
Smol Vision Language Models: Notes

Marija Brkic

May 27, 2025

1 Hugging Face Blog 1: Link to the blog

One-shot learning: one example and the model needs to learn from it (example: digit recognition model and then you give him one input it has not seen before and it has to recognize it later). Zero-shot learning: a model has to solve a problem without being specifically trained to do so.

Vision Language model as an input has images and text(combined) and as an output it has text. It is a multimodal model: multiple input types. It could be used for chatting about images, image recognition via instructions, visual question answering, document understanding, image captioning, segmentation.

We can differ models, training datasets and the way they encode images. (If you want to chat with the model like a chatbot it has to be fine-tuned for it)

-Grounding feature: reduces hallucinations (this means that the output is linked to a trusted external source like some database or document found online in order to have accurate, verifiable and context-aware output. Made up outputs that make no sense are called hallucinations) There are a couple of leaderboards online that show ranked VLMs available online: Vision Arena(not good it has been down for a while) where users vote between two models that are not known to them/Open VLM Leaderboard: models are ranked according to some metrics. The metrics are available in VLMEvalKit, another evaluation suite is LMMS-Eval which provides a command line for evaluating hugging face models.

Both leaderboards are limited to the submitted models, they need to be changed in order to add a new model.

Evaluation benchmarks: (WHAT ARE THE METRICS)

MMMU: A Massive Multi-discipline Multimodal Understanding and Reasoning Benchmark for Expert AGI: Link to the video explaining it

This is a set of 11.5K multimodal challenges that requires college-level subject knowledge across different disciplines such as arts and engineering.(This is a dataset)

MMBench: 3000 single choice questions over 20 different skills.(This is a dataset). The paper also proposes a strategy called CircularEval where answer choices are shuffled and the model has to correctly answer each time. In both of the previous cases the problem is multiple choice and a model needs to chose the right one. To evaluation metrics are I assume the same as any classification problem.

In order to train a Vision language model we can use multiple ways, but the point is to somehow combine image and textual input with a certain type of encoder and transfer it to decoder that will generate text. (well there is a part in between the encoder and decoder).

Usually it is a certain image encoder(like a CNN-my guess), and then an embedding projector to align image and text representations(like a dense neural network(FC)) and at the end a text decoder.

2 LLaVA: Link to the original paper presenting LLaVA

Visual Instruction Tuning(the name of the paper):

The idea of this paper is to use LLM(GPT-4) to generate multimodal language-image instruction-following data (that means that they take image and add some caption or some sort of instruction and want to generate description from it and they are trying to make the model as conversational as possible -; that means longer texts). LLaVA = Large Language and Vision Assistant. It uses vision encoder and an LLM for generating visual understanding. They also make comparison with multimodal GPT-4.

So the point is that up until this paper all of the problems have been solved separately like a separate

network was trained for segmentation, captioning chatting and so on, and they had only vision input. Here they want to add a textual input that describes the problem and now this model can solve more problems. (All of this in vision) They want to combine this visual aspect with LLMs.

They make next contributions:

1. Multimodal-instruction following data: transferring data to appropriate type for this using ChatGPT/GPT-4
2. Large Multimodal Models(LMM): end-to-end model from image to text- i visual encoder (CLIP- i GO INTO THAT), language decoder Vicuna. Fine tuning the whole model. (SoTA = state of the art)
3. Benchmarks

Note that visual instruction tuning is different from visual prompt tuning: the first one is aimed to improve instruction following abilities of a model, and the second one improves parameter-efficiency in model adaptation(that means give the model a prompt that gives a bit of help without changing the model a lot).
—THIS PAPER WAS WRITTEN IN OVERLEAF AND THEIR FIGURES GOT IN BETWEEN THE TEXT AND THEY DIDN'T FIX IT. IT BOTHERS ME—

The first example they are showing(Table 2) gives an example of giving instructions to GPT and three response types, however IT IS NOT SUPPORTED WITH AN IMAGE, the only input in the model was a text. In order to increase the data they prompted GPT-4 to give a set of questions X_q for each X_v image and caption X_c . This way they expand image text pair. (I did not really get what is the GPT creating here is it a caption or answers to questions)

Look at the architecture but it is very simple they first encode the image with CLIP and then project that with just some W projection matrix into word embedding space (I think that means). This could be combined in different ways, but I don't think I necessarily need that.

Training: See the paper but at the beginning its image + question, and then sequentially other questions(NOT SURE).

$$p(\mathbf{X}_a | \mathbf{X}_v, \mathbf{X}_{\text{instruct}}) = \prod_{i=1}^L p_{\theta}(\mathbf{x}_i | \mathbf{X}_v, \mathbf{X}_{\text{instruct}, < i}, \mathbf{X}_{a, < i}),$$

Figure 1: Joint Probability distribution function

- θ are trainable parameters
- $X_{\text{instruct}, < i}$ are all instruction tokens before the current token prediction x_i
- $x_{a, < i}$ are all answer tokens before the current token prediction x_i

So L is the length of all answers (but I think all tokens: CHECK).

- Stage 1: Pre-training for Feature Alignment
- Stage 2: Fine-tuning End-to-end

Link to the YouTube video explaining LLaVA and fine-tuning it, very good explained
LLaVA GIT repository: LLaVA git.....there should be a DEMO but it is not working it makes me sad.

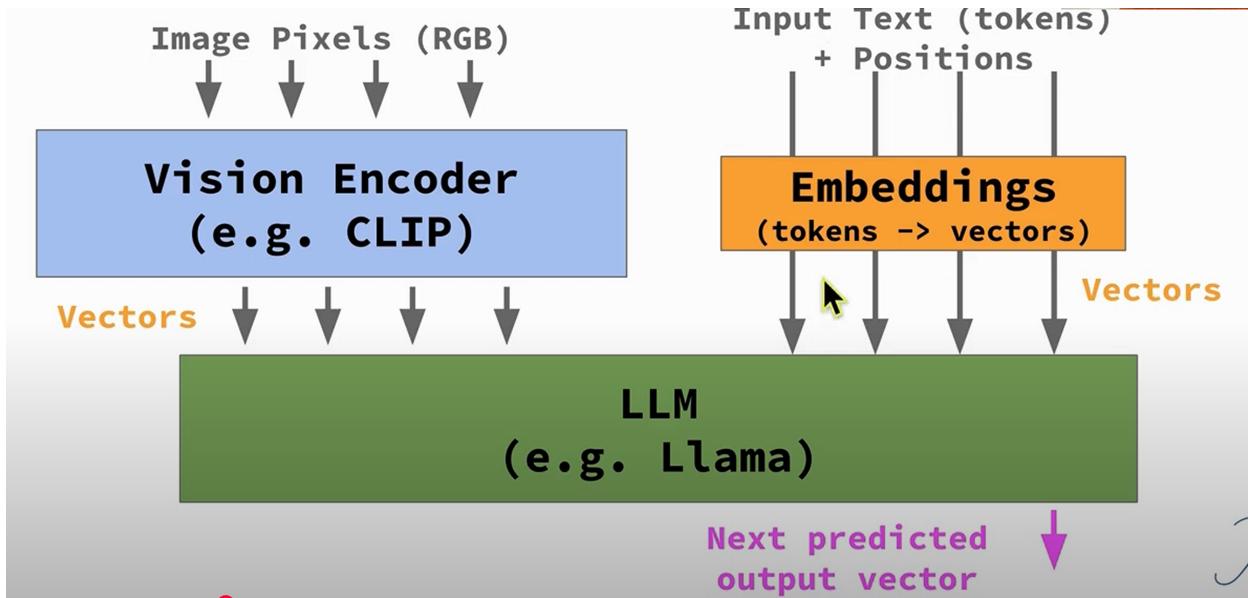


Figure 2: LLaVA architecture

GO OVER TRANSFORMERS

Video:

Covering llava 1.5, llava 1.6, idefix, using a custom dataset of chess pieces, fine-tuning llava 1.6: mistral 7B version (7 billion parameters: i need less) and yt 34B(a lot bigger).

Before his fine-tuning for chess board GPT works much better.

Architecture:

Vision encoder(CLIP) is transferring image to some vectors, and input text that is also embedded into vectors(i guess that can be any word to vector architecture but it can maybe also be some type of encoder or like a transformer based but i don't think it is....if it was only text in that case the whole network would be transformer based).

All the input goes in in parallel....the prediction is output vector and we have to decode it.

Vision encoder:

We can split an image into patches (16x16 ex) and we do an embedding where we have basically a matrix that multiplies a patch and the output is the vector representation of it. parameters of that matrix are trainable, and at times will be trained and at times they will not be. This vector now goes to Vision Encoder which makes use of attention(that is a transformer term, will look into it a bit more). Feed forward and attention layers.....this gives the output representation of vectors that are in sync with the meaning of a patch. We also input a position of patches at the same time when we embed a patch.

Let's assume that we want to use a pretrained LLM(ex: llama) and a vision encoder(ex: CLIP) that means that the output of the vision encoder is not necessarily aligned for the llm input. That means that we need an adapter what in the first paper is only a matrix (Linear transformation, but in the later paper is MLP and nonlinearity is added). This means that at some point we keep llm fixed and vision encoder fixed and we train only adapter.

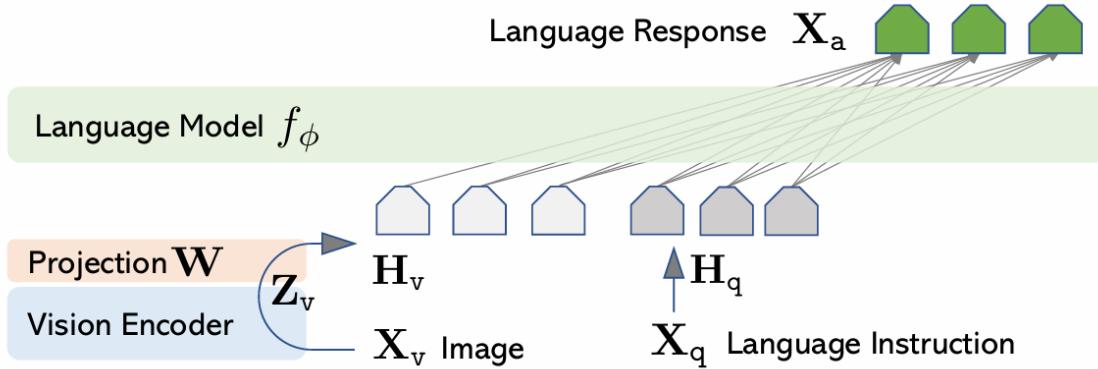


Figure 1: LLaVA network architecture.

Figure 3: LLaVA architecture: image from the Visual Instruction Tuning paper(the original one)

LLaVA 1.5:

CLIP encoder + llama 2 llm: Training PART 1:

- freeze the vision encoder and an llm and train the adapter
- use text and image pairs like online datasets THIS IS TRAINING AND NOT FINE TUNING

Training PART 2:

- unfreeze everything and train it together on the synthetic data
- use text and image pairs but our own and I believe this part is basically fine-tuning—yup

For this we use image + questions about it (for synthetic data). We use chatGPT for creating those questions. LLaVA uses GPT-4 for creating questions, and not the vision model, this is language only model.

Questions generating:

So to GPT-4 we don't give any images but text representation of an image and it has to give us detailed descriptions or smth like that based on our caption and bounding boxes which are the prompt inputs to the model. The bounding boxes are for all objects and we have their labels. That way we get detailed descriptions, and we can have question/answer format.

Link to the paper Improved Baselines with Visual Instruction Tuning this is LLaVA 1.5 /HD but has MLP improvement and also image batching. And if I understood correctly uses specific scientific topics.

THE POINT IS ORIGINAL LLAVA DOESN'T HAVE MLP, BUT 1.5 HAS IT, IT IS CLEAR IN THE PAPER.

LLaVa 1.6:

We have better building blocks:

- uses mistral llm 7B or yi 34B
- bigger CLIP or younger version of CLIP
- MLP for adapter instead of linear layer - nonlinearity
- larger image can be an input

IDEFICS model: NO IDEA A COMPLETELY DIFFERENT ARCHITECTURE AND TRAINING I WILL SKIP FOR NOW.

Now there comes an implementation of fine-tuning which is present in both video and blog but I might come back to that later, for now I am moving on with explaining CLIP and transformers in detail!

3 CLIP: Contrastive Language-Image Pre-Training OpenAI git repository, Learning Transferable Visual Models From Natural Language Supervision Paper

I am looking at the video: CLIP paper video

So this is incredible actually CLIP could be used as a zero-shot classifier and it works very well on many different images and completely different tasks and datasets:

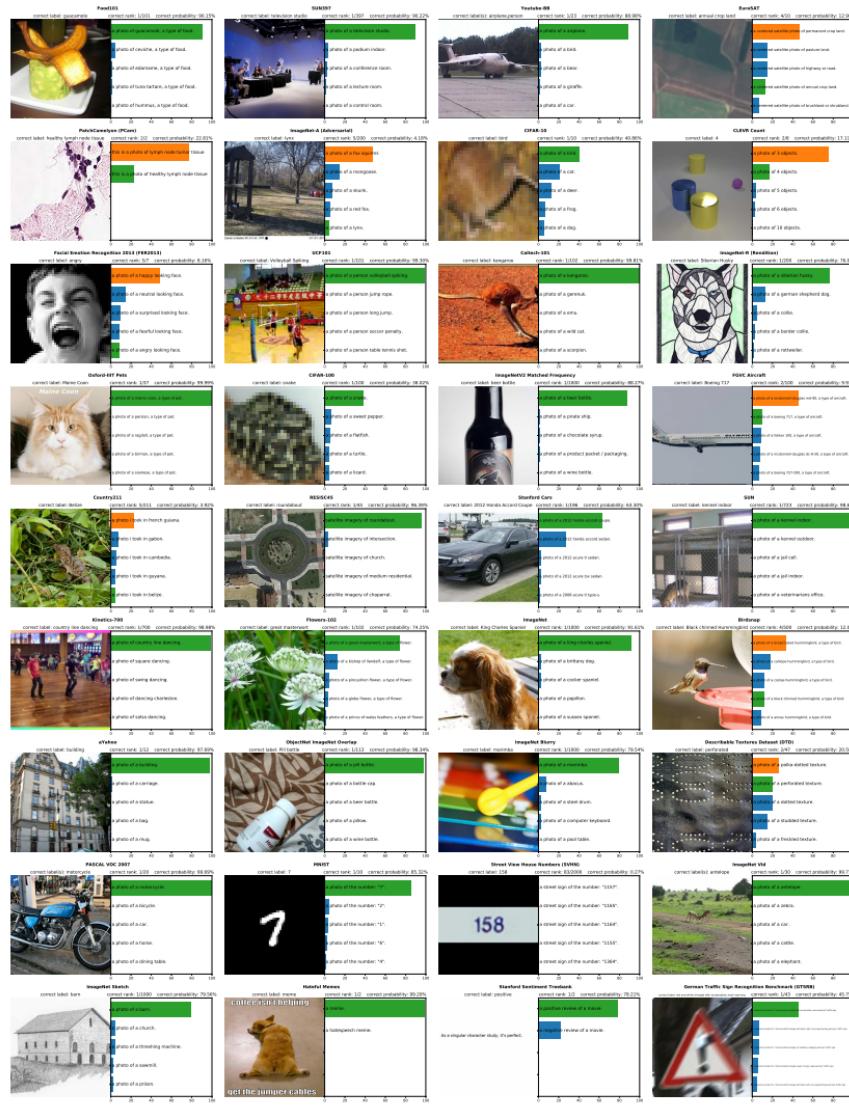


Figure 21. Visualization of predictions from 36 CLIP zero-shot classifiers. All examples are random with the exception of reselecting Hateful Memes to avoid offensive content. The predicted probability of the top 5 classes is shown along with the text used to represent the class. When more than one template is used, the first template is shown. The ground truth label is colored green while an incorrect prediction is colored orange.

Figure 4: CLIP Paper results

It is using different test datasets for labeling images, choosing the right label. Zero-shot means it is not trained on these dataset. Also you can notice that the labels are quite weird.

CLIP takes images and text and connects them in a NON GENERATIVE WAY. A model that can represent

images and text very well. How can we connect images and text. Let's say we have images with text like a cat and a text My little cat. It's very common, wherever you take an image it usually goes with a label. So we can take the image and predict text from image. This is a model that is labeling images and a representer of image.

An input in this network is an image and an output is a text like My little cat and that network does not only represent pixels like raw values but it is actually trying to capture a meaning in that image and describe it in words. In the cat case it has to recognize a concept of a cat.

There have been papers who predict captions and the next graph shows their comparisons: This is for

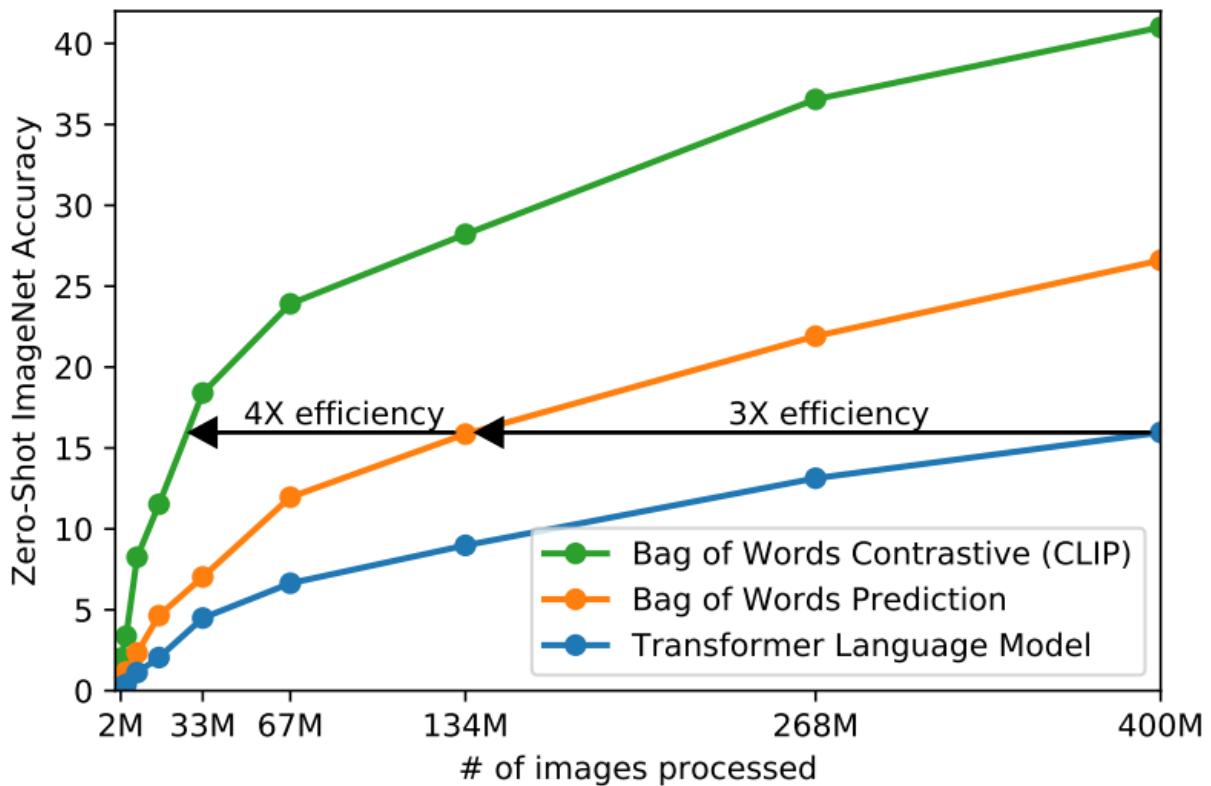


Figure 5: CLIP Paper model comparison

zero-shot accuracy. Transformer language model (this is evaluated on ImageNET), and bag of words is used. Bag of words is predicting only words without an order but it is still captioning some sense what is actually happening in an image. Obviously the best result is CLIPContrastive. This approach is not only predicting text but it is using some input text as well(MAYBE, THIS GOES AGAINST WHAT I'VE LEARNED SO FAR).

the thing is we don't really want a classifier? Or we do but I want to know how likely is a phrase.....but that is still a classifier.

Ok the point is if we phrase a label in a certain way we might get a better classifier. Maybe the phrase 'a photo of a dog' is more informative than a 'dog' and we get less noise. So those labels and descriptions are filled out prompts basically and this model is now just a regular classifier. A zero-shot classifier.

CLIP does the following: takes the image and passes it to image encoder and that gives us a vector in latent space. We get a representation of each image(a vector). The same for the text. I_1 is a vector of the first image and T_1 is a vector representation of the first text and the first image is described by the first text.

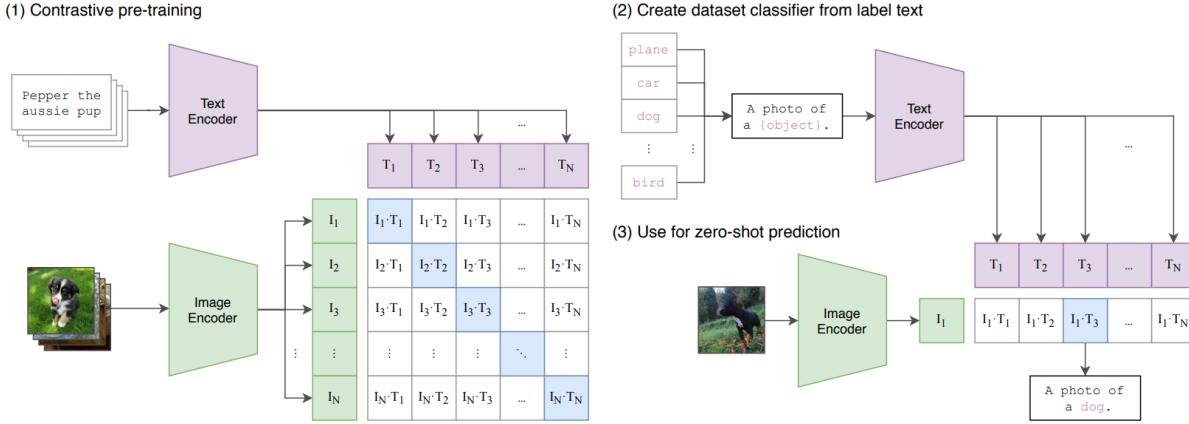


Figure 1. Summary of our approach. While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time the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

Figure 6: CLIP architecture

Now we are not predicting the text anymore from the image but we ask which of the texts is most appropriate for this image and this is contrastive learning. Of course since this is training we know that, and it increases that closeness.....but isn't this a classifier????????? You get a classification of an inner product(a matrix) and you get a softmax for both directions: exactly it's a classification problem in two different directions. It would be good to have larger minibatches. Ok and inferencing does the same thing. But we assume that we have all of the labels and we engineer a prompt. And we see what is the closest representation. It is zero shot because (if i understood it correctly) you can now insert an image of whatever and it should generate a probability of a label.

OK COOL BUT THIS IS STILL JUST A CLASSIFICATION AND IT STILL HAS TO HAVE A LABEL. THE POINT IS THAT YOU TRAIN AN ENCODER SO IT WORKS FOR THE TEXT.....

This is not a specific classification because it is connecting specific images with specific text like it's not just a dog but it is a pup....so it is not batching all the dogs together, as a classifier would.

Text encoder: text encoder in here is a transformer(DO THAT) but i suppose it can be some other stuff. It uses a representation only on that one word that is a label.

Image encoder:

They tested various networks:

- ResNET variations: Link to ResNET paper :

A ResNET has skip-connections and this prevents vanishing or exploding gradients. It uses a residual function:

$$H(x) = F(x) + x \quad (1)$$

where:

- x is the input to a layer or a block
- $F(x)$ is the output of a few stacked layers like CONV-BatchNorm-ReLU
- $x + F(x)$ is the residual connection added ELEMENT WISE

I understand why it helps with vanishing but not why it helps with exploding.

- ViT which is a visual transformer.

They also work on prompt engineering, but this is another topic.

Linear probing.....not now, I don't think it is very important.

The point is embedding is some cnn.

They get very good results for zero-shot...it is better than ResNET 50 that is exclusively trained for the certain task! That is actually amazing. It even becomes state of the art for this one dataset that does not have a lot of examples for classes. For a lot of models CLIP actually outperforms them if they are pretrained with a small amount of labeled data.

You can even do linear probing with clip but I did not go into detail of that.

4 Transformers: Link to the Attention is all you need paper, Link to the YouTube video that helps explaining it

Let's say we want to translate a sentence- i , but could be basically used for anything else.

First we need some word embedding- j , this could be any embedding I guess like word to vector:

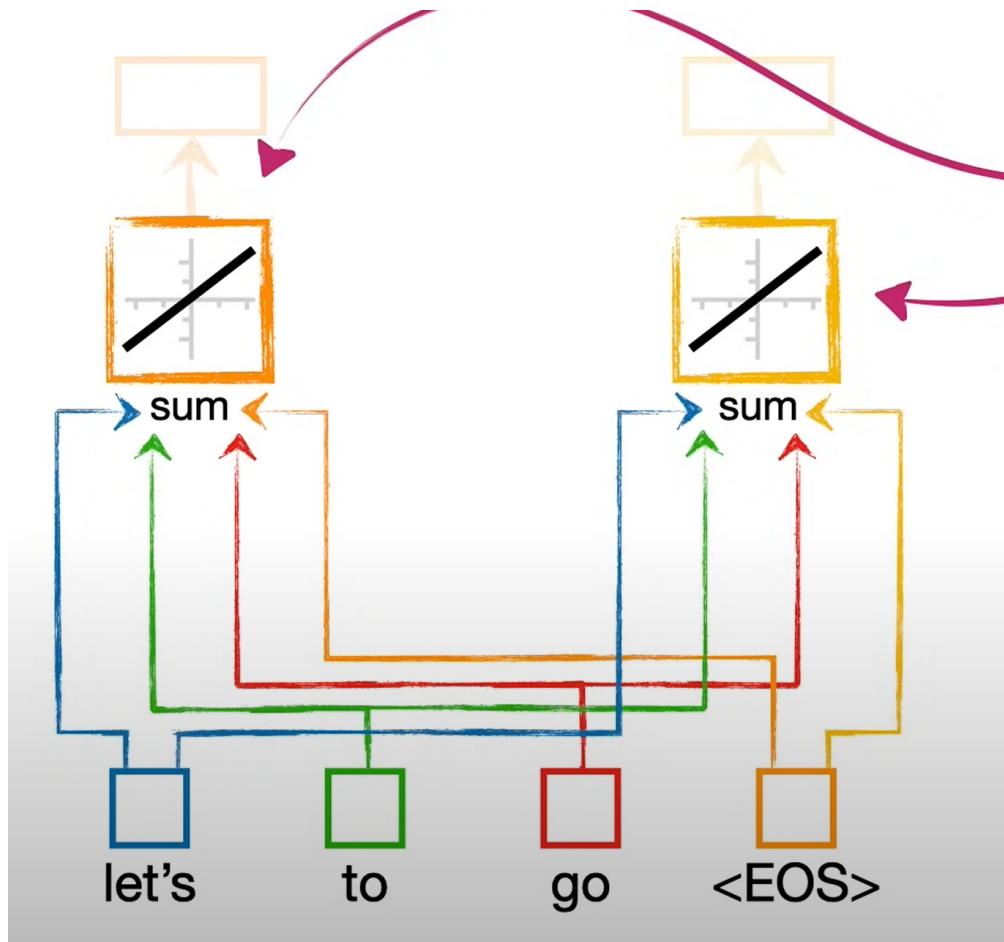


Figure 7: Word embedding (from the video)

Now we need to pay attention to the position of each word and we do that by adding a position value for each word. The position value is taken from alternating sin and cos functions with increasing wave length. As an x value we take a position of the word in the sentence like 1,2,3.....and an output is a value between -1 and 1 and we add that to the word embedding:

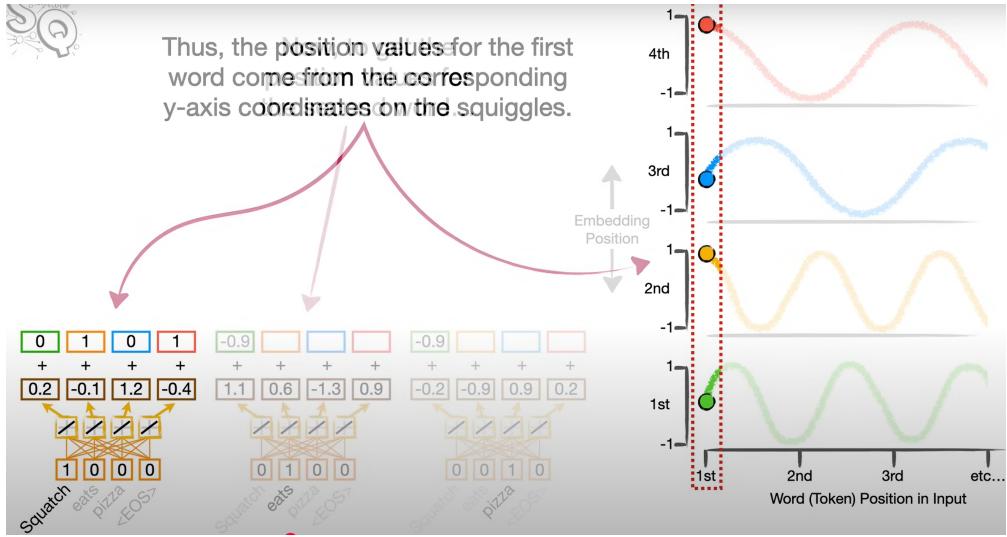


Figure 8: Position adding (from the video)

Since sin and cos are periodic functions it is possible that two words have some of those values the same but that is why they are increasing the wave length.

Now the next part is very tricky and it has to do with attention. So basically we can't only look at the order of the words but we would like to follow relationships between the words. Like if i say 'Marija is very pretty and her sister is not' we want our model to understand that 'her' refers to 'Marija'. So we use something called attention which basically captures how close two vectors are in vector space. We add a self-attention layer (that is the same weights for all of the words, they are not new for each word). We have a linear matrix for calculating a part called **Attention Query**, **Attention Key**, and **Attention Value**. For example if i have words 'let's' and 'go' and i want to find a self-attention values for the word 'let's' I have to do it for all the other words, including the word 'let's' itself. We want to find similarities between the word 'let's' and all the other words and for that we use a query part for the word 'let's' and a key part for all the other words and we find a dot product between all of those. Now, that does not have to necessarily be a dot product it could technically be any similarity measure like cosine similarity or whatever. Now we take those new values and put them in SoftMax in order to get like a percentual similarity to each word. And now we multiply those percents with the attention value of each word and that is a self-attention output of a word. We repeat that for all the words:

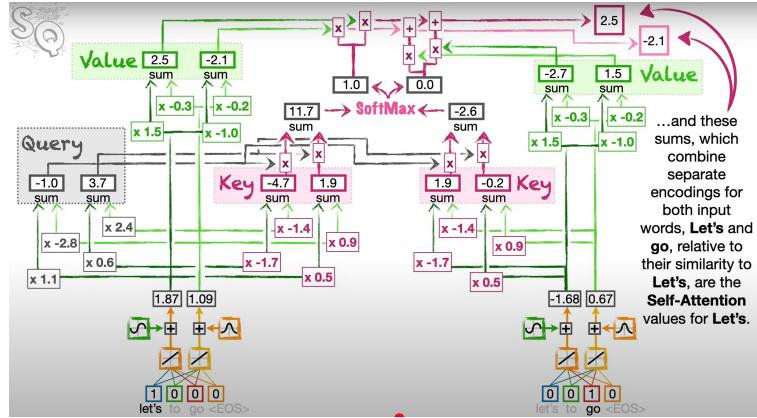


Figure 9: Self-attention (from the video)

After that we add a residual skip connections to the output of the self-attention layer. The output of that is the final encoder block of the transformer neural network.

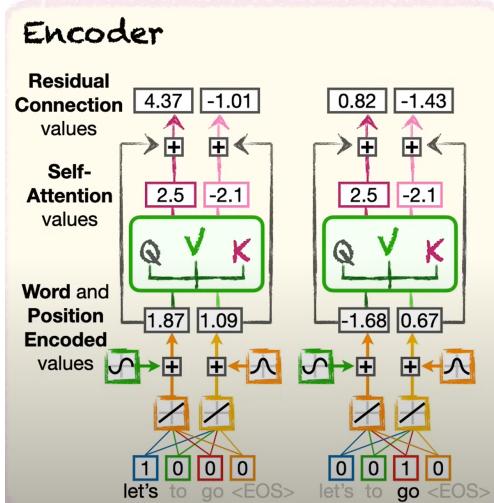


Figure 10: Encoder (from the video)

This is of course pretty nice because all of this could be done in parallel so that means that we can use GPU. That is cool because for example a recurrent network also could do the translation but it has to process word by word sequentially, not in parallel, therefore it is a lot slower.

After that we can do the same thing for the decoder whose input is the translated sentence but in that case we first start with '*EOS*' or '*SOS*'. It also has a self-attention block. After all that we have to have some sort of attention between a encoder and decoder and that is the same form of attention block, with query, key and values and it is called **Encoder-Decoder Attention block**. We use a query of the decoded translated word and a key of encoded words. That is also an input to softmax and that is multiplied with an attention-value of encoded words. Then we can also add residual connections. Finally we put that as an input into a fully connected layer and that to softmax so we can get an output that should be the translated word again. We repeat the process until the output is '*EOS*', or in training until we expect it.

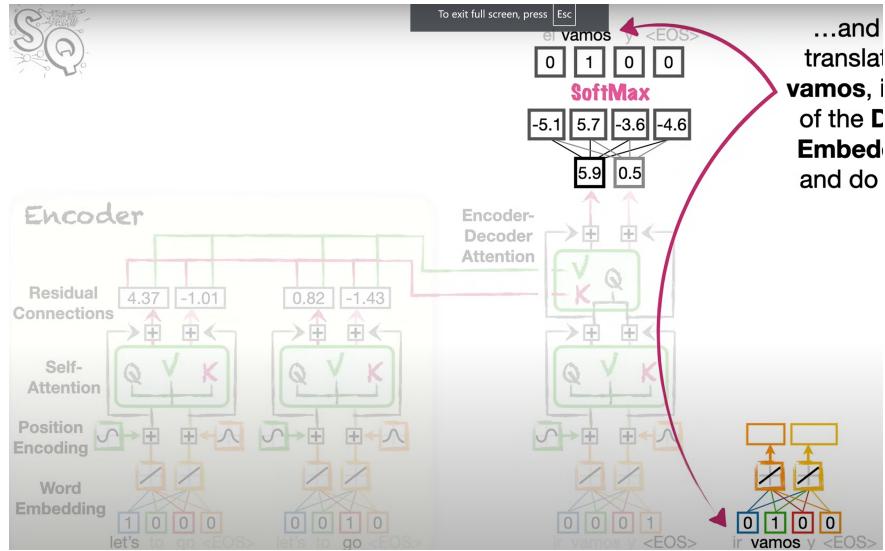


Figure 11: transformer architecture (from the video)

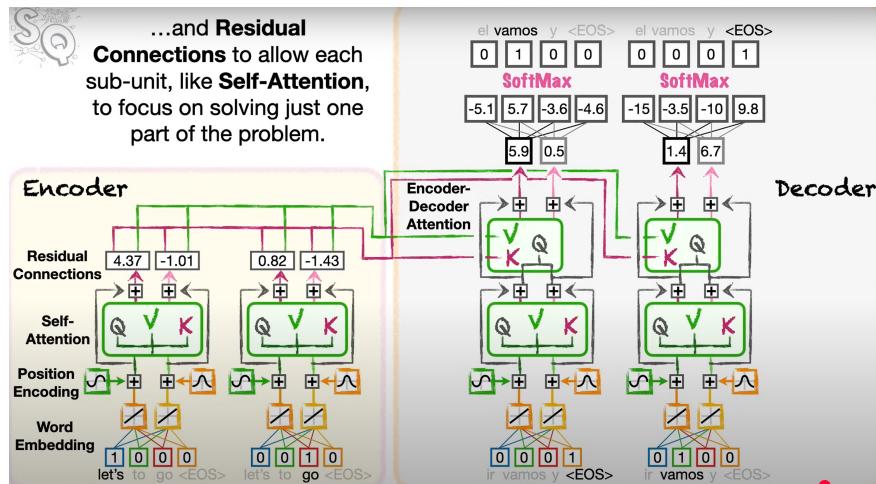


Figure 12: transformer architecture (from the video)

5 Hugging face Blog 2(2025): Link to the blog

Smaller and more powerful models YEY!

Multimodal Retrieval Augmented Generation(RAG) and multimodal agents.

5.1 New VLMs

5.1.1 Any-to-any models

They can take any modality input and output like an image, text, audio. They do that by transferring one modality to another which is super cool. Any-to-any multimodal llm paper, Any-to-any generation via composable diffusion paper

These models have a separate encoder for any type of modality and they transfer them into the same vector space. Decoders go back to the modality of choice. We have chameleon by meta which was a first attempt into this: Link to the model, Lumina-mGPT. The most advanced (by the publishing of this blog) is Qwen 2.5 Omni. It uses a thinker-talker architecture, where thinker is a text generating part and a talker is generating audio response. MiniCPM-o 2.6 is another any to any multimodal model that processes speech, vision and text. Janus-Pro-7B. They collected any to any models in here: Link to any to any models. Link to the hugging face explanation of any to any models.

5.1.2 Reasoning Models

5.1.3 Smol yet Capable Models

The idea here is to shrink some models, possibly through distillation(TAKE A LOOK AT THAT). This reduces compute costs, simplifies deployment, and makes new use cases like local execution with private documents. When we talk about small vision language models we usually talk about models with less than 2 billion parameters, that can be run on consumers GPU.

SmolVLM is a type of small vision language models where they did not try to shrink other larger models but to custom make smaller one with like 265M, 500M, 2.2B parameters. FOR EXAMPLE VIDEO UNDERSTANDING WITH A MODEL OF 500M. My man HuggingFace has an iPhone app, HuggingSnap, that can do this video understanding with small model on customers devices. (I CAN NOT DOWNLOAD IT WITHOUT UPDATING MY SOFTWARE-DOES NOT WORK).

Another model is gemma3-4b-it by Google DeepMind. It has 128k token context window (the size of an input in tokens). Works with 140+languages. This model is from Gemma 3 family which is obviously very good and they distilled their largest model that is the best on Chatbot Arena to 1B parameters.

Another model they are mentioning is Qwen2.5-VL-3B-Instruct that can do a lot of tasks like object localization on an image, or document understanding or agentic tasks with the context length up to 32k tokens. Take a look at the paper for testing some of the models.

5.1.4 Mixture-of-Experts as Decoders

Mixture of experts (MoEs): I am not sure if I understood this correctly but if I did this model activates only the relevant sub model, 'expert' for the given data modality. Because of the smaller network, these models are somewhat faster. They converge quickly! That is cool. They need more memory because the whole model needs to be on GPU?????? Did not understand that. So like only a part of the model is activated, even in inferencing. This is not like a dropout where you train only a part this is basically ok my input is text, activate the part that is specializes for the text and use only that part for both training and inferencing.

They make these models by adding these MoE layers instead of feed forward(fully connected) layers in a regular transformer architecture. Some models are linked in a blog, I will not go into that for now, now i need more on smol vlms.

5.1.5 Vision-Language-Action Models

These are models now active in robotics. Those type of models are known as Vision language Action models(VLA). So this is basically another VLM just the output is a text in which the action of a robot is based. The input is still same like an image or a text. So now we have actions and also current state of the machine, or like an environment state(like a state machine). Ok I will not go very deeply into this but sounds cool.

5.1.6 Capabilities: Object detection, segmentation, counting

So an input is an image and a task text and an output of one of these models, like PaliGemma is a structured text with localization tokens, they help with segmentation or detection. The text input has a task prefix and

an object that should be segmented or detected or whatever is next. Like 'segment stripped cat' or 'detect bird on the roof'. For detection the output is the bounding box(LOVE THIS) and for

6 Temporal dimension processing

If I want to process only some of the frames it would be good to make a difference between frames and to exclude similar ones.

So far, as far as I am concerned, this has been done in multiple ways:

- Pixel level difference (L1 or L2 norm) An Effective Approach to Detect Lesions in Color Retinal Images
- First feature extraction using some CNN (like in CLIP they first use visual encoder that gives some embeddings and than they can compare the embeddings like cosine similarity or any vector similarity) AND IMPROVEMENT OF THAT WITH FRAME VOYAGER. The paper: Frame-Voyager: Learning to Query Frames for Video Large Language Models proposes a model called Frame Voyager used in Video large Language Models. They explain that using all of the video frames might increase the token length of a model (especially in our case where we are not able to use a lot input tokens?!?!) and that can lead to 'lost in the middle' and hallucinations issue and introduce significant computational complexity. This issue is proposed in the paper: Lost in the Middle: How Language Models Use Long Contexts and basically means that models with large inputs work better if the most relevant information is at the beginning, but if it is somewhere in the middle of an input it gets lost.

The paper goes on to explain that some paper propose uniform sampling of frames and then using subset of frames, or text-frame matching(text matching calculates semantic similarities using CLIP between queries of each frame). The first model is not the best because it can obviously skip some important information. The second one can exclude some temporal reasoning, like the causal relations in the video. The approach of this paper is basically ranking problem. They train a model to extract the combination of the most relevant frames. They get a ranking score for each frame. (THIS COULD BE USEFUL TO TRY OUT).

- Scene Boundary Detection: detects major visual scene changes and keeps keyframes from new scenes. Proposed in the paper: Hierarchical Boundary-Aware Neural Encoder for Video Captioning. This is based on RNN and LSTMS which I have gone over already, but I did not write it here. This paper is quite old, from the beginning time of video captioning. They present a time boundary-aware LSTM cell: it can discover discontinuities in the input video and enables the encoding layer to modify its temporal connectivity. They input structure into input data (I AM NOT SURE I UNDERSTAND IT COMPLETELY).
- Reinforcement Learning for Frame selection proposed in the paper: End-to-End Dense Video Captioning with Masked Transformer. They talk about video description in two parts: event proposal and event description(event proposal and captioning modules). Those two modules could be trained separately and generate best descriptions for best event proposals. The other way of training is alternate training. I think they propose a training where they train a decoder for captioning and a mask for importance of events at the same time but they train a mask based on the quality of the description. They incorporate this in transformer based network and they do not use RNNs and they also say they are the first ones that do not use RNNs.
- Motion Based Filtering based on optical flow: Sequence to Sequence – Video to Text I described this in the notebook already....will come back to rewrite it later but in this paper optical flow is not used to extract a part of the video but to capture temporal reasoning of the video, which probably might somehow be used in this case too. I just want to keep this approach in mind also.

- Semi optimal policy: proposed in the paper: Scalable Frame Sampling for Video Classification: A Semi-Optimal Policy Approach with Reduced Search Space (This is a relatively new paper which is nice). This paper works on the topic of video classification but still chooses $N \leq T$ frames from video with T frames. When we want to choose from a $\binom{T}{N}$ search space, this paper proposes reducing the search space from $O(T^N)$ to $O(T)$, based on independently estimated value of each frame using per-frame confidence.

This paper explains that video frames are often redundant and that it would be good to choose them based on information they carry. The most advanced models still use constant timeframe or uniform sampling or something like that. They say that the 'optimal policy π_0 ' would be the one that would actually select the optimal N frames (the most informative ones). This could be possible to get but that would mean to search over T^N space which is not very nice. They also mention that several papers proposed reinforcement learning for solving this problem: AdaFrame: Adaptive Frame Selection for Fast Video Recognition, AdaFocus V2: End-to-End Training of Spatial Dynamic Networks for Video Recognition, OCSampler: Compressing Videos to One Clip with Single-step Sampling, 2D or not 2D? Adaptive 3D Convolution Selection for Efficient Video Recognition, Multi-Agent Reinforcement Learning Based Frame Sampling for Effective Untrimmed Video Recognition, but their search space is still $O(T^N)$. The purpose of this video is to reduce the searchspace itself. We need to consider joint distribution of the frame values. They say that the frames could be observed separately????????? If the length of the video is big and if the frame rate is not extremely high. Based on those observations they propose semi-optimal policy π_s that selects the top N frames based on estimated value of each frame using per-frame confidence. Afterwards they use one of the reinforcement learning samplers, but with the reduced space. CHECK WHAT A FEATURE POOLING IS!!!!

They want to separate each frame and they want to make that the importance of one frame is not related to the importance of another. Again they say that this approach is only possible if the video is not very short and there is no high frame frequency. However, in the opposite case extraction of frames would not even be necessary. They measure the importance of one frame relative to another based on applying Gaussian smoothing kernel on their difference? Then they measure confidence scores of each frame...something for classification.....

Ok now great I want to try my own approach. What if I use for start 2D cross correlation between frames. So cross-correlation function given as:

$$(F_i \star F_j)[m, n] = \sum_h \sum_w F_i[h, w] \cdot F_j[h + m, w + n] \quad (2)$$

for discrete signals. This tells us how similar two frames are. If this value is big than that means that they are similar, and we want to choose only not similar frames. Now we can normalize this by:

$$\text{Corr}_{ij}[m, n] = \frac{\sum_h \sum_w F_i[h, w] \cdot F_j[h + m, w + n]}{\|F_i\|_F \cdot \|F_j\|_F} \quad (3)$$

Where:

$$\|F_i\|_F = \sqrt{\sum_h \sum_w F_i[h, w]^2} \quad (4)$$

is called Frobenius norm, and we use it in regular scaling, as any other vector. Now the correlation values are all between -1 and 1 and as a similarity score we can use:

$$\text{sim}(F_i, F_j) = \max_{m, n} (\text{Corr}_{ij}[m, n]) \quad (5)$$

and as dissimilarity score we can use:

$$\text{diff}(F_i, F_j) = 1 - \max_{m, n} (\text{Corr}_{ij}[m, n]) \quad (6)$$

Wonderful now this absolutely did not work!

Now i tried scaling it a bit differently:

$$\begin{aligned}\tilde{F}_1 &= \frac{F_1 - \mu_{F_1}}{\sigma_{F_1} + \varepsilon}, & \tilde{F}_2 &= \frac{F_2 - \mu_{F_2}}{\sigma_{F_2} + \varepsilon} \\ \text{Corr}(F_1, F_2)[m, n] &= (\tilde{F}_1 \star \tilde{F}_2)[m, n] \\ \text{diff}(F_1, F_2) &= \frac{\max_{m,n} (\text{Corr}(F_1, F_2)[m, n])}{HW}\end{aligned}\tag{7}$$

where:

- μ_{F_1} and μ_{F_2} are means of F_1 and F_2
- σ_{F_1} and σ_{F_2} are standard deviations of F_1 and F_2
- ε is a small constant for numerical stability (10^{-5}) but I think it might be better without it
- HW is the image size

After that i also tested picking relevant frames with using CLIP: openai/clip-vit-large-patch14 and that makes the embeddings of the size 512. Than we use cosine similarity between those:

$$\text{cosine_similarity}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|}\tag{8}$$

where:

- $\mathbf{u} \cdot \mathbf{v}$ is a dot product between vectors \mathbf{u} and \mathbf{v}
- $\|\mathbf{u}\|$ and $\|\mathbf{v}\|$ are Euclidean norms of vectors \mathbf{u} and \mathbf{v} respectively

Finally I got better result with using CLIP just because the execution lasted 18.12s, and for my manual correlation approach it took 398.04s. However, while observing the video, my approach actually gave better results in terms of choosing relevant frames.

7 Paper: SmolVLM: Redefining small and efficient multimodal models and a video from one of its authors: Link to the Open-source Multimodality video

The lady explains multimodality but I understood that already. Open model means it can be used for commercial purposes. She focuses on VLMs(image/video and text as an input and text as an output). The most famous one is GPT-4.

Vision Language Models	Multimodal Retrieval	Zero-shot Vision
Qwen2VL, Molmo, Pixtral, Llama-Vision, Llava, PaliGemma...	ColPali, MCDSE, ColQwen	CLIP, SigLIP, GroundingDINO, OWL, GroundingSAM..

Figure 13: VLMs summary

What is open-source- you can use them locally and guaranty a complete privacy.

I absolutely did not understand the distillation after student models part (GO OVER BOTH IN DETAIL).

CLIP was explained first but I have that written already

SigLIP is very used today: probabilities are not summed to 1.

Ok I can not write about all of them: go to the video if you need any.

TAKE A LOOK AT THE SEGMENT ANYTHING MODEL 2

Basically this model segments an object in an image but as a text input you need to give it a bounding box, and this is for the first version of this model, but in the second one they have an object tracking in the video and obviously you don't need to give a bounding box for every frame but they track some other way they have this memory system that I want to take a look at but she said it's very useful but very complex.

She also talks about LLaVA, the exact paper I read about and she says that that was kind of the first Vision Language Model that actually worked.

Explaining pretraining and fine-tuning LLaVA. Image encoders could be different, as well as llms.

7.0.1 PaliGema

I think very similar to LLaVA. We still have a image encoder(SigLIP) and a lm(Gemma). They project image and text and concatenate and pass to gemma. This one was used for segmentation. She says that VLMs are better.

Qwen2-VL—in that point state of the art. Comes in multiple sizes. Similar architecture I think. Pixtral, Molmo.

7.0.2 Leaderboard

Open VLM Leaderboard Link for the Open VLM leaderboard

They have a couple of benchmarks like math solving, document understanding, BLINK benchmark: how a model responds to things it sees for a very short time, like humans do. There is a lot benchmarks, and every new model usually use different ones. That is kind of confusing. So basically they test only on certain tasks and then check different benchmarks and finally average them and compare based on that, but that is not something that is necessarily very good and it is questionable whether models could be compared this way. So on each set of benchmarks basically different model is the best.

Again the leaderbord that did not work before Link to the Vision Arena leaderboard still not working.

I did not understand a part of what she said like if our prompt is not included then the model might not perform well on some prompts??

Arena is the one that offers you two models and outputs—it gives you a 'battle' and you choose.

7.0.3 What has changed since LLaVA: Recent Advancements

Multiple image encoders: MiniGemini, mPLUG-DocOwl 1.5, BRAVE. One image encoder is CNN based and another one is Vision Transformer based. Usually they concatenate the output and pass to the projection layer. MiniGemini also has a prompt as an output that could be passed to stable-diffusion in order to get another image, that is very cool.

BRAVE has multiple image encoders and then they pass the outputs of all of them to the model that chooses the best outputs and then they are concatenated and passed to decoder or lm or whatever.

Another advancement is having interleaved input: Make references to images through text? I did not understand. AAAA include image in the text!!!! VIDEOS: **Video-LLaVA**: they take every X frame-downsample and pass it as separate image tokens or as one video token (to the interleaved thing).

LongVU downsamples videos pretty smartly: They pass frames to DINOv2 model and it extracts image features and then compare them and it gets rid of redundant frames based on similarity. But they also compare frames to the text prompt (after comparing them with one another). This is the state of the art. BUT SHE SAYS ALSO THAT WE USUALLY JUST TOOK A FEW FRAMES UNIFORMLY CHOSEN.

Another advancement is different pretraining and post training(fine-tuning) setups. Whether image encoder is pretrained and llm. There is one paper that train everything. Or mix something like first fix an image encoder, but later retrain it. That is done(retraining of a vidion encoder) in case we need to align the encoding to the specific domain(like if you are doing with MRAs or some other medical imaging).

Another thing is zero-shot vision tasks!

Also VLMs could be used for retrieval and that could be used for Multimodal RAG. This is basically description of documents.

You can use any open-source Hugging Face model and fine-tune it. PaliGEMMA is very used.

7.1 Video processing

Qwen2.5VL can handle long context and is adapted to dynamic FPS rates, as the model is trained with videos with different frame rates. Through extended multimodal RoPE, it understands the absolute time positions of frames, and can handle different rates and still understand the speed of the events happening in real life. Another model is Gemma 3, which can accept video frames interleaved with timestamps in text prompt, e.g. “Frame 00.00: jimage;..”, and is very performant for video understanding tasks.

Paper about Qwen2.5-VL Technical Report (DID NOT READ IT)

8 Paper SmolVLM: Redefining small and efficient multimodal models

So larger models are not very practical because they require a lot of resources and could be used on mobile and edge devices. SmolVLMs is not only about architecture but also tokenization way. SmolVLM-256M, uses less than 1GB GPU memory during inference and outperforms the 300-times larger Idefics-80B model, despite an 18-month development gap. The largest model, at 2.2B parameters, rivals state-of-the-art VLMs consuming twice the GPU memory. (Curated data mean selected data so this is like a way of sampling).

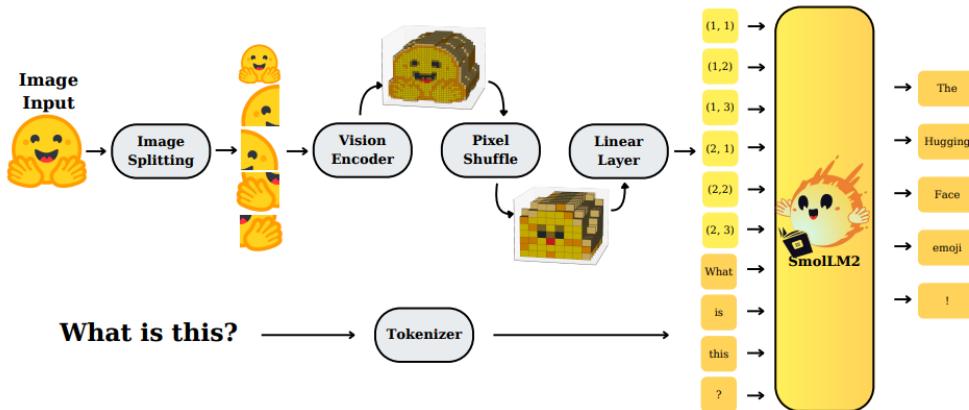


Figure 2 | SmolVLM Architecture. Images are split into subimages, frames are sampled from videos, and then encoded into visual features. These features are first rearranged via a pixel-shuffle operation, then mapped into the LLM input space as visual tokens using an MLP projection. Visual tokens are then concatenated/interleaved with text embeddings (orange/red). This combined sequence is passed to the LLM for text output.

Figure 14: SmolVLM architecture taken image from the paper

The only difference that I see so far in comparison to regular VLMs is that the image is split at the beginning.

You have all the Relative work on VLMs in the paper.

- **Compact yet Powerful Models:** We introduce SmoVLM, a family of powerful small-scale multimodal models, demonstrating that careful architectural design can substantially reduce resource requirements without sacrificing capability.
- **Efficient GPU Memory Usage:** Our smallest model runs inference using less than 1GB GPU RAM, significantly lowering the barrier to on-device deployment.
- **Systematic Architectural Exploration:** We comprehensively investigate the impact of architectural choices, including encoder-LM parameter balance, tokenization methods, positional encoding, and training data composition, identifying critical factors that maximize performance in compact VLMs.
- **Robust Video Understanding on Edge Devices:** We demonstrate that SmoVLM models generalize effectively to video tasks, achieving competitive scores on challenging benchmarks like Video-MME, highlighting their suitability for diverse multimodal scenarios and real-time, on-device applications.
- **Fully Open-source Resources:** To promote reproducibility and facilitate further research, we release all model weights, datasets, code, and a mobile application showcasing inference on a smartphone.

Figure 15: Contributions

First thing they try is to investigate optimal capacity allocation between vision encoders and language models: Finding 1. Compact multimodal models benefit from a balanced encoder-LM parameter allocation, making smaller vision encoders preferable for efficiency.

They adopt a self attention architecture where they combine visual embeddings with textual ones and process them together in a lm. Ok the problem is that the small LM they want to use SmoLLM2 has a 2k token limit, and if they use SigLIP-B/16 for visual encoding of an image 512x512 they get an 1024 token output which is probably too much so they need to reduce that somehow. They increased Base of RoPE Bounds Context Length and fine-tuned the model on long-context texts and short-context ones.

I really don't understand the token thing. But i think the thing is context window size which is the textual input size(in tokens). It would be good to have a compression method for tokens. One proposed one is **pixel shuffle** whose idea is to extend spatial dimension but reduce number of visual tokens per one dimension: I DO NOT UNDERSTAND THE POINT. But balancing token sizes in videos and images is crucial because images need higher resolution and videos can work with smaller resolutions per frame so they can process longer sequences, but both would be ideal. The idea for images is to split image in a few parts and process each separately. This approach proved effective in maintaining image quality without excessive computational overhead. For videos, however, we found that strategies such as frame averaging, negatively impacted performance. Consequently, frame averaging was excluded from SmoVLM's final design, and video frames were instead rescaled to the resolution of the image encoder. The video frame averaging was proposed by NVILA: Efficient Frontier Visual Language Models .

They also included positional tokens for image parts.

Finding 6. System prompts and media intro/outro tokens significantly improve compact VLM performance, particularly for video tasks. During SFT, only train on completions.

SmoVLMs are called smol because of tiktok like a smol cat.....

Curated Training Data: By carefully selecting and organizing the training data, they ensured that the model learns effectively from a smaller dataset, which contributes to its efficiency.

- Modular Architecture

SmoVLM separates the model into components: visual encoder, language model, and a multimodal connector.

This helps reduce the overall size and allows reusing off-the-shelf components (like pre-trained LLMs).

- Efficient Visual Tokenizer (SigLIP)

Instead of using large image encoders like ViT-G, they use SigLIP, a smaller image model that still encodes visual info well.

It converts images into a small number of visual tokens.

- Lightweight Language Backbone

For the language side, they use small LLMs like Phi-2 and Mistral-7B, depending on the model size.

These are fine-tuned to understand text in a multimodal context.

- Multimodal Connector

The "bridge" between image and text. It transforms visual features into something the language model can understand.

They use a tiny MLP (multi-layer perceptron) instead of heavy transformers for this.

9 Qwen2.5-VL Technical Report

In this paper they process a video by adding a temporal component of absolute time into a prompt (textual input at the beginning)

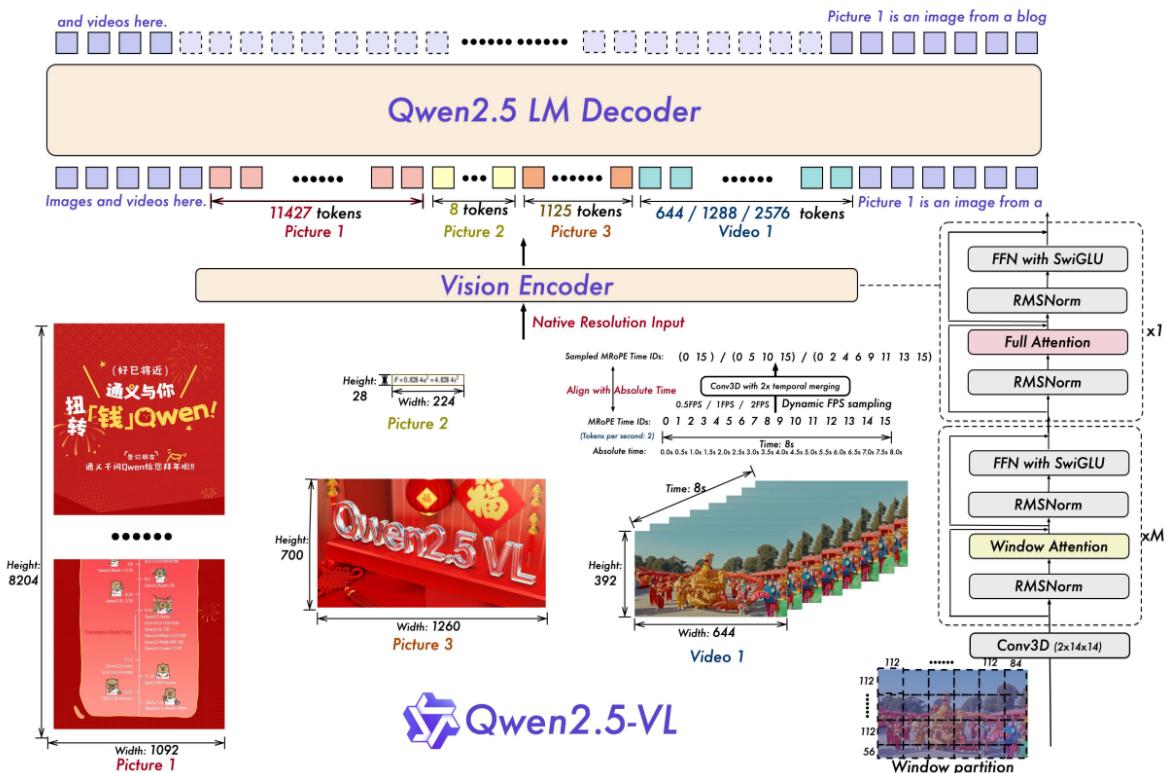


Figure 16: Temporal components

10 Another paper on SmolVLMs: Small Vision-Language Models: A Survey on Compact Architectures and Techniques

They will explain knowledge distillation, lightweight attention mechanism and modality pre-fusion(no idea what that is). So the point of all of these models is efficiency and less resources needed.

Ok they are first talking about regular VLMs and they are used in: visual question answering, image captioning, cross-modal retrieval and document understanding. All of those models require significant resources for both training and inference, which means that they can't be used in environments with less resources, like mobile phones or edge devices or embedded systems(I LOVE THAT). Because of that SmolVLMs emerged, and the idea is that they should use less resources but with good performance and computational efficiency.

10.1 Historical Overview

10.1.1 CLIP: Contrastive Language-Image Pre-training

I went over this already but this is one of the first multimodal models that combines images and text.-It is a training through natural language supervision. CLIP performs zeros-shot classification on over 30 different datasets. CLIP struggles with fine-grained classification (this is like a very picky classification, inside of a class, example: instead of saying this is a bird it has to choose a species of a bird), abstract reasoning(like finding patterns), and out-of-distribution generalization(a seen task but the distribution changes), particularly in handwritten digit recognition. Some more downsides the point is needed improvement.

10.1.2 Vision-and-Language Transformers (ViLT) ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision

This approach eliminates CNNs altogether. Images are divided into 32x32 patches, linearly projected, and combined with textual embeddings. This type of visual embedding lacks fine-grained object representation. This limits detailed semantic understanding.

10.1.3 VirTex VirTex: Learning Visual Representations from Textual Annotations

Learns visual representation using semantically rich captions, instead of large labeled datasets (so I guess instead of using): ResNet-50 visual backbone and a bidirectional transformer-based textual head.

10.1.4 FLAVA FLAVA: A Foundational Language And Vision Alignment Model

Vision and language alignment model. It has a transformer based architecture. This basically uses the previous architecture, ViT for the image encoding, and the same type encoding for the textual design. It is a very hard learning task if I understood it correctly because everything is learned at the same time.

10.1.5 SimVLM SimVLM: Simple Visual Language Model Pretraining with Weak Supervision

No idea what is happening.

10.1.6 BLIP BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation

11 Attention: Link to the video

If you would use an encoder decoder network to encode a whole text that would not be very good because the whole text would be shown as one vector (let's say that you tried an LSTM) and now you would probably

want to encode words or parts of text separately and to make some similarity between them, which i guess is the whole point of attention. In LSTMs the first word might become forgotten. That might change the whole meaning. So long term memory part might loose some parts. So it would be good that every word has its own encoding and then make connections between them. Embedding layer of a text is still like a very simple network such as word2vec basically. A context vector is the embedding after the embedding layer. And in LSTM all of the input is combined into context vector.

We start decoding with EOS_t or SOS_t , BUT we will still be using LSTM outputs, so we have a long term memory output and short term memory. We want a similarity score from the first step of the encoder and the first step of the decoder (the short term memory output) SO it would make sense to make some sort of cosine similarity which is technically what we are going to do. Just before that we will add some matrices. But also a similarity of a secodn output of an encoder and first output of a decoder. Different attention algorithms use different ways to compare similarities—so a cross corelation is a good idea after all. But here we will use cosine similarity: The result is between -1 and 1. You can also just use a dot product. We will

$$\text{Cosine Similarity} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

Figure 17: Cosine similarity

use that. -1 means that embeddings are completely opposite. The score for 'go' and EOS_t is higher which is why we want the word go to have more influence on the first word that comes out of the decoder. We do a little softmax-probability for decoding. So now we multiply those probabilities with the outputs again, and we sum them. Now that goes to FCL, as well as the output of a decoder together. After that we use softmax and we get 1 for the output word. I don't know if I understood it completely but the idea is there and I understood it when he used this with transformers. Now we move on with decoding until we get EOS_t at the output (in training until we expect it). Now we don't need LSTMs anymore at all.

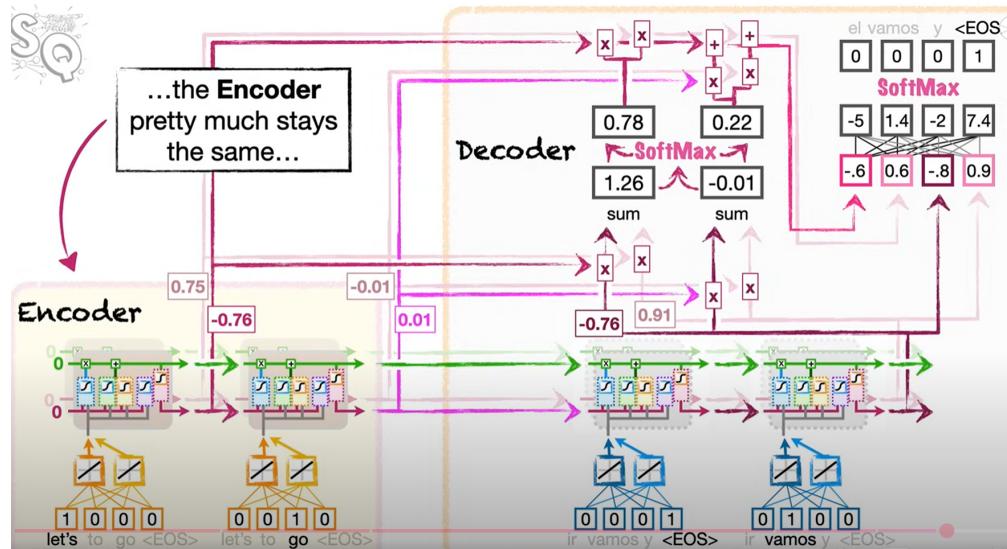


Figure 18: Attention

12 GO BACK TO THE SmoVLM PAPER IT HAS A PART ON VIDEOS BUT NOW I AM GOING THE REMAINING FEW LINKS BECAUSE ONE OF THEM IS ABOUT VIDEO PROCESSING AND THEN I WILL COME BACK TO IT

13 Blog SmoVLM - small yet mighty Vision Language Model

SmoVLM is actually one instance of a small vision language model and it is from HuggingFace. Some small vlms include: PaliGemma 3B, moondream2, and Qwen2VL. They present their model called SmoVLM with 2 billion parameters. COOL THING: Hugging Face offers open training pipelines that means that you can use their architecture but also change it if you wish to.

They present three models:

- SmoVLM-Base: good for downstream fine-tuning(train it for special jobs)
- SmoVLM-Synthetic: fine-tuning on synthetic data(it is also trained on synthetic data)
- SmoVLM Instruct: fine-tuned instruction variant(no need to fine-tune it it is good at following instructions for different tasks)

These models are open-source and they are integrated in the transformers library! They used The Cauldron and Docmatix datasets for training.

They also have a demo for the Instruct model: Demo: WORKS VERY GOOD IT'S COOL!!!!

13.1 Architecture

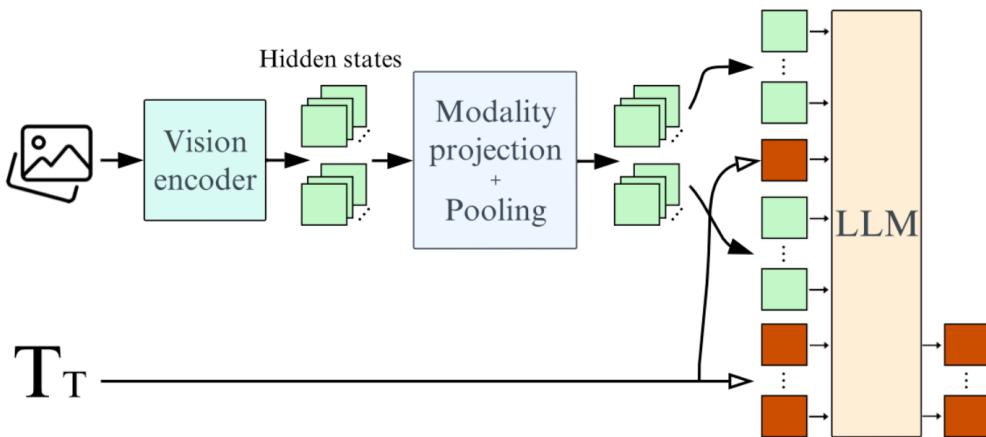


Figure 19: SmoVLM Architecture

First they followed the same architecture from Idefics3, they used the same transformers, and added a few changes.

READ(BECAUSE THE SAME IMAGE IS TAKEN FORM THESE PLACES AND EXPLAINED BETTER): What matters when building vision-language models? and Building and better understanding vision-language models: insights and future directions but the pooling part means that the output of the encoder is first projected to smaller dimensions (let's say with a matrix, but it's for sure something much more complicated) and then you do the pooling so you reduce the vector size by taking maximum values (max pooling) or average some of them (average pooling).

The changes they made in the original architecture:

- They replaced Llama 3.1 8B with SmolLM2 1.7B for the language model, so a smaller architecture
- They reduced the compressed visual information by reducing it 9 times using pixel shuffle strategy.
- They use patches 384x384 because it is divisible by 3, which they need for the pixel shuffle
- The vision encoder is SigLIP with patches 384x384 and inner patches of 14x14.

This next part is chat gpt explanation of pixel shuffle: it is used in super-resolution tasks, like making images sharper. Ok so what this basically does is it makes more channels from one channel(hight x width) and then processes them separately. They use this to compress (reduce resolution) visual patches. So you take a patch of 384x384. A 3x compression reduces this in 9 channels of the size 128x128. So every 3x3 box from the original image is split in 9 pixels(3x3) and each goes to another channel (OFC THIS IS IF WE ARE TALKING ABOUT GRayscale).

A patch is a splitting of an image, ok I knew that. So every channel is divided in block of the size 14x14. Actually in the paper it says YOU FIRST SPLIT THE IMAGE AND THEN YOU

Models have positional encoding in order to know which patch is where. The positions are encoded into vector and added to the embedding of a patch. FIND OUT WHERE IS THIS ADDED!!!!!!!

Ok back to the paper they offer a benchmark:

Model	MMMU (val)	MathVista (testmini)	MMStar (val)	DocVQA (test)	TextVQA (val)	Min GPU RAM required (GB)
SmolVLM	38.8	44.6	42.1	81.6	72.7	5.02
Qwen2-VL 2B	41.1	47.8	47.5	90.1	79.7	13.70
InternVL2 2B	34.3	46.3	49.8	86.9	73.4	10.52
PaliGemma 3B 448px	34.9	28.7	48.3	32.2	56.0	6.72
moondream2	32.4	24.3	40.3	70.5	65.2	3.87
MiniCPM-V-2	38.2	39.8	39.1	71.9	74.1	7.88
MM1.5 1B	35.8	37.2	0.0	81.0	72.5	NaN

Figure 20: SmolVLM Benchmarks

SmolVLM has the best memory usage for one and two images. It encodes a image of the size 384x384 to 81 tokens(vectors). That means that it can encode an image and a textual prompt into 1.2k tokens, while Qwen2-VL uses 16k tokens. This is why in the next graph for Qwen2-VL the memory increases a lot for two images, and not that much at all for SmolVLM and PaliGemma that uses similar approach. Because the image encoding takes a very small space:

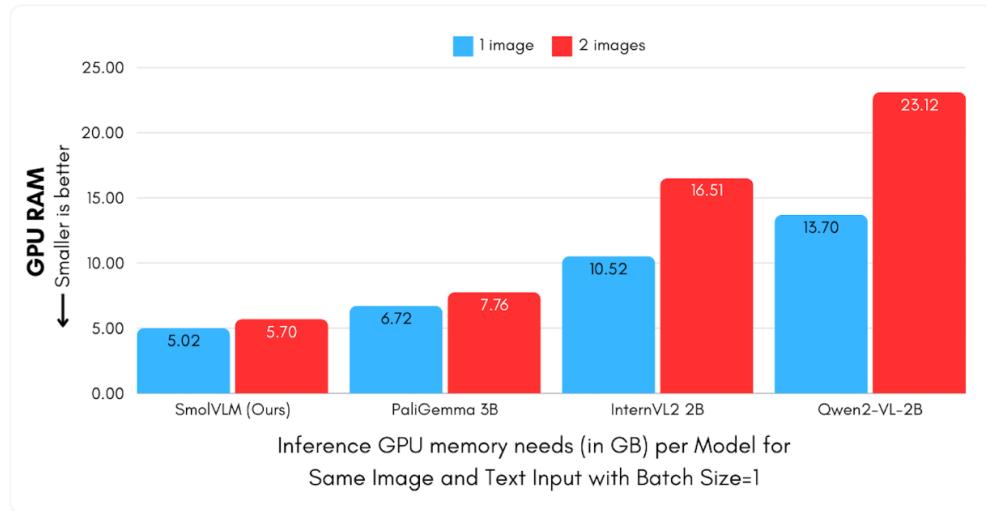


Figure 21: SmoLVM Memory usage

This model is a lot faster in comparison to Qwen2-VL

13.2 Video

They made a video processing pipeline: Link to their GitHub repository where they extract up to 50 evenly sampled frames, avoiding internal frame resizing. That means no resizing or anything while they process a video AND THEY DO NOT HAVE A SMART WAY OF EXTRACTING VIDEO FRAMES!!! Other models used are InterbVL2 and Video LLaVA. They got ok results but this could be done better. This has a problem with time.

13.3 Evaluation: VLMEvalKit

They also have in the blog a line to integrate evaluation kit into fine-tuning. For example, to evaluate on MMMU dev validation set and MathVista mini and store the results in a folder called smol.

13.4 Use with transformers

You have a code—very simple just import transformers. They also gave a code of testing the model, I will try it.

13.5 Training Details

They first had to extend the context of a SmoLVM2 which is a language model and they will explain why but I guess that 81 token input is too long for this model. I have already written datasets they used! Exactly the SmoLVM2's pre-trained context window is too small for VLM, because images are embedded into many tokens and we want to support many images. They extended it by increasing the RoPE based value from 10k to 273k. NO IDEA WHAT DOES THIS MEAN THEY ARE FOLLOWING A CERTAIN GUIDELINE: “Scaling Laws of RoPE-based Extrapolation”. Now they are explaining how they trained the model and I don't really know what they are talking about but I think those are just some datasets.

They tracked the training over 25 checkpoints and in each checked:

- General multimodal understanding (MMMU and MMStar) which are the most comprehensive benchmark.

- Document and text-based visual question answering (DocVQA and TextVQA)
- Mathematical Reasoning (MathVista)
- Diagram understanding (AI2D)

So for a final metric they made some weighted averaging over these metrics and based on that chose the optimal checkpoint.

13.6 Fine-tuning with TRL

They gave a notebook: Link to the notebook for fine-tuning on VQAv2 dataset. There some tips and tricks on how to save up even more memory to fit the training in L4. They use LoRA and QLoRa, No idea what they are. Anyway you need like 16 GBs of VRAM. THIS MEANS YOU CAN USE COLAB YEEEEEEEEEEY! SmolVLM also comes with TRL integration so you can apply Direct Preference Optimization (DPO) easily through the CLI. Also no idea What does that mean....

13.7 IMPORTANT: HOW TO CITE THE BLOG

@articlemarafioti2025smolvlm, title=SmolVLM: Redefining small and efficient multimodal models, author=Andrés Marafioti and Orr Zohar and Miquel Farré and Merve Noyan and Elie Bakouch and Pedro Cuenca and Cyril Zakka and Loubna Ben Allal and Anton Lozhkov and Nouamane Tazi and Vaibhav Srivastav and Joshua Lochner and Hugo Larcher and Mathieu Morlon and Lewis Tunstall and Leandro von Werra and Thomas Wolf, journal=arXiv preprint arXiv:2504.05299, year=2025

14 Medium blog: SmolVLM: A Small Yet Mighty Vision Language Model

Once again the 2.2 billion parameter model is the best, but smaller ones work cool too. Most of the people have Sharpay Evans thinking 'bigger is better', but they are not always correct. Except Sharpay of course. I think this blog is just paraphrasing the previous one.

Now this next part is specifically for videos:

So we have three model sizes:

- SmolVLM2–2.2B: Our flagship model for robust vision and video tasks.
- SmolVLM2–500M: A leaner version packing near-star performance with only a fraction of the parameters.
- SmolVLM2–256M: An experimental, ultra-compact model pushing the limits of small-scale video understanding.

They talk about the way these models are good, not very important. They mention a couple of demos, but no links and they give a code for testing and fine-tuning- TRY IT.

Available demos are:

- iPhone Video Understanding–app
- VLC Media Player Integration–still in development
- Video Highlight Generator–I TRIED IT, DIDN'T WORK, try here Apps NO THIS IS COOL BUT NONE OF IT IS FOR VIDEO DESCRIPTION BUT THERE IS ONE FOR MAKING A VIDEO FROM AN IMAGE!

15 Hugging Face Blog on videos: SmolVLM2: Bringing Video Understanding to Every Device

Again the goal is to make models that would be accessible across all devices and use cases (SMALLER). Three sizes: 2.2B, 500M, 256M. All available here: [Link to models](#). A demo for SmolVLM2.2B is available here: [SmolVLM2: The Smallest Video Model Ever DOESN'T WORK RIGHT NOW FOR ME I'LL TRY LATER](#).

The two smaller models, 256M and 500M ones are(in the February last year) the smallest video language models ever released. Video-MME is the go-to scientific benchmark in videos(because it has videos from 11 seconds to 1 hour, it has multiple data modalities including audio and subtitles, and high quality expert annotations. [Link: Video-MME](#)).

15.1 SmolVLM 2.2B

This model workd very well with images: solving math problems with images, reading text in photos, understanding complex diagrams, and tackling scientific visual questions. In Video-MME it's better than all 2B models (but the best models are still large ones)

Read the Apolo paper and blog: Apollo: An Exploration of Video Understanding in Large Multimodal Models: [THIS IS THE BLOG](#). You also have available code for testing: try it, it's very simple.

15.2 500M and 256M

HuggingFaceTB/SmolVLM2-500M-Video-Instruct works almost as good as the 2.2B one and it's a lot smaller. The smallest one: SmolVLM2-256M-Video-Instruct This one is still in the development and it's nice to explore it.

15.3 Demo applications

- iPhone Video Understanding-not released but it's using SmolVLM 500M
- VLC media player integration, using using SmolVLM2, also in progress, but there is a prototype: [Link](#). In this you can navigate through a video and find relevant parts.

You can find tutorials how to use SmolVLM2 with transformers. And then how to do the inference as well. Also for MLX but I have no idea what that is.

There is also a notebook for fine tuning: Fine-tuning notebook, but it is invalid. It says also it should be good to do the fine-tuning of the whole model instead of QLoRA or LoRA. TAKE A LOOK AT WHAT THAT IS!

16 I will try inference for both video and image now!

So on image inference testing I tried SmolVLM-Instruct and it works crazy bad.....like I inserted images of digits(from MNIST but more of them) and it said blocks.....

SmolVLM2.2B for video description is cool, this model is splitting into 1s shots and I assume takes one frame from each second.

After running the code from the GitHub the description for my video:

```

Describe this video
Assistant: The video features a person standing in front of a whiteboard with various notes and diagrams written on it. The person is wearing a white t-shirt with the word "NOS" printed on it. The whiteboard is filled with text and diagrams, including mathematical equations, scientific notations, and diagrams. The person appears to be speaking or presenting, as indicated by their hand gestures and the way they are looking at the camera. The background is a plain white wall, and there are no other objects or people visible in the video. The person's movements are deliberate and controlled, suggesting they are engaged in a presentation or lecture. The video does not show any significant changes in the person's appearance, the whiteboard, or the background throughout the video.
Model loaded in 9.53 seconds.

```

Figure 22: The output of SmolVLM2.2B on my video

The inference took 55.15.

The next image is the output for Chanel video:

```

Describe this video
Assistant: The video begins with a close-up of a bottle of Chanel Coco Mademoiselle perfume, featuring a transparent cap and a label with the brand name and location. The bottle is placed on a dark surface, with a gold chain and a small pendant visible in the background. The scene transitions to a woman in a black dress descending a grand staircase, carrying a black handbag. She enters a room with a wooden door, which has a sign reading "COCO MADEMOISELLE" above it. The woman looks around the room, which has a warm, dimly lit atmosphere. The video then shows a close-up of the woman's face, with the "COCO MADEMOISELLE" sign visible in the background. The final scene features a close-up of the perfume bottle against a white background, with the text "coco mademoiselle" written above it. The video concludes with a close-up of the perfume bottle.
Model loaded in 10.85 seconds.
1 inferences completed in 67.53 seconds.
Average inference time: 67.53 seconds.
PS C:\Users\marij\TelecomSudParis\video-model-research> []

```

Figure 23: The output of SmolVLM2.2B on Chanel video

THIS NEXT PART IS WROENG BTW:

In `transformers->utils->image_utils.py->make_batched_videos` a list of videos is converged into a single video.

For video reading: `transformers->utils->image_utils.py->read_video_opencv` it says if you want to sample frames in a different way than one each second you should define your own `'sample_indices_fn'` function. SO A PROBLEM:

If I want to choose frames based on similarity but I want the similarity to be calculated after vision encoding how can i do that???????

OK I WILL GET BACK TO THIS CODE FOR NOW I WILL MOVE TO PYSCENEDETECT!

17 PySceneDetect

PySceneDetect detects changes in scenes in a few different ways:

- Adaptive Content Scene Detection: HSL color space (hue, saturation, lightness)(device independent), rolling average of differences, dynamic thresholding (the rolling average is there so some fast camera changes could be ignored)
- Content-Aware Scene Detection: HSL differences combined with filtering(maybe like edge extraction)
- Content-Aware Scene Detection: perceptual hashing (hashes an image so i guess encodes it somehow and then looks for differences between hashes)
- Content-Aware Scene Detection: YUV and uses Y component and basic thresholding
- Threshold Scene Detection: average intensity thresholding (I am not sure what intensity is it using here???????)

THESE IMAGE PROCESSING TASKS ARE SUPER COOL, THEY CAN USE OTSU THRESHOLDING OR SOME EROSION OR SMTH LIKE THAT!

You have a code how to install and use. And also added changes.

I will try to use python api.

Additionally, content aware detector can track changes in edges(this is the filtering part 100%, it could be done by Kenny! much better), the good starting point for thresholds are also given.

Most of the methods actually use threshold-based detector that compares the intensity of a frame with a threshold. AVERAGES RGB! NOT GOOD!!!!!!!!!!!! IT IS DEVICE DEPENDENT.

Histogram detector is in Y space(luminance). HISTOGRAMS ARE NORMALIZED AND COMPARED USING CORRELATION THAT IS AMAZING! And that should be lower or higher than a threshold.

Perceptual Hash Detector—there is a certain hashing algorithm, it also converts the frame to grayscale.
NOW I WILL LOOK AT THE CODE I TRY TO ACTUALLY USE IT BUT I'M DOING PYTHON API!
Whole scenedetect package i think is split into .py files with different classes.

WORKS!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

AdaptiveDetector so far works the best. It split that one scene correctly, but that is honestly against what I would expect because it is averaging(rolling but still) differences over frames so I would expect it to be less sensitive to subtle changes.

HistogramDetector also works the same.

I am trying to reduce the threshold for ContentDetector!

ThresholdDetector: It does not work very well. The default threshold does not work at all and it needs to be higher which is unexpected. I assume it is because of the set minimum length in frames of one shot, and if the threshold is too low it is extracting way too short shots. But I also can not seem to find the right threshold.

I can not find a function that is returning indexes for one or a few frames so I will do it manually, but I should check that still.

18 Video summary

So I used HashDetector from PySceneDetector (but all of them gave me the same result for the Chanel video) and I took the first frame from the scenes from the output. I concatenated all of them into a new video and that was my new input into SmolVLM. However, SmolVLM still took only one of the frames and described it, rather than the whole new video, so the output description is:

```
Frame from 00:00:  
  
Describe this video  
Assistant: The video begins with a close-up of a bottle of Chanel No. 5 perfume, which is a classic fragrance known for its elegant and sophisticated scent. The bottle is a transparent glass cylinder with a gold cap, and it is adorned with a gold chain that is draped over it. The background is dark, which makes the bottle stand out prominently. The bottle is positioned on a black surface, which contrasts with the bottle's gold color. The video then transitions to a close-up of a hand holding a small, round, gold-colored object. The object appears to be a key, and it is being held in front of the Chanel No. 5 perfume bottle. The hand is positioned in such a way that the key is prominently displayed in the frame, and the bottle is still visible in the background. The video concludes with the hand still holding the key, and the bottle of Chanel No. 5 perfume is still visible in the background.  
Model loaded in 9.81 seconds.  
1 inferences completed in 33.56 seconds.  
Average inference time: 33.56 seconds.  
PS C:\Users\marij\TelecomSudParis\video-model-research> █
```

Figure 24: The output of SmolVLM2.2B on a sum of a Chanel video

Qwen does pretty much the same thing:

```
warnings.warn(
Prediction: ['The video features a close-up shot of a perfume bottle with the label "COCO MADEMOISELLE" from the brand Chanel. The bottle is placed on a surface, and the background is dark, creating a dramatic and elegant atmosphere. The lighting highlights the intricate details of the perfume bottle, such as the gold chain and the label. The overall composition of the video is simple yet sophisticated, with a focus on the Chanel perfume bottle.']
Model loaded in 20.92 seconds.
1 inferences completed in 63.70 seconds.
Average inference time: 63.70 seconds.
PS C:\Users\marij\TelecomSudParis\video-model-research> []
```

Figure 25: The output of Qwen on a sum of a Chanel video

19 Apollo: An Exploration of Video Understanding in Large Multimodal Models The paper done by Meta, but is also done at the end of last year

They say that video LMMs are not researched enough and they are talking about sampling, architectures and so on....So they say that they proved that fps sampling during training is preferable to uniform frame sampling. (The difference is that one of them is based on time(so a frequency is defined) and the other one is you take every (example) 5th frame-has nothing to do with time). A model they are proposing is called Apollo: Apollo-3B that hs 55.1 on LongVideoBench(CHECK IS THAT A METRIC OR A BENCHMARK OR WHAT IS IT), and Apollo-7B (at the time) is state of the art for 7B models with 70.9 MLVU and 63.3 on Video-MME. (now on 7B is Video-LLaVA LOOK AT IT).

Take a look at the referenced paper I think there might be a few useful ones.

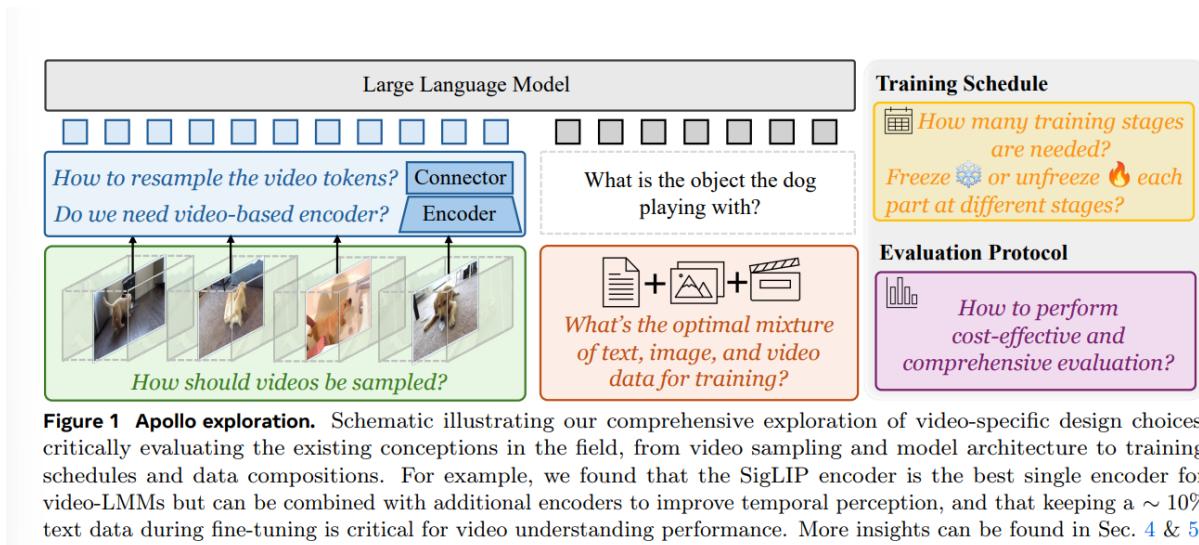


Figure 1 Apollo exploration. Schematic illustrating our comprehensive exploration of video-specific design choices; critically evaluating the existing conceptions in the field, from video sampling and model architecture to training schedules and data compositions. For example, we found that the SigLIP encoder is the best single encoder for video-LMMs but can be combined with additional encoders to improve temporal perception, and that keeping a $\sim 10\%$ text data during fine-tuning is critical for video understanding performance. More insights can be found in Sec. 4 & 5.

Figure 26: Apollo Exploration

You can see at the description of this image it say they proved that the best vision encoder for video is SigLIP (SmolVLM uses that) but it can be somehow improved! They also worked with smaller LMMs and assumed that the same relative results will be on larger ones. What they worked on: addressing essential aspects of video-language modeling, such as video sampling and encoding methods, token resampling and integration strategies, and data compositions. SmolVLM also uses fps but I think that is not really enough.

Apollo models exhibit strong performance across various scales. Notably, Apollo-3B surpasses most existing 7B models, achieving scores of 58.4 (+12.8) on Video-MME (w/o sub.), 68.7 (+6.9) on MLVU, and 62.7 (+14.1) on ApolloBench. Apollo-7B attains impressive scores of 61.2 (+0.6) on Video-MME (w/o sub.), 70.9 (+5.4) on MLVU, and 66.3 (+2.4) on ApolloBench, making it competitive with 30B models.

Our contributions are as follows:

1. We conduct a systematic exploration of the video modeling design space for Large Multimodal Models, uncovering critical factors that drive performance and providing actionable insights for future research.
2. We identify Scaling Consistency, where design decisions effective for smaller LMMs and datasets are transferred effectively to larger ones, reducing computational costs and enabling efficient experimentation.
3. We address evaluation inefficiencies by curating ApolloBench, a subset of existing benchmarks that cuts evaluation time by 41 \times while offering detailed insights into temporal reasoning and perception tasks.
4. We introduce Apollo, a family of LMMs that achieves state-of-the-art results across video understanding multiple benchmarks. Notably, Apollo-3B surpasses nearly all 7B models, while Apollo-7B variant is state-of-the-art among models with less than 30B parameters.

Figure 27: Apollo Contributions

I will skip to video sampling then I will return to the benchmarks.

The point is that most models have maximum number of frames they can use, and in here they only mention frame per second and uniform sampling methods. And maximum number of frames is usually set due to vision encoder and LLM context window.....so i guess they are still concatenating frames.

They say uniform sampling is bad because of video speed...I guess because of dynamic of a video. For a shorter video, N uniformly sampled frames represent a slower playback (more frames per second of actual content), while for a longer video, those same N frames represent a faster playback. So this sampling method is not really helping and like it is making different dynamics of videos. On the other hand they can use N fps and sample M frames this way and encode them together?

Finding 2: fps sampling is preferable over uniform sampling during model training and inference.

Finding 3: There is a trade-off between tps and fps, with 8-32 tokens per frame being optimal.