# Multi Modal LLMs

By Madhava Prasath

# Additional Reads on @himanshu’s Blog where he covered VAE really well

Diffusion Models - bit …

Made with Notion

**Diffusion Models - bit by bit | Notion**

author : @himanshustwts

Himanshu Dubey's Notion on Notion

# Link here

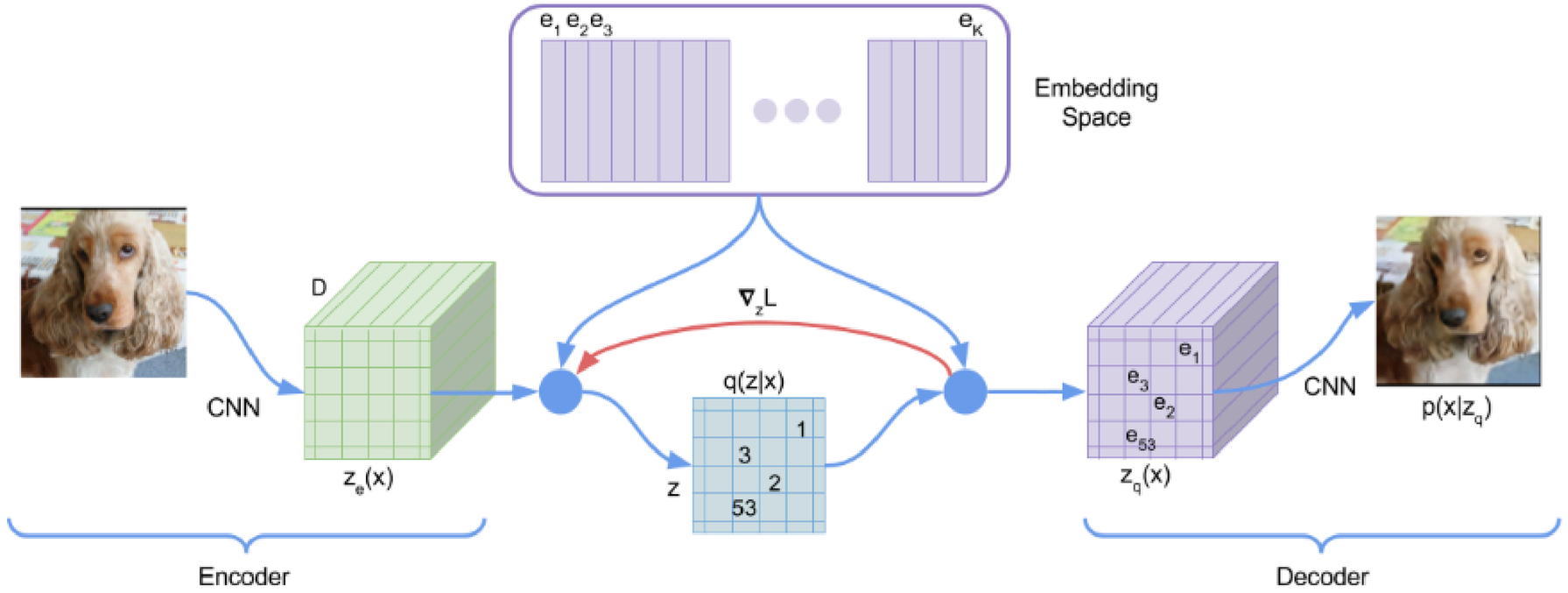
# Table of Contents in Today’s lecture

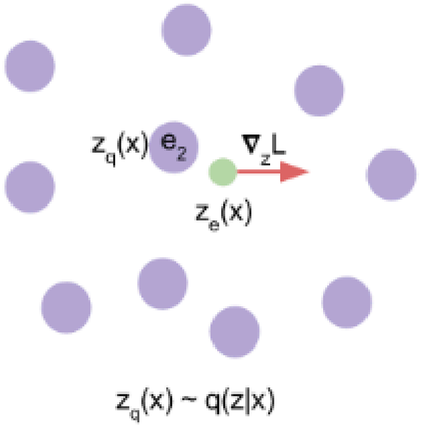
## Prerequisites :

- VQ-VAE
- Decoder mask on LLMs

- Video Poet
- Flamingo
- Llava 3D

$$L = \log p(x|z_q(x)) + \|\text{sg}[z_e(x)] - e\|_2^2 + \beta\|z_e(x) - \text{sg}[e]\|_2^2,$$




K = 512

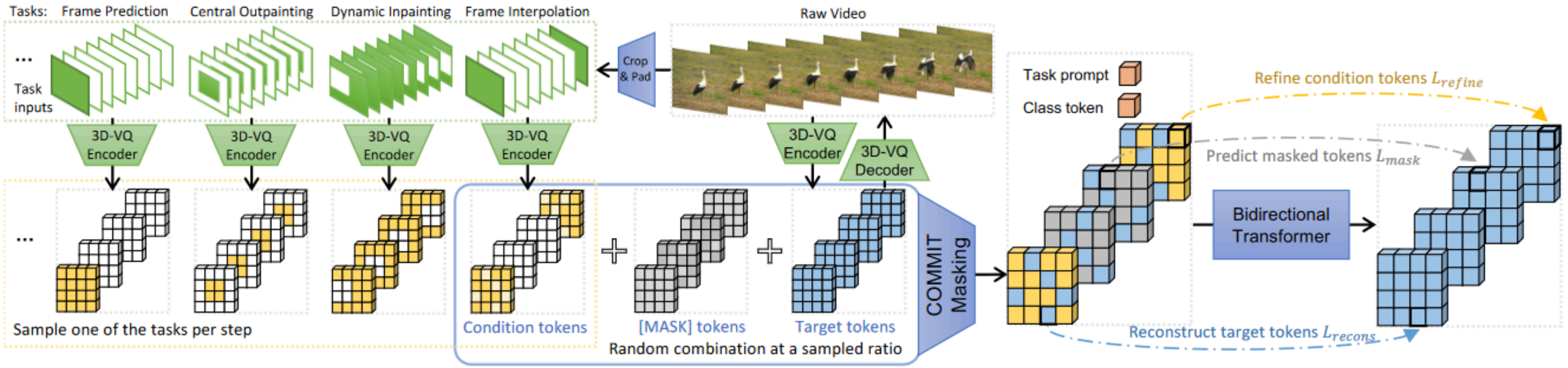
# Why VQ-GAN Matters?

- **Stable Diffusion uses it**
- **A whole new paradigm - Latent Diffusion Models**

# Decoder mask on LLMs

If I get one dollar for each time discussing it I would be rich

# Mag-VIT


Tasks: Frame Prediction
Central Outpainting
Dynamic Inpainting
Frame Interpolation
Task inputs
3D-VQ Encoder
Crop & Pad
Raw Video
3D-VQ Encoder
3D-VQ Decoder
Sample one of the tasks per step
Condition tokens
[MASK] tokens
Target tokens
Random combination at a sampled ratio
COMMIT Masking
Task prompt
Class token
Refine condition tokens $L_{refine}$
Predict masked tokens $L_{mask}$
Bidirectional Transformer
Reconstruct target tokens $L_{recons}$

At a sampled mark ratio, we randomly replace target tokens $\mathrm{z}_i$, with either 1) the condition token $\tilde{\mathrm{z}}_i$, if the corresponding supervoxel of $\mathrm{z}_i$ contains condition pixels; or 2) the special `[MASK]` token, otherwise. Formally, we compute the *multivariate* conditional mask $\mathbf{m}(\cdot \mid \tilde{\mathbf{z}})$ as

$$\mathrm{m}(\mathrm{z}_i \mid \tilde{\mathrm{z}}_i) = \begin{cases} \tilde{\mathrm{z}}_i & \text{if } \mathrm{s}_i \leq s^* \wedge \neg\, \mathrm{ispad}(\tilde{\mathrm{z}}_i) \\ \texttt{[MASK]} & \text{if } \mathrm{s}_i \leq s^* \wedge \mathrm{ispad}(\tilde{\mathrm{z}}_i) \\ \mathrm{z}_i & \text{if } \mathrm{s}_i > s^* \end{cases} \tag{2}$$

$$\mathcal{L}(\mathbf{V};\theta) = \mathop{\mathbb{E}}_{\rho,\hat{\mathbf{V}}} \mathop{\mathbb{E}}_{\mathbf{m}\sim p_{\mathcal{M}}} \left[ \sum_{i} -\log p_\theta(\mathbf{z}_i \mid [\rho, \mathbf{c}, \overline{\mathbf{z}}]) \right] \tag{3}$$

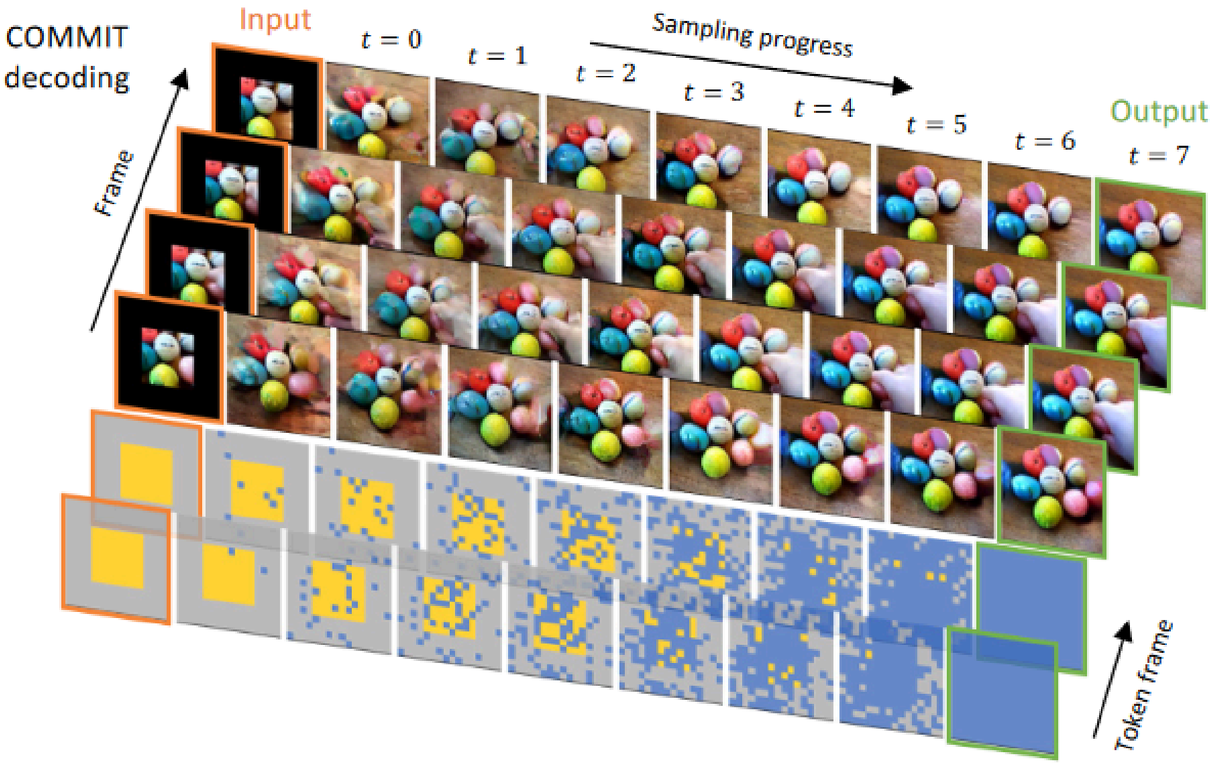
We can decompose the loss in Eq. (3) into three parts according to Eq. (2): $\mathcal{L}_{\text{refine}}$ refines the task-specific condition tokens, $\mathcal{L}_{\text{mask}}$ predicts masked tokens , and $\mathcal{L}_{\text{recons}}$ reconstructs target tokens. Let $\overline{\mathbf{c}} = [\rho, \mathbf{c}, \overline{\mathbf{z}}]$ for simplicity,

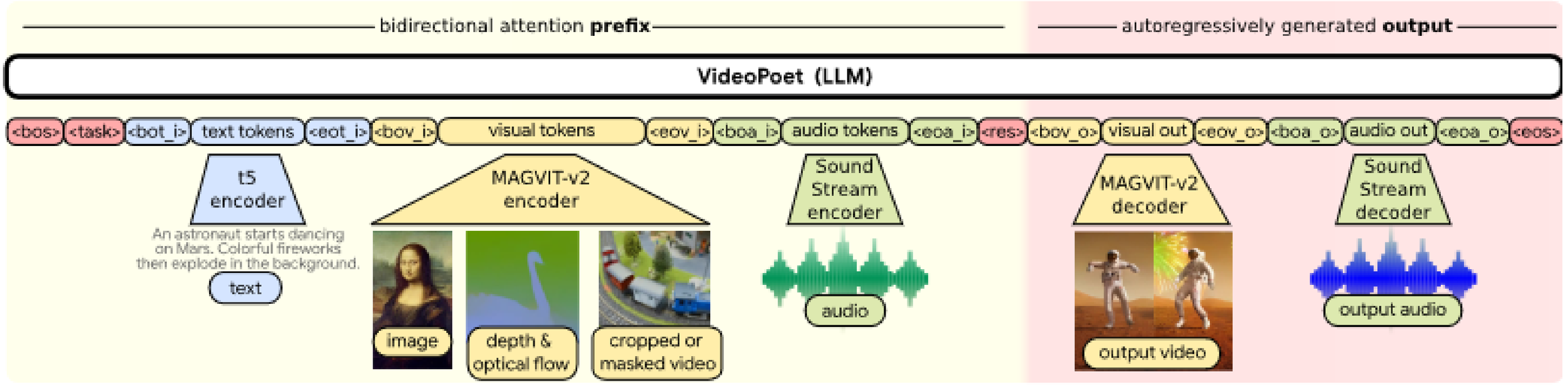
$$\begin{aligned} \sum_{i=1}^{N} -\log p_\theta(\mathbf{z}_i \mid [\rho, \mathbf{c}, \overline{\mathbf{z}}]) &= \underbrace{\sum_{\overline{\mathbf{z}}_i = \tilde{\mathbf{z}}_i} -\log p_\theta(\mathbf{z}_i \mid \overline{\mathbf{c}})}_{\text{Refine condition tokens } \mathcal{L}_{\text{refine}}} \\ &+ \underbrace{\sum_{\overline{\mathbf{z}}_i = \texttt{[MASK]}} -\log p_\theta(\mathbf{z}_i \mid \overline{\mathbf{c}})}_{\text{Predict masked tokens } \mathcal{L}_{\text{mask}}} + \underbrace{\sum_{\overline{\mathbf{z}}_i = \mathbf{z}_i} -\log p_\theta(\mathbf{z}_i \mid \overline{\mathbf{c}})}_{\text{Reconstruct target tokens } \mathcal{L}_{\text{recons}}} \end{aligned} \tag{4}$$

**Algorithm 1** Non-autoregressive Decoding by COMMIT

**Input**: prefix $\rho$ and $\mathbf{c}$, condition $\tilde{\mathbf{z}}$, steps $K$, temperature $T$
**Output**: predicted visual tokens $\hat{\mathbf{z}}$

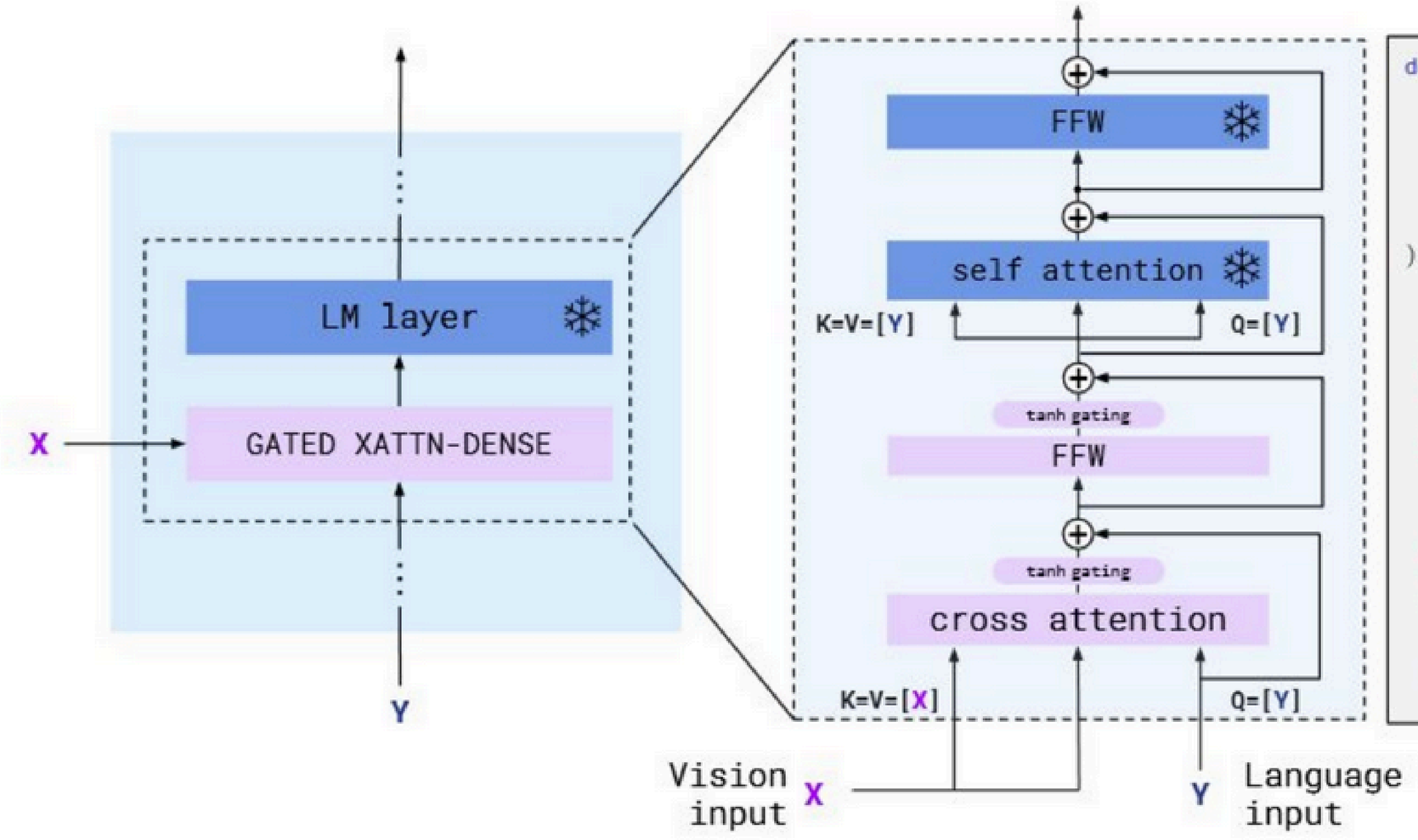
$\mathbf{s} = \mathbf{0}$, $s^* = 1$, $\hat{\mathbf{z}} = \mathbf{0}^N$
**for** $t \leftarrow 0, 1, \cdots, K-1$ **do**
  $\overline{\mathbf{z}} \leftarrow \mathbf{m}(\hat{\mathbf{z}} \mid \tilde{\mathbf{z}}; \mathbf{s}, s^*)$
  $\hat{\mathrm{z}}_i \sim p_\theta(\mathrm{z}_i \mid [\rho, \mathbf{c}, \overline{\mathbf{z}}])$, $\forall i$ where $\mathrm{s}_i \leq s^*$
  $\mathrm{s}_i \leftarrow p_\theta(\hat{\mathrm{z}}_i \mid [\rho, \mathbf{c}, \overline{\mathbf{z}}])$, $\forall i$ where $\mathrm{s}_i \leq s^*$
  $\mathrm{s}_i \leftarrow \mathrm{s}_i + T(1 - \frac{t+1}{K})\,\mathrm{Gumbel}(0, 1)$, $\forall i$ where $\mathrm{s}_i < 1$
  $s^* \leftarrow$ The $\lceil \gamma(\frac{t+1}{K})N \rceil$-*th* smallest value of $\mathbf{s}$
  $\mathrm{s}_i \leftarrow 1$, $\forall i$ where $\mathrm{s}_i > s^*$
**end for**




bidirectional attention **prefix**
autoregressively generated **output**
VideoPoet (LLM)
<bos>
<task>
<bot_i>
text tokens
<eot_i>
<bov_i>
visual tokens
<eov_i>
<boa_i>
audio tokens
<eoa_i>
<res>
<bov_o>
visual out
<eov_o>
<boa_o>
audio out
<eoa_o>
<eos>
t5 encoder
An astronaut starts dancing on Mars. Colorful fireworks then explode in the background.
text
MAGVIT-v2 encoder
image
depth & optical flow
cropped or masked video
Sound Stream encoder
audio
MAGVIT-v2 decoder
output video
Sound Stream decoder
output audio

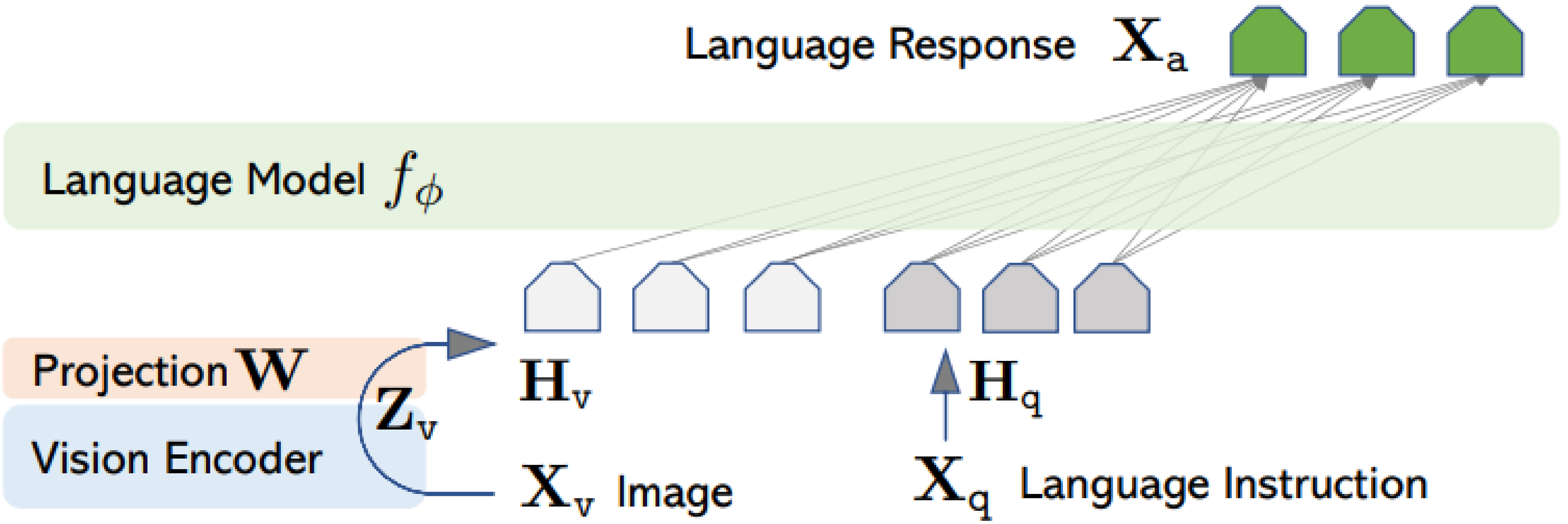
**Lets say you are not rich as Google and You have Vision Encoder and LLM you have to align it**

# Flamingo

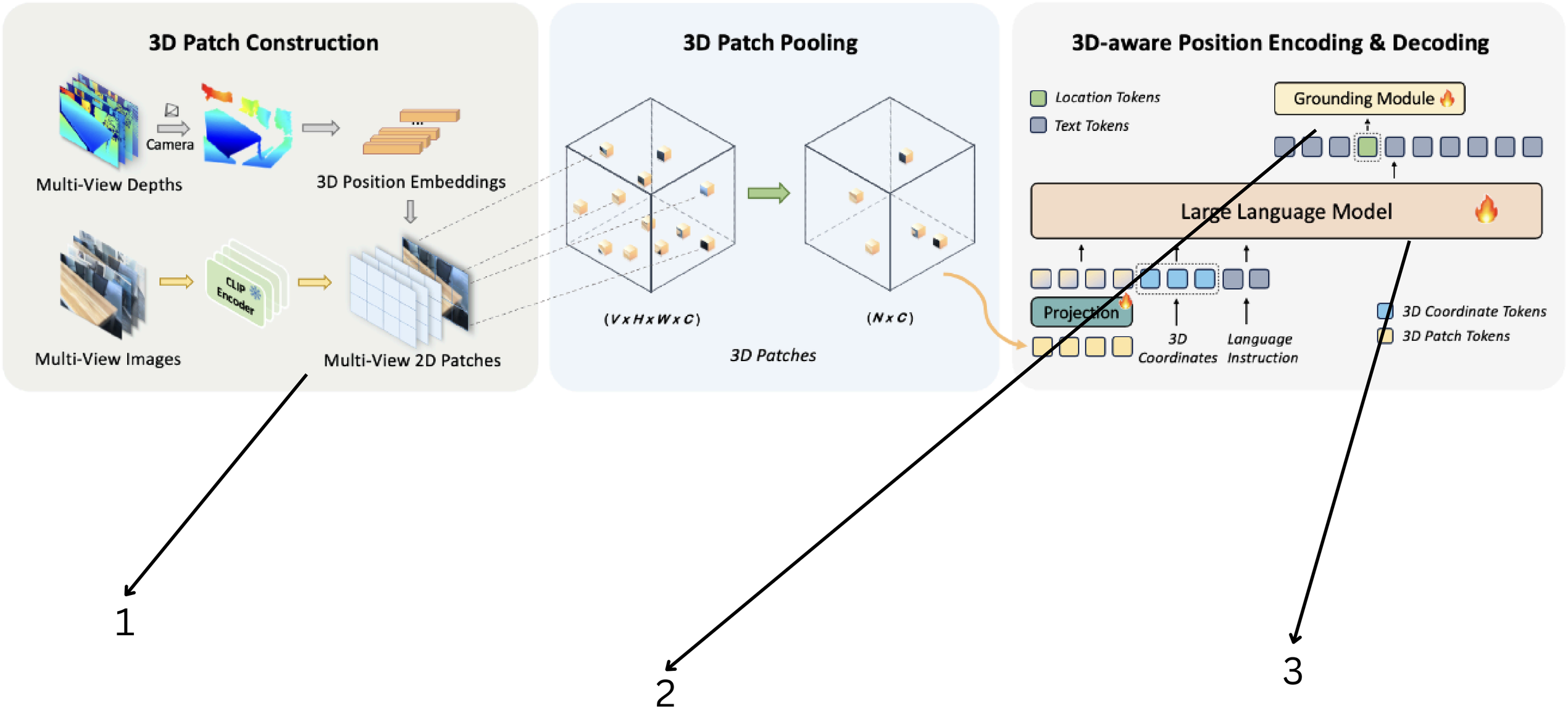



```
def gated_xattn_dense(
    y,  # input language features
    x,  # input visual features
    alpha_xattn, # xattn gating parameter – init at 0.
    alpha_dense, # ffw gating parameter – init at 0.
):
  """Applies a GATED XATTN-DENSE layer."""

  # 1. Gated Cross Attention
  y = y + tanh(alpha_xattn) * attention(q=y, kv=x)
  # 2. Gated Feed Forward (dense) Layer
  y = y + tanh(alpha_dense) * ffw(y)

  # Regular self-attention + FFW on language
  y = y + frozen_attention(q=y, kv=y)
  y = y + frozen_ffw(y)
  return y  # output visually informed language features
```

# Llava Generally


Language Response $\mathbf{X}_a$
Language Model $f_\phi$
Projection $\mathbf{W}$
Vision Encoder
$\mathbf{Z}_v$
$\mathbf{H}_v$
$\mathbf{H}_q$
$\mathbf{X}_v$ Image
$\mathbf{X}_q$ Language Instruction

# Llava - 3D


3D Patch Construction
Camera
Multi-View Depths
3D Position Embeddings
CLIP Encoder
Multi-View Images
Multi-View 2D Patches
3D Patch Pooling
(V x H x W x C)
(N x C)
3D Patches
3D-aware Position Encoding & Decoding
Location Tokens
Text Tokens
Grounding Module
Large Language Model
Projection
3D Coordinates
Language Instruction
3D Coordinate Tokens
3D Patch Tokens
1
2
3

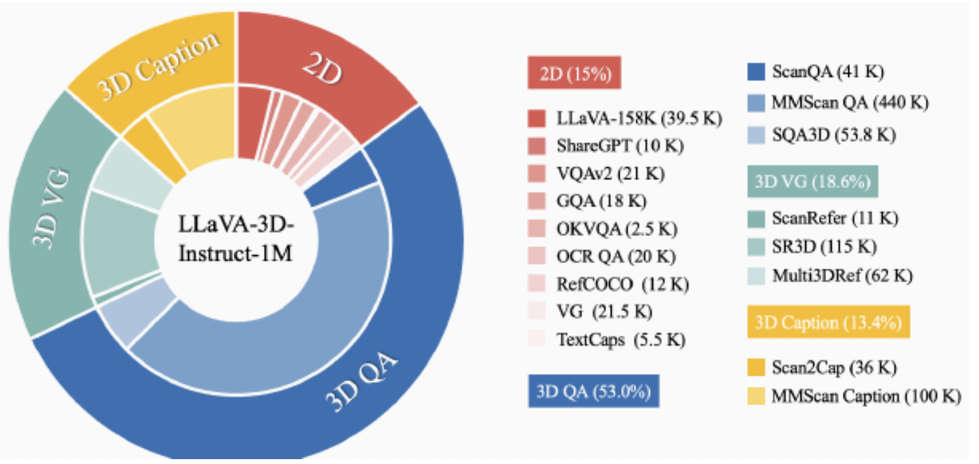
### 3.2. 3D Patch

Our 3D Patch representations are built upon the 2D patch features $X'_v$ extracted from multi-view images with CLIP visual encoder to leverage the strong visual-semantic alignment. To construct the 3D Patches, we inject the 3D position information into the aforementioned 2d patches so that the 3D Patches can explicitly model 3D spatial information while preserving the semantic information from 2D patches. As illustrated in left block of Fig. 2, given the multi-view 2D patch features $X'_v \in \mathbb{R}^{V\times c\times w\times h}$, we obtain their 3D positions $P \in \mathbb{R}^{V\times 3\times w\times h}$ in the 3D world, using nearest neighbor depth and known camera intrinsic and extrinsic parameters. The 3D positions $P$ are then encoded into 3D position embeddings $P' \in \mathbb{R}^{V\times w\times h\times d}$ through a learnable two-layer MLP, which are subsequently added to the 2D patch visual tokens, resulting in the 3D patches $X'_{3D} \in \mathbb{R}^{V\times w\times h\times d}$:

$$X'_{3D} = X'_v + \text{MLP}(P') \tag{1}$$

**Stage 1: 3D Patch Language Alignment.** During the first training stage, we use the region-level and scene-level caption data that describe spatial relationships among 3D objects to align the 3D patches with the LLM for enhanced 3D spatial comprehension. At this stage, the input multi-view images used are selected from sequences that correspond to specific regions or entire scenes. We freeze the vision encoder and LLM parameters, and only train the projection layer and 3D position embedding layer, encouraging efficient alignment between 3D patch features and text space. Since 3D patches are derived from CLIP features augmented with 3D positional information, the alignment between 3D Patch and LLM converges rapidly.

**Stage 2: Task Instruction Tuning.** During the instruction-tuning stage, LLaVA-3D is optimized to respond to complex 3D V&L tasks and maintain its inherent 2D image reasoning and instruction-following capabilities. To facilitate this capability, we collect the **LLaVA-3D-Instruct-1M** dataset, a hybrid collection of 2D and 3D data specifically tailored for instruction tuning. The overall distribution of the dataset collection is shown in Fig 3, more details about the instructional tuning datasets are listed in the appendix. The 2D Data of LLaVA-3D-Instruct-1M is derived from existing LLaVA-1.5 instruction tuning data, ensuring the preservation of 2D image comprehension and vision-language conversation abilities. When tuning with 2D data, we keep the vision encoder frozen and jointly train the projection layer and LLM. The 3D Data of LLaVA-3D-Instruct-1M, on the other hand, comprises data from diverse 3D QA, 3D dense captioning, and 3D grounding tasks. During the 3D data instruction tuning, the 3D position embedding layer will be added to jointly train with the other modules. Additionally, for tasks where the instruction contains 3D coordinate information or requires 3D bounding box outputs, the corresponding encoding and decoding modules will be trained together. During instruc-




tion tuning, the 3D data pathway includes the 3D position embeddings and 3D patches, while the 2D data pathway is the original LLaVA. All modules except for the 3D position embeddings to construct 3D patches are shared across 2D and 3D data. This training setup ensures that LLaVA-3D is capable of processing both 2D and 3D visual tokens effectively, and is adaptive to various task formulations and

## 5.1. Implementation Details

LLaVA-3D is built upon the LLaVA-1.5-7B, utilizing their pre-trained weights from the HuggingFace library. For 3D tasks, we add the 3D position embeddings to the 2D patch visual tokens, and utilize the voxelization pooling strategy to reduce 3D patch number before passing the input visual tokens to the projection layer and LLM. The number of views $V$ is set to be 20 and voxel size is set to 0.2m. Due to the LLM context length limitation, the maximum number of 3D patch tokens after 3D-aware pooling is set to 3096. For 2D tasks, LLaVA-3D functions the same as LLaVA. All experiments are conducted on 8 × 80G A100 GPUs. We train our model for 1 epoch with a learning rate of 1e-3 and a batch size of 32 in stage 1, and fine-tune on the collected LLaVA-3D-Instruct-1M dataset, with a learning rate of 2e-5 and a batch size of 16 in stage 2.

# Cheap self promotion - :)

## A Comprehensive Survey of Mamba Architectures for Medical Image Analysis: Classification, Segmentation, Restoration and Beyond

SHUBHI BANSAL*, Indian Institute of Technology Indore, India
SREEHARISH A*, R.M.D. Engineering College, Kavaraipettai, India
MADHAVA PRASATH J*, R.M.D. Engineering College, Kavaraipettai, India
MANIKANDAN S*, R.M.D. Engineering College, Kavaraipettai, India
SREEKANTH MADISETTY*, Jio Platforms Limited, India
MOHAMMAD ZIA UR REHMAN*, Indian Institute of Technology Indore, India
CHANDRAVARDHAN SINGH RAGHAW*, Indian Institute of Technology Indore, India
GAURAV DUGGAL, Birla Institute of Technology & Science Pilani, India
NAGENDRA KUMAR†, Indian Institute of Technology Indore, India

Paper Link

Github link here

## Awesome Mamba Papers on Medical Domain

This repository includes all the referenced papers, along with the sections covered in the survey titled **A Comprehensive Survey of Mamba Architectures for Medical Image Analysis: Classification, Segmentation, Restoration, and Beyond.** If you encounter any issues with the links or notice any errors, please feel free to reach out via email at 21204028@rmd.ac.in

While some papers may be repeated, our objective is to highlight the papers available in each specific sections. For the best experience, we recommend keeping this repository open alongside the topics you are currently reading. If you wish to explore further, this resource can serve as a useful tool.

**Any doubts ?**

**Feel free to ping me on discord @maddy**

**or**

**Linkedin - Madhava Prasath**