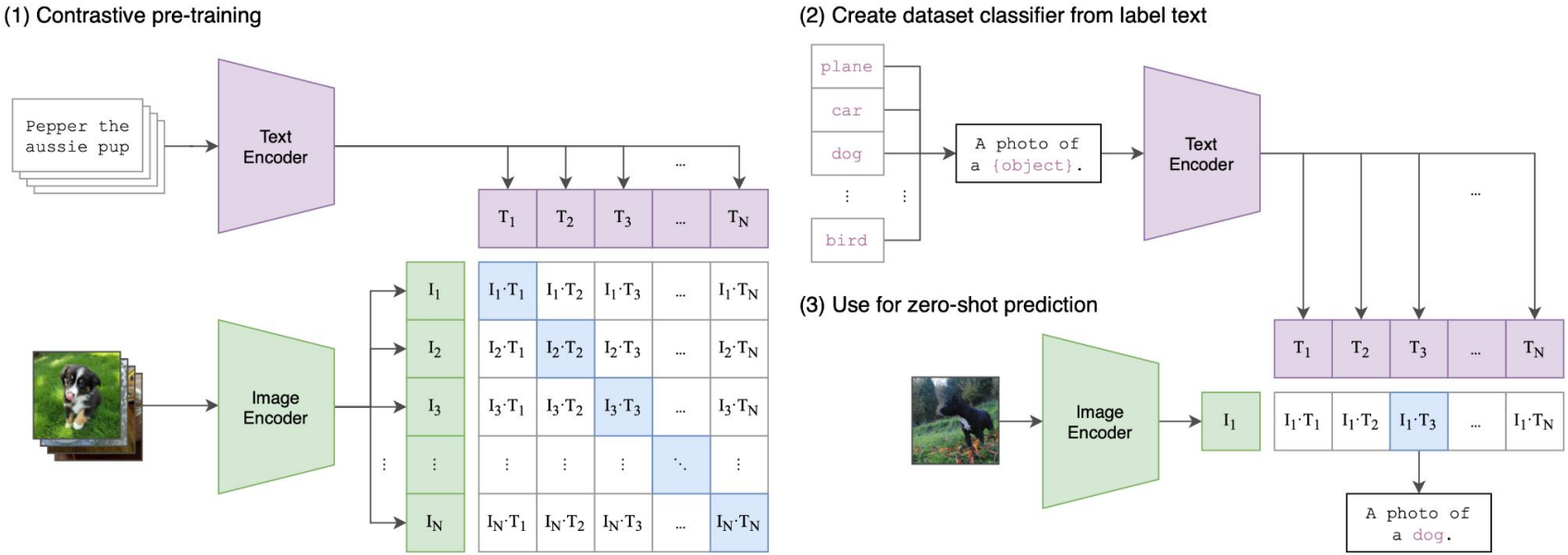




Swin Transformer: Hierarchical Vision Transformer using Shifted Windows (https://arxiv.org/pdf/2103.14030)

CLIP: Contrastive Language-Image Pre-Training

The model comprises a text encoder, an image encoder, and a projection head.
Text encoder: GPT-2, **Image encoder (two variants):** 1) modified ResNet 2) modified ViT



Learning Transferable Visual Models From Natural Language Supervision, OpenAI 2021 (https://arxiv.org/pdf/2103.00020)

Contrastive train: InfoNCE

Given a context vector $\mathbf{c}$, the positive sample should be drawn from the conditional distribution $p(\mathbf{x}|\mathbf{c})$, while $N-1$ negative samples are drawn from the proposal distribution $p(\mathbf{x})$, independent from the context $\mathbf{c}$. For brevity, let us label all the samples as $X = \{\mathbf{x}_i\}_{i=1}^N$ among which only one of them $\mathbf{x}_{\text{pos}}$ is a positive sample. The probability of we detecting the positive sample correctly is:

$$p(C=\text{pos}|X,\mathbf{c}) = \frac{p(x_{\text{pos}}|\mathbf{c})\prod_{i=1,\ldots,N;i\neq\text{pos}} p(\mathbf{x}_i)}{\sum_{j=1}^N \left[p(\mathbf{x}_j|\mathbf{c})\prod_{i=1,\ldots,N;i\neq j} p(\mathbf{x}_i)\right]} = \frac{\frac{p(\mathbf{x}_{\text{pos}}|c)}{p(\mathbf{x}_{\text{pos}})}}{\sum_{j=1}^N \frac{p(\mathbf{x}_j|\mathbf{c})}{p(\mathbf{x}_j)}} = \frac{f(\mathbf{x}_{\text{pos}},\mathbf{c})}{\sum_{j=1}^N f(\mathbf{x}_j,\mathbf{c})}$$

where the scoring function is $f(\mathbf{x},\mathbf{c}) \propto \frac{p(\mathbf{x}|\mathbf{c})}{p(\mathbf{x})}$.

The InfoNCE loss optimizes the negative log probability of classifying the positive sample correctly:

$$\mathcal{L}_{\text{InfoNCE}} = -\mathbb{E}\left[\log \frac{f(\mathbf{x},\mathbf{c})}{\sum_{\mathbf{x}'\in X} f(\mathbf{x}',\mathbf{c})}\right]$$

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

Contrastive train: InfoNCE

The symmetric contrastive InfoNCE loss used in CLIP

$$-\frac{1}{2|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}\left(\overbrace{\log\frac{e^{t\mathbf{x}_i\cdot\mathbf{y}_i}}{\sum_{j=1}^{|\mathcal{B}|}e^{t\mathbf{x}_i\cdot\mathbf{y}_j}}}^{\text{image}\to\text{text softmax}}+\overbrace{\log\frac{e^{t\mathbf{x}_i\cdot\mathbf{y}_i}}{\sum_{j=1}^{|\mathcal{B}|}e^{t\mathbf{x}_j\cdot\mathbf{y}_i}}}^{\text{text}\to\text{image softmax}}\right)$$

So, to compute the loss we:

1) Normalize the embeddings.

2) Form a similarity matrix by taking dot products of the embeddings and multiplying every entry by the temperature.

3) Apply cross-entropy over both the columns and the rows of that similarity matrix.

4) Add the two losses and divide the sum by 2.

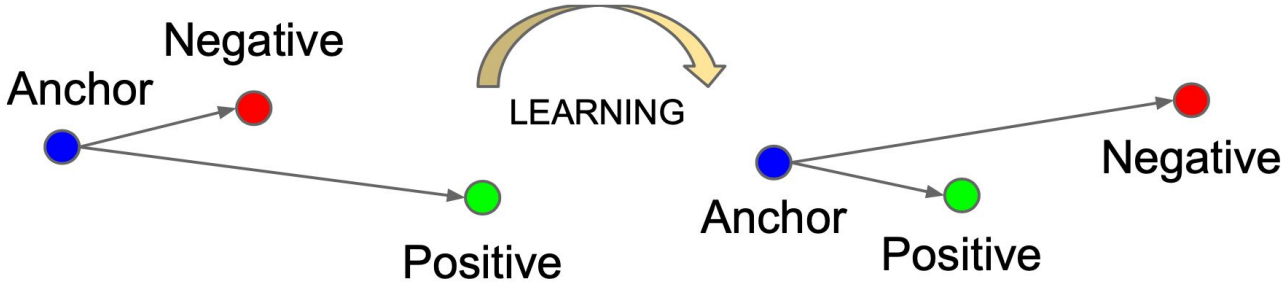
```
# image_encoder - ResNet or Vision Transformer
# text_encoder  - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = np.linalg.norm(np.dot(I_f, W_i), axis=1)
T_e = np.linalg.norm(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

Learning Transferable Visual Models From Natural Language Supervision (https://arxiv.org/pdf/2103.00020)

Contrastive train: Triplet Loss

Given one anchor input $\mathbf{x}$, we select one positive sample $\mathbf{x}^+$ and one negative $\mathbf{x}^-$, meaning that $\mathbf{x}^+$ and $\mathbf{x}$ belong to the same class and $\mathbf{x}^-$ is sampled from another different class. Triplet loss learns to minimize the distance between the anchor $\mathbf{x}$ and positive $\mathbf{x}^+$ and maximize the distance between the anchor $\mathbf{x}$ and negative $\mathbf{x}^-$ at the same time with the following equation:

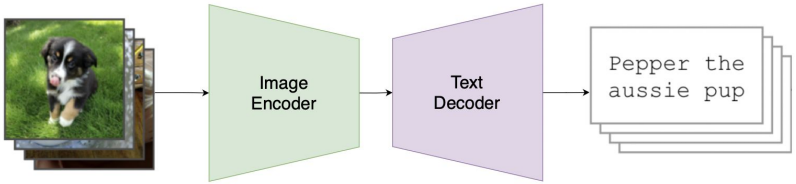
$$\mathcal{L}_{\text{triplet}}(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-) = \sum_{\mathbf{x} \in \mathcal{X}} \max\left(0, \|f(\mathbf{x}) - f(\mathbf{x}^+)\|_2^2 - \|f(\mathbf{x}) - f(\mathbf{x}^-)\|_2^2 + \epsilon\right)$$

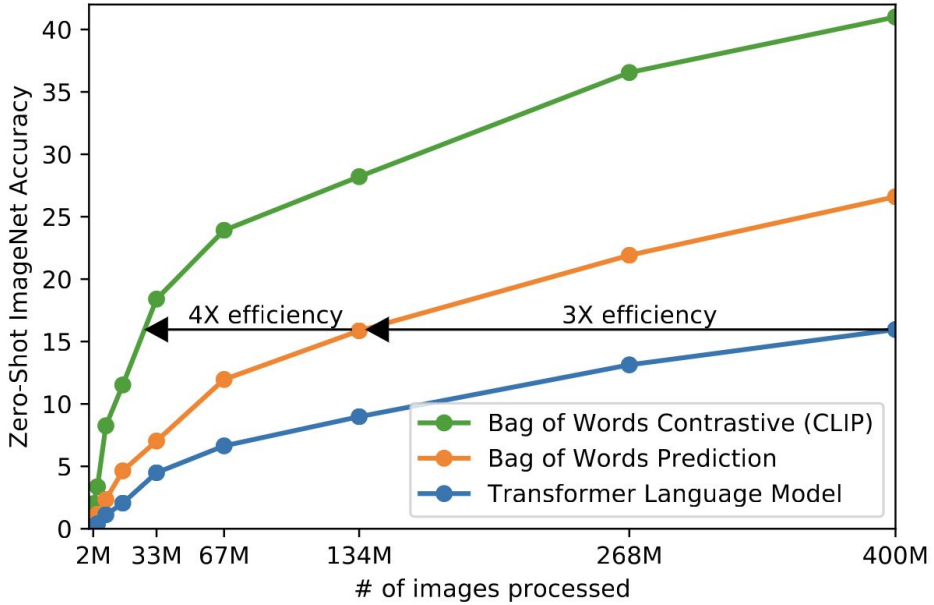


FaceNet: A Unified Embedding for Face Recognition and Clustering (https://arxiv.org/abs/1503.03832)

CLIP: Contrastive Language-Image Pre-Training

Why not train a single network that takes an image as input and produces text as output?



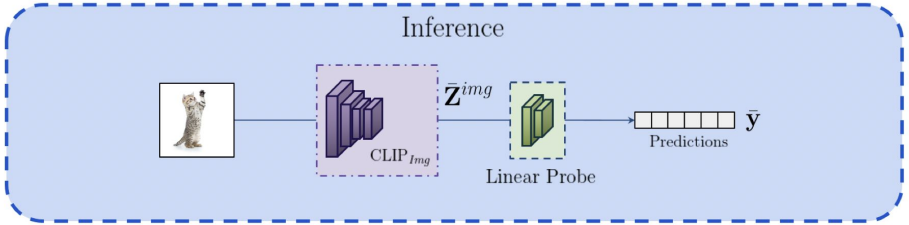


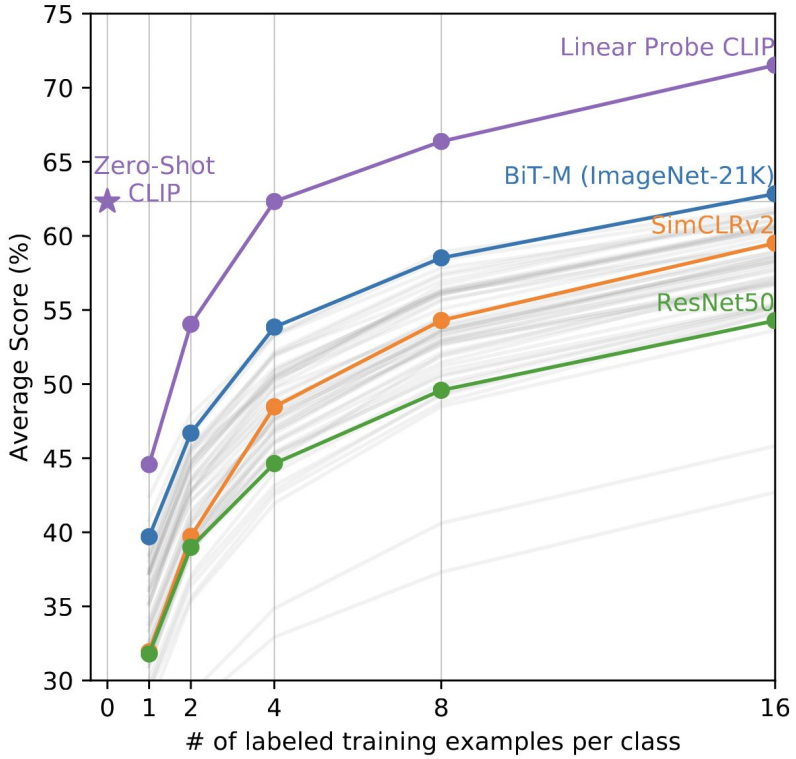
Learning Transferable Visual Models From Natural Language Supervision (https://arxiv.org/pdf/2103.00020)

Linear Probing

Для оценки добавляем линейный классификатор к выходам модели, замораживаем все веса модели и учим веса только этого линейного классификатора. Такой метод не требует большого количества данных для обучения.

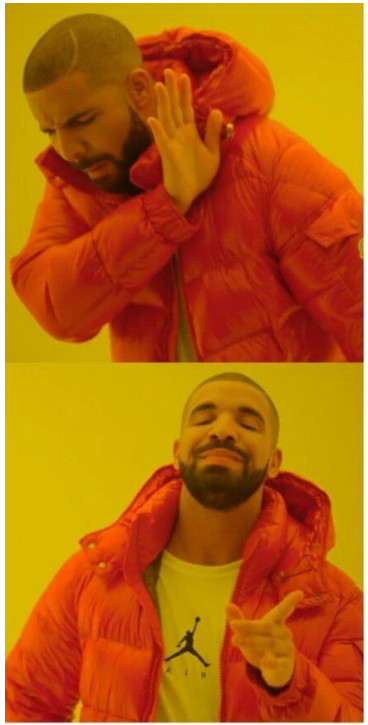
Можно ставить 1-shot learning, 2-shot learning, 3-shot learning и так далее. Разница в том сколько для обучения модели используется классов, например по 1, 2, 3 и так далее изображений на класс.





Learning Transferable Visual Models From Natural Language Supervision (https://arxiv.org/pdf/2103.00020)

SigLIP



CLIP: SoftMax

$$-\frac{1}{2|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}\left(\overbrace{\log\frac{e^{t\mathbf{x}_i\cdot\mathbf{y}_i}}{\sum_{j=1}^{|\mathcal{B}|}e^{t\mathbf{x}_i\cdot\mathbf{y}_j}}}^{\text{image}\to\text{text softmax}}+\overbrace{\log\frac{e^{t\mathbf{x}_i\cdot\mathbf{y}_i}}{\sum_{j=1}^{|\mathcal{B}|}e^{t\mathbf{x}_j\cdot\mathbf{y}_i}}}^{\text{text}\to\text{image softmax}}\right)$$

SigLIP: Sigmoid

$$-\frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}\sum_{j=1}^{|\mathcal{B}|}\underbrace{\log\frac{1}{1+e^{z_{ij}(-t\mathbf{x}_i\cdot\mathbf{y}_j+b)}}}_{\mathcal{L}_{ij}}$$

В отличие от CLIP, который использует softmax для нормализации косинусного сходства между парами изображений и текстов, SigLIP заменяет softmax на сигмоиду. Это позволяет убрать необходимость глобального нормирования (softmax) при вычислении потерь. Распределить вычисления между GPU без синхронизации на этапе подсчёта потерь.

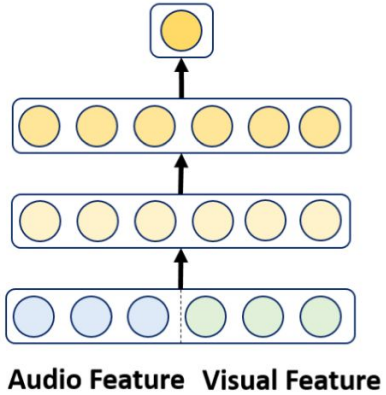
Ключевое отличие :

1) **CLIP:** Использует softmax для нормализации косинусного сходства. Это требует, чтобы все пары (изображения + тексты) находились на одном устройстве для вычисления вероятностей.
2) **SigLIP:** Использует сигмоиду для каждой пары независимо, что позволяет обрабатывать данные на разных устройствах параллельно.

Благодаря этому мы можем эффективно скейлить наши батчи до абсурдных размеров в 1млн пар!!! (Как говорится в статье)

Sigmoid Loss for Language Image Pre-Training (https://arxiv.org/abs/2303.15343)

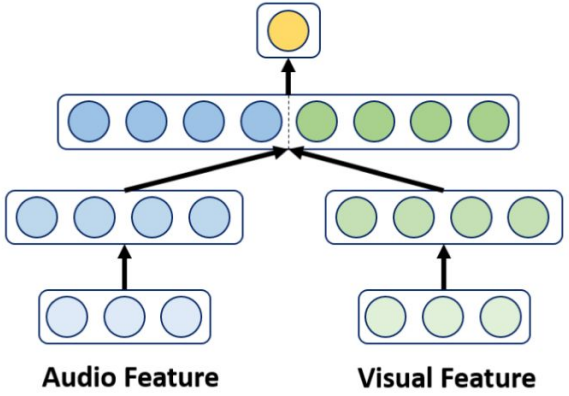
Fusion approaches



(a) Early Fusion

After feature extraction, embeddings are concatenated and treated as one big vector:

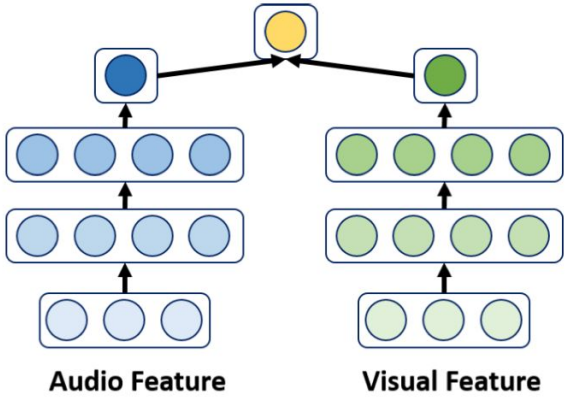
concat([x_text, x_visual]) → MLP/Transformer → softmax



(b) Model-level Fusion

Let each modality “mature” into high-level features, then merge inside the model for a few layers.

Cross-attention: Queries from text, Keys/Values from the image.



(c) Late Fusion

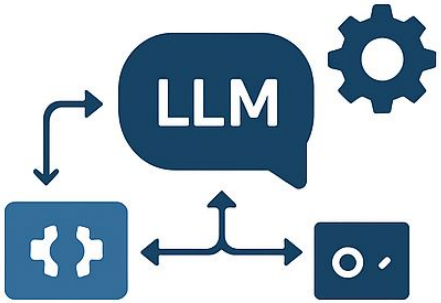
At the end, after separate classifiers. You mix the output distributions or logits

Mean or weighted mean of **logits** or probs, meta-classifier (stacking) on top of concatenated logits.

Multi-modal Conditional Attention Fusion for Dimensional Emotion Prediction (https://arxiv.org/pdf/1709.02251)

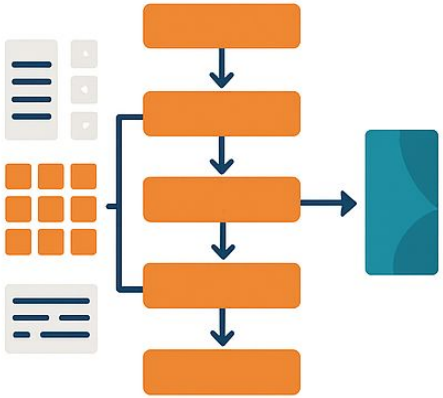
Architecture designs

Tool-Augmented LLM
(Orchestrator)



LLM calls CV APIs

End-to-End Multimoolal LLM



Jointly trained on mixed data

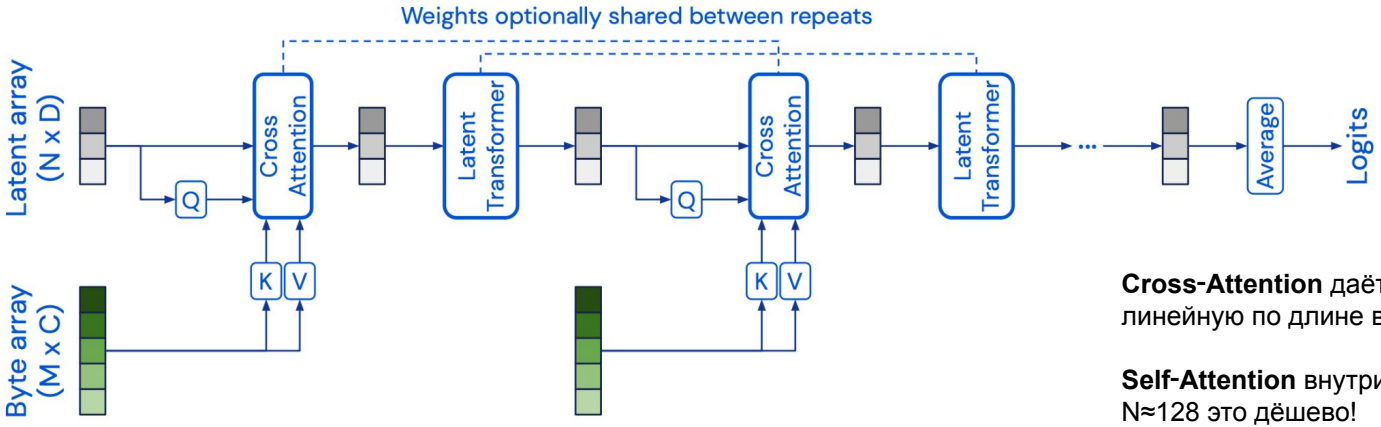
Modality Bridging with Pretrained Models

Align latent spaces

Perceiver: iterative CA

Latent Array (N × D) - Меньшее фиксированное хранилище (например, N = 64 или 256). Его векторы — *параметры*.
Byte array (M x C) - *Сырой длинный вход* — патчи картины 224×224 → 12 288 токенов.

1) **Iterative Cross-Attention -** На каждой итерации: **Q = латенты**, **K V = входные токены**. Каждый латент вытягивает нужные фичи из огромного входа. **Веса можно шэрить** между итерациями (эффективнее по памяти).
2) **Повтор L раз -** Схема Cross → Latent → Cross → … L циклов, пока латенты не соберут достаточно информации.
3) **После усредняем** латенты → MLP head → логиты



Cross-Attention даёт сложность **O(M · N)**, линейную по длине входа.

Self-Attention внутри латентов — **O(N²)**, но при N≈128 это дёшево!

Jaegle, Andrew, et al. "Perceiver: General perception with iterative attention." 2021 (DeepMind) - https://arxiv.org/pdf/2103.03206

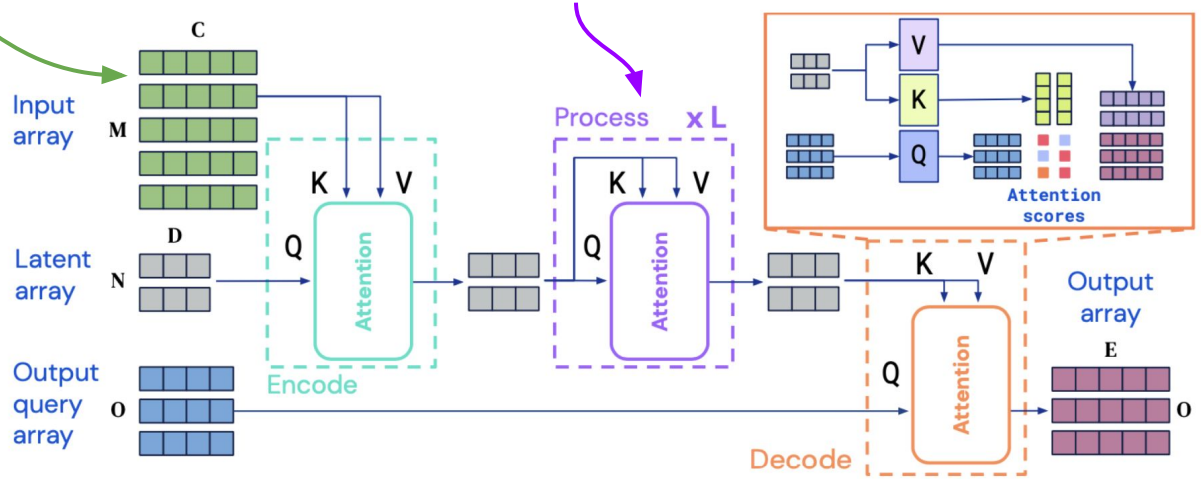
Perceiver IO: CA on input/output

Проблема: perceiver брал **огромный вход** $M \times C$, переводил его в **N** латентов через итеративный cross-attention и, в конце, **усреднял** латенты → MLP → логиты. Что отлично для **классификации**, но **не годится для произвольных выходов** — картинок, текста, множества символов.

Encode CA: Q = *latents*, $K\,V$ = *input* — вытягиваем данные (прям как в *perceiver)*

Process: обычный Transformer на латентах (N небольшой, квадратичность терпима).

Decode CA: Q = ***output-queries***, $K\,V$ = *latents*. *Число запросов O и их структура сами задают форму вывода.*



Сложность всё ещё $O(M \cdot N + N^2 + O \cdot N)$ ⇒ **линейна** по входу *и* выходу.

Jaegle, Andrew, et al. "Perceiver io: A general architecture for structured inputs & outputs." 2021 (DeepMind) - https://arxiv.org/pdf/2107.14795

Flamingo

- The first goal of the authors is to leverage pre-trained language models -> **Chinchillah by DeepMind.**

- On the vision side, **the authors pretrain a vision encoder with a contrastive text-image CLIP.** The role of this model is to extract rich semantic spatial features from the given images. They use NFNet (https://arxiv.org/abs/2102.06171)

- The second goal was to bridge these two models harmoniously, by **freezing the weights of these models** and link them via two learnable architectures.

- **The Perceiver Resampler** receives spatiotemporal features from the Vision Encoder and outputs a fixed-size set of visual tokens.

- The visual tokens are then used to condition the frozen LM using freshly initialised **cross-attention layers** that are interleaved between the pre-trained LM layers. These layers offer LM a way to incorporate visual information for the NTP task.

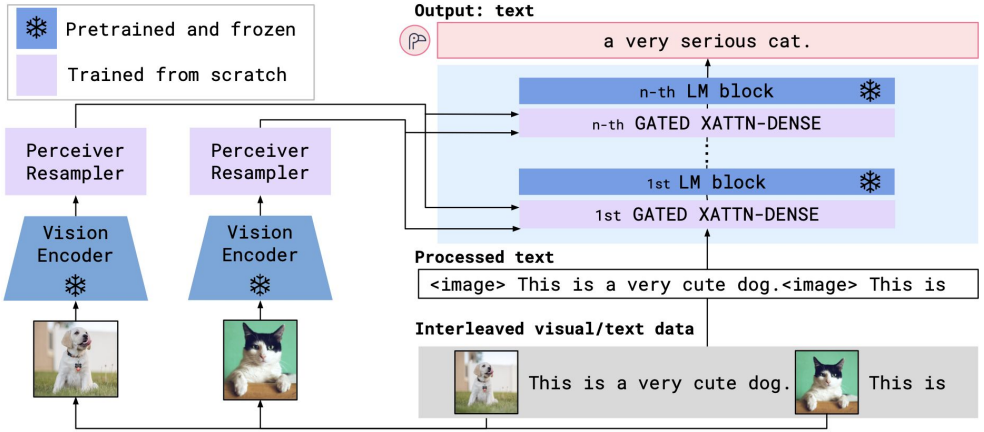
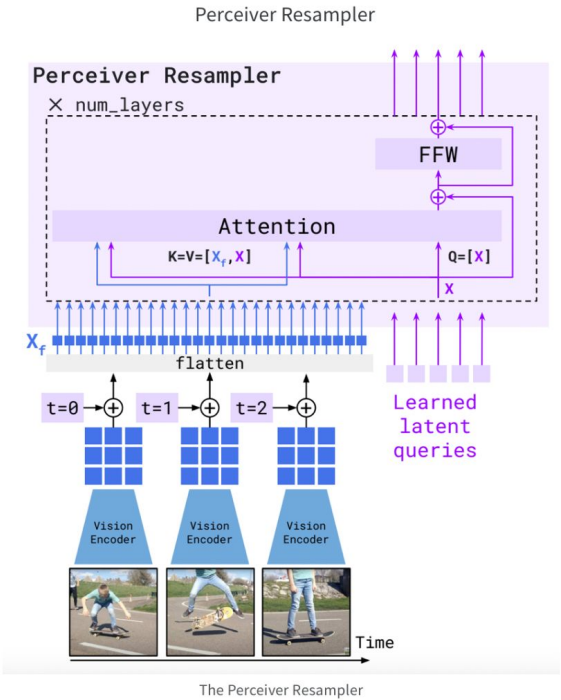


Figure 3: **Flamingo architecture overview.** Flamingo is a family of visual language models (VLMs) that take as input visual data interleaved with text and produce free-form text as output.

Flamingo: a visual language model for few-shot learning. NeurIPS (2022) - https://arxiv.org/pdf/2204.14198

Flamingo



Pseudo-code of the Perceiver Resampler

```
def perceiver_resampler(
 x_f, # The [T, S, d] visual features (T=time, S=space
 time_embeddings, # The [T, 1, d] time pos embeddings.
 x, # R learned latents of shape [R, d]
 num_layers, # Number of layers
):
 """The Perceiver Resampler model."""
 # Add the time position embeddings and flatten.
 x_f = x_f + time_embeddings
 x_f = flatten(x_f) # [T, S, d] -> [T * S, d]
 # Apply the Perceiver Resampler layers.
 for i in range(num_layers):
    # Attention.
    x = x + attention_i(q=x, kv=concat([x_f, x]))
    # Feed forward.
    x = x + ffw_i(x)
return x
```

Flamingo: a visual language model for few-shot learning. NeurIPS (2022) - https://arxiv.org/pdf/2204.14198

Flamingo

Visual and text tokens go through each decoder block, which has two parts:

1. **Gated cross-attention** (Q = text tokens, K/V = visual tokens from the resampler),
2. **Frozen LM layer**.

A small feed-forward network follows the cross-attention, and LayerNorm is applied to all Q, K, V inputs as well as to the FFN inputs.

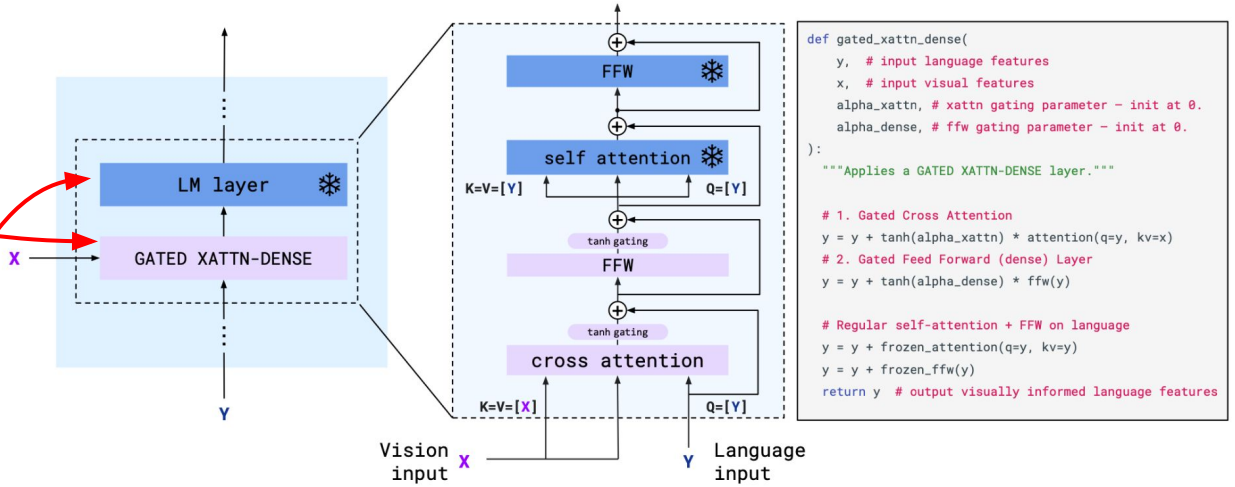


Figure 4: **GATED XATTN-DENSE layers.** To condition the LM on visual inputs, we insert new cross-attention layers between existing pretrained and frozen LM layers. The keys and values in these layers are obtained from the vision features while the queries are derived from the language inputs. They are followed by dense feed-forward layers. These layers are *gated* so that the LM is kept intact at initialization for improved stability and performance.

To understand better, recommend this one explanation by WandB blog:
https://wandb.ai/gladiator/Flamingo%20VLM/reports/DeepMind-Flamingo-A-Visual-Language-Model-for-Few-Shot-Learning--VmlldzoyOTgzMDI2

Flamingo: a visual language model for few-shot learning. NeurIPS (2022) - https://arxiv.org/pdf/2204.14198

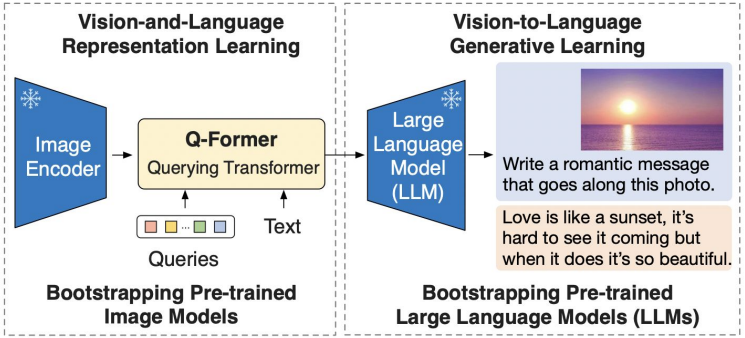
BLIP-2 (Q-Former)

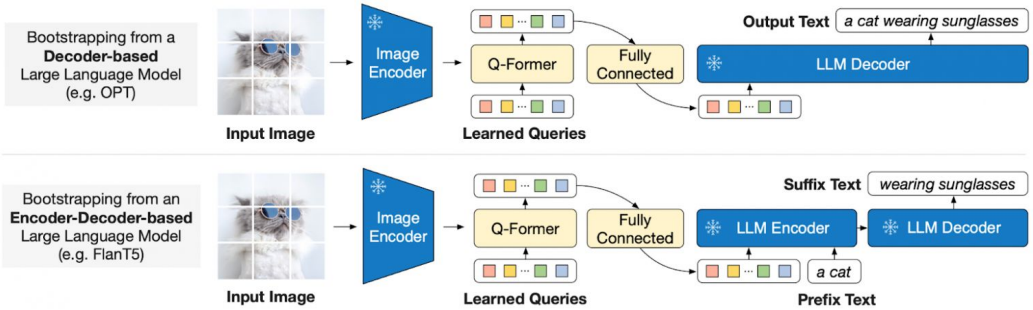
Image Encoder (e.g. a pre-trained **ViT** or **Swin**) produces M visual feature vectors.

Q-Former (Querying Transformer)
- Holds K learnable query tokens.
- Repeats N blocks of:
    1) **Self-Attention** among the K queries.
    2) **Cross-Attention** (Q = queries, K/V = image features) to fetch relevant visual content.
    3) **Feed-Forward** + residual/LayerNorm.

Outputs K adapted query embeddings.

LLM (e.g. GPT-3)
- Receives the K query embeddings as its “prefix” tokens.
- Generates text (captions, answers, continues prompts) grounded in the image.





BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. CVPR (2023) - https://arxiv.org/pdf/2301.12597

LLAVA

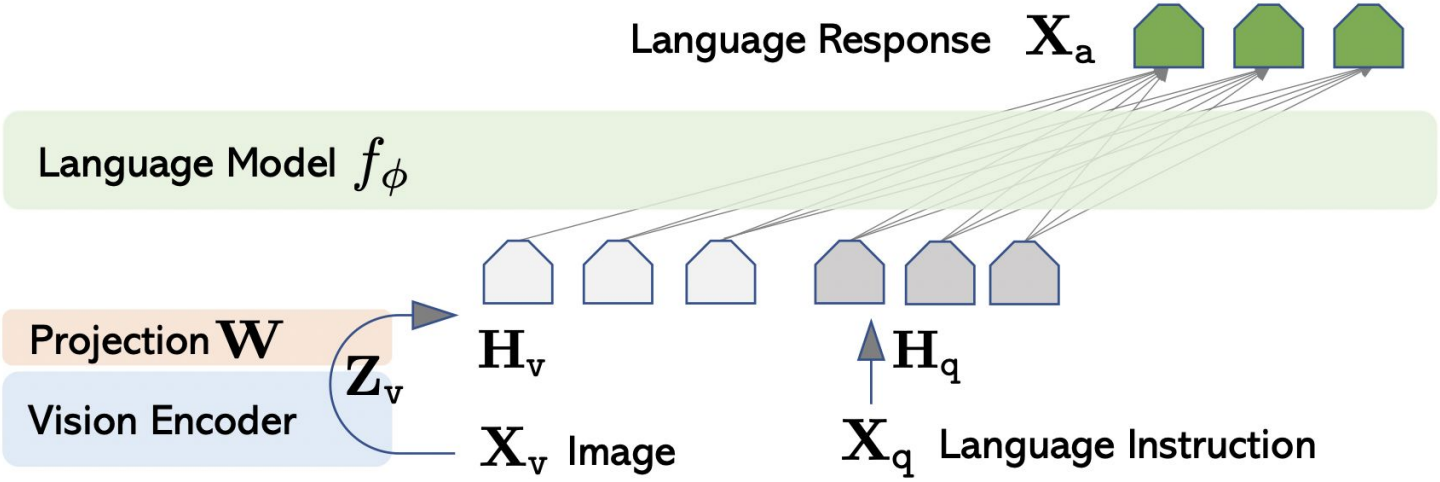


Figure 1: LLaVA network architecture.

FROMAGE

Language model — **OPT-6.7B**
Visual encoder— **CLIP ViT-L/14**
Training — 18k steps with bs = 180, 1 day on 1×A6000

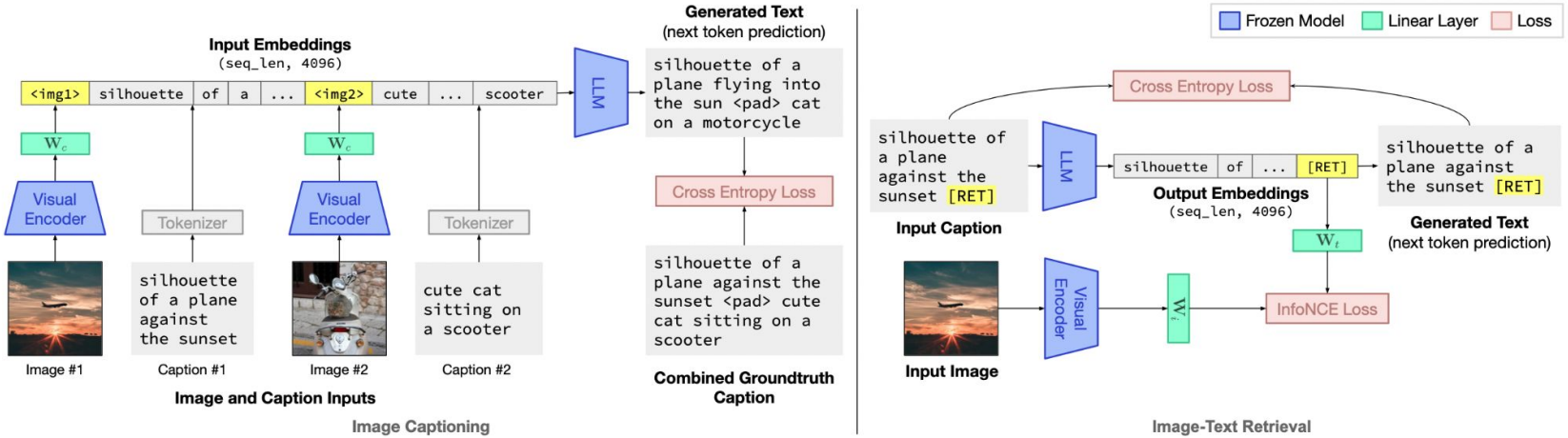


Figure 2. Overview of the FROMAGe architecture. FROMAGe is a model trained on image-text pairs for image captioning and image-text retrieval. It is capable of processing arbitrarily interleaved image and text inputs, and producing interleaved images and text as outputs.

Grounding Language Models to Images for Multimodal Inputs and Outputs. 2023 - https://arxiv.org/pdf/2301.13823

GILL

Language model — **OPT-6.7B**
Visual encoder— **CLIP ViT-L/14**
Image generator — **SD 1.5**
Trainable parameters— 50M, Training — 20k steps with bs = 200, 2 days on 2×A6000

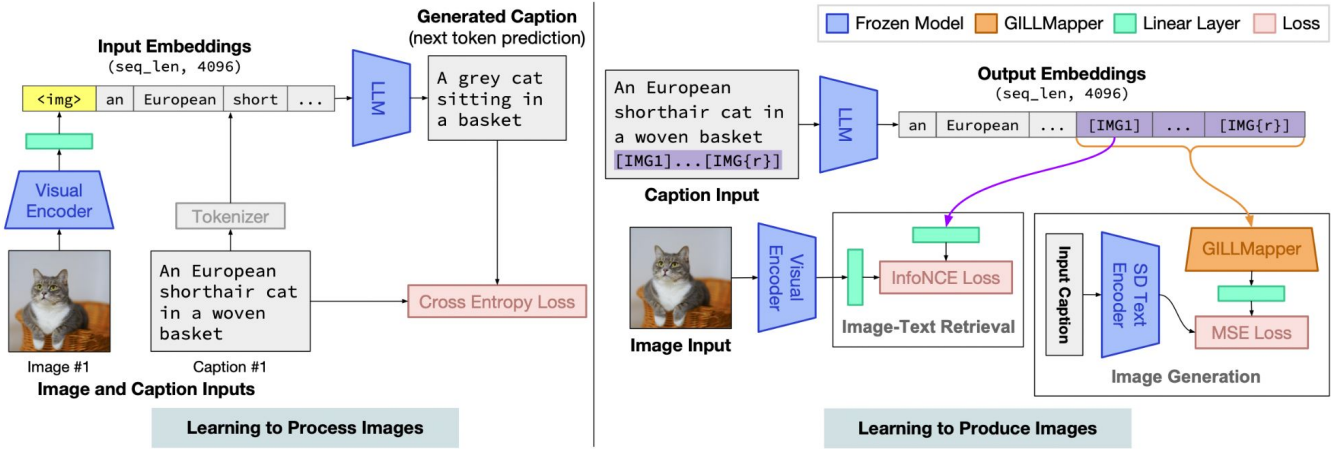


Figure 2: GILL model architecture overview. It is trained with a captioning loss to learn to process images (left), and losses for image retrieval and image generation to learn to produce images (right).

Generating Images with Multimodal Language Models, NeurIPS 2023 - https://arxiv.org/pdf/2305.17216

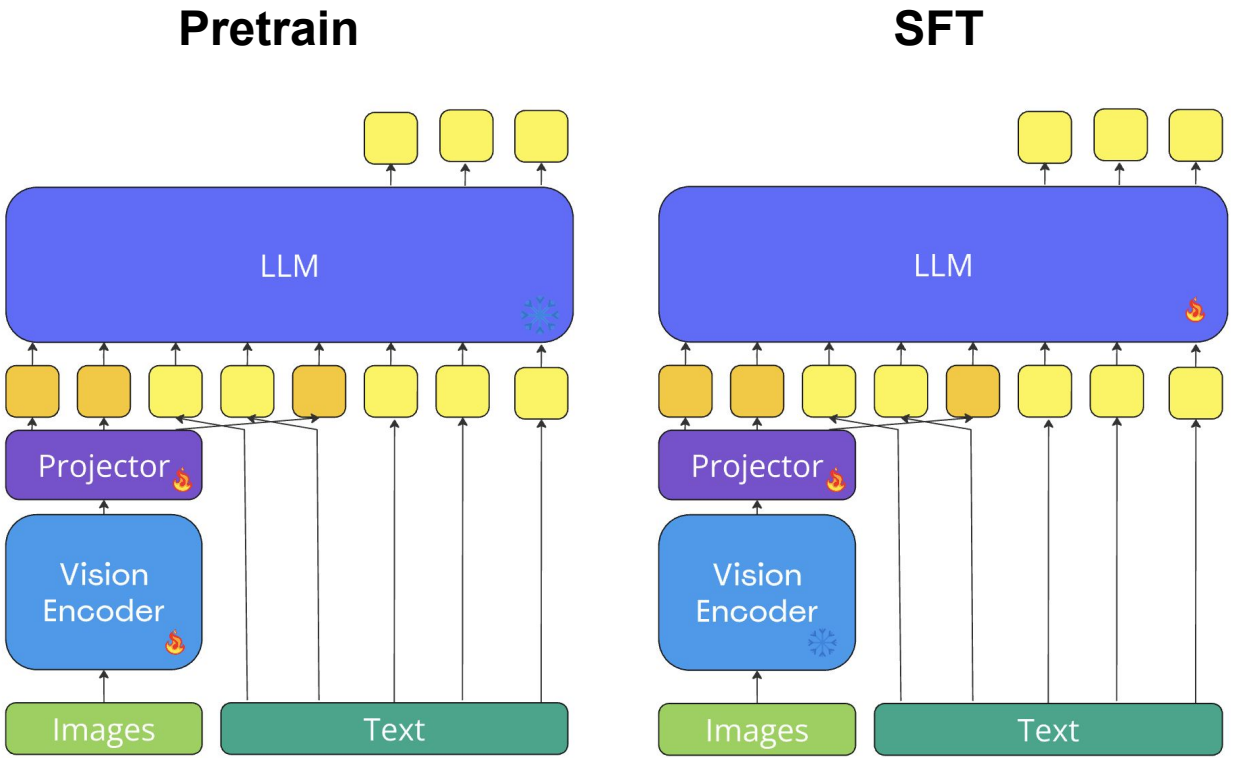
How to train this models?

Training on the **predict next tokens** based in the previous context (masked)

t2i (text→image): генерация изображения по текстовому описанию.

i2t (image→text): генерация подписи (caption) к картинке.

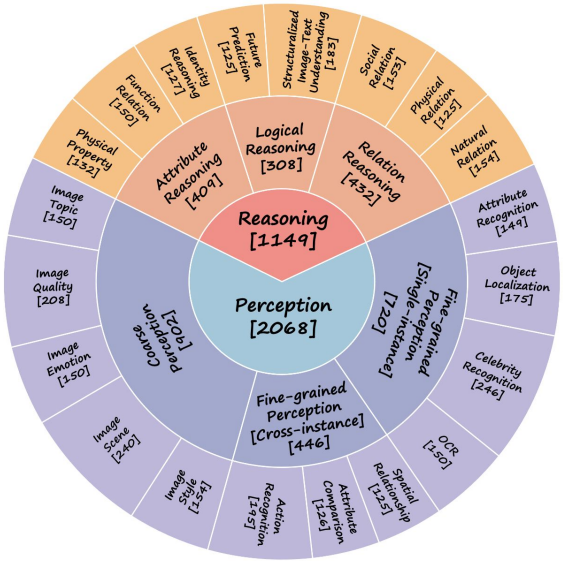
t2t (text→text): языковое моделирование.



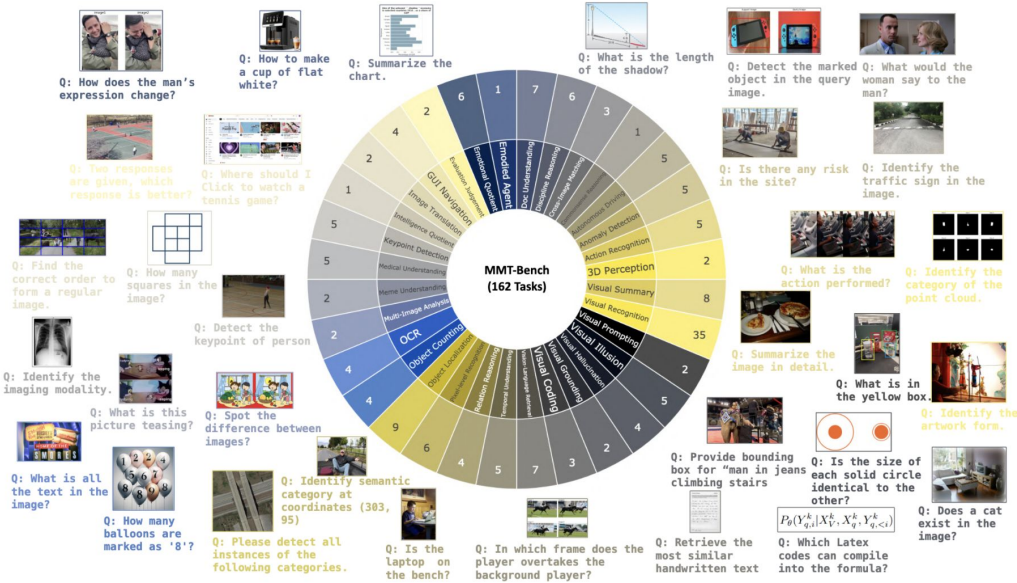
The image is taken from https://blog.deepschool.ru/cv/v-llm/

Benchmarks

MMBench



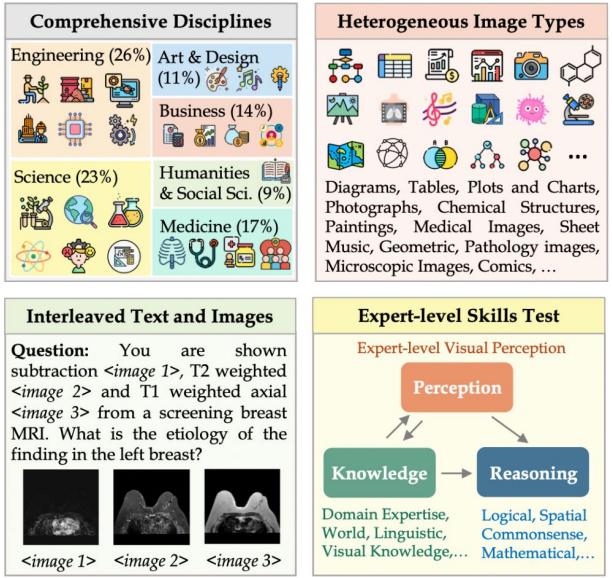
MMT-Bench



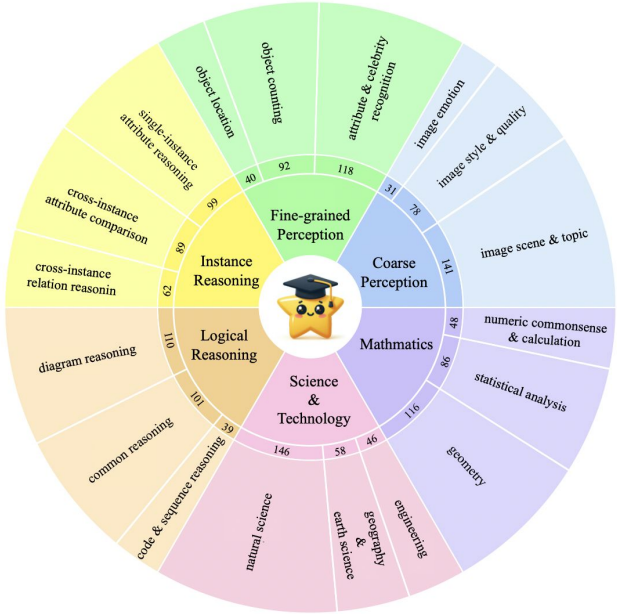
Generating Images with Multimodal Language Models, NeurIPS 2023 - https://arxiv.org/pdf/2305.17216

Benchmarks

MMMU



MMStar



MMMU: A Massive Multi-discipline Multimodal Understanding and Reasoning Benchmark for Expert AGI, 2024 - https://arxiv.org/pdf/2311.16502
Are We on the Right Way for Evaluating Large Vision-Language Models?, 2024 - https://arxiv.org/pdf/2403.20330

More datasets

Viz-Wiz - Real-world photo-description questions asked by people with low vision.

MM-Vet - A challenging multimodal dataset of open-ended queries, with answers judged by GPT-4.

POPE - A suite for detecting hallucinations in multimodal conversational agents.

LLaVA-Bench - A small, synthetic dialogue benchmark built from image captions (generated via GPT-4) as introduced in the LLaVA paper.

TextVQA - An OCR-focused benchmark where questions revolve around reading and interpreting text embedded in images.

ScienceQA - Multiple-choice questions covering lecture content across various STEM domains.

VQA v2 - Open-ended visual questions requiring deep scene understanding and logical reasoning.

Contacts

My TG-blog

My TG

KNOWLEDGE CLUB