mobileye

Driving AI

2024

Navigating The Path to Autonomous Mobility

Prof. Amnon Shashua, CEO

Prof. Shai Shalev-Schwartz, CTO

How to Solve Autonomy

- Reaching a real “full self driving” system (eyes-off)
- While maintaining a sustainable business

* Subject to defined Operational Design Domain and products specifications

How to Solve Autonomy

	Sensors	AI Approach	Cost	Modularity	Geographic Scalability	MTBF
Waymo	Lidar-centric	CAIS	✗	✗	?	✓
Tesla	Camera only	End-to-end	✓	✓	✓	?
Mobileye	Camera-centric	CAIS	✓	✓	✓	?

How to Solve Autonomy

	Sensors	AI Approach	Cost	Modularity	Geographic Scalability	MTBF
Waymo	Lidar-centric	CAIS	✗	✗	?	✓
Tesla	Camera only	End-to-end	✓	✓	✓	?
Mobileye	Camera-centric	CAIS	✓	✓	✓	?

Which is more likely to succeed?

End-to-End Approach

Premise

No glue code

Unsupervised data **alone** can reach sufficient MTBF

Reality

Glue code shifted to offline

Rare & correct vs. common & incorrect

"AV alignment" problem

Really?

- Calculator
- Shortcut learning problem
- Long tail problem

“No Glue Code”: AV Alignment Problem

End-to-end aims to maximize $\mathrm{P}[y|x]$ where y is the future trajectory human would take, denoted y, given the previous video, denoted x
This learning objective prefers 'common & incorrect' over 'rare & correct’

Examples:

1. Most drivers slow down at a stop sign but do not come to a full stop
   - Rolling stop ≡ common & incorrect
   - Full stop ≡ rare & correct
2. “Rude drivers” that cut in line
3. Reckless drivers

This is why RLHF is used in LLMs: the reward mechanism differentiates between ‘correct’ and ‘incorrect’

Glue code shifted to offline

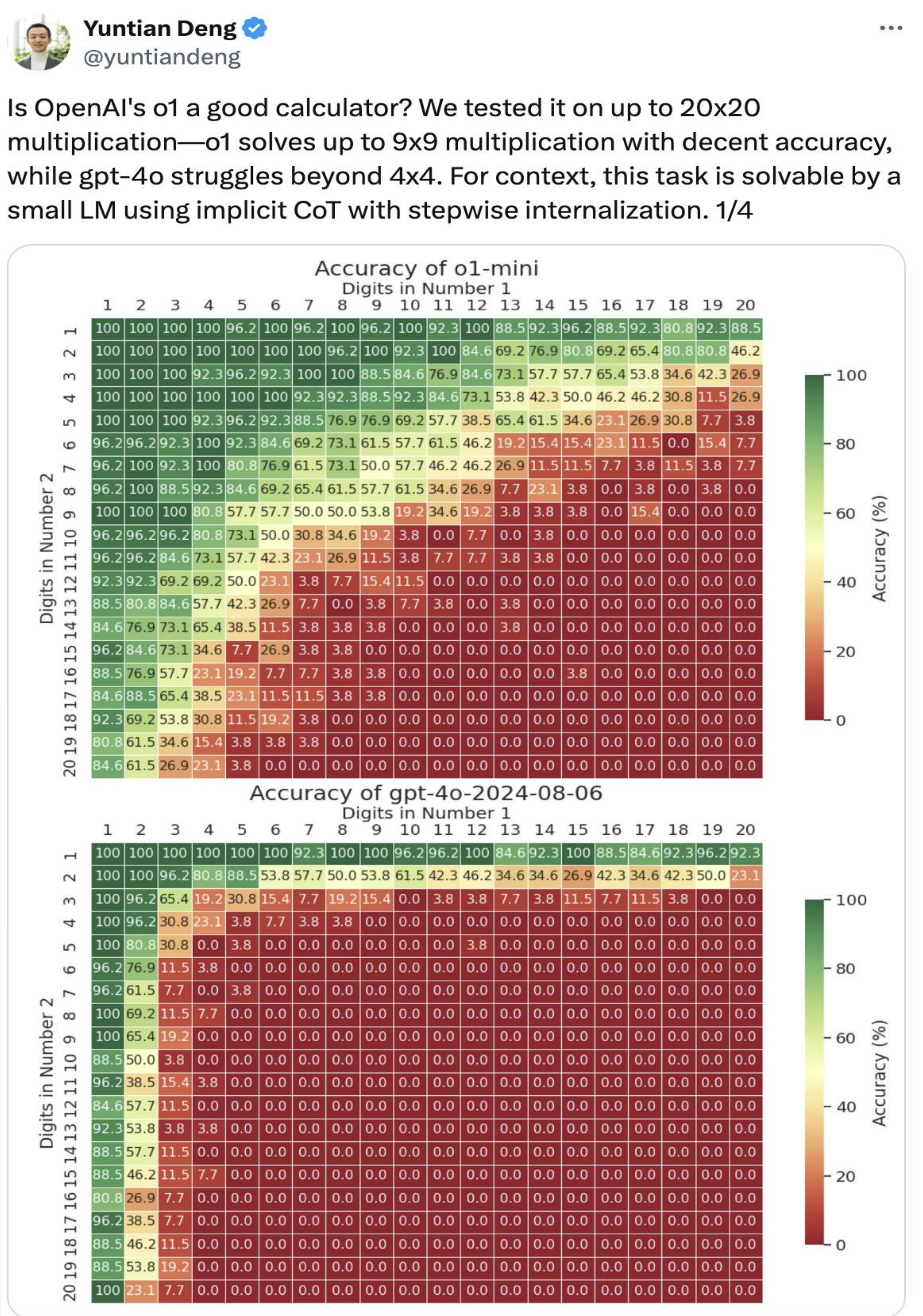
Can Unsupervised Data Alone Reach High MTBF?

Calculators

End-to-end learning from data often misses important abstractions and therefore doesn't generalize well

Example

Learning to multiply 2 numbers, a task where even the largest LLMs struggle



https://x.com/yuntiandeng/status/1836114401213989366

Can Unsupervised Data Alone Reach High MTBF?
Calculators

End-to-end learning from data often misses important abstractions and therefore doesn't generalize well

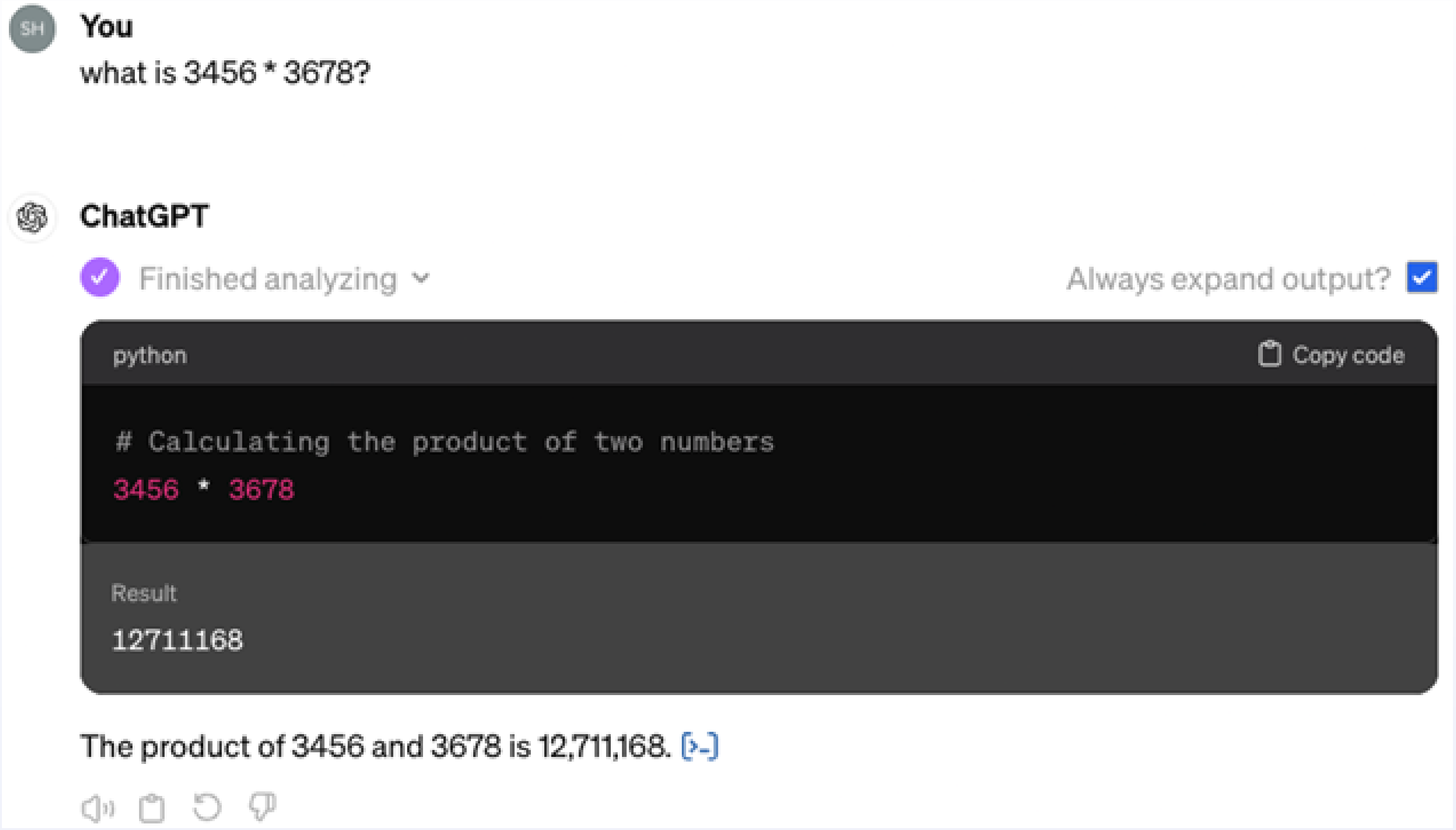
Example

Learning to multiply 2 numbers, a task where even the largest LLMs struggle

What can be done?

- Provide tools to LLMs
- → Compound AI Systems (CAIS)

ChatGPT
Call a tool
(calculator)



Can Unsupervised Data Alone Reach High MTBF?
Shortcut Learning Problem

Relying on different sensor modalities is a well-established methodology for increasing MTBF

The question: How to fuse the different sensors?

The "end-to-end approach": Just feed all sensors into one big network and train it

"The Shortcut Learning Problem"

When different input modalities have different sample complexities, end-to-end Stochastic Gradient Descent struggles in leveraging the advantages of all modalities

Can Unsupervised Data Alone Reach High MTBF?

Shortcut Learning Problem

Consider 3 types of sensors

Camera

Radar

Lidar

Suppose that each system has inherent limitations that cause a failure probability of ϵ, where ϵ is small (e.g., one in 1000 hours)

Additionally, assume that the failures of the different sensors are independent

We compare two options

- Low level, end-to-end, fusion (train a system based on the combined input)
- CAIS: Decomposable training of a system per each modality, followed by high-level fusion

Which option is better?

Shortcut Learning Problem: A Simple Synthetic Example

Distribution: all variables are over {+1, -1}, and data is created by the following simple generative model:

$$y \sim \mathrm{B}\left(\frac{1}{2}\right),\ \ \mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3 \sim \mathrm{i.i.d.}\,\mathrm{B}(\epsilon),\ \ x_1 = y\ \ r_1, x_2 = y\, r_2, x_4, x_5 \sim i.i.d.\,B\left(\frac{1}{2}\right), x_3 = y\, r_3\, x_4\, x_5$$

This is a simple model of fusion between Lidar, Radar, Camera systems with the following properties:

- The 3 systems have uncorrelated errors (modeled by $\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3$) of level ϵ
- x_1 and x_2 are ”simpler” systems (modeling radar and lidar), while the product of $x_3\ x_4\ x_5$ equals to $y\ r_3$, and therefore is a “complicated to learn” system (modeling the camera)

Theorem:

- Can easily reach error of $O(\epsilon^2)$ with decomposable training of 1-hidden-layer FCN + majority
- End-to-end SGD training will be “stuck” at an error of ϵ for T/ϵ where T is the time complexity of learning the complicated system (camera) individually

What happened? Isn’t end-to-end always better?

Shortcut learning problem: End-to-end SGD struggles to leverage systems with different sample complexities

Can Unsupervised Data Alone Reach High MTBF?

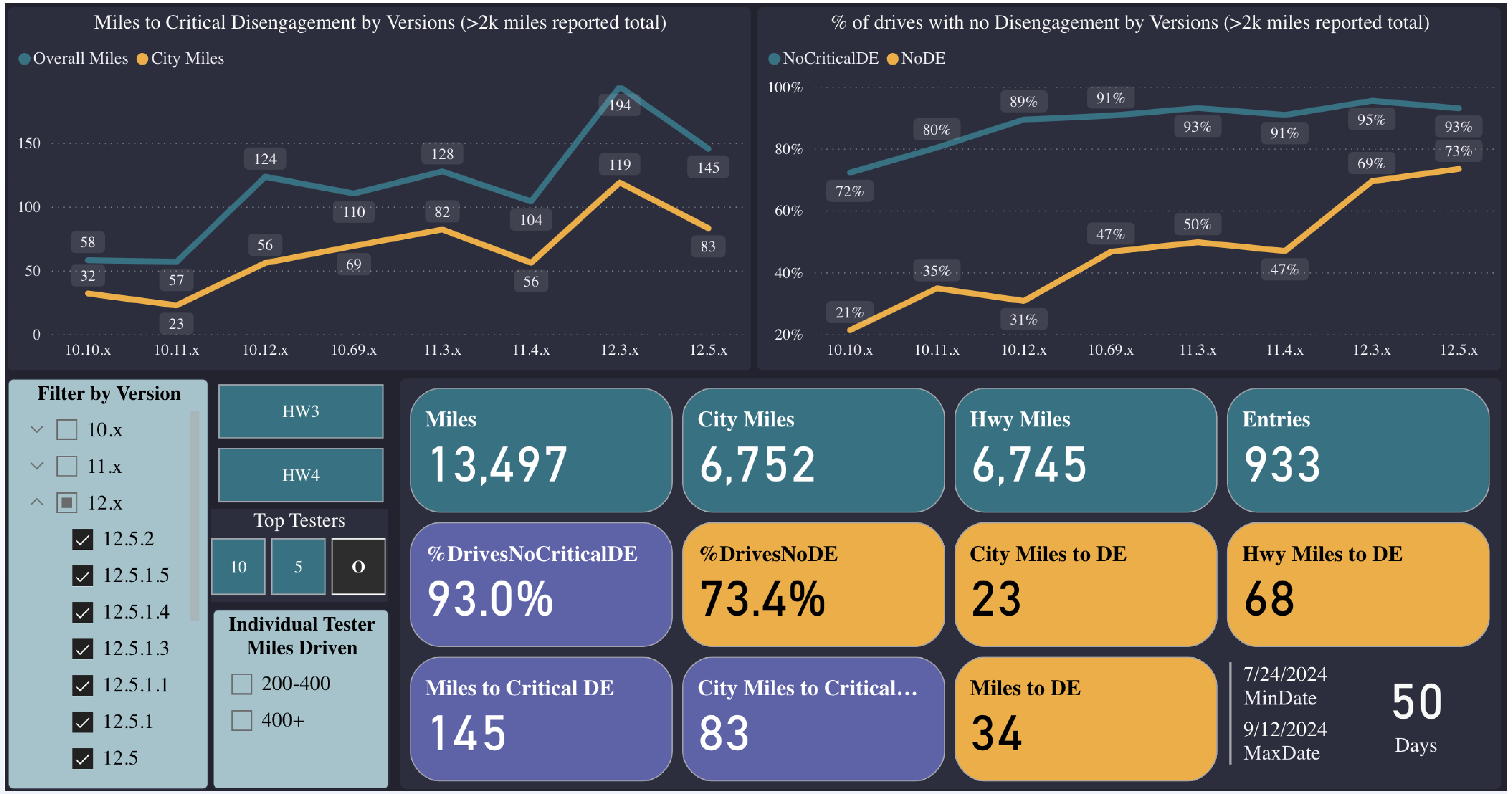
The Long Tail Problem

In the optimistic scenario, a few rare events reduce the probability mass considerably

In the pessimistic scenario, each rare event has minimal impact on the probability mass

Long Tail of Tesla FSD

- TeslaFSDtracker indicates that reducing variance solely through a data pipeline results in incremental progress



*teslafsdtracker.com - public data on Tesla's recent 12.5.x

During testing, AMCI drivers had to intervene over 75 times while FSD was active, resulting in an average of once every 13 miles. In one instance, the Tesla Model 3 ran a red light in the city during nighttime even though the cameras clearly detected the lights. In another situation with FSD (Supervised) enabled on a twisty rural road, the car went over a double yellow line and into oncoming traffic, forcing the driver to take over. One other notable mishap happened inside a city when the EV stopped even though the traffic light was green and the cars in front were accelerating.

Here's how Guy Mangiamele, Director of AMCI Testing, put it: "What's most disconcerting and unpredictable is that you may watch FSD successfully negotiate a specific scenario many times–often on the same stretch of road or intersection–only to have it inexplicably fail the next time."

AMCI released a series of short videos which you can watch embedded below (just try to ignore the background music.) The clips show where FSD (Supervised) performed very well, like moving to the side of a narrow road to let incoming cars pass, and where it failed.

https://insideevs.com/news/735038/tesla-fsd-occasionally-dangerously-inept-independent-test/

How to Solve Autonomy

	Sensors	AI Approach	Cost	Modularity	Geographic Scalability	MTBF
WAYMO Waymo	Lidar-centric	CAIS	✗	✗	?	✓
TESLA Tesla	Camera only	End-to-end	✓	✓	✓	?
mobileye Mobileye	Camera-centric	CAIS	✓	✓	✓	?

The Bias-Variance Tradeoff in Machine Learning

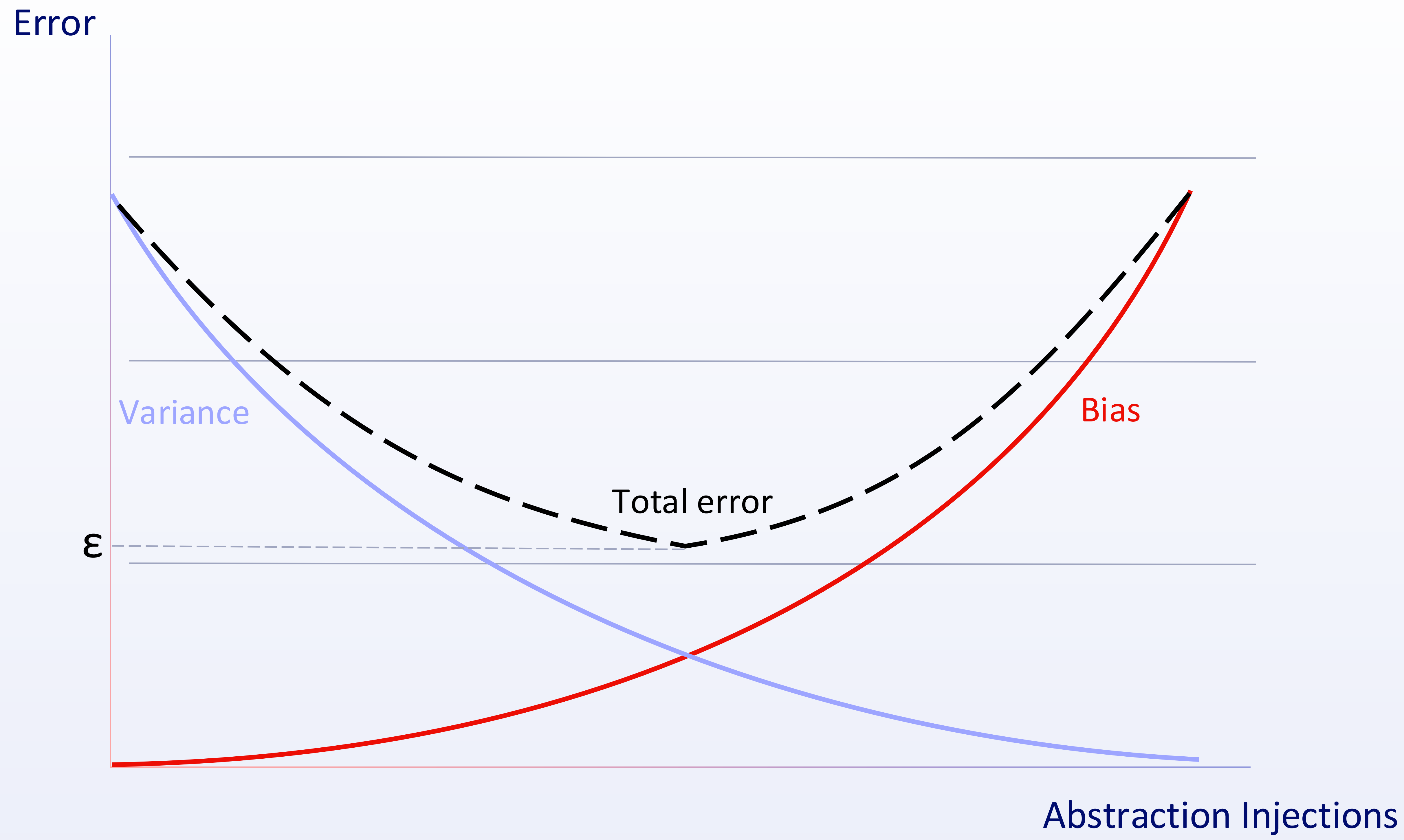
Bias ('approximation error')

The learning system cannot reflect the full richness of reality

Variance ('generalization error')

The learning system overfits to the observed data, and fails to generalize to unseen examples

Total error



Mobileye Compound AI System (CAIS)

AV Alignment

RSS

Separates correct from incorrect

Reaching Sufficient MTBF

Abstractions

- Sense / Plan / Act
- Analytic calculations: RSS, time-to-contact…

Redundancies

Sensors | Algo | High level fusion

Mobileye Compound AI System (CAIS)

AV Alignment

RSS

Separates correct from incorrect

Extremely Efficient AI
(Shai will cover)

Reaching Sufficient MTBF

Abstractions

- Sense / Plan / Act
- Analytic calculations: RSS, time-to-contact…

Redundancies

Sensors | Algo | High level fusion

PGF

High Level Fusion: How to Perform

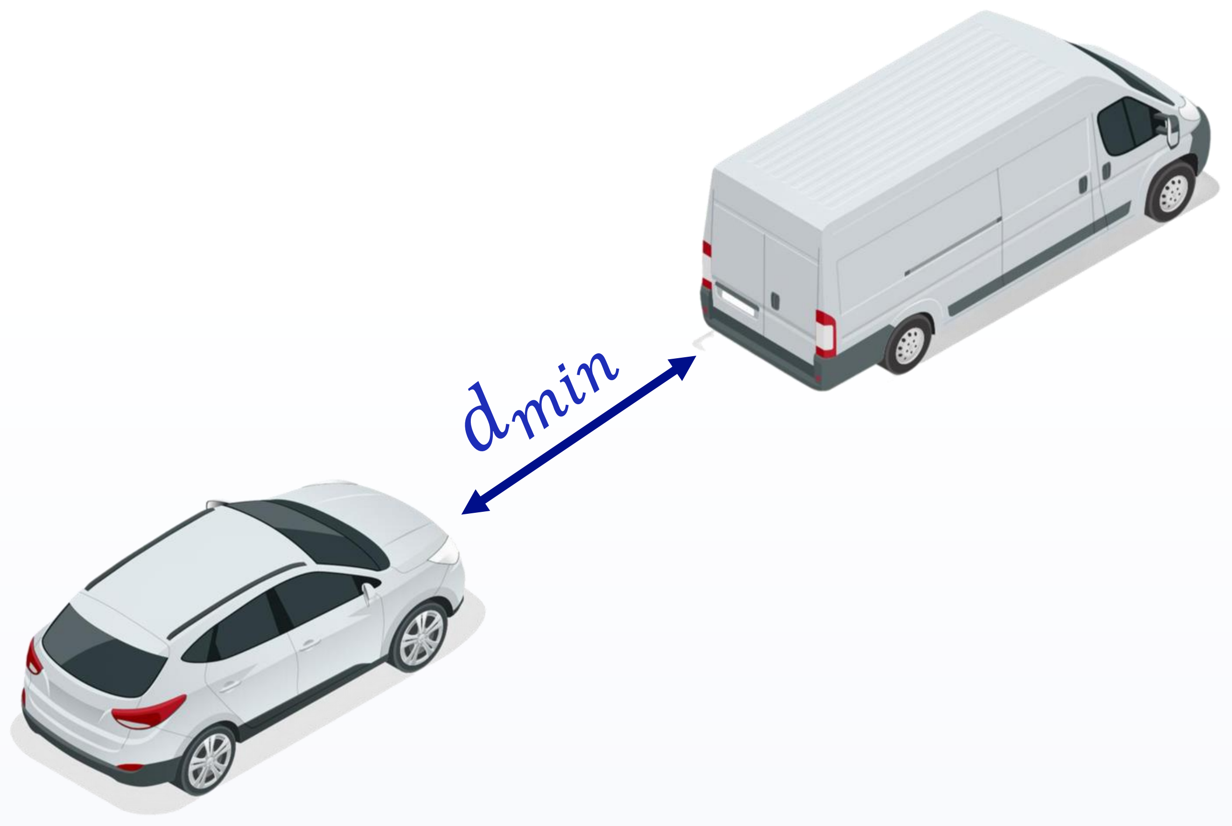
Consider a simple case

We are following a lead vehicle, and we have 3 sensors

Camera

Radar

Lidar



If there are contradictions between the sensors, where some dictate a strong braking while others not, what should we do?

Majority: 2 out of 3 (2oo3)

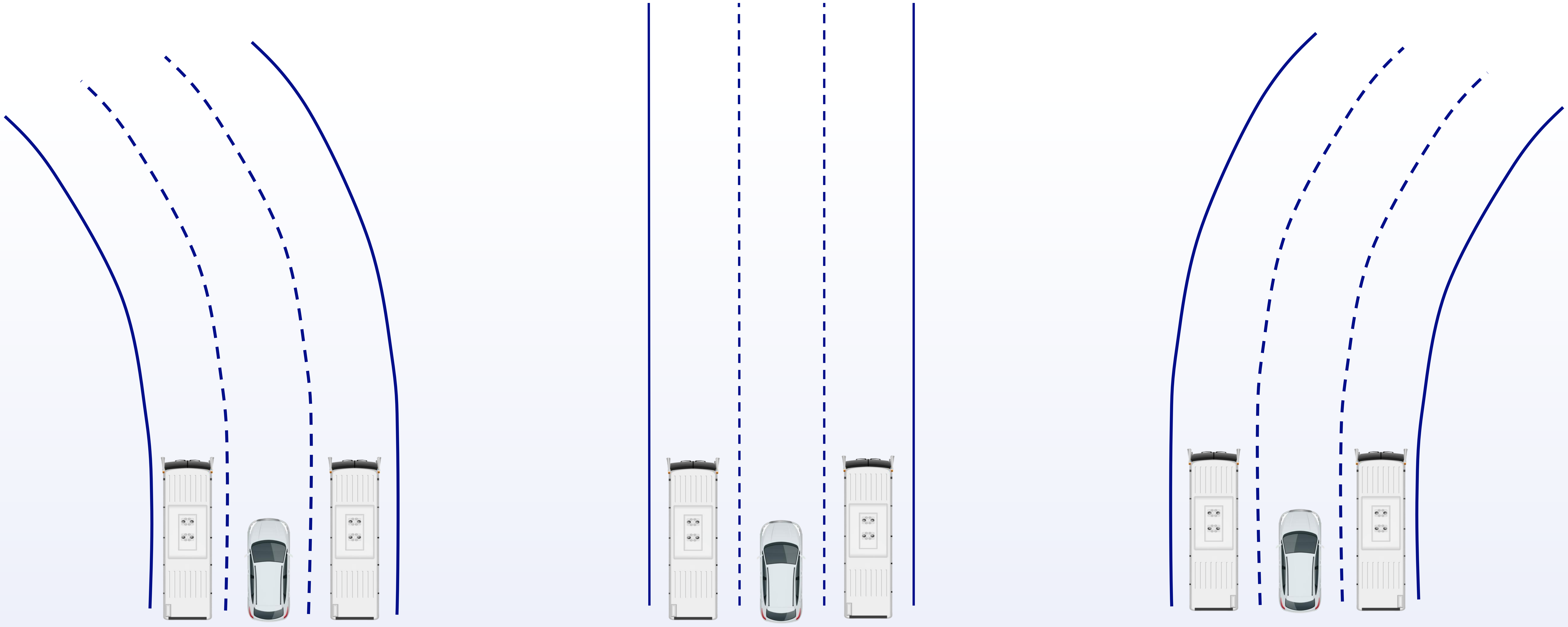
Property of majority

If each modality has an error probability of at most ϵ, and the errors are independent, then - majority vote has an error probability of $O(\epsilon^2)$

Majority is Not Always Applicable

Now consider 3 systems, each one predicts where is our lane

Majority is not defined for non-binary decisions, **so what can be done?**



The Primary-Guardian-Fallback (PGF) Fusion

We propose a general approach for generalizing the majority rule to non binary decisions

We build 3 systems

- Primary (P) – Predicts where the lane is
- Guardian (G) – Checks if the prediction of the primary system is valid or not
- Fallback (F) - Predicts where the lane is

Fusion:

- If Guardian dictates Primary is valid, choose valid
- Otherwise, choose Fallback

Theorem: The PGF has the same property of the majority rule

If the failure probability of each system is at most ϵ and these probabilities are independent, then the fused system has an error of $O(\epsilon^2)$

Proof Consider 3 systems, P, G, F, standing for Primary, Guardian, and Fallback. The Primary and Fallback systems are regular SDSs that output a trajectory. The Guarding system gets the trajectory of the primary system and outputs a Boolean, $\hat{\epsilon}_G(P)$, where the goal of this Boolean is to estimate the value of $\epsilon(P)$.

The PGF fusion system is:

$$\phi(P, G, F) = \begin{cases} F & \text{if } \hat{\epsilon}_G(P) \\ P & \text{else} \end{cases}$$

That is, the fused system selects the Fallback trajectory if the Guardian systems estimates that the Primary system fails, and otherwise the fused system selects the Primary system.

Let's analyze the probability that the fused system will err.

$$\begin{aligned}
&\mathbb{P}[\epsilon(\phi(P, G, F))] = \mathbb{P}[\epsilon(P) \wedge !\hat{\epsilon}_G(P)] + \mathbb{P}[\epsilon(F) \wedge \hat{\epsilon}_G(P)] \\
&= \mathbb{P}[\epsilon(P) \wedge !\hat{\epsilon}_G(P)] + \mathbb{P}[\epsilon(F) \wedge \hat{\epsilon}_G(P) \wedge \epsilon(P)] + \mathbb{P}[\epsilon(F) \wedge \hat{\epsilon}_G(P) \wedge !\epsilon(P)] \\
&\leq \mathbb{P}[\epsilon(P) \wedge !\hat{\epsilon}_G(P)] + \mathbb{P}[\epsilon(F) \wedge \epsilon(P)] + \mathbb{P}[\epsilon(F) \wedge \hat{\epsilon}_G(P) \wedge !\epsilon(P)] \\
&\leq \epsilon^2 + \epsilon^2 + \epsilon^2
\end{aligned}$$

■

Mobileye Compound AI System (CAIS)

Extremely Efficient AI

AV Alignment

RSS

Separates correct from incorrect

Reaching Sufficient MTBF

Abstractions

- Sense / Plan / Act
- Analytic calculations: RSS, time-to-contact…

Redundancies

Sensors

Algo

High level fusion

Extremely Efficient AI

- Transformers for Sensing and Planning at x100 efficiency
- Inference chip (EyeQ6H): Design for efficiency
- Efficient labeling by Auto Ground Truth
- Efficient modularity by teacher-student architecture

Prologue

6 AI Revolutions

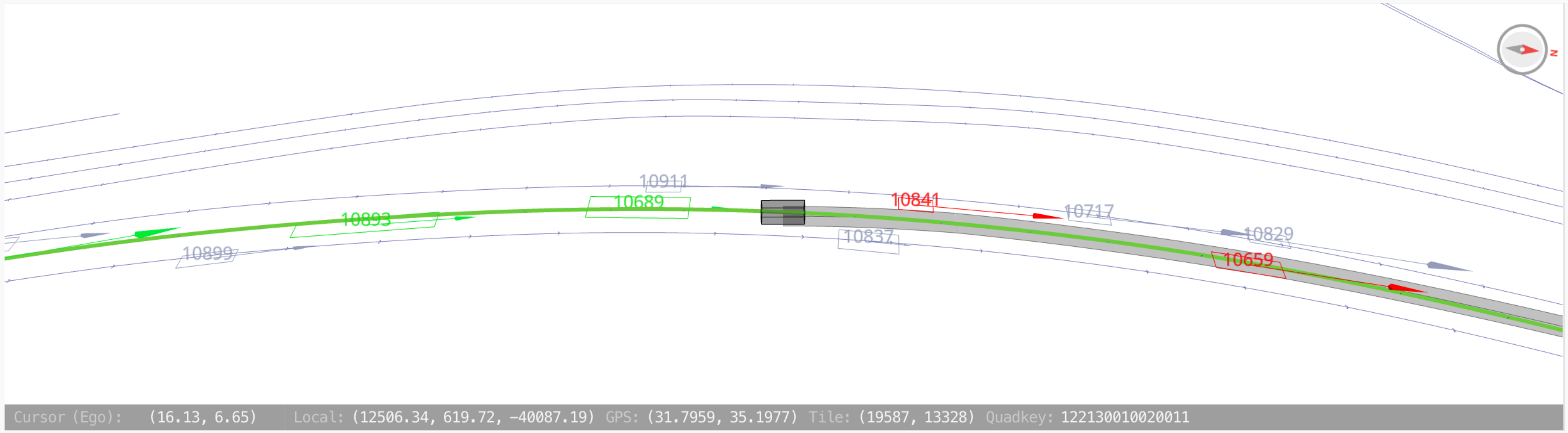
- Machine Learning
- Deep Learning
- Generative AI
- Universal Learning
- Sim2Real
- Reasoning

Transformers

Pre-Transformers: Object Detection Pipeline

Clustering and max suppression

2D to 3D



Three Revolutions of Generative Pretrained Transformers (GPTs)

Tokenize everything

Generative, Auto-regressive

Transformer architecture: ’Attention is all you need’

Three Revolutions of Generative Pretrained Transformers

01

Tokenize everything

Input: Transcribe each input modality (e.g., text, images) into a sequence of tokens

Output: Transcribe each output modality as a sequence of tokens and employ generative, auto-regressive models with suitable loss function

Accommodates: Complex input and output structures (e.g., sets, sequences, trees)

Object detection pipeline example:

Input
Single image

’Tokenized’ input
Sequence of image patches

‘Tokenized’ output
Sequence of 4 coordinates determining the location of the objects in the image

Three Revolutions of Generative Pretrained Transformers

02

Generative, Auto-regressive

Previous approach: Classification or regression with fixed, small size, outputs (e.g., ImageNet)

Current approach: Learn probabilities for sequences of arbitrary length (e.g., sentence generation)

Key Features: Chain Rule – Models sequence dependencies

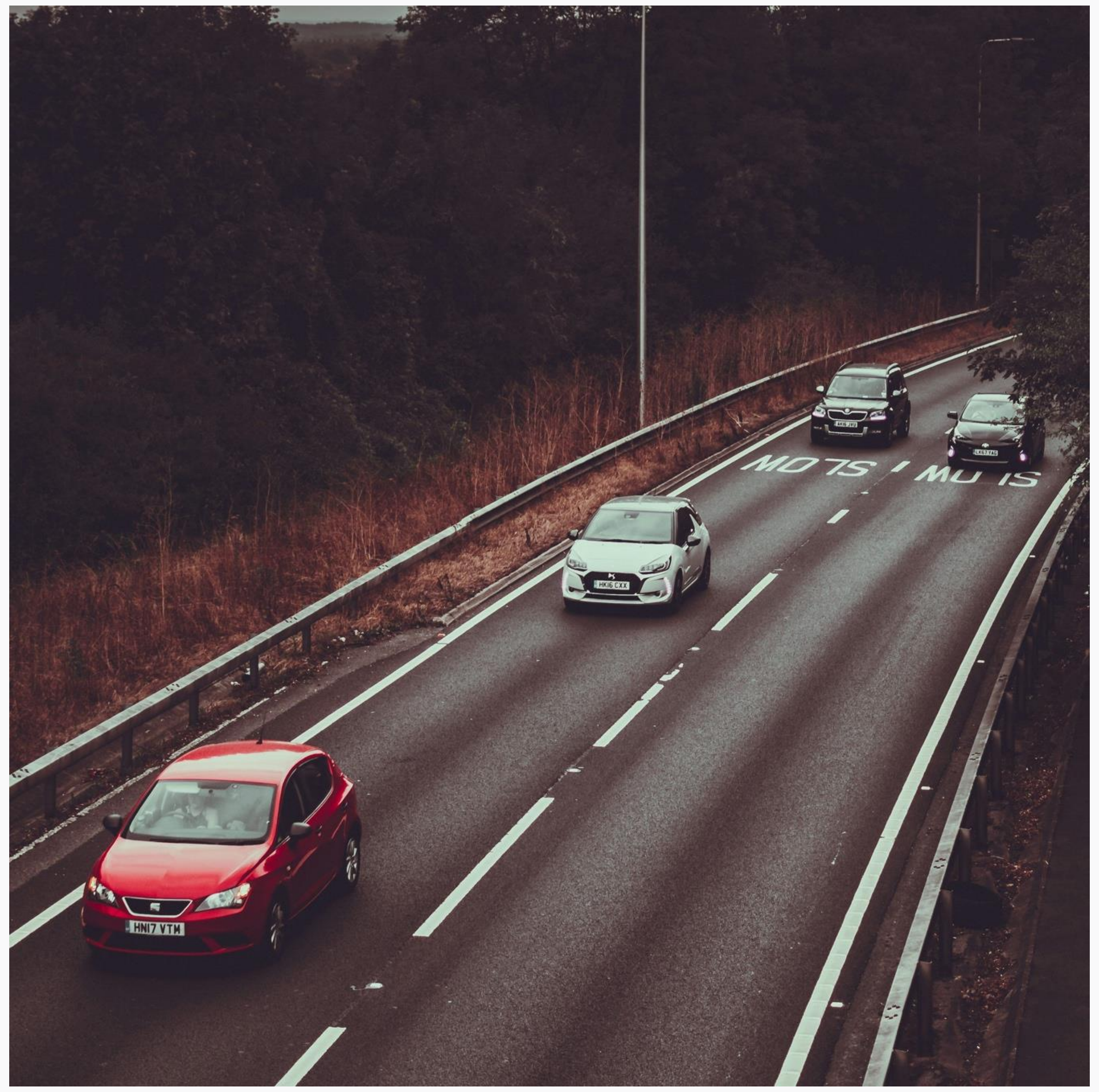
Generative – Fits data using maximum likelihood

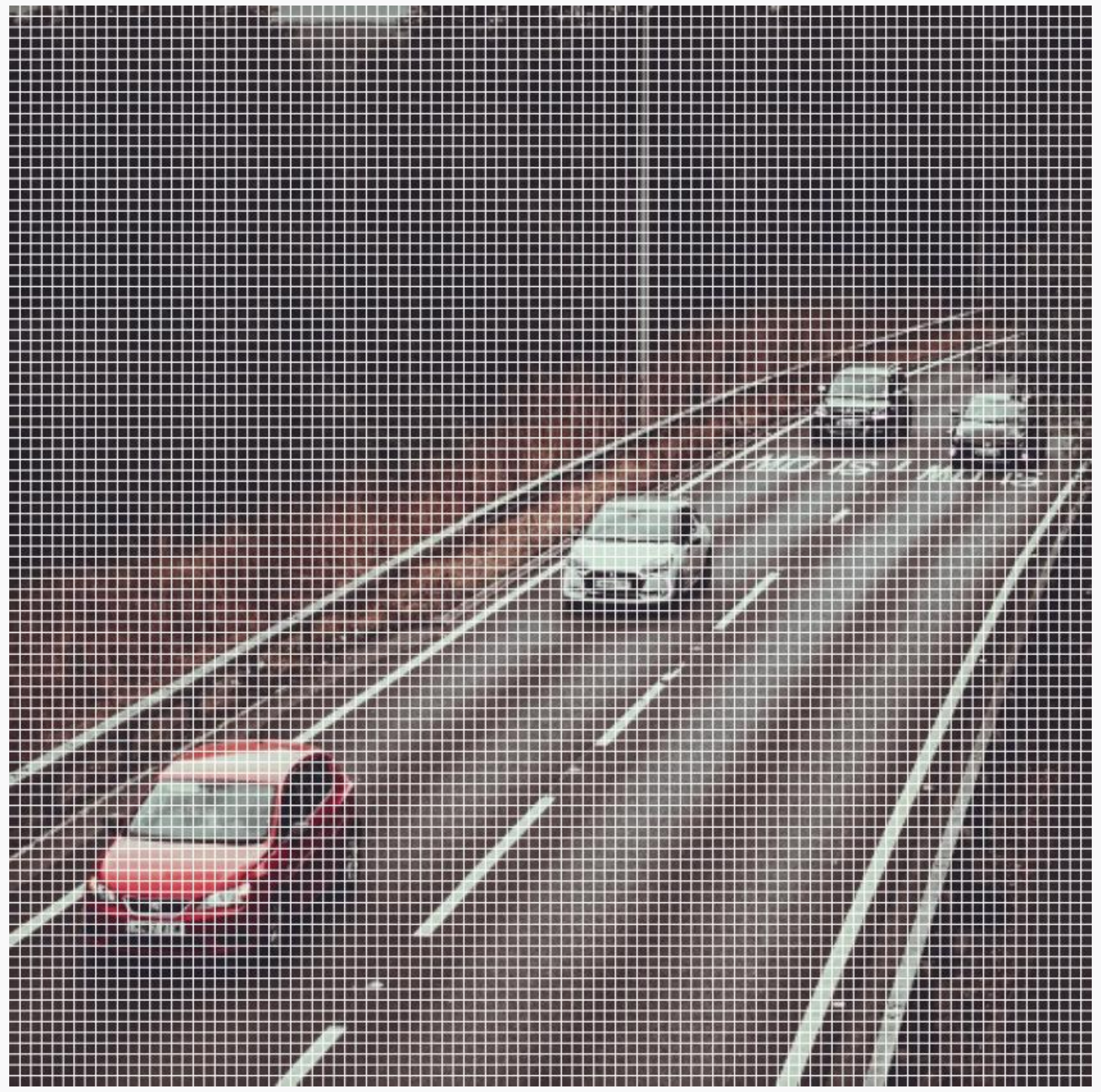
Enables: Self-supervision (e.g., future words in a document)

Handles uncertainty (multiple valid outputs by learning $\mathrm{P}[y|x]$)

Three Revolutions of Generative Pretrained Transformers

Example: Consider a 1000x1000 pixel image containing 4 vehicles, with the image divided into 10x10 pixel patches. What are the probabilities for identifying vehicle positions when not using the chain rule compared to when using the chain rule?





List of 4 coordinates per vehicle $\left(x_{1,1}, y_{1,1}, x_{1,2}, y_{1,2}, \dots, x_{4,1}, y_{4,1}, x_{4,2}, y_{4,2}\right)$

Without using the chain rule

$$P(vehicles|I) = P\left(x_{1,1}, y_{1,1}, x_{1,2}, y_{1,2}, \dots, x_{4,1}, y_{4,1}, x_{4,2}, y_{4,2} | I\right)$$

$$\mathrm{Dim} = 10^{32}$$

Using the chain rule

$$P(vehicles|I) = P\left(x_{1,1}|I\right) * P\left(y_{1,1}|x_{1,1}, I\right) * \cdots * P\left(y_{4,2}|x_{1,1}, \dots, x_{4,2}, I\right)$$

$$\mathrm{Dim} = 100$$

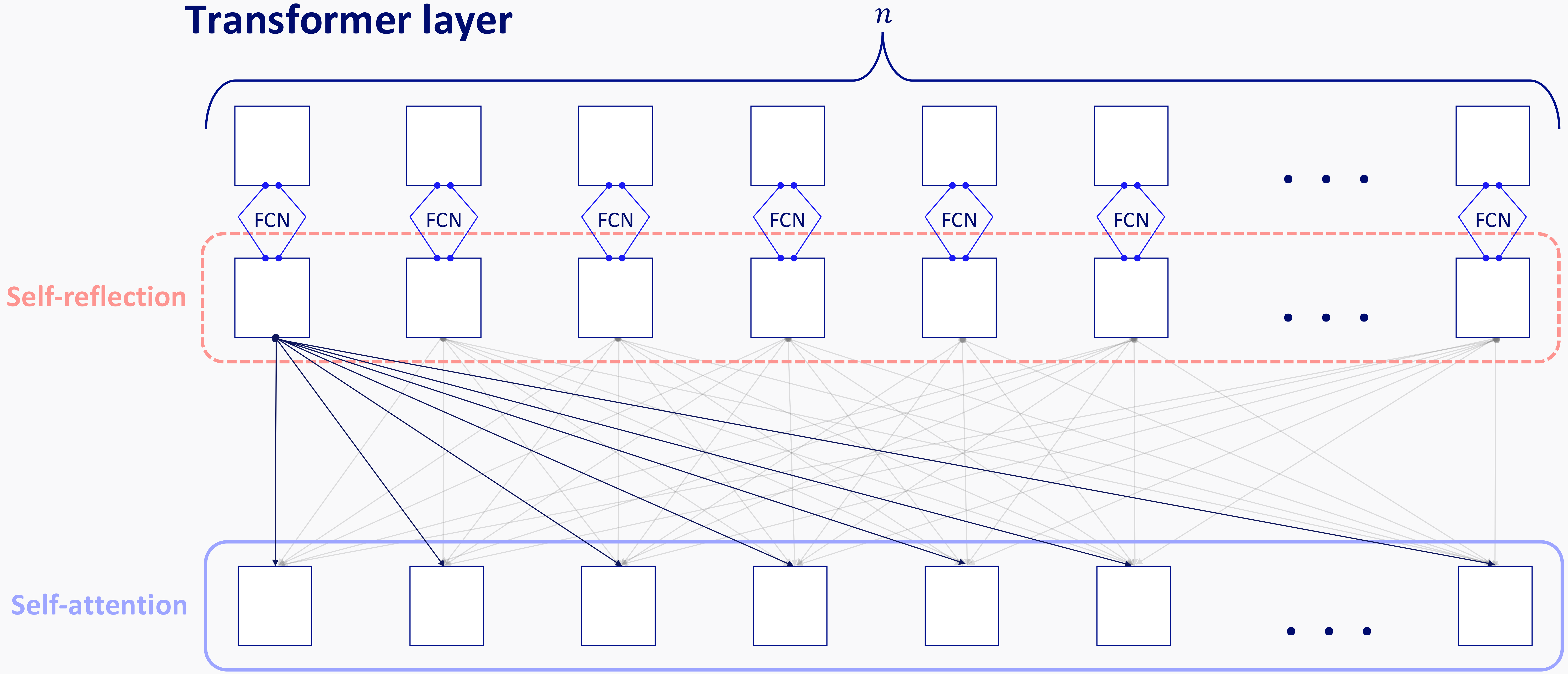
Three Revolutions of Generative Pretrained Transformers

03

Transformer architecture: ’Attention is all you need’

Tailored for problem of predicting $P[token_{n+1} |\ token_n, token_{n-1}, \ldots, token_0]$

Transformer layer



Transformers Layer: Group Thinking Analogy

Imagine a team discussing a project

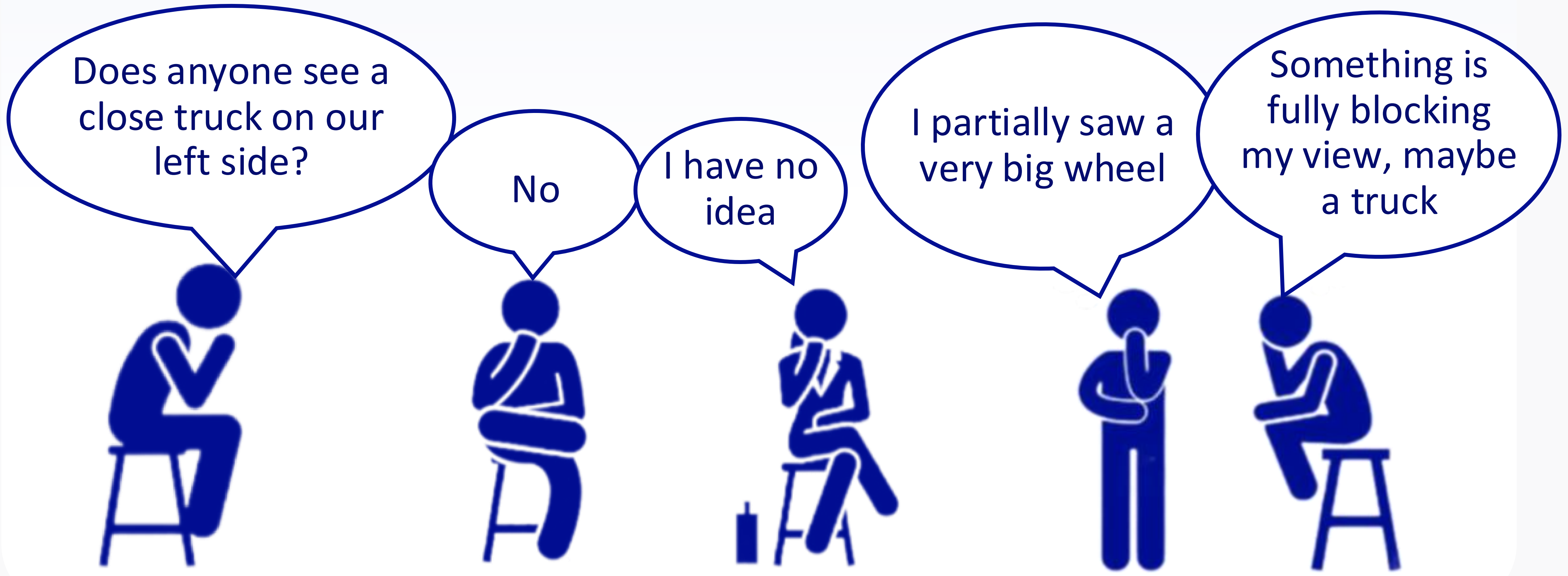
- Each person has their own area of expertise
- they all contribute to the overall outcome
- Everyone is working simultaneously rather than one after another

Self-reflection

Each participant takes time alone to process ideas and organize their thoughts

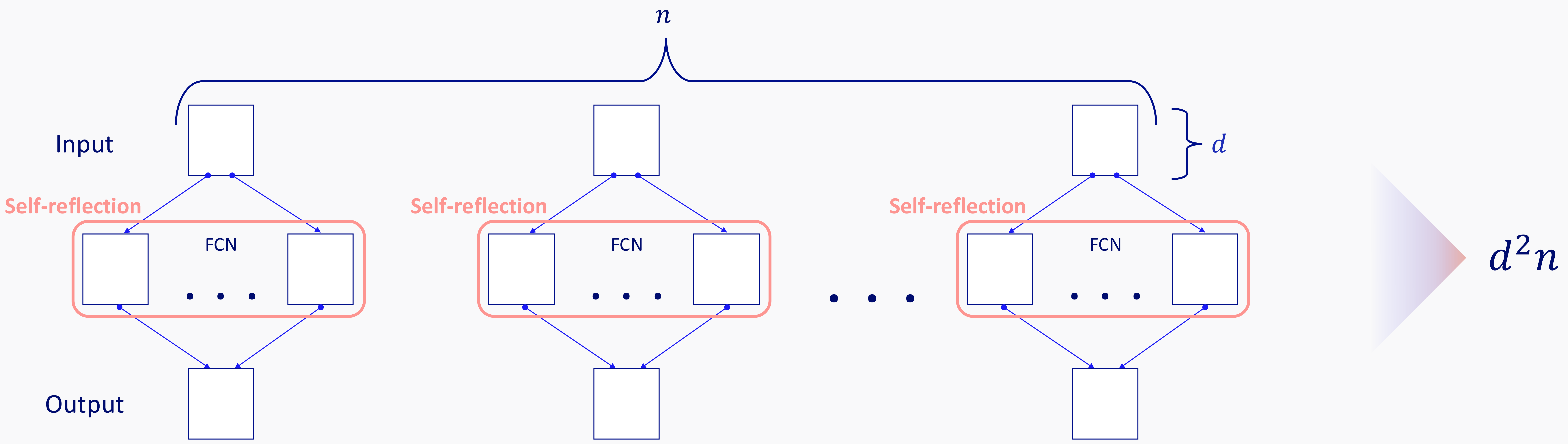
Self-attention

Each member listens to others and responds in real-time, adjusting their input based on important points raised



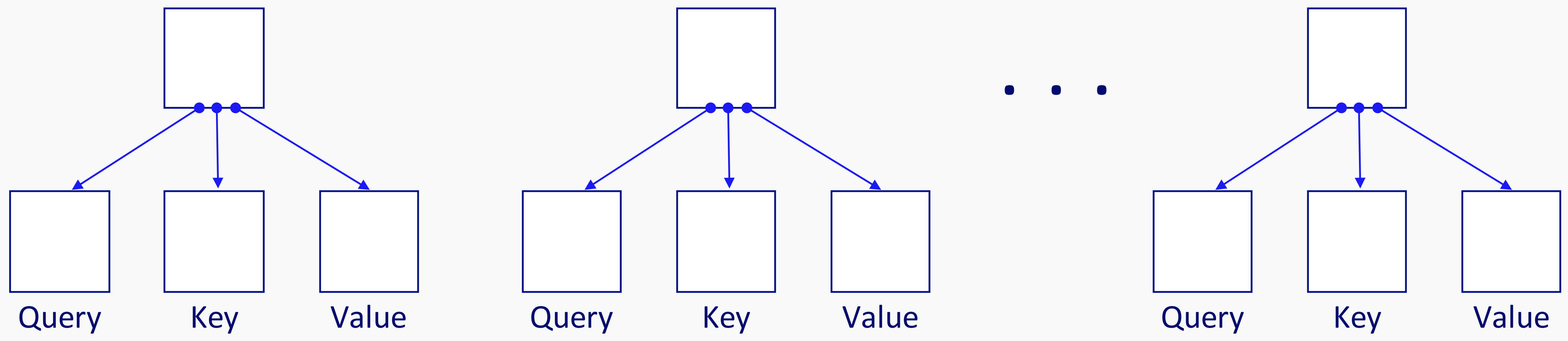
Transformers Layer: Self-Reflection

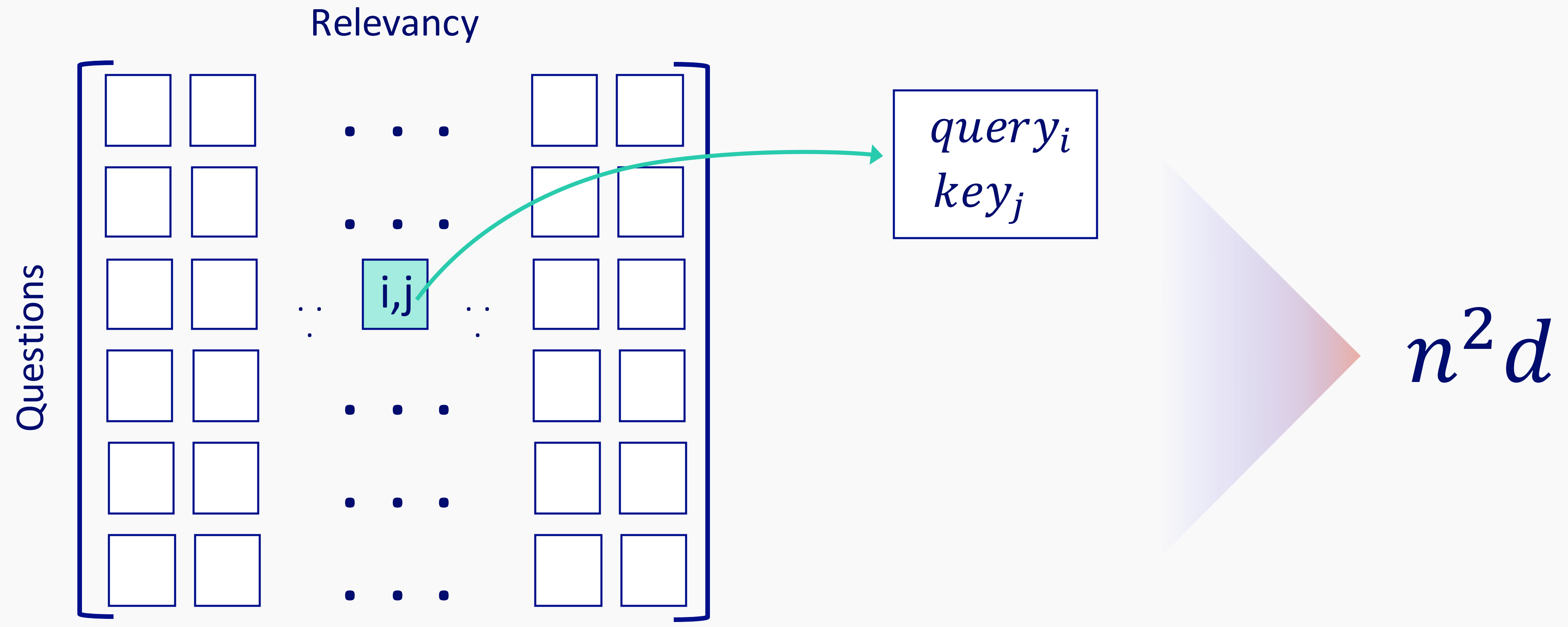
- Each token individually processes its ‘knowledge’ using a multi-layer-perceptron, without interacting with other tokens



Transformers Layer: Self-Attention

- Each token send ‘query’ to the other tokens, which respond with values if their ‘key’ match the ‘query’
- The querying token then averages the received values, facilitating inter-token connectivity



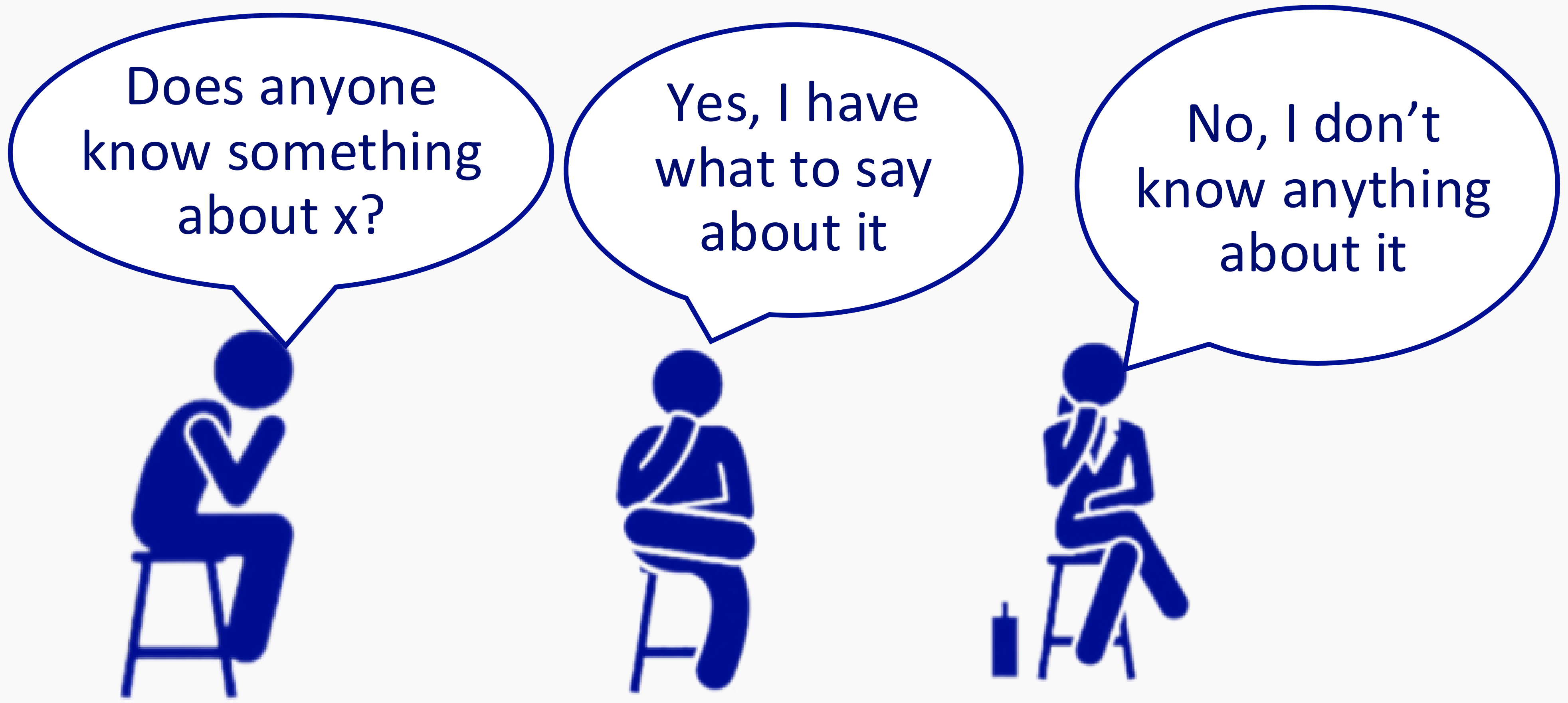


Example from the Group Thinking Analogy

Person i asks: “Does anyone knows something about x?”

Person j responds: “Yes, I have what to say about it”

Person j' responds: ”No, I don’t know anything about it”

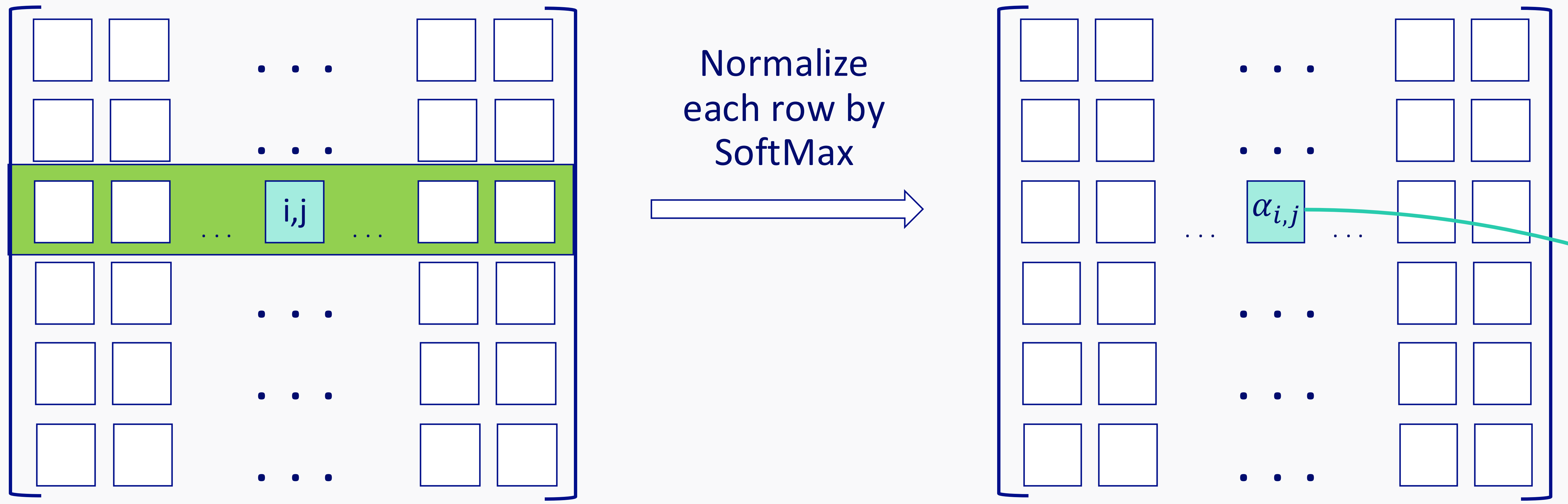


Transformers Layer: Self-Attention

Normalizes Scores: It converts raw attention scores into **normalized probabilities**

Probability Distribution: Each set of attention scores is transformed so that their probabilities sum to 1

Focus Mechanism: This allows the model to weigh different parts of the input differently, focusing more on relevant parts based on the probabilities



Message i gets from the group

$$\sum_j \alpha_{i,j} V_j$$

Indicates how much i wants to pay attention to j

Transformers: Complexity

$$\boldsymbol{L * (n\, d^2 + n^2 d)}$$

#layers

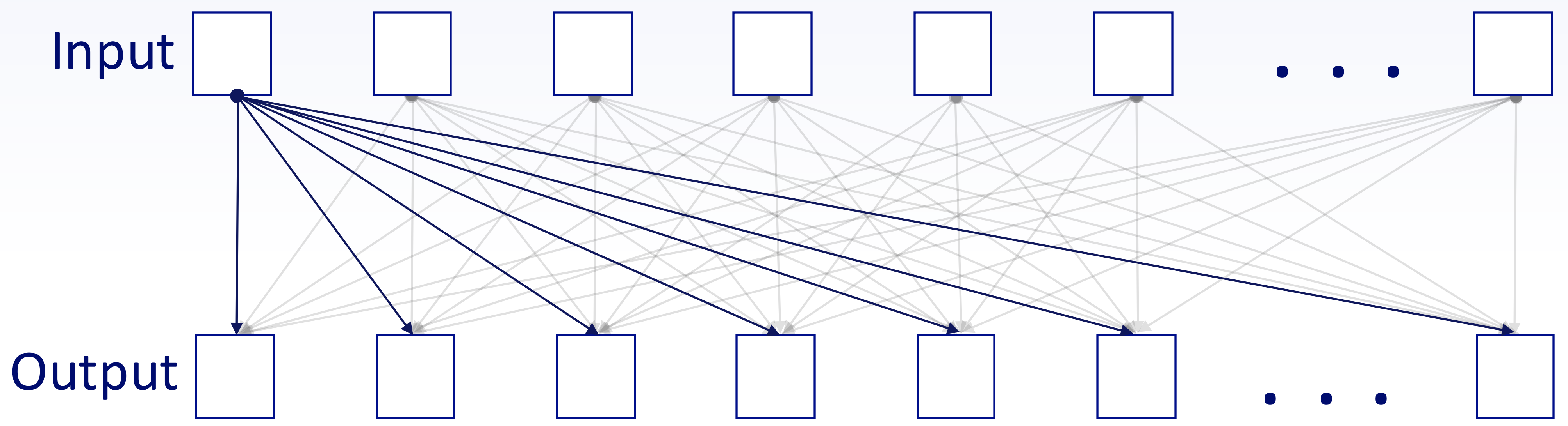
Self reflection

Self attention

Cost per layer for alternative architectures:

Fully Connected Network (FCN)

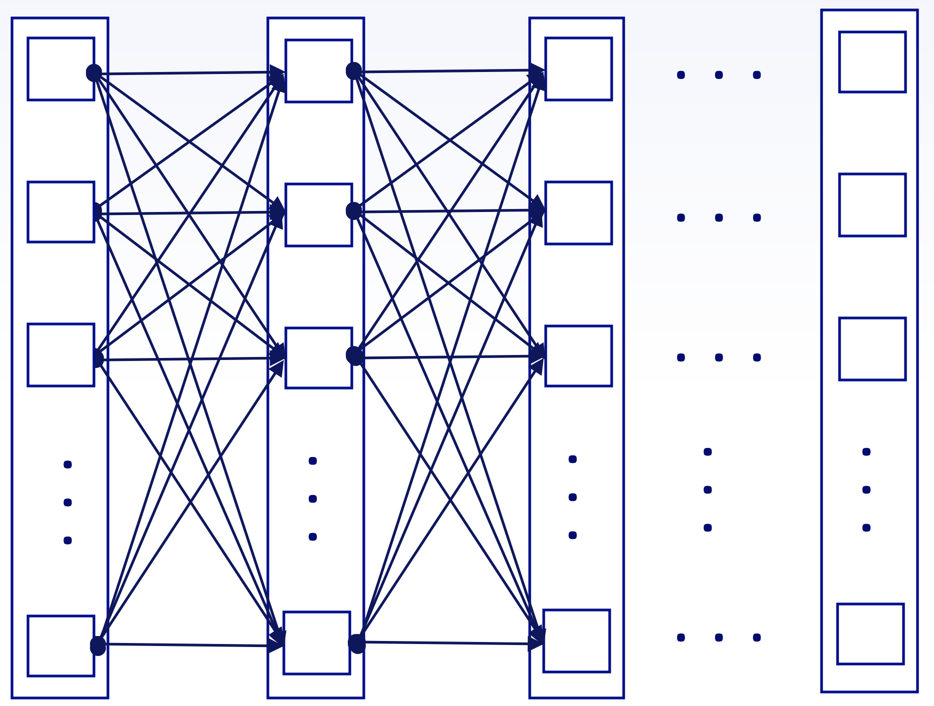
Flatten nd values



Connections: $\mathbf{d}^2 \boldsymbol{n}^2$

Recurrent Neural Network (RNN)

'Talks' only with previous token



Connections: $\boldsymbol{nd}^2$

‘Effective Sparsity’ of Transformers

Transformers

Sparser
$d^2n + n^2d$

Any modality

Denser, but effectively selects only a **few** past tokens for communication

Fully Connected Network (FCN)

d^2n^2 Connections

Convolutional Neural Networks (CNNs)

Sparsity specific to images

Long-Short-Term-Memory (LSTM)

Recurrent Neural Networks (RNN)

Markov sparsity context represented by a state vector

The 3 Revolutions Enable a Universal Solution

Handle all types of inputs

Deals with uncertainty (by learning probability)

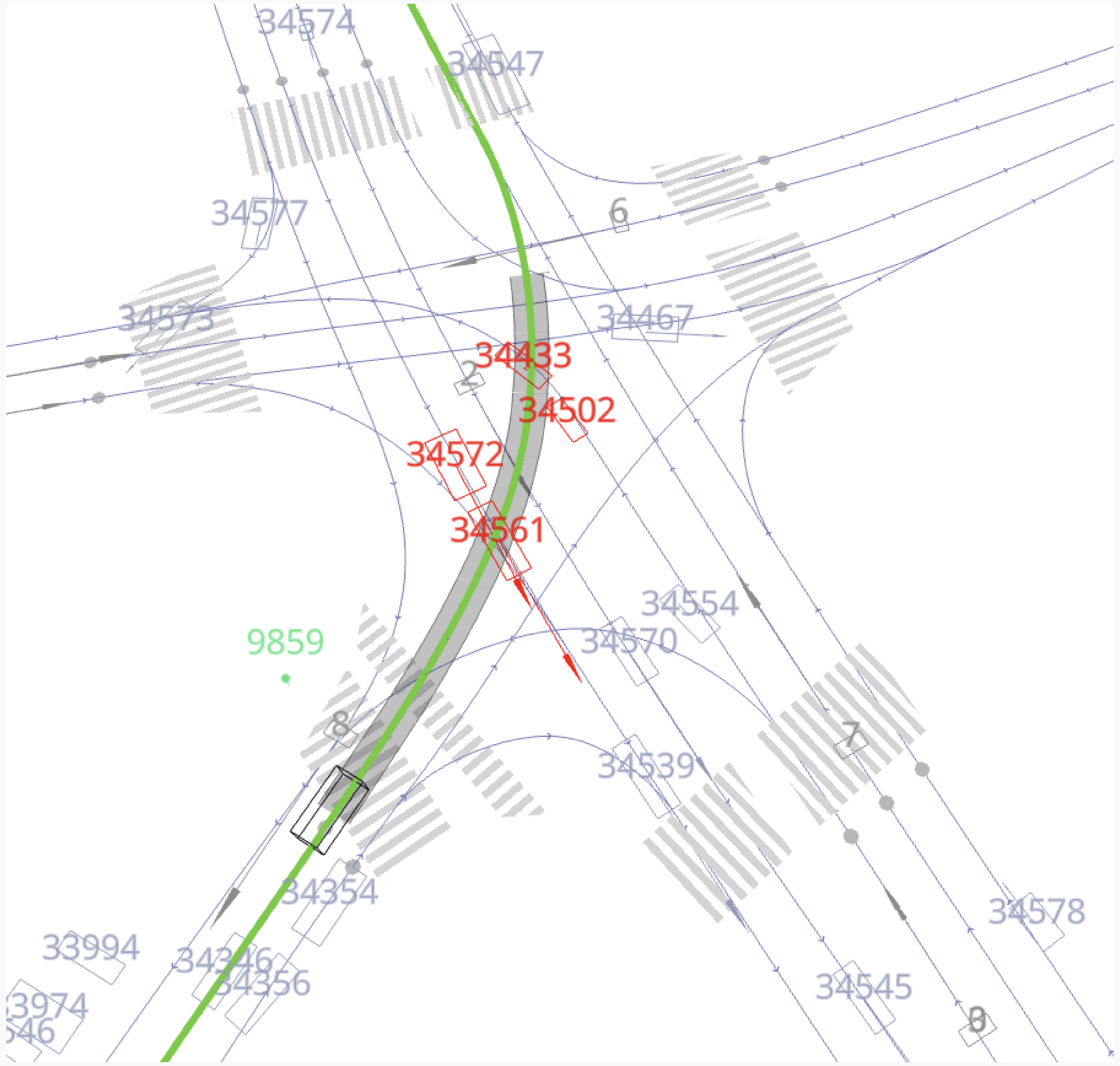
Enables all types of outputs

The ultimate learning machine?

A Transformer End-to-end Object Detection Network

Input: images

Output: all objects



A Transformer End-to-end Object Detection Network

The 5 "Multi" problems

Multi-camera: surround

Multi-frame: : from multiple time stamps

Multi-object: needs to output all (vehicles, pedestrians, hazards, …)

Multi-scale: needs to detect far and close objects at different resolutions

Multi-lanes: needs to assign objects to relevant lanes / crosswalks

- **Universality of Transformers**
  - Encode image patches (from different cameras, different frames, and different resolutions) as tokens
  - Encode objects as a sequence of tokens (for each object: position, velocity, dimensions, type)
  - Apply a Transformer to generate the probability of output tokens given input tokens in an Auto-Regressive manner

Network Architecture: Vanilla Transformer

- **CNN backbone for creating image tokens**:
  - $C = 32$ high resolution images are converted to 32 images of resolution 20x15 yielding $N_p = 300$ "$pixels$" per image, and $d = 256$ channels
- **Encoder:**
  - We have $N = C * N_p = 9600$ "$image\ tokens$", each at dimension $d = 256$
  - A vanilla transformer network with L layers requires $O(L * (N^2 d + d^2 N))$
  - Encoder alone requires around 100 TOPs (assuming 10Hz, L=32)
- **Decoder:**
  - Predict a sequence of tokens representing all the objects (hundreds of tokens)
  - A vanilla AR decoding is **sequential,** and with KV cache, each iteration involves compute of at least $O(LNd)$ per token prediction (but the real issue is IO of LNd here)
  - Around 100Mb per token prediction!

Vanilla Transformers are Not Effiecient

Transformers are a brute force approach with limited way to utilize prior knowledge

This is the "dark side" of universality

Self-connectivity: nd^2

GPT3

$d = 12288 \quad n = 2048$

$nd^2 = 317B$

Inter-connectivity: $n^2 d$

$n^2 d$

In AV $n \approx 10^4$,which becomes a bottleneck

We pay both

- Sample complexity (d is large as it needs to handle all the information in each token)
- Computational complexity of inference (n, d are large)

- (both issues are known in the literature, and general mitigations such as "mixture-of-experts" and "state-space-models" were proposed)

What About End-to-End From Pixels to Control Commands

Weaknesses of transformers

01 Brute force

02 The learning objective (of learning $\mathrm{P}[y|x]$) prefers ‘common & incorrect’ $\boldsymbol{y}$ over 'rare & correct’ $\boldsymbol{y}$

03 **Questionable whether it can reach sufficiently high MTBF**

- Misses important abstractions and therefore doesn’t generalize well
- The Shortcut Learning Problem

04 (as part of CAIS, our e2e architecture has an additional head that outputs control commands directly as well, which is fine as a low MTBF redundant component)

Mobileye Compound AI System (CAIS)

AV Alignment

RSS

Separates correct from incorrect

Reaching Sufficient MTBF

Abstractions

- Sense / Plan / Act
- Analytic calculations: RSS, time-to-contact…

Redundancies

Sensors | Algo | High level fusion

Implications

- Must output Sensing State
- Each subsystem must be super efficient because we don't have a single system

Extremely Efficient AI

Transformers for Sensing and Planning at x100 efficiency

Inference chip (EyeQ6H): Design for efficiency

Efficient labeling by Auto Ground Truth

Efficient modularity by teacher-student architecture

STAT: Sparse Typed Attention

Vanilla transformer: $n^2 d + \mathrm{d}^2\mathrm{n}$

STAT:

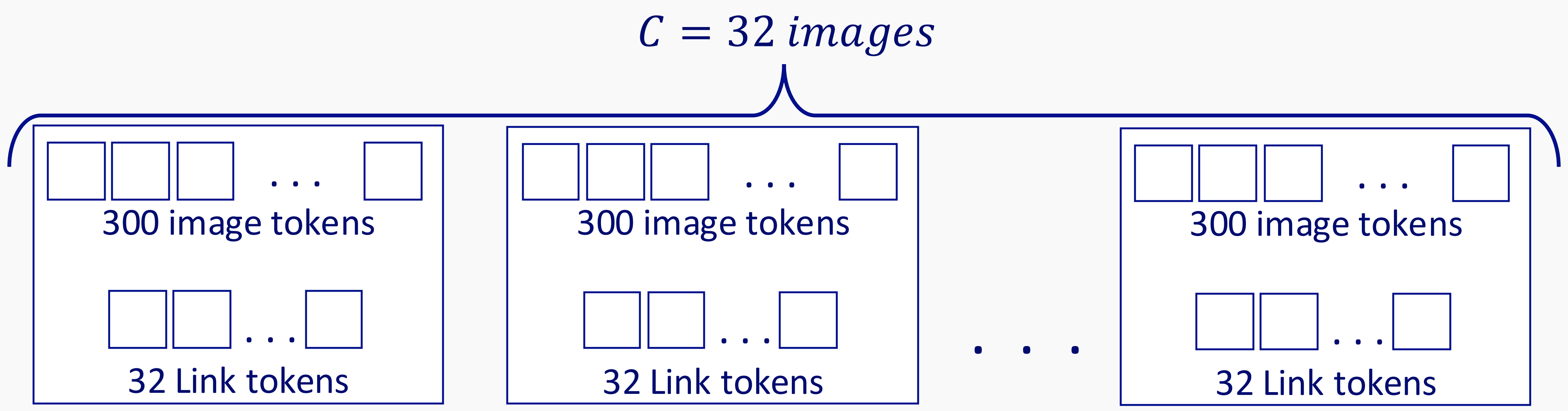
- **Token Types**: Each token has a “type”
- **Dimensionality**: of embeddings and self-reflection matrices may vary based on the token type.
- **Token Connectivity**: The connectivity between tokens is sparse and depends on their types
- **Link Tokens**: We add “link” tokens for controlling the connectivity
- **Inference Efficiency**: For our end-to-end object detection task, STAT is x100 faster at inference time and at the same time slightly improves performance

STAT: Sparse Typed Attention

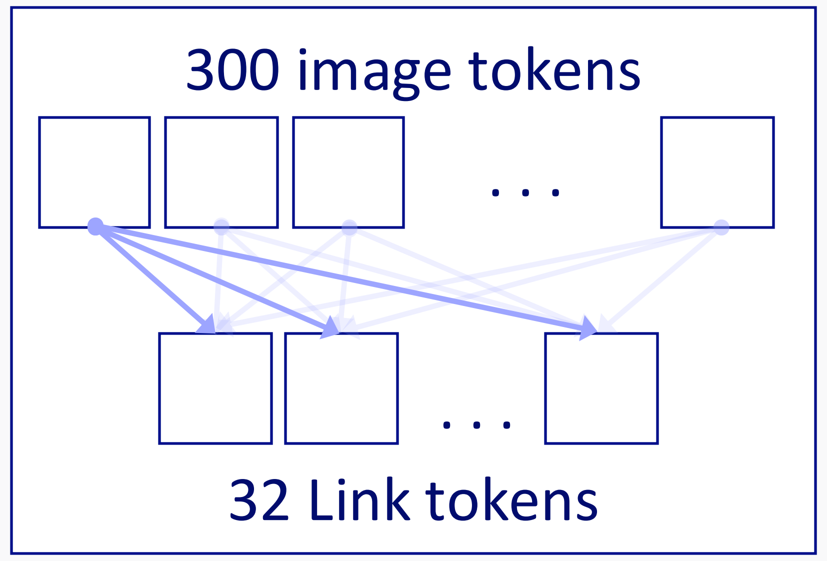
Vanilla transformer: $n^2 d + \mathrm{d}^2 \mathrm{n}$

STAT Encoder for Object Detection:

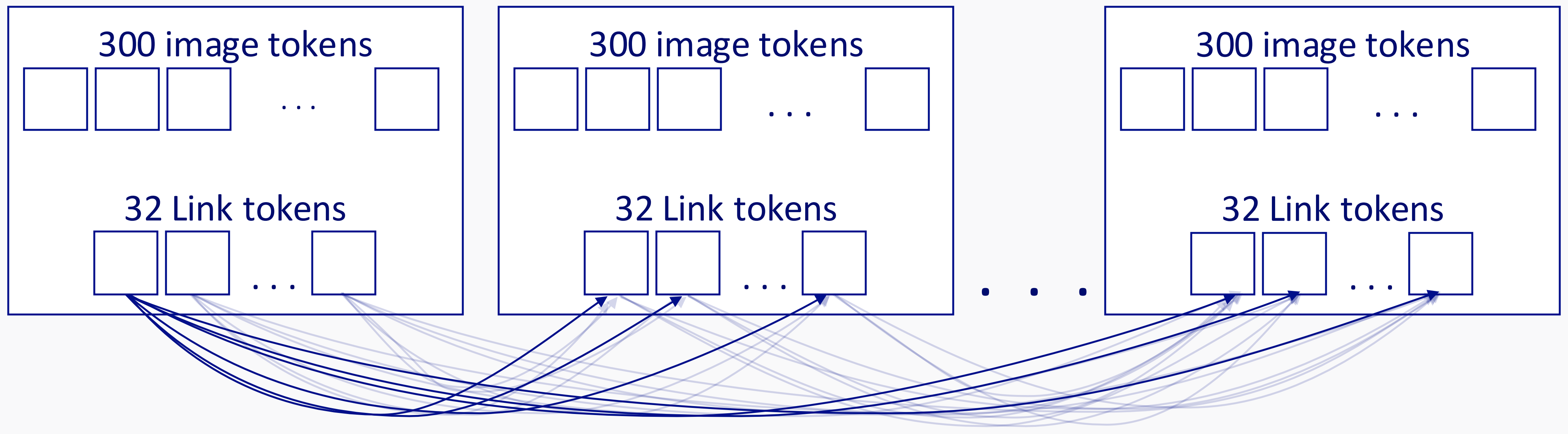
- **Token types:**
  - Image tokens: recall, we have $C = 32$ images each with $N_p = 300$ "$pixels$" , yielding 9600 image tokens
  - We add $N_L = 32$ "$Link$" tokens per image
- **STAT Block**:
  - Within each image, Cross Attention between the 300 image tokens and the 32 link tokens ($C * N_p * N_L * d$)
  - Across images, full self attention between all link tokens $(C * N_L)^2 d$
  - Compared to $\left(C * N_p\right)^2 d$ in vanilla transformers, we get a factor improvement of $(\frac{N_p}{N_L}) * \min(C, \frac{N_p}{N_L})$, which is approximately **x100 faster** in our case
- **Performance**: For our end-to-end object detection task, STAT is not only x100, but also improves performance (we enlarge the expressivity of the network while making it much faster at inference time)



Cross attention



Cross image



Parallel Auto-Regressive (PAR)

We need to detect all objects in the scene: What is the order?

Auto-Regressive: It doesn't matter due to the chain rule !

Price of sequential decoding

- Sequential decoding is costly on all modern deep learning chips (due to IO)
- We added un-needed "fake uncertainty" (what is the order)

DeTR (**DET**ection **Tr**ansformer, Facebook AI, May 2020)

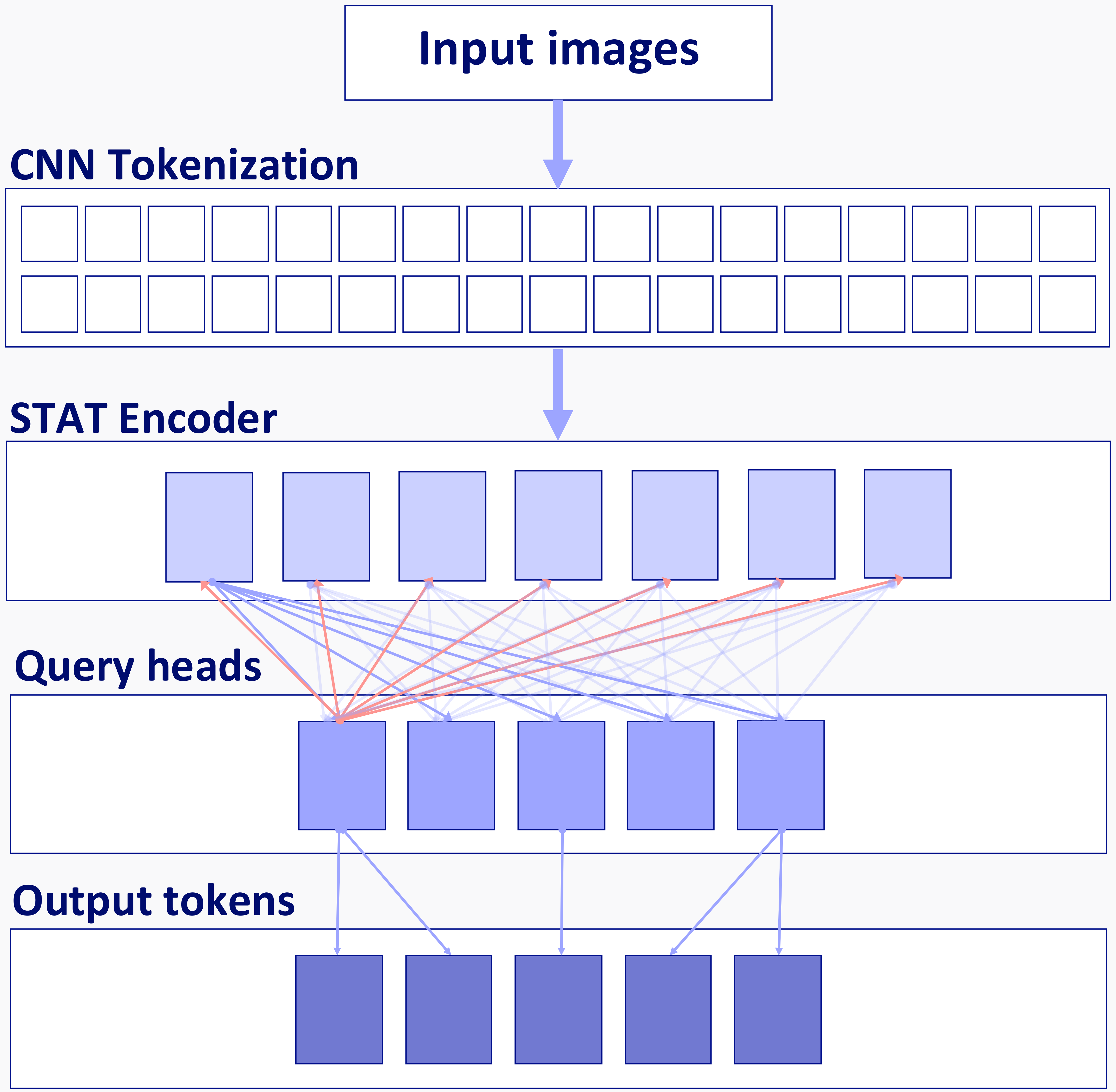
- Output all objects in parallel
- Hungarian matching to determine the relative order between the network's predictions and the order of the ground truth
- **Problem**: Doesn't deal well with true uncertainty
  - The "truck and trailer" problem
  - Streets which can be 1 or 2 lanes, etc.

"Truck and trailer" problem

Paris streets

Parallel Auto-Regressive (PAR)

- The decoder contains query heads which perform cross attention with the encoder's link tokens entirely in parallel
- Each query head outputs, auto-regressively, 0/1/2 objects (independently and in parallel to the other query heads)
- ➞ dealing only with "true uncertainties" and not with "fake uncertainties"



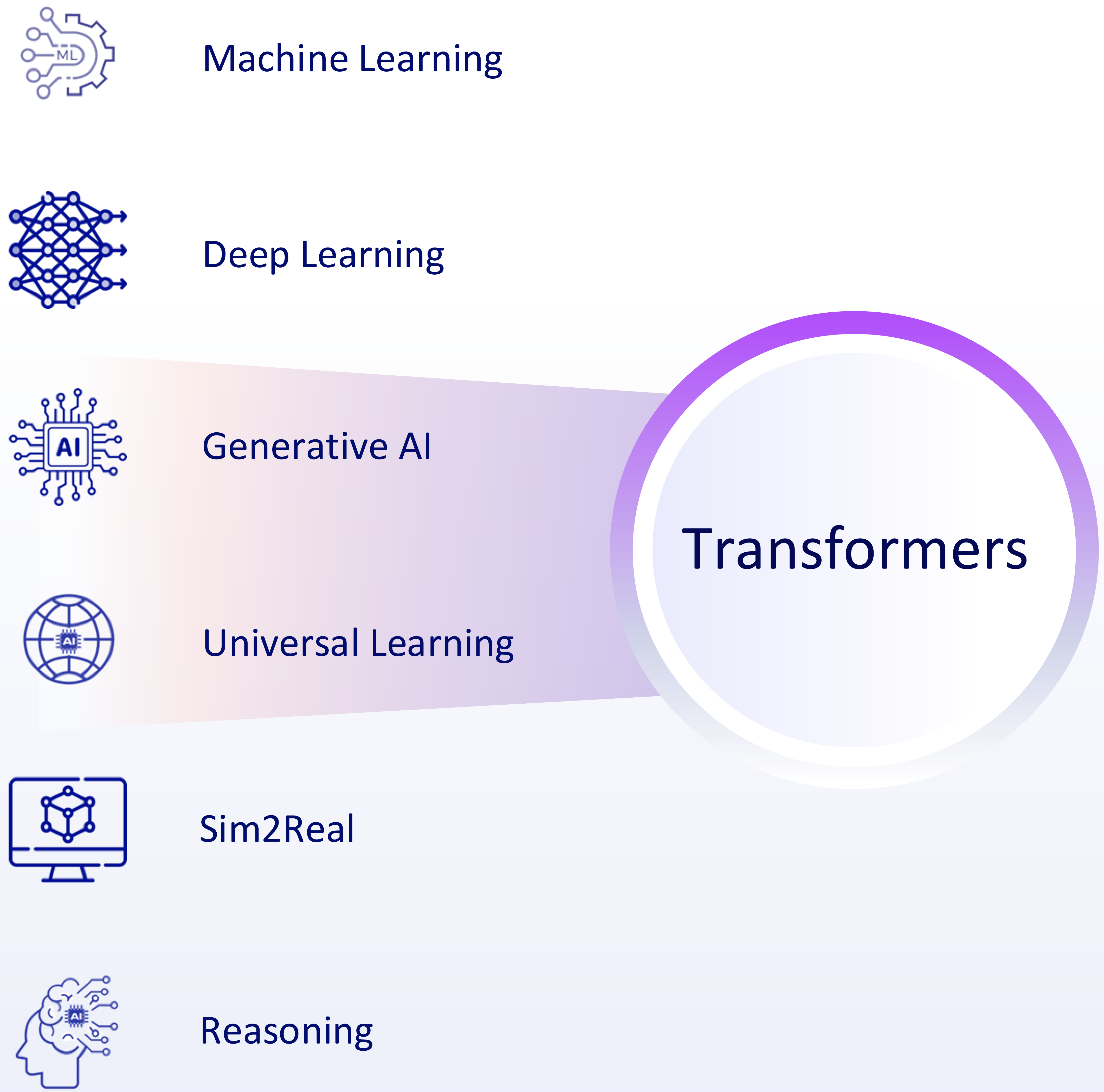
Intermediate Summary

Transformers revolutionized AI

- **The good**
  - Universal, generative, AI
- **The bad**
  - Can’t separate “correct & rare” from “wrong & common”
  - Miss important abstractions
  - Questionable when very high accuracy is required
- **The ugly**
  - Brute force approach, unnecessarily expensive

Working <u>smarter</u> with transformers

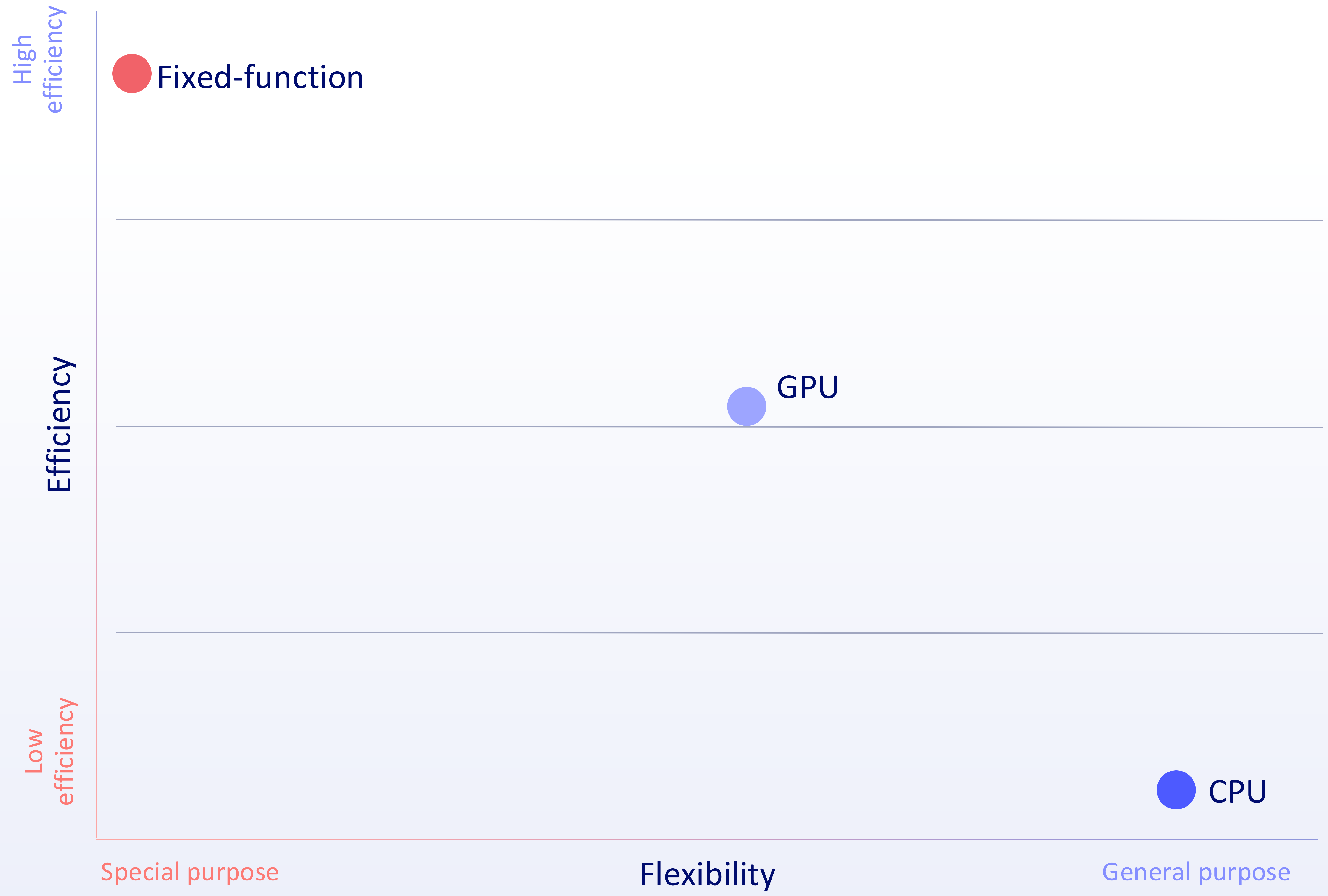
- **STAT**: x100 faster & better accuracy
- **PAR**: x10 faster & embrace uncertainty only when it is needed



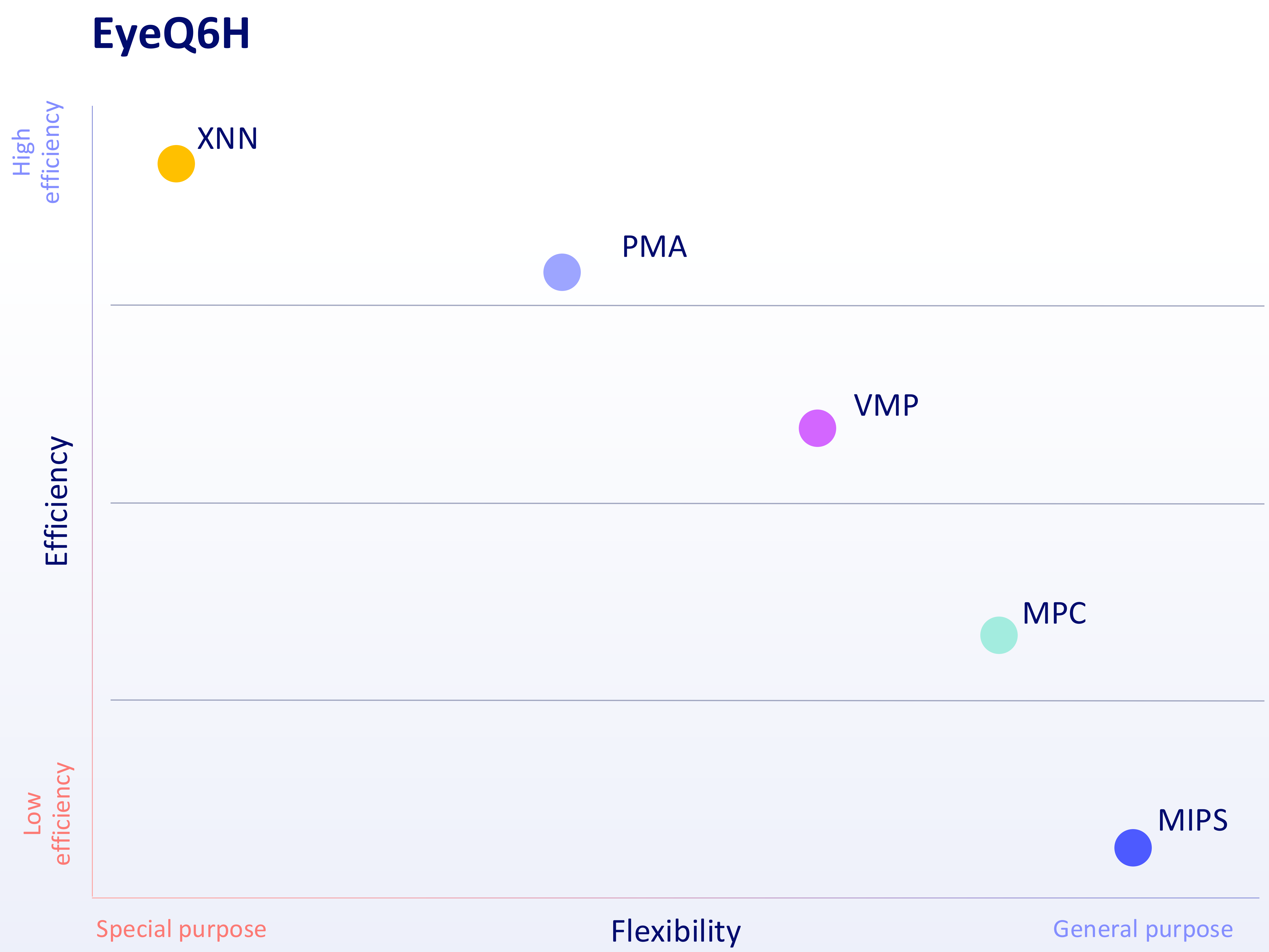
Extremely Efficient AI

- Transformers for Sensing and Planning at x100 efficiency
- **Inference chip (EyeQ6H): Design for efficiency**
- Efficient labeling by Auto Ground Truth
- Efficient modularity by teacher-student architecture

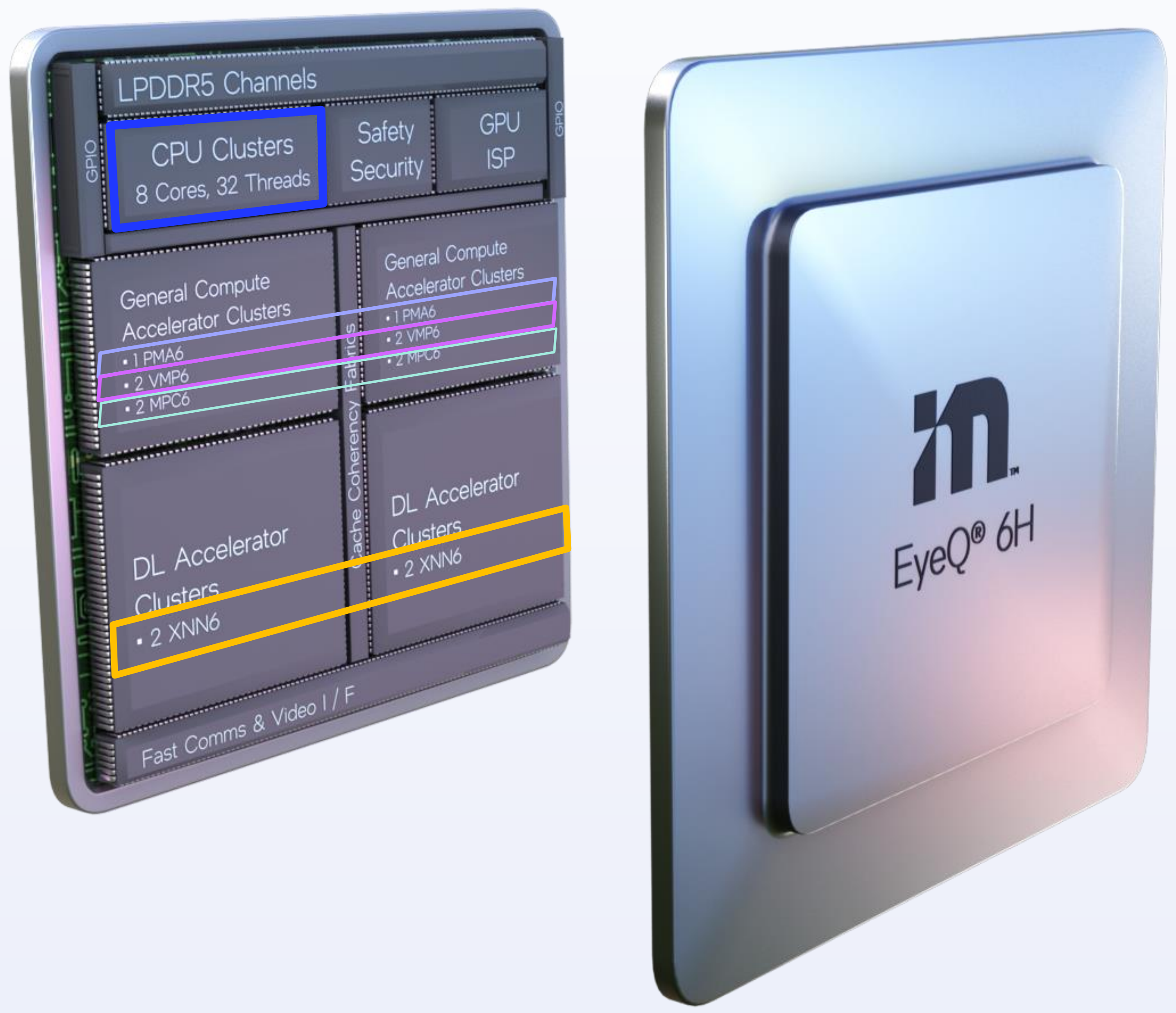
Hardware Architectures Tradeoff: Flexibility vs. Efficiency



EyeQ6 High: 5 Distinct Architectures



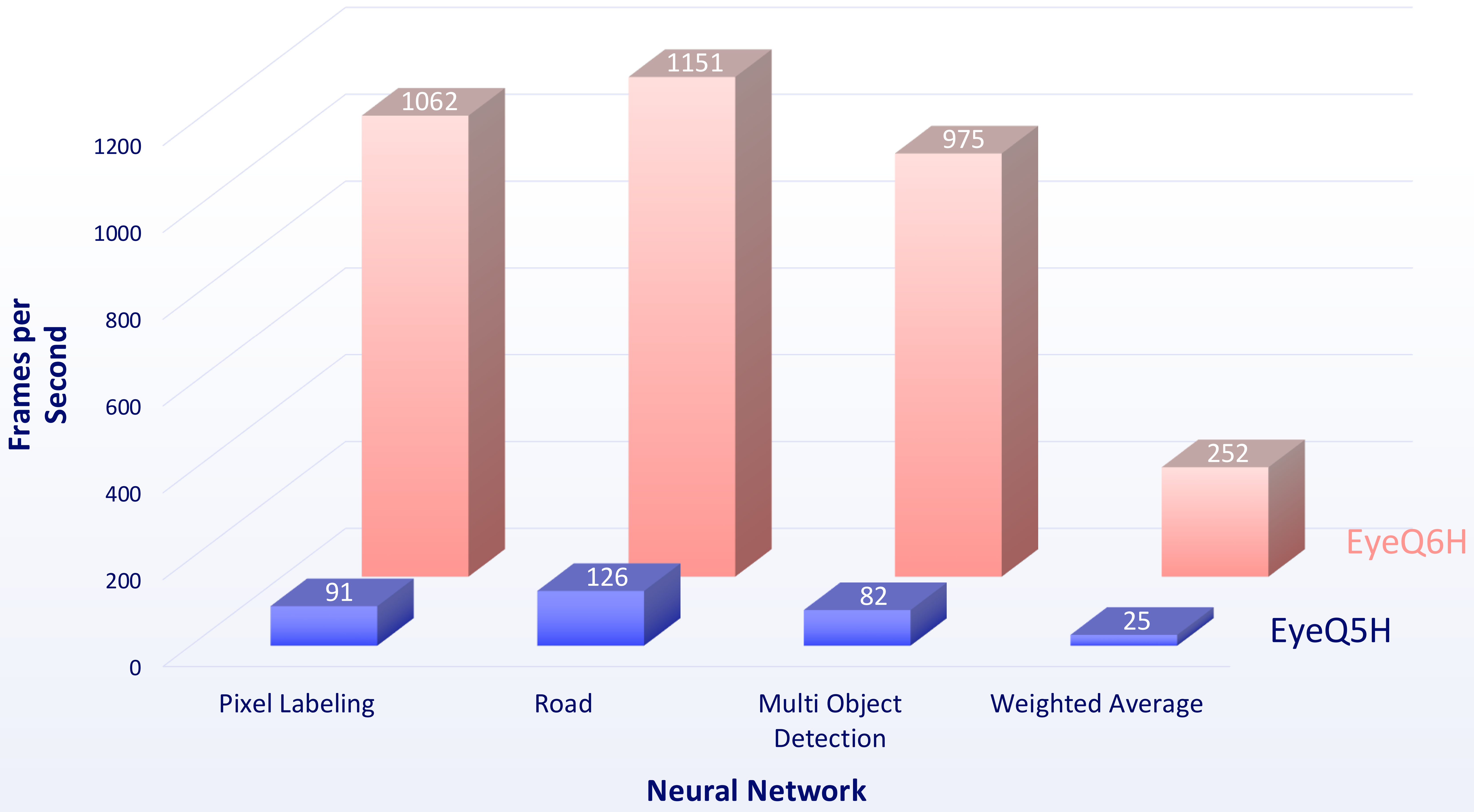
- Address Mobileye's high efficiency and flexibility needs
- Enable accelerating range of parallel compute paradigms



5 Distinct Architectures: Enhanced Parallel Processing

- **MIPS**
  - A general-purpose CPU
- **MPC**
  - A CPU specialized for thread level parallelism
- **VMP**
  - Very-Long-Instruction-Width (VLIW) – Single-Instruction-Multiple-Data (SIMD)
  - Designed for data-level parallelism of fixed points arithmetic (e.g., converge the 12-bit raw image into a set of 8-bit images of different resolutions and tone-maps)
  - Basically, performs operations on vectors of integers
- **PMA**
  - Coarse-Grain-Reconfigurable-Array (CGRA)
  - Designed for data-level parallelism including floating point arithmetic
  - Basically, performs operations on vectors of floats
- **XNN**
  - Dedicated to fixed functions for deep learning: convolutions, matrix-multiplication/fully-connect, and related activation post-processing computations: Excels in CNNs, FCNs, Transformers

EyeQ6H vs. EyeQ5H: 2x in TOPS, But 10x in FPS!



EyeQ5H

16 TOPS (int 8)

27W (max)

EyeQ6H

34 TOPS (int 8)

33W (max)

mobileye

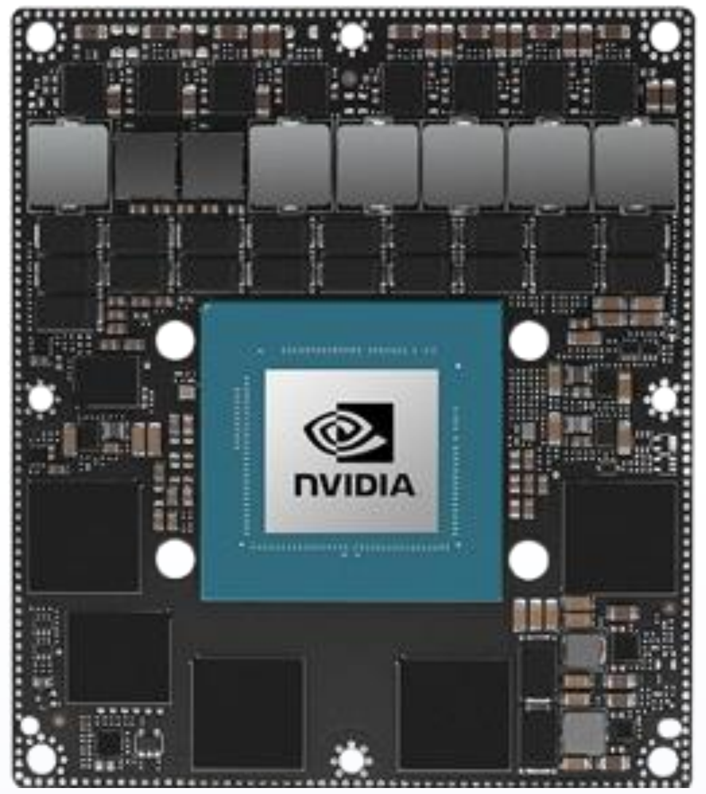
EyeQ6H vs. Orin: It's Not All About TOPS

Theoretical TOPS

34 TOPS (int8)

Factor x8

275 TOPS (int8)



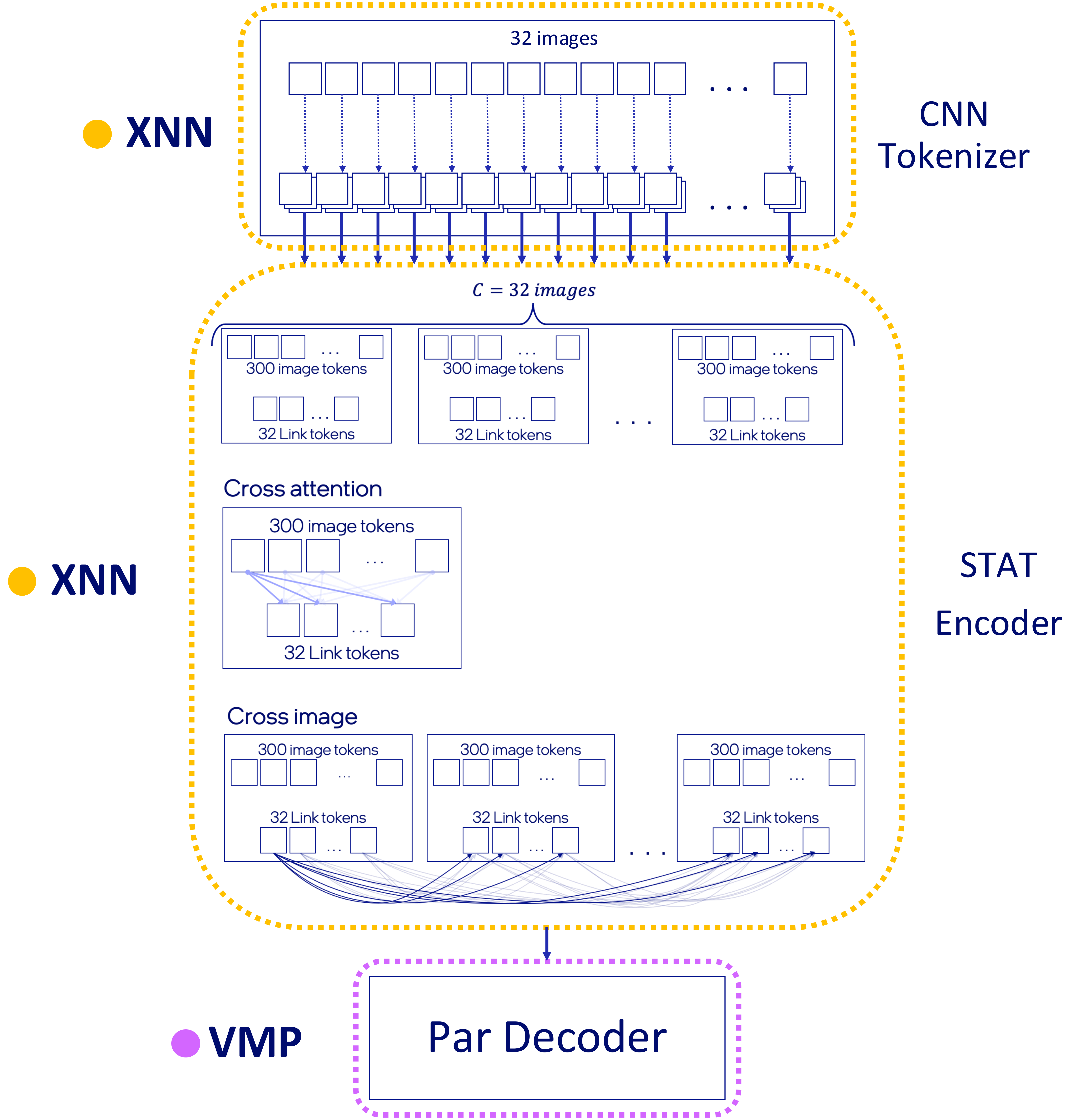
Frames per Second for ResNet50

Only factor x2

Conclusion

- TOPS are a poor measure for compute capabilities

End-to-End Sensing State Network



Extremely Efficient AI

- Transformers for Sensing and Planning at x100 efficiency
- Inference chip (EyeQ6H): Design for efficiency
- **Efficient labeling by Auto Ground Truth**
- Efficient modularity by teacher-student architecture

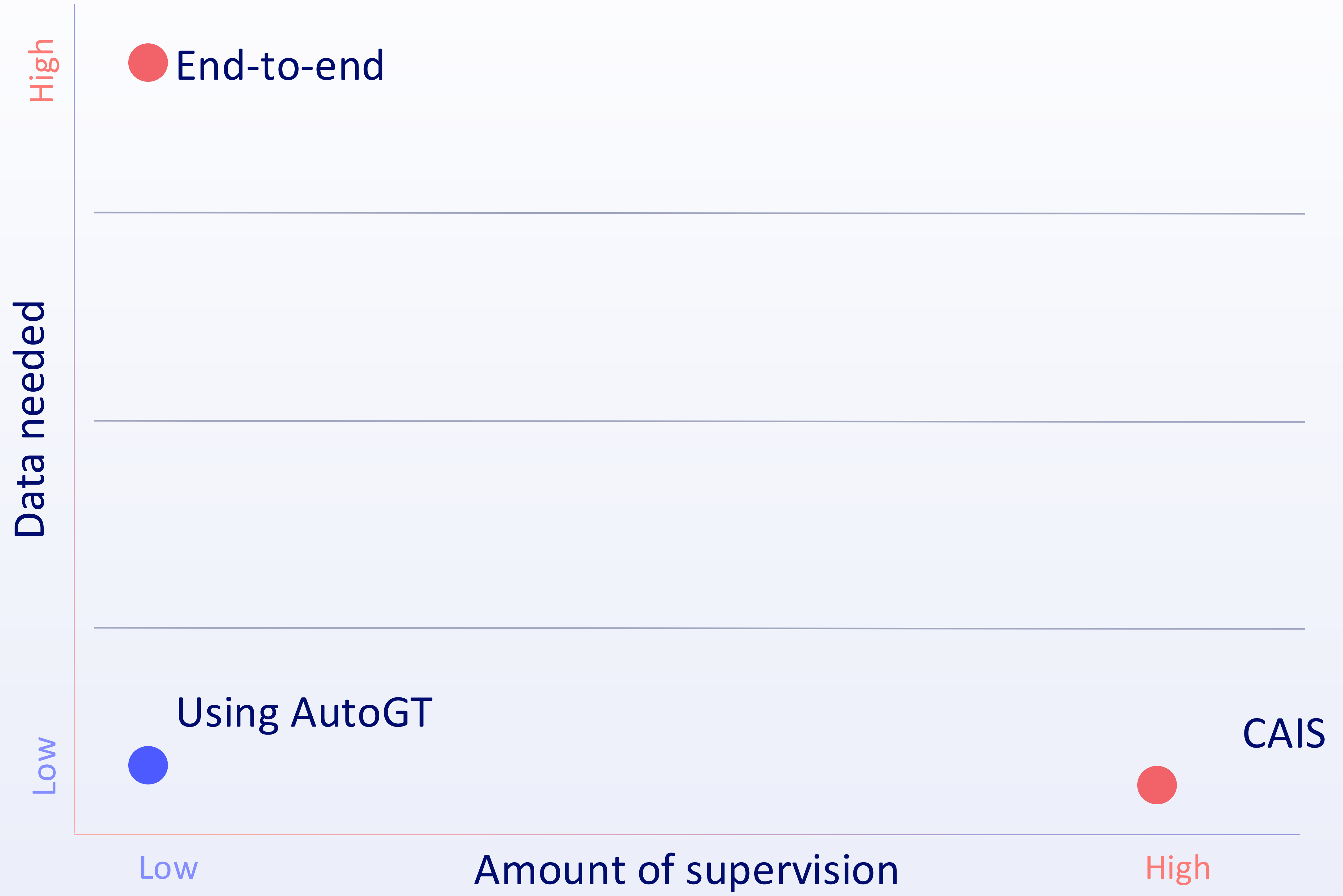
Automatic Ground Truth: CAIS vs. End-to-End

Compound AI System

- **Injecting abstractions**: Sensing State, RSS, PGF, etc.
- **Need to label data**: Normally does through supervised learning

End-to-end solution

- **Much more data**
- **Unsupervised**



Automatic Ground Truth: How to Reduce #Labels

Easier problem to solve

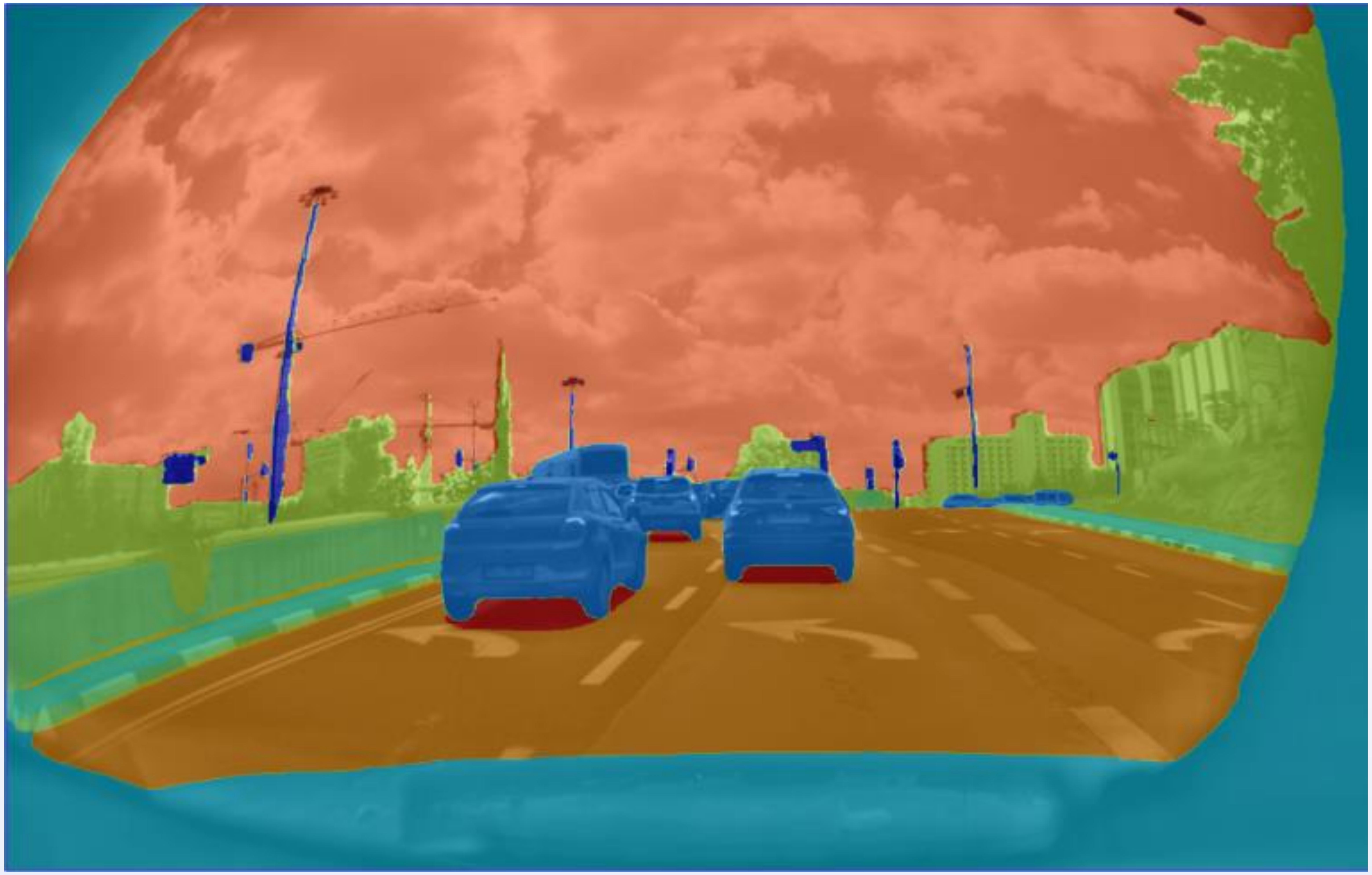
- Since the future is known
    - Kinematics become easier
    - Circumvent temporary occlusions
    - Can focus on short range + tracking
- Powerful (expensive) sensor (e.g., 360° Lidar)

The future is known

Offline compute

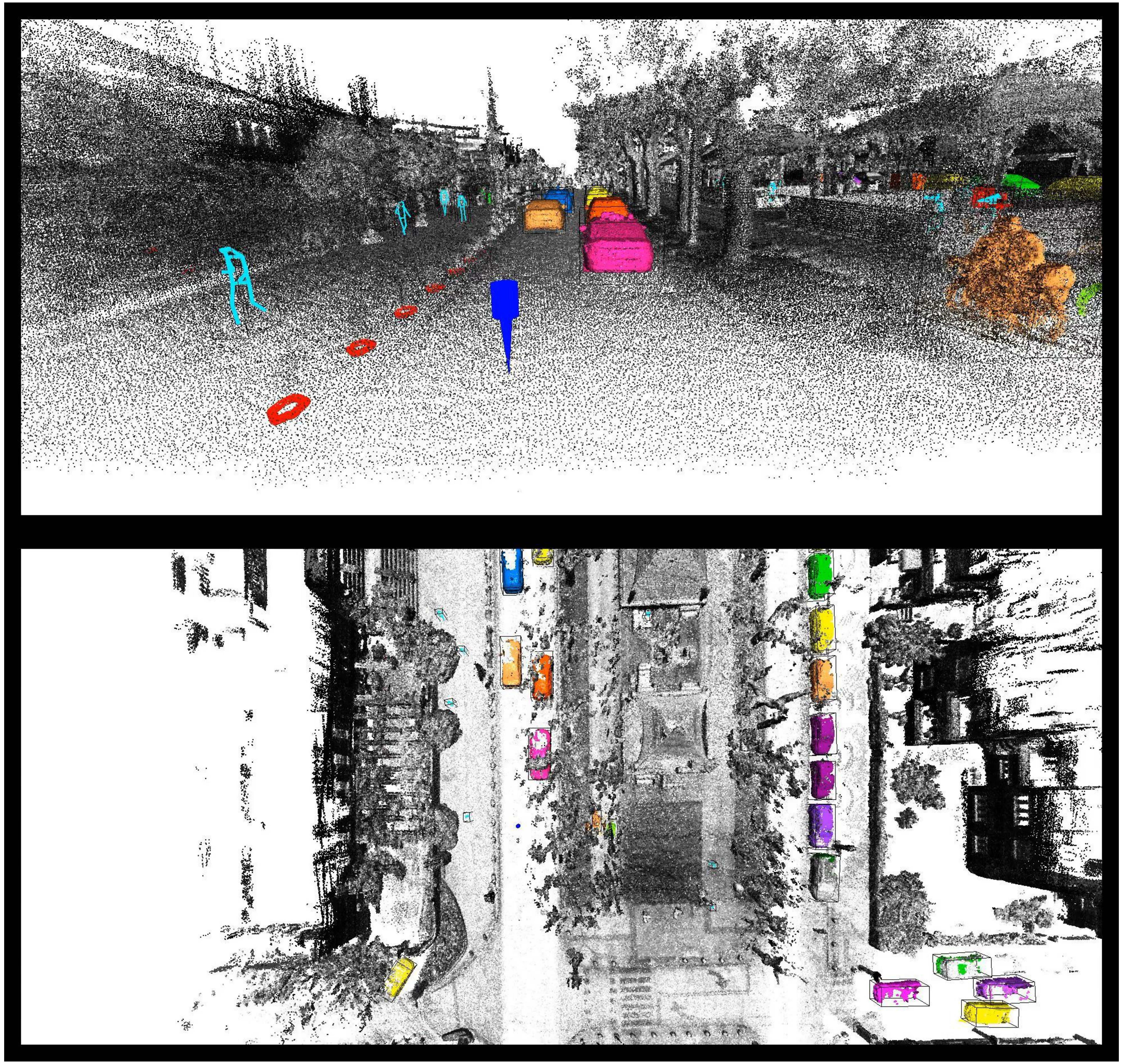
- Train foundation model on large unsupervised data
- Supervised fine tuning on a smaller number of labels

Automatic Ground Truth: Foundation Model



Automatic Ground Truth

Final Product



Extremely Efficient AI

- Transformers for Sensing and Planning at x100 efficiency
- Inference chip (EyeQ6H): Design for efficiency
- Efficient labeling by Auto Ground Truth
- **Efficient modularity by teacher-student architecture**

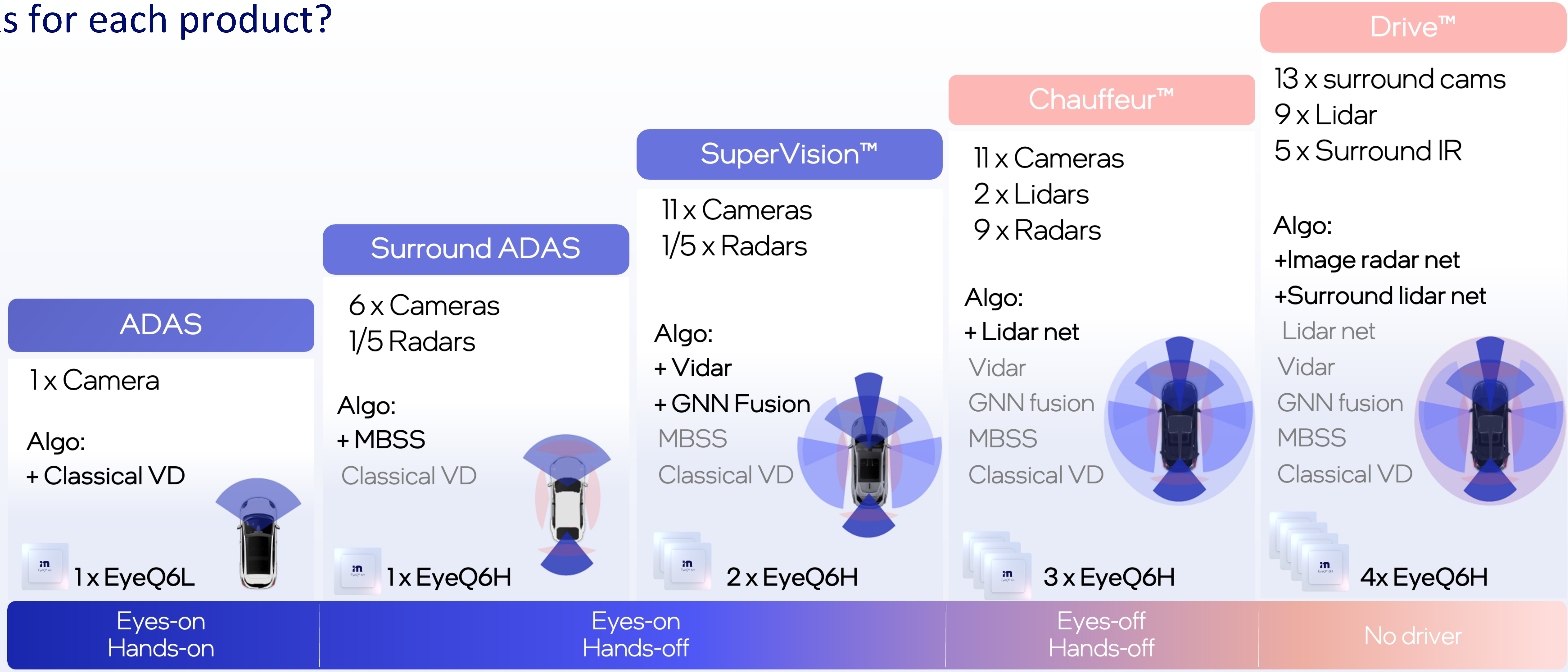
Designing for a Modular Product Portfolio

While Leveraging Data Across All Products

Mobileye's technology path: Modularity

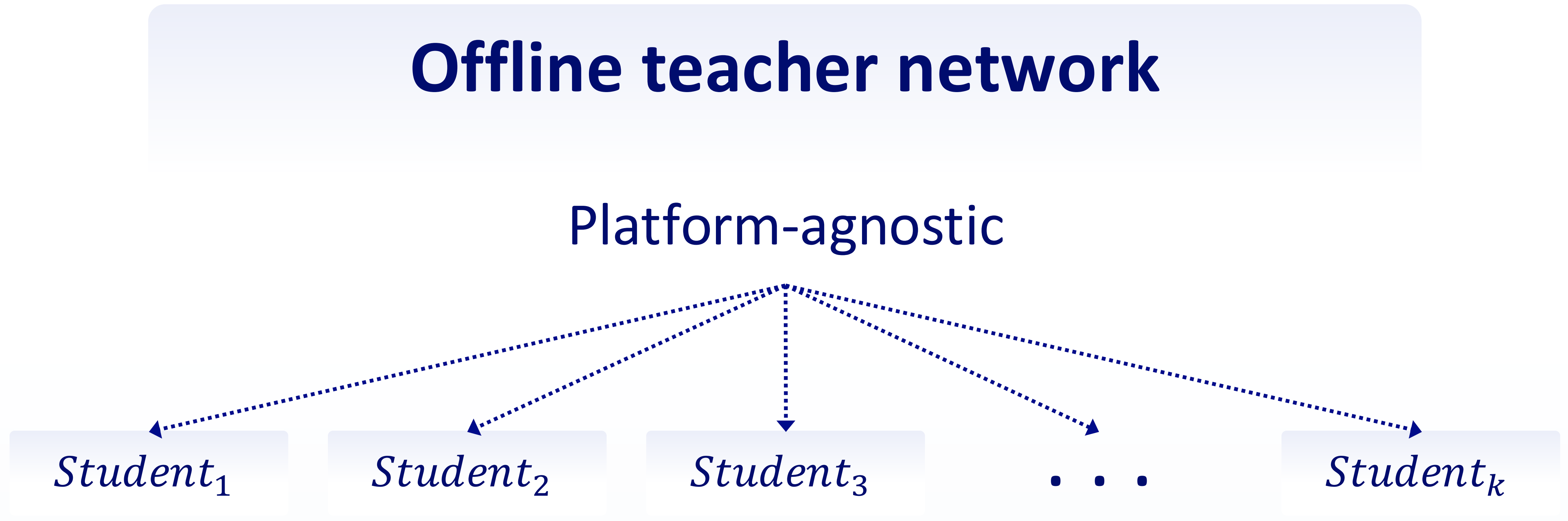
Challenge

- How to create a unified development framework that eliminates the need for separate stacks for each product?



Designing for a Modular Product Portfolio

While Leveraging Data Across All Products



Teacher → Student

- **EyeQNAS** (Neural Architecture Search): Determine architecture optimally per each chip
- **Distillation**: A training framework for imitating the teacher network by a student network

Summary

CAIS

AV Alignment

RSS: Separates correct from incorrect

Reaching sufficient MTBF

Abstractions

- Sense / Plan / Act
- Analytic calculations: RSS, time-to-contact…

Redundancies

Sensors | Algo | High level fusion

Extremely efficient AI

- Transformers for Sensing and Planning at x100 efficiency
- Inference chip (EyeQ™6H): design for efficiency
- Efficient labeling by Auto Ground Truth
- Efficient modularity by teacher-student architecture

mobileye™

Driving AI

2024