# Driving through Generative video Pretraining:

# VaVIM-VaVAM

Matthieu Cord
Sorbonne Université
June 19, 2025
Joint work with the valeo.ai team:


arXiv > cs > arXiv:2502.15672

Computer Science > Computer Vision and Pattern Recognition

[Submitted on 21 Feb 2025]

**VaViM and VaVAM: Autonomous Driving through Video Generative Modeling**

Florent Bartoccioni, Elias Ramzi, Victor Besnier, Shashanka Venkataramanan, Tuan-Hung Vu, Yihong Xu, Loick Chambon, Spyros Gi
Odabas, David Hurych, Renaud Marlet, Alexandre Boulch, Mickael Chen, Éloi Zablocki, Andrei Bursuc, Eduardo Valle, Matthieu Cord

# World Model
## Mechanism

Driving world models work by predicting the future on sensors' streams, e.g., future frames in videos of cameras

Driving world models generate those future data: they are

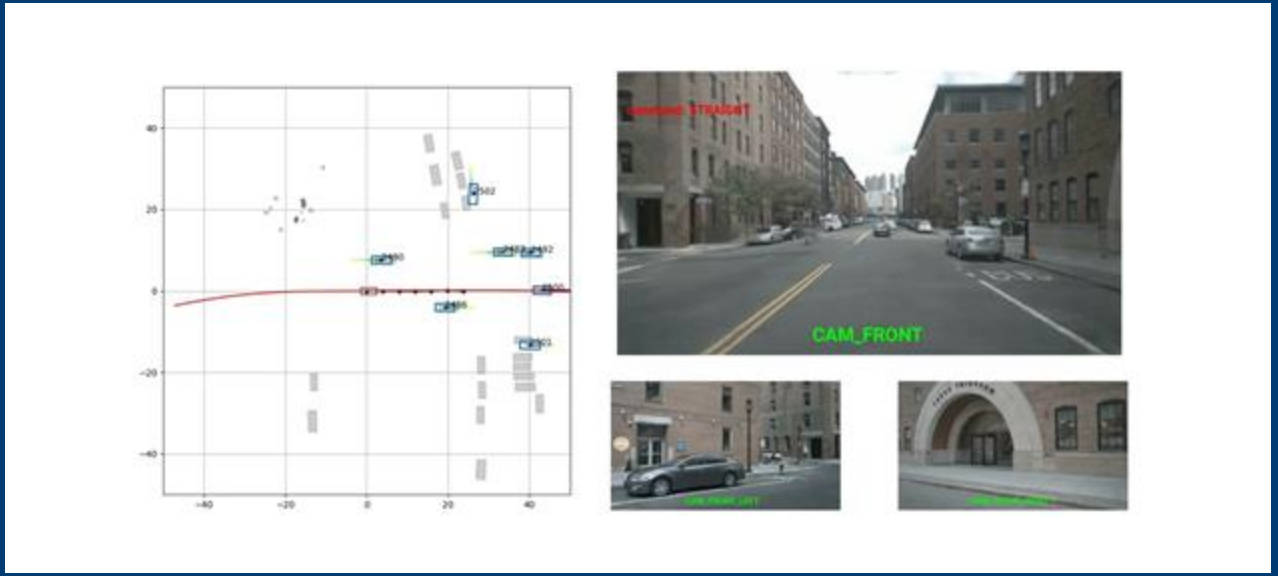
Generative AI models


# World Model

## Purpose

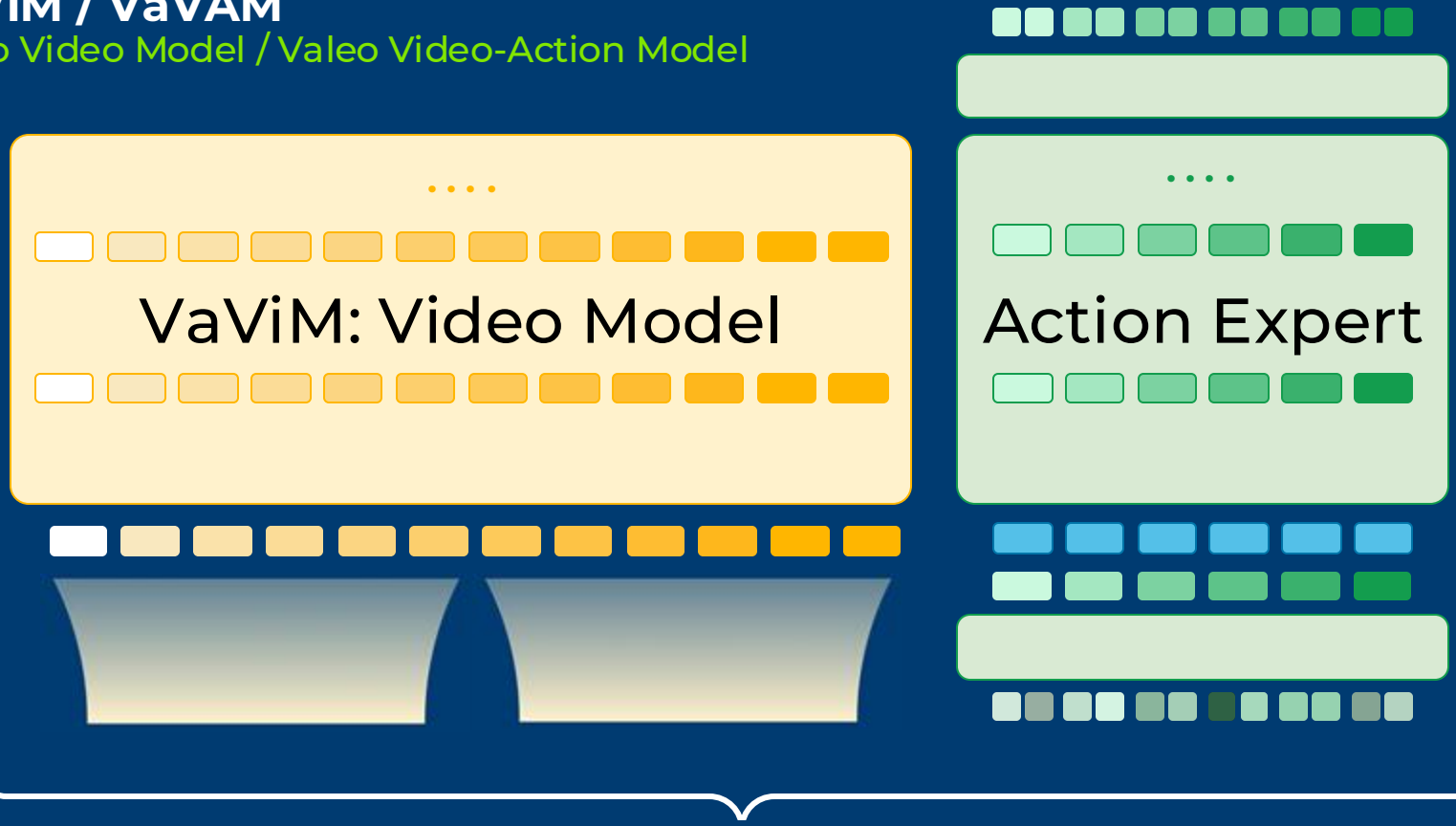
A driving world model allows a driving agent to decide on the best actions to reach the desired outcome

UniAD Failure Case



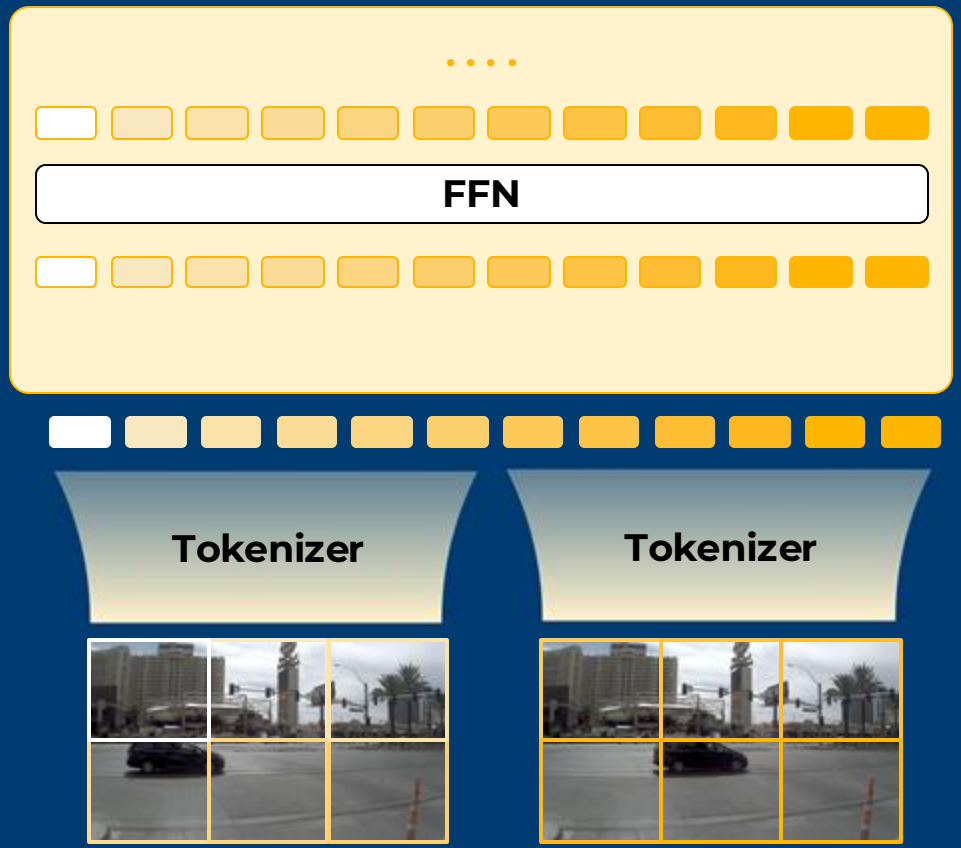
# VaVIM / VaVAM

Valeo Video Model / Valeo Video-Action Model



# VaVIM / VaVAM

## Valeo Video Model / Valeo Video-Action Model



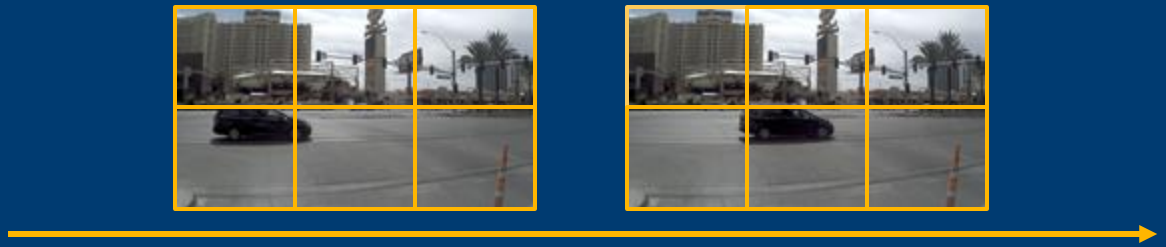

VaViM is our video world model

- Trained on 1800+ hours of YouTube front-cam videos
- Discretizes video into text-like “tokens”
- Predicts future with GPT-like autoregression
- Predicted tokens can be decoded into video or used *per se* for decisions

# VaVIM in detail

Let’s predict future video frames as if we were predicting the next word on a phrase

Step 1: lay out the frames in sequence and cut them into patches



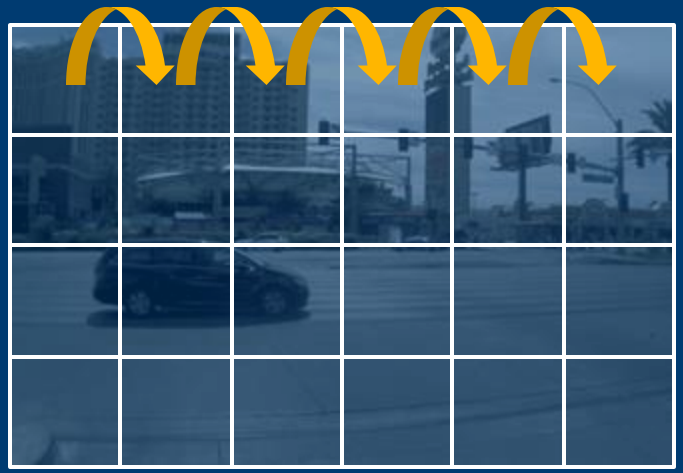
# Discrete Tokenization

## From RGB images to a discrete sequence


(*) VQ-VAE, LFQ, FSQ, etc...

# Autoregressive generation

## Overview

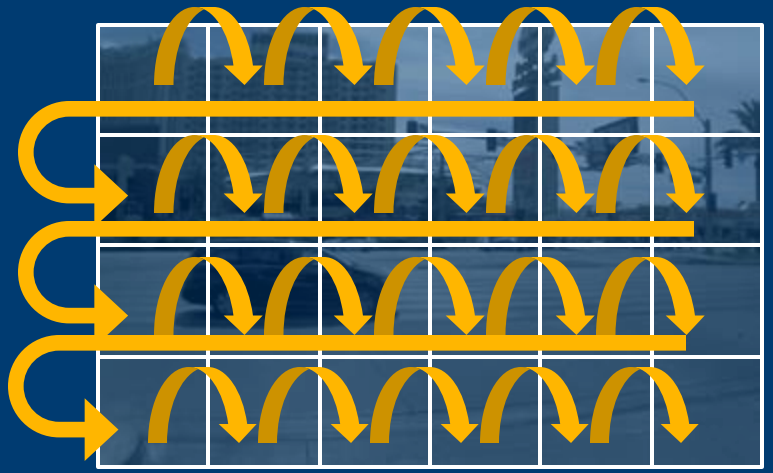


# Autoregressive generation

## Overview

# Autoregressive generation

## Overview



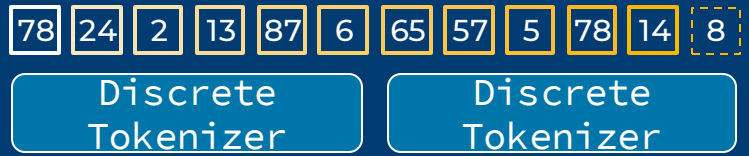
# Learning the video model

## Next-token prediction - teacher forcing supervision

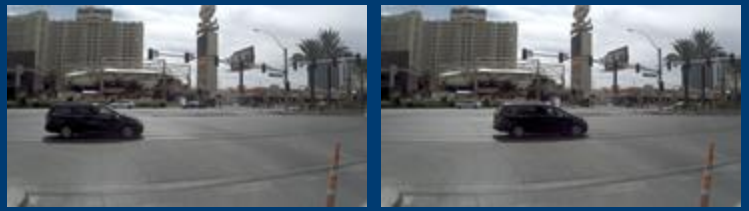
GT Tokens (left shift)
[h*w]


Logits + Softmax
[h*w, Vocab. size]


Input Tokens
[h*w]




Input Video



(*) GPT, LLaMA, DeepSeek, etc...

VaViM predicts and generates future scenarios

# World Models

## Quality of generation scale with the model size

Example of long video generation, beyond training context length

### VaViM-S (200M)

Pedestrian mistaken for a car, unrealistic scene generation


### VaViM-L (1B)

Pedestrian generated, realistic scene generation

# Limitations
## VaViM-1

Limitation 1) ImageNet Tokenizer

→ Cosmos Tokenizer (20M+ hours of diverse videos)

Limitation 2) Generation is slow: generating 4 frames = +10s

→ Reduce the number of tokens
→ Change the AR sampling (MaskGit-style)

One frame is currently 18x32 = 576 tokens → 4 future frames = 2304 tokens !!!

# Reducing # of tokens

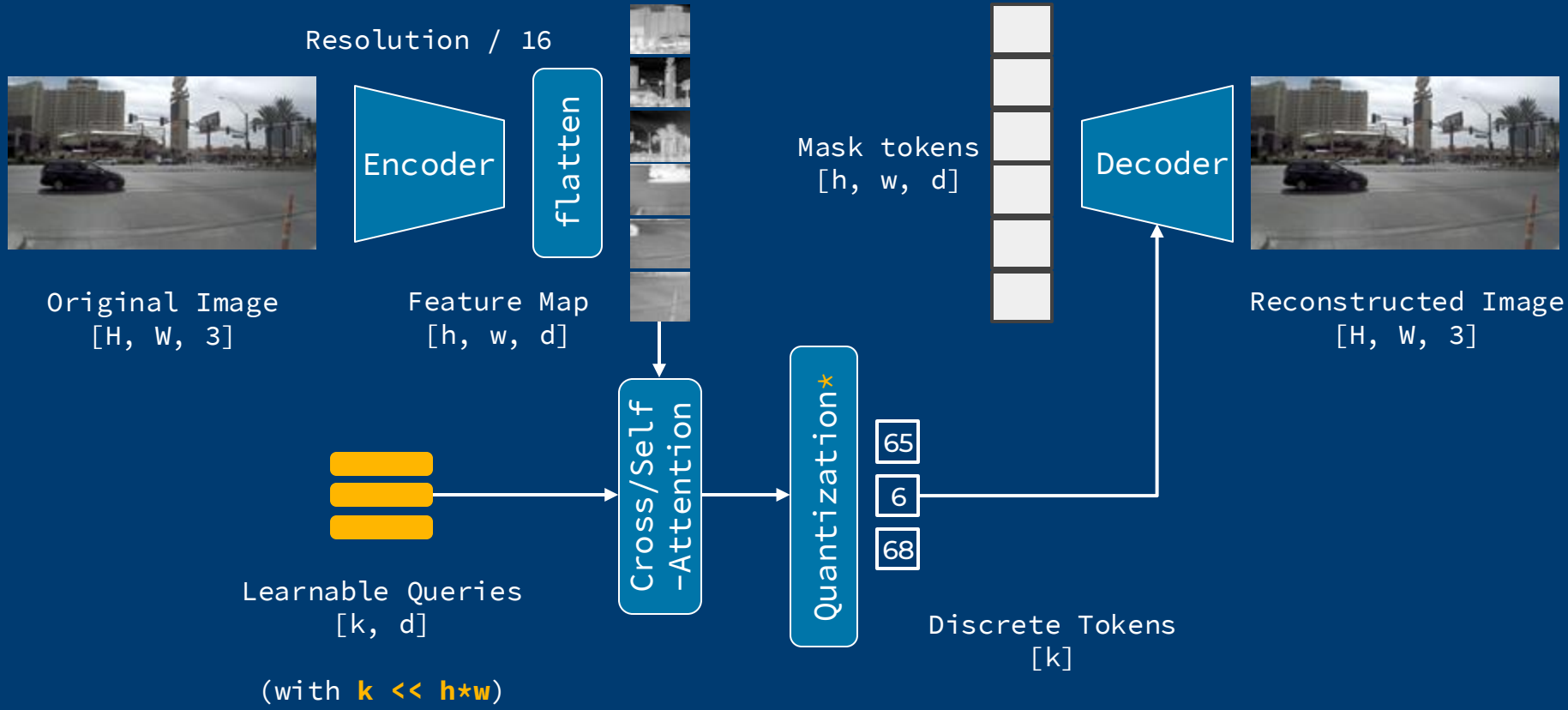
# Discrete Tokenization

## From RGB images to a discrete sequence


(*) VQ-VAE, LFQ, FSQ, etc...

# 1D tokenizer

## Moving out from the grid




TiTok: An Image is Worth 32 Tokens for Reconstruction and Generation (Yu et al., 2024)
FlexTok: Resampling Images into 1D Token Sequences of Flexible Length (Bachmann et al., 2025)

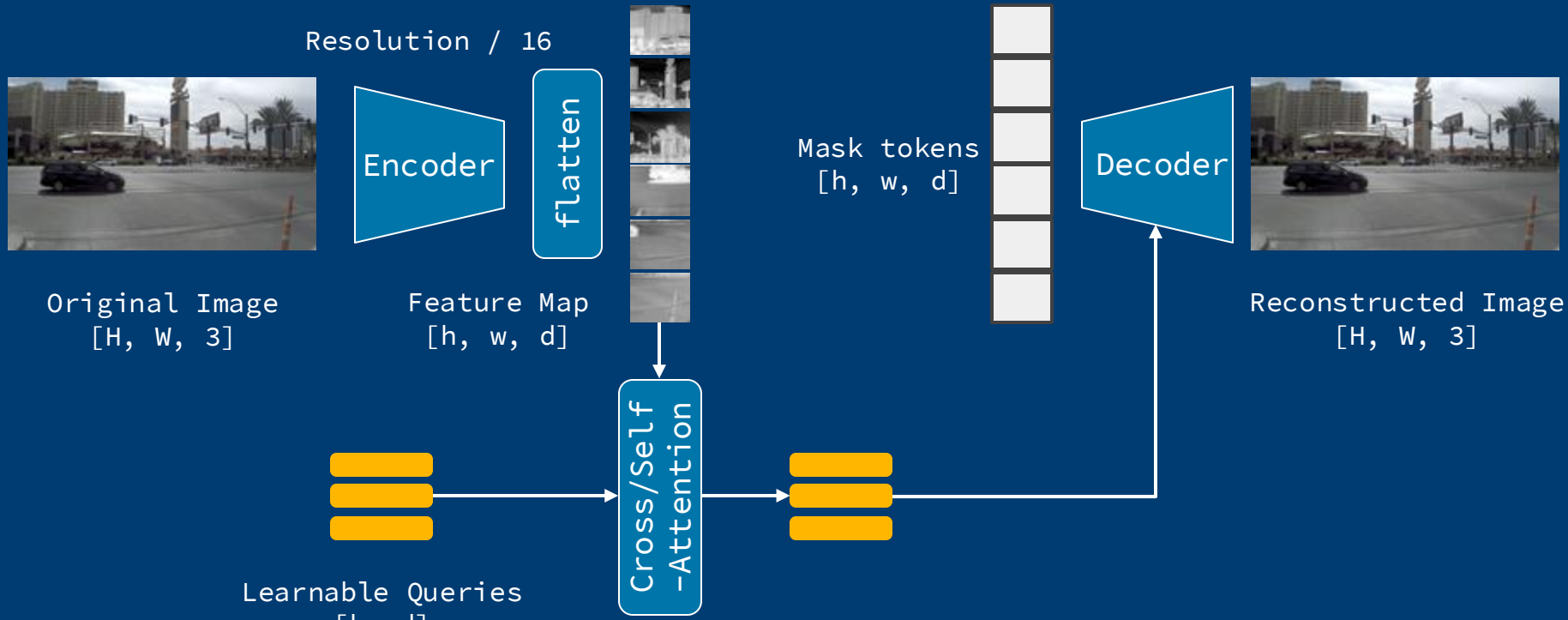
# 1D tokenizer - discrete - 256 tokens

Well... be the judge

# 1st change
# Going Continuous

# 1D Tokenizer - Continuous Tokenization

What if we remove the quantization layer



# 1D tokenizer - continuous - 256 tokens

Works well !

# 1D tokenizer - continuous - 32 tokens

Works well too !

# 1D tokenizer - continuous - 32 tokens

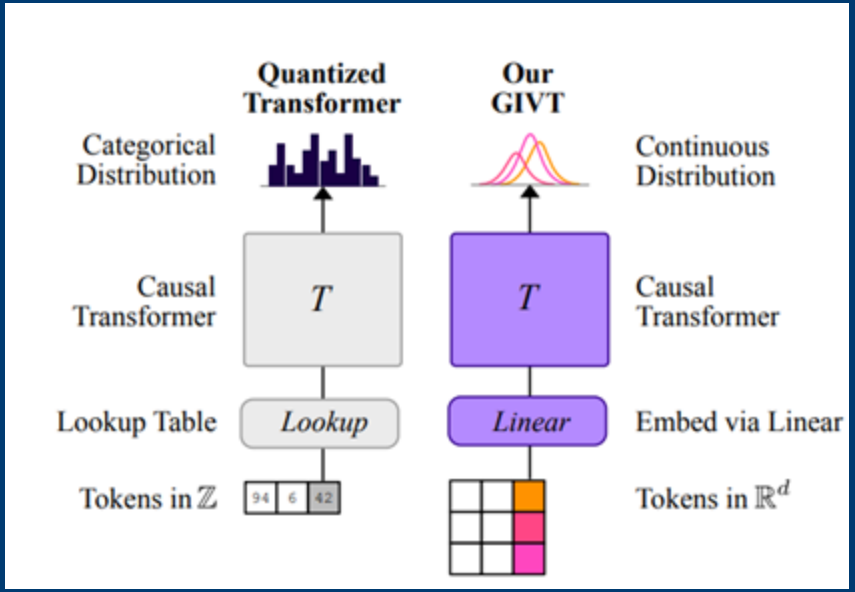
## Results on driving data (nuScenes)

| | # tokens | MSE↓ | SSIM↑ | PSNR↑ | rFID (dinov2-L)↓ |
|---|---|---|---|---|---|
| | | | | | |
| LLamaGen - VQGAN | 256 | 0.0039 | 24.7291 | 0.8242 | 141.8907 |
| Cosmos - FSQ | 256 | 0.004 | 24.7684 | 0.8184 | 175.7658 |
| Cosmos - continous | 256 | 0.0013 | 29.7131 | 0.9058 | 67.7483 |
| FlexTok - FSQ* | 256 | 0.0136 | 19.1557 | 0.7289 | 523.2096 |
| FlexTok - FSQ* | **32** | 0.0191 | 17.7005 | 0.6778 | 557.3641 |
| FlexTok - continuous* | 256 | **0.0009** | **31.0313** | **0.9291** | **49.6797** |
| FlexTok - continuous* | **32** | 0.0018 | 28.0202 | 0.9078 | 84.2266 |

(*) Ours Flextok-like

# AR video model - Moving from discrete to continuous

How to train AR model with continuous tokens ?




**Discrete:**

- Output = Softmax
- Transformer head dim = |vocabulary|
- Loss = cross-entropy[one-hot, categorical distrib]

**Continuous:**

- Output = GMM (mixture of k *d-dimensional* diag-cov gaussian)
- Transformer head dim = 2*k*d + k
  - k * d means
  - k * d std
  - k mixture weights
- Loss = NLL[GT vector, GMM distrib]

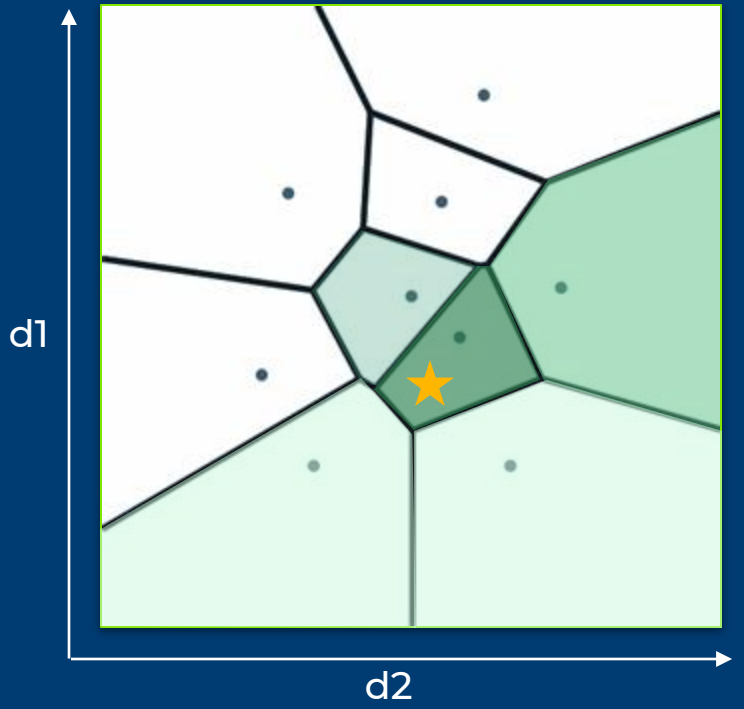
GIVT: Generative Infinite-Vocabulary Transformers (Tschannen et al., 2024)

# AR video model - Moving from discrete to continuous

## Inference - sampling of discrete token vs sampling of continuous vector

Discrete codebook + Softmax

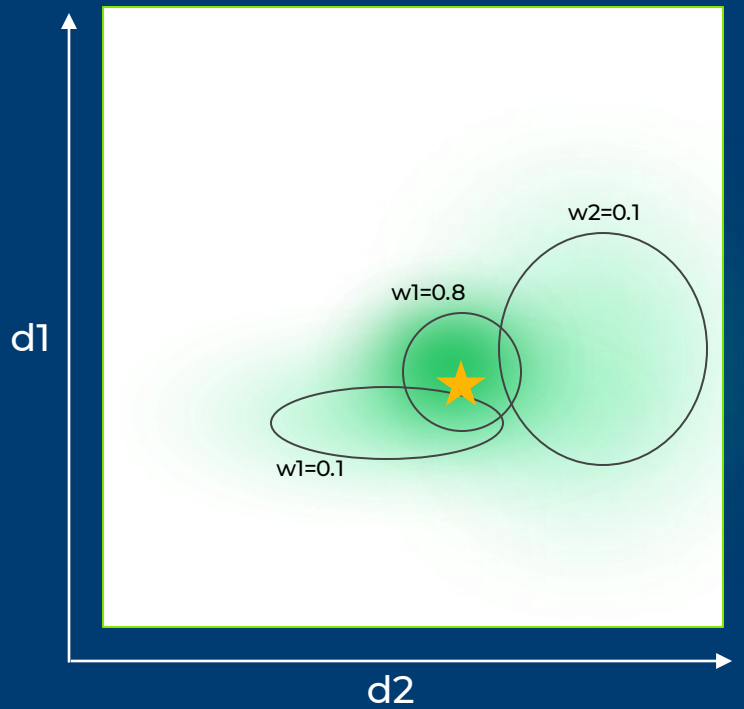
Distrib of next token =
probabilities over candidates
voronoi cells




True next token

Continuous latents + GMM head

Distrib of next token =
mixture of gaussians (e.g., k=3)



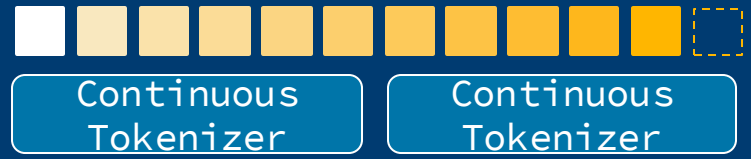
# Learning the video model - we don't change many things

Discrete tokenizer, softmax, CE -> Continuous tokenizer, GMM, NLL

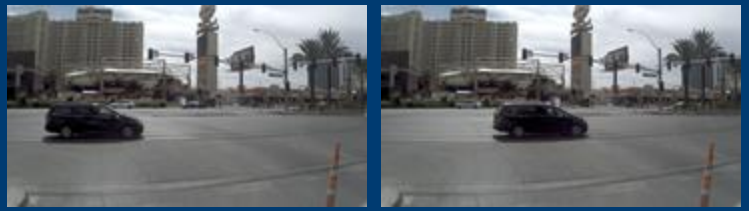
GT Tokens (left shift)
[h*w, d]


GMM (k mixtures)
[h*w, 2*k*d + k]


Input Tokens
[h*w, d]




Input Video

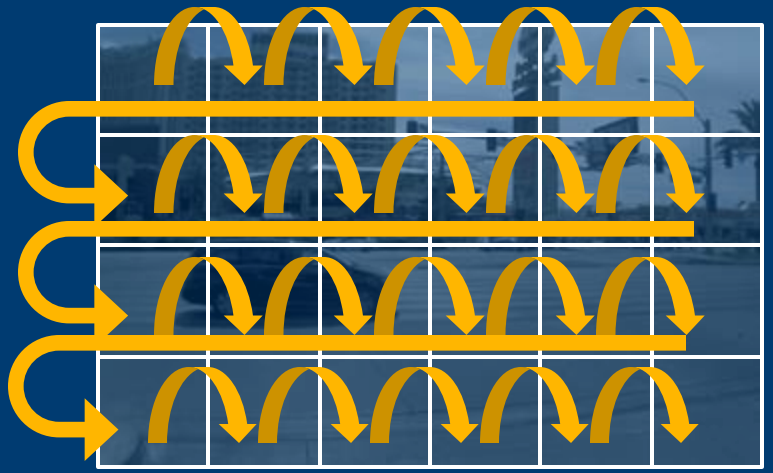


(*) GPT, LLaMA, DeepSeek, etc...

# 2nd change
# Patchify tokens

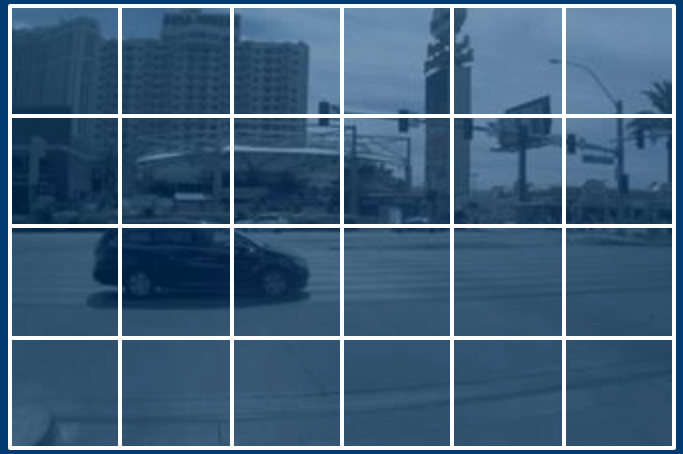
# Patchify

Reduce # of generation steps



# Patchify

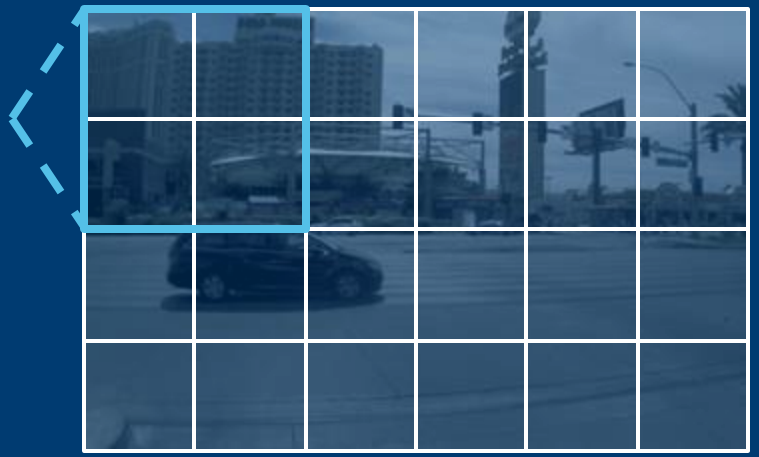
Reduce # of generation steps



# Patchify

## Reduce # of generation steps

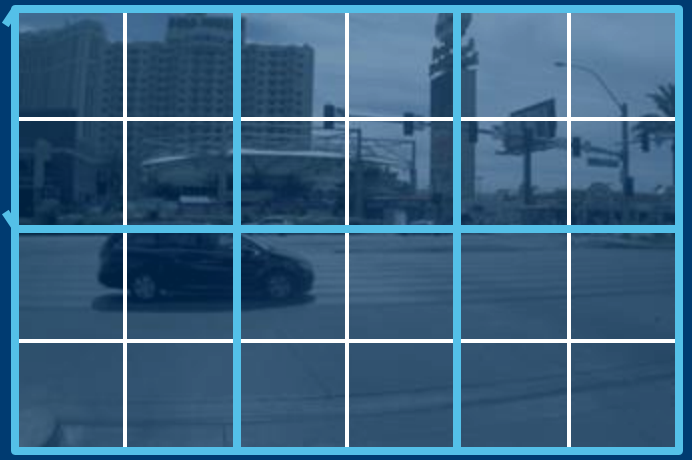
Example with 2x2 patch



# Patchify

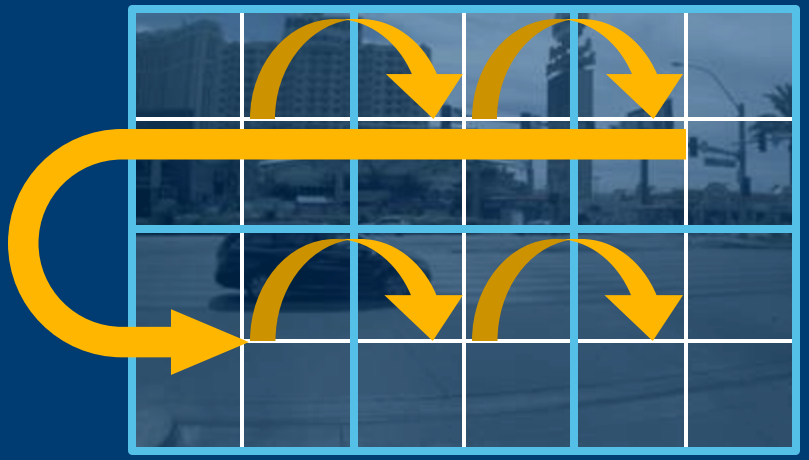
## Reduce # of generation steps

Example with 2x2 patch



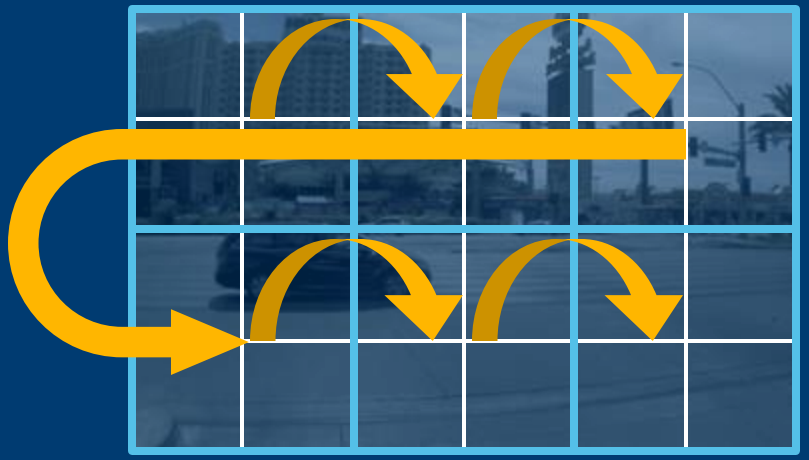
# Patchify

Reduce # of generation steps



# Patchify

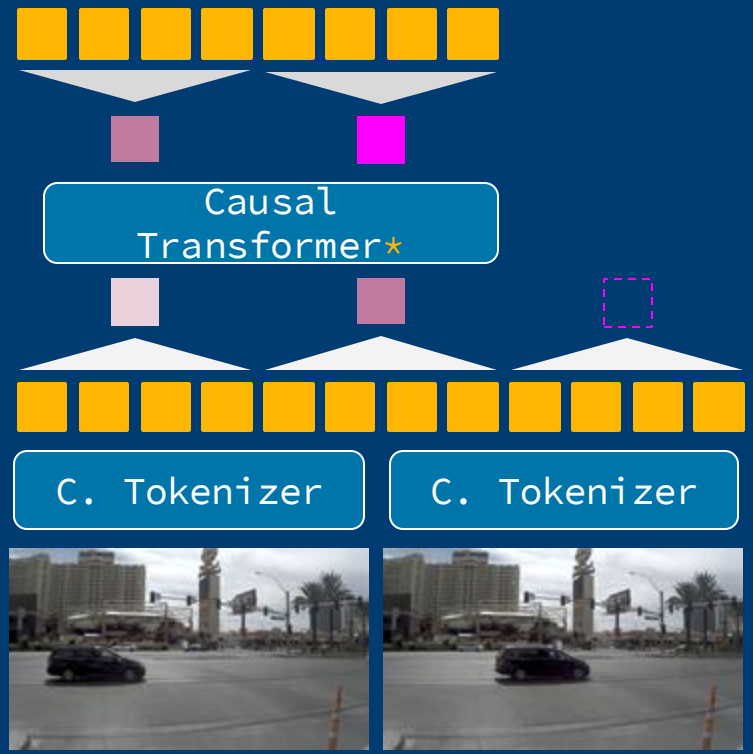
## Reduce # of generation steps



Issue: if we generate 4 tokens at the same time, they are generated independently and are not coherent

# Patchification

## Simple Patchify + Unpatchify




Patch embeddings
[h*w / $p^2$, d]

Input Tokens
[h*w, d]

Input Video

Issue: if we generate 4 tokens at the same time, they are generated independently and are not coherent

(*) GPT, LLaMA, DeepSeek, etc...

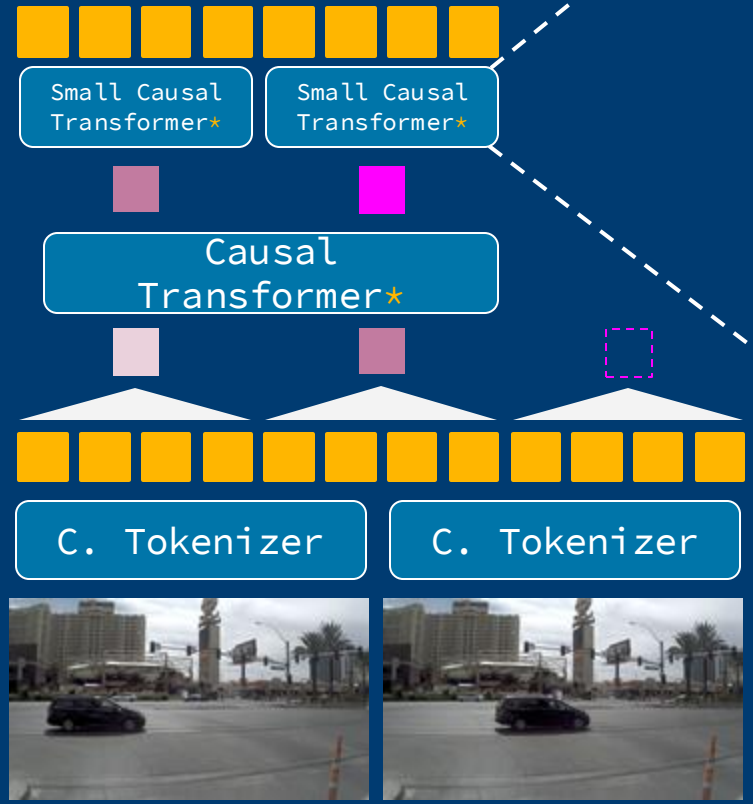
# Patchification

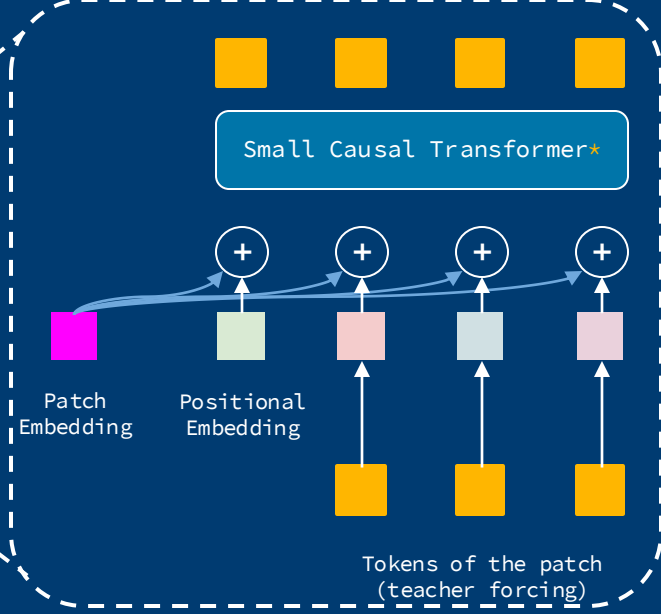
## Simple Patchify + Unpatchify

Patch embeddings
[h*w / p², d]

Input Tokens
[h*w, d]

Input Video



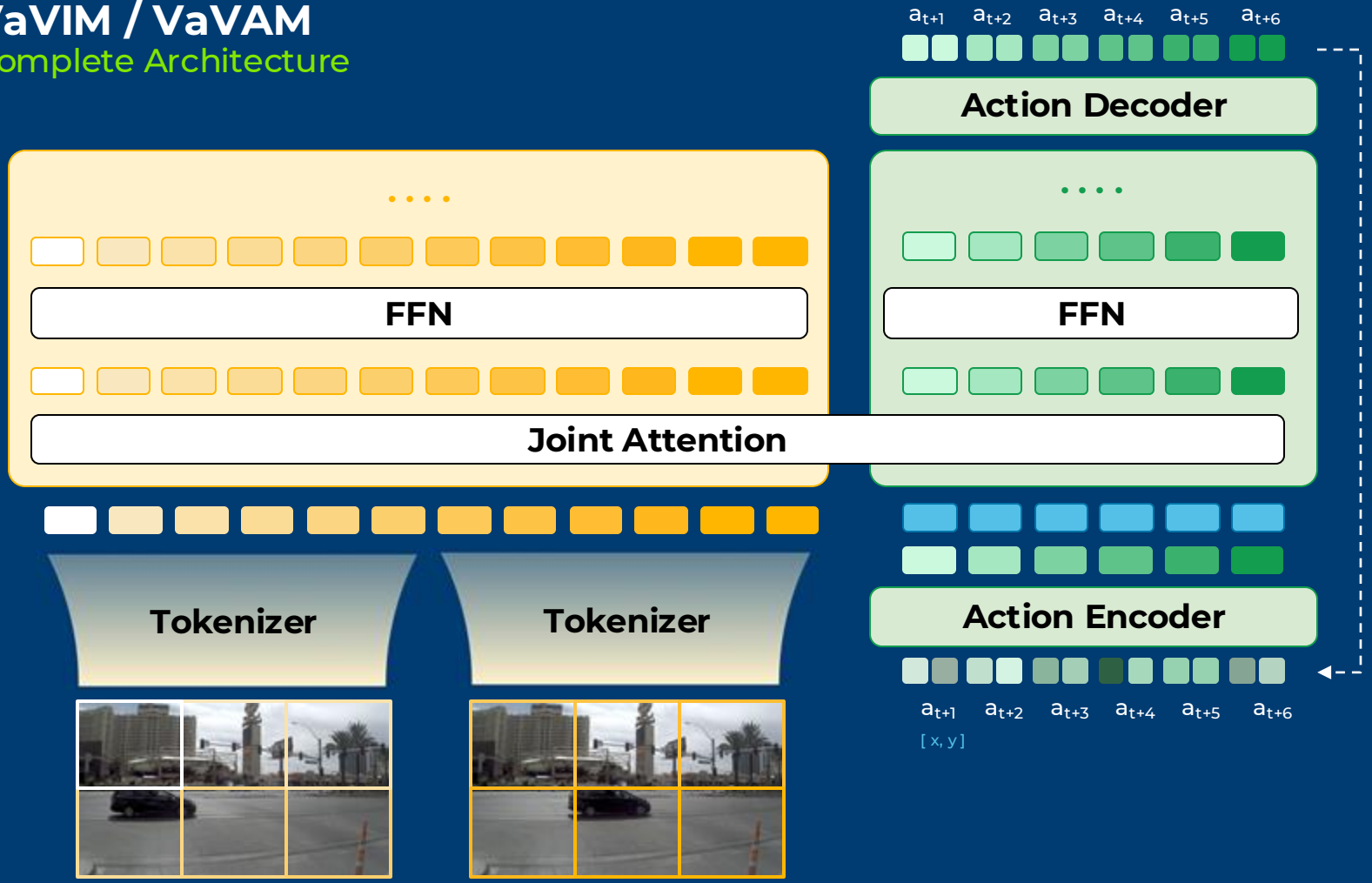




(*) GPT, LLaMA, DeepSeek, etc...

# How to use the features for VaViM to drive ?

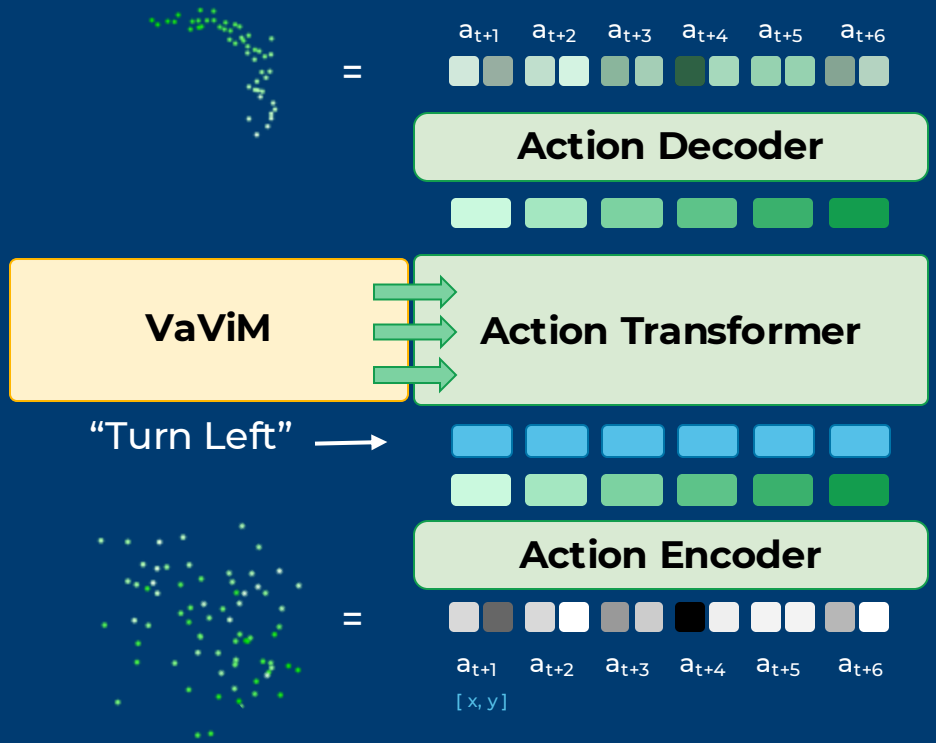
Valeo

# VaVIM / VaVAM
## Complete Architecture



# VaVAM in detail

## VaVAM = VaViM + Action expert



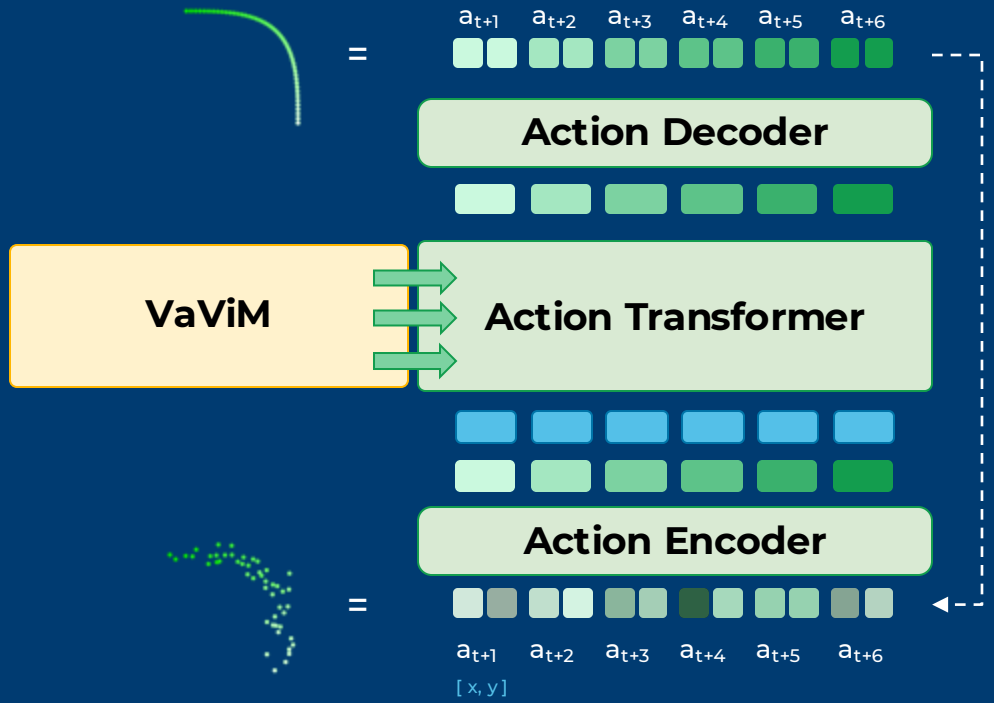

Train an action expert on VaViM's embeddings

The action expert estimates the agent's decisions

Model: Flow-matching action expert [NeurAD K. Black et al. ]

# VaVAM in detail

## VaVAM = VaViM + Action expert

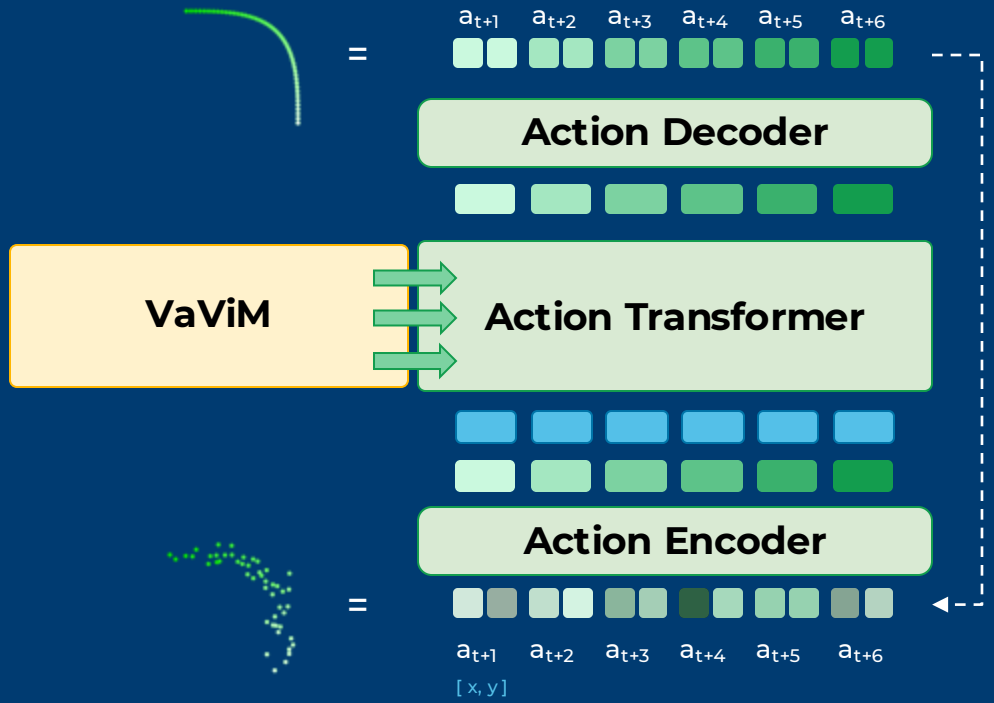


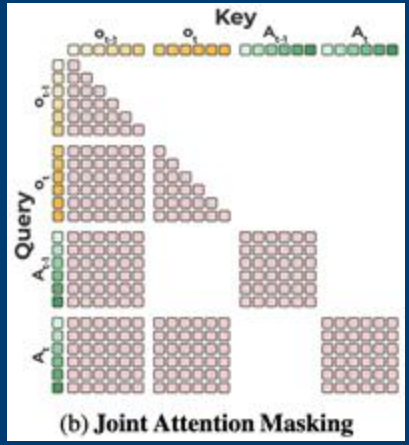

Train an action expert on VaViM's embeddings

The action expert estimates the agent's decisions exploiting the temporal contexts of multiple frames that are crucial for understanding dynamic scenarios

# VaVAM in detail

## VaVAM training




Video model kept frozen during the training of the action expert

Data and Training Strategy: nuPlan nuScenes (standard driving datasets), providing synchronized camera and ego-trajectory data, enabling supervised training of action models

Only front cam, 4s video clips at 2 FPS => 8 512×288 frames per clip

# VaVAM experiment on NeuroNCAP

To evaluate safety-critical behavior beyond open-loop metrics, we use [NeuroNCAP], a photorealistic, NeRF-based simulator supporting data-driven closed-loop evaluation. Unlike synthetic carla or view-reprojection systems, NeuroNCAP produces novel views from real data and inserts **adversarial agents to mimic critical Euro NCAP scenarios**: ego-lane obstacles, frontal collisions, and cross-traffic.

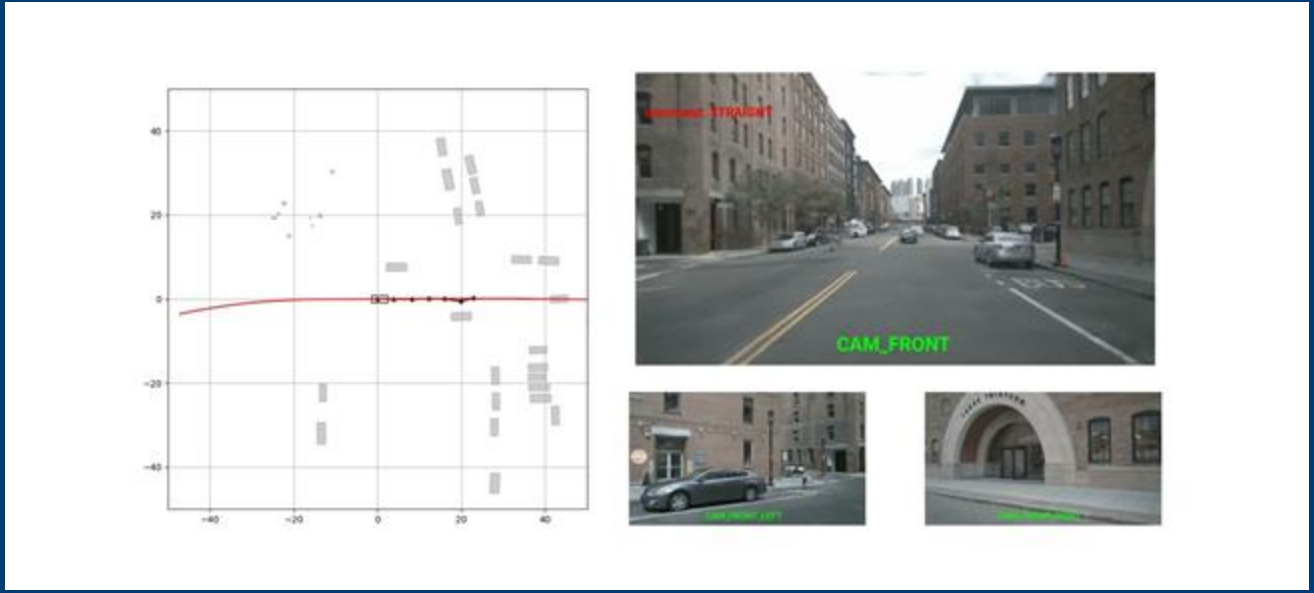
Driving decisions are executed in simulation, with observations updated accordingly. The ego-vehicle and the adversarial agents are initialized so that, under constant speeds and steering angles, a collision would occur in 4 seconds.

Predicted trajectories are converted into low-level control commands (steering, throttle, brake) via an LQR controller implemented within the NeuroNCAP simulator.

# VaVAM experiment on NeuroNCAP

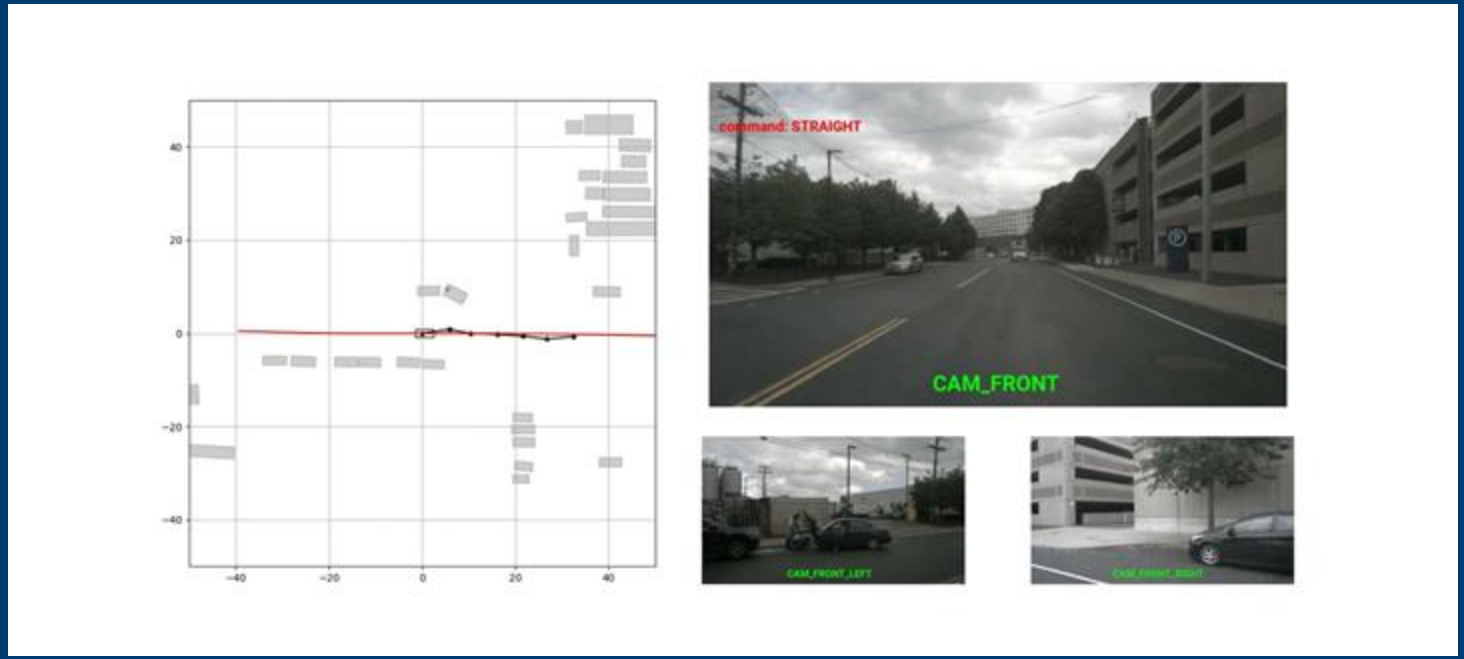
A driving world model allows a driving agent to decide on the best actions to reach the desired outcome

VaVAM



# VaVAM experiment on NeuroNCAP

VaVAM uses VaViM for end-to-end trajectory decision



# VaVAM experiment on NeuroNCAP

NeuroNCAP metrics: the collision rate (lower is better) and the NeuroNCAP Score(NNS) (higher is better) which is derived from the collision rate and severity: zero collisions give a perfect score of 5.0, which is lowered for more collisions or collisions at higher speeds

* Static scenarios, sensitive to annotation-dependent post-processing
† Side scenarios, sensitive to multiview-cameras
‡ Baseline reproduced by us

| MODEL | POST-PROC. | NEURONCAP SCORE ↑ | | | | COLLISION RATE (%) ↓ | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | AVG. | STAT.* | FRONTAL | SIDE† | AVG. | STAT.* | FRONTAL | SIDE† |
| Baselines — Trained with hand-labeled annotations, 360° View | | | | | | | | | |
| BASE-U | N/A | 2.65 | 4.72 | 1.80 | 1.43 | 69.90 | 9.60 | 100.00 | 100.00 |
| BASE-V | N/A | 2.67 | 4.82 | 1.85 | 1.32 | 68.70 | 6.00 | 100.00 | 100.00 |
| UNIAD | ✗ | 0.73 | 0.84 | 0.10 | 1.26 | 88.60 | 87.80 | 98.40 | 79.60 |
| VAD | ✗ | 0.66 | 0.47 | 0.04 | 1.45 | 92.50 | 96.20 | 99.60 | 81.60 |
| UNIAD | ✓ | 1.84 | 3.54 | 0.66 | 1.33 | 68.70 | 34.80 | 92.40 | 78.80 |
| UNIAD‡ | ✓ | 2.08 | 3.58 | 1.18 | 1.48 | 61.10 | 31.20 | 78.80 | 73.20 |
| VAD | ✓ | **2.75** | **3.77** | 1.44 | **3.05** | **50.70** | **28.70** | 73.60 | **49.80** |
| VAM — Trained on raw data, Front-cam only | | | | | | | | | |
| VAM | ✗ | 2.62 | 3.13 | **2.67** | 2.07 | 52.70 | 47.20 | **50.00** | 60.80 |

# Conclusion

- End-to-end driving system that combines generative video pretraining with action learning from demonstrations
- VaVAM achieves strong closed-loop performance and sets a new state of the art in safety-critical scenario
- Future directions include reward-based learning, multi-camera inputs, and **improved VaViM tokenization**


https://valeoai.github.io

- Main contributors:
  - Florent Bartoccioni
  - Elias Ramzi
  - Shashanka Venkataramanan
  - Eduardo Valle

Valeo