From Autoencoder to Physical AI

With the Example of the Automated Vehicle

A simplified introduction for non-experts

July 2026

MCG Management Consult GmbH

Contents

Contents 2
Introduction 3
1. The autoencoder: compression as a learning principle 4
The basic idea 4
What autoencoders are good for 5
2. The variational autoencoder: from copy to creation 6
The trick: a region instead of a point 6
The reward: a map without holes 7
3. The road to generative AI: GAN, diffusion, latent diffusion 8
Stage 1: the GAN – forger versus inspector 8
Stage 2: diffusion – generating in small steps 8
Stage 3: latent diffusion – the VAE returns 9
4. A closer look: diffusion in brief, and the difference between decoder and generator 11
Diffusion in brief: practice with an answer key 11
Decoder and generator: same blueprint, different teacher 11
5. Where this is heading: faster, more controllable, smarter (status: July 2026) 13
Front 1: faster – the shortcut across the map 13
Front 2: more controllable – from dice roll to tool 14
Front 3: smarter – thinking time instead of just model size 15
Outlook: from finished video to a world you can walk into 16
6. The building blocks at work: physical AI in the automated vehicle (SAE L3–L5) 17
Where object and situation recognition attach 17
The sentinel: SOTIF and the standards landscape 18
Prediction is generation: region instead of point 18
Generative AI and planning: three roles 19
Generative safety assurance: turning unknown into known 20
Behind planning: behavior on two time scales 22
7. Around the driving function: the transformer rides along, the fleet learns (status: July 2026) 24
The transformer rides along: language models at the wheel 24
Three quiet helpers on board 26
The data factory: teacher, student, and the sentinel as curator 26
AI in the development process: assistance, not evidence 27
Sources 29

Introduction

Generative artificial intelligence has arrived in everyday life. Software writes text, paints pictures on request, and turns out video that is hard to tell apart from camera footage. From the outside it looks like magic: you type a sentence, and a machine creates something that did not exist a moment earlier. This document opens the black box. The real surprise is that behind the magic there is no impenetrable mountain of formulas, only a handful of startlingly simple ideas that build on one another, and every one of them can be explained with everyday pictures: bullet points and summaries, maps, an art forger and the inspector who hunts him, a Polaroid developing out of the speckle. No prior knowledge required; every technical term is introduced exactly where it is first needed.

So why does a story about creation begin with a word as clunky as “autoencoder”, a machine whose entire job is to copy its own input? Because it is the common ancestor of the whole family. The apparently pointless copying already contains the core of everything that follows: compress, separate the essential from the disposable, restore. Every later chapter develops that single idea: from copying to creating, from image to video, from video to a world you can walk into, from that world to the automated vehicle. Understand Chapter 1 and you hold the key to the rest; everything after it is extension work.

For orientation: this document brings together a seven-part series of explanations of generative artificial intelligence, written to be understood without prior training. It runs from the autoencoder (compressing data and restoring it) through the variational autoencoder (VAE; loosely: an autoencoder that learns regions instead of fixed points), generative adversarial networks (GANs; loosely: a generative network trained in competition with an inspecting network) and diffusion models (generating by removing noise step by step), to current developments and expectations around thinking time (inference-time scaling) and world models. Chapter 6 then transfers the building blocks to physical AI in the automated vehicle, and Chapter 7 adds the roles AI plays around the driving function: from the language model on board, through the learning fleet, to the development process.

The chapters build on one another and use the same visual language throughout: the code map (violet), the networks doing the work (green), random points (yellow-orange), learning signals (amber boxes), the inspector (pink), checked, safe states (light green), and warnings and fallback levels (salmon). Read from the front and you meet the same building blocks again and again, right up to the double punchline: in practically every modern image generator an autoencoder is still literally at work, and at the end the building blocks leave the screen and steer a real vehicle safely through traffic.

1. The autoencoder: compression as a learning principle

The basic idea

Autoencoders are among the most elegant ideas in machine learning, mostly because the principle survives being explained with an everyday situation. Picture someone reading a long article who is allowed to write down only three bullet points. A second person then has to reconstruct the article from those bullets alone. Compare the result against the original and you can see immediately how good the bullets were: the closer the reconstruction, the better the essentials were captured. That is exactly what an autoencoder does, except with data instead of articles, and both "people" are parts of the same neural network.

An autoencoder has three parts. The **encoder** compresses the input (say, an image with a million pixel values) step by step down to a tiny bottleneck, the so-called **code** – perhaps only 32 numbers. The **decoder** then tries to restore the original image from those few numbers. At first the task sounds pointless ("just output what came in"), but the bottleneck in the middle is what forces the actual learning: the network *has* to decide which information matters, because nothing else fits through.

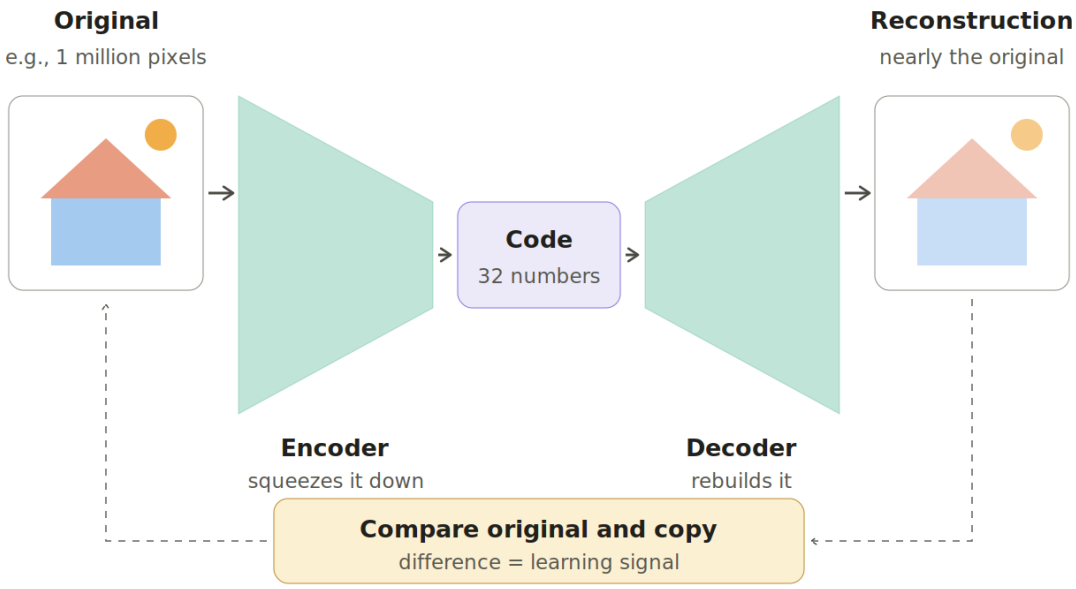


Figure 1: How an autoencoder is built: encoder, code (the bottleneck), and decoder – comparing the original with the copy provides the learning signal.

The learning works without any human labeling of the data, which is one of its great advantages. After each attempt the network simply compares its reconstruction against the original, measures the difference (the *reconstruction error*), and adjusts encoder and decoder together so the error shrinks. After a few thousand repetitions it has learned to squeeze its training data through the bottleneck remarkably well. The code in the middle is then a kind of learned essence of the data: for faces it holds things like head pose, lighting or face shape, without anyone ever having taught the network those words.

What autoencoders are good for

Three typical applications: **data compression** (the compressed representation needs far less memory), **denoising** (train the network to reconstruct clean images from noisy ones and it learns what "clean" means), and perhaps the most important one, **anomaly detection**. The logic is disarmingly simple. An autoencoder trained only on normal examples can only reconstruct normal things well. Show it something it has never seen – an unusual credit card transaction, an atypical machine noise, a strange camera image – and the copy fails, sending the reconstruction error through the roof. That high error is the alarm.

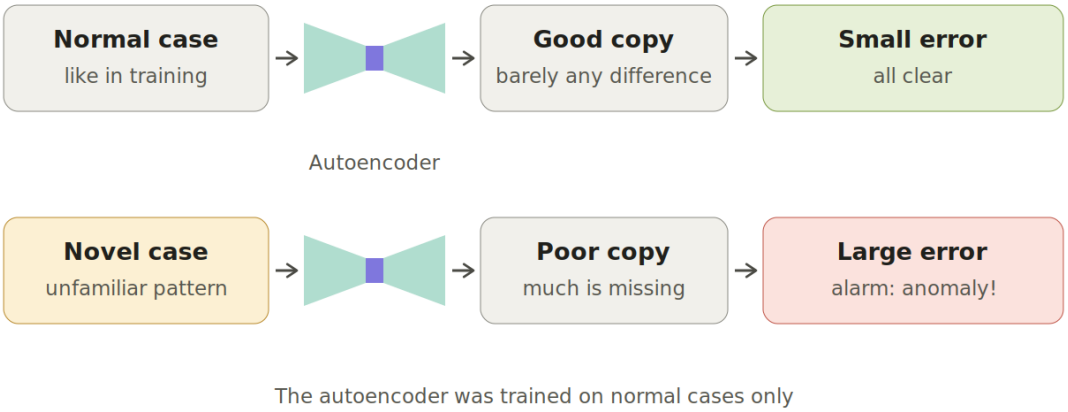


Figure 2: Anomaly detection with an autoencoder: normal input is reconstructed well (small error), novel input poorly (large error = alarm).

Two more things are worth knowing. First, the compression is lossy: the reconstruction is never perfect, it is the network's "recollection" of what mattered. Second, there is a famous refinement, the *variational autoencoder* (VAE). It arranges the code so that you can also "invent" new points there and generate entirely new, plausible images from them, which is how the autoencoder became one of the ancestors of today's generative AI. The next chapter belongs to it.

Key takeaway: An autoencoder is a neural network that learns to squeeze data through a bottleneck and restore it, and in passing learns what is essential in that data, what is disposable, and what is unusual.

2. The variational autoencoder: from copy to creation

The trick: a region instead of a point

The variational autoencoder (VAE) is the logical continuation of the story, and the starting point is a weakness of the ordinary autoencoder. Picture the code space as a map: the encoder assigns every training image an exact point on it. The problem is what lies between those points – no man's land. The decoder has only ever learned to convert back the points actually visited, so pick an arbitrary point in between and you usually get pixel mush. As a tool for *generating* new images, the map is useless.

The variational autoencoder repairs exactly that, with a surprisingly small trick: the encoder no longer delivers **a point, but a small region** – more precisely a center and a spread (for the advanced reader: a mean and a standard deviation). On every training pass a **random point** is drawn from that region, and only that point travels into the decoder.

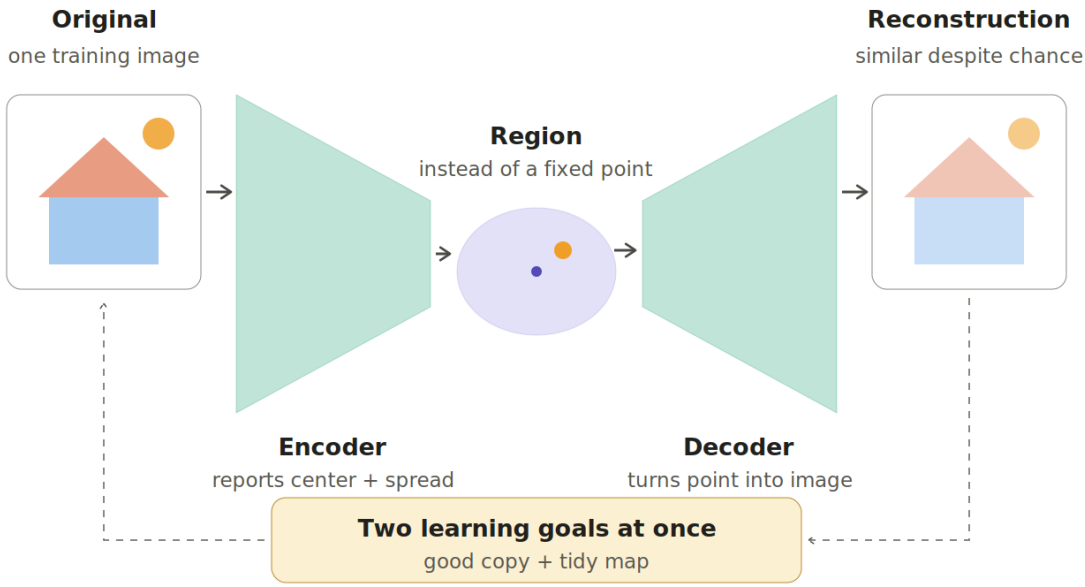


Figure 3: The VAE: the encoder delivers a region (center + spread); a random point drawn from it travels into the decoder.

Why does this built-in blur help? Because the decoder never knows which point from the region will be drawn, it has to deliver a similarly good image for *all* points in the neighborhood. Neighboring codes can no longer stand for completely different things, so the map is forced to become smooth and connected. On top of that comes the second training objective from the diagram, a kind of tidiness rule (technically: the KL divergence). It gently pulls all regions toward a common, standardized area around the center of the map so they overlap and no holes appear. The two objectives are in tension, incidentally: perfect copies *or* a perfectly tidy map, and training looks for the best compromise.

The reward: a map without holes

The reward for that effort shows when you lay the two code maps side by side. With the ordinary autoencoder an arbitrarily chosen point usually leads nowhere; with the VAE *every* point in the ordered area yields a new, plausible image. The reconstruction tool has become a generation tool.

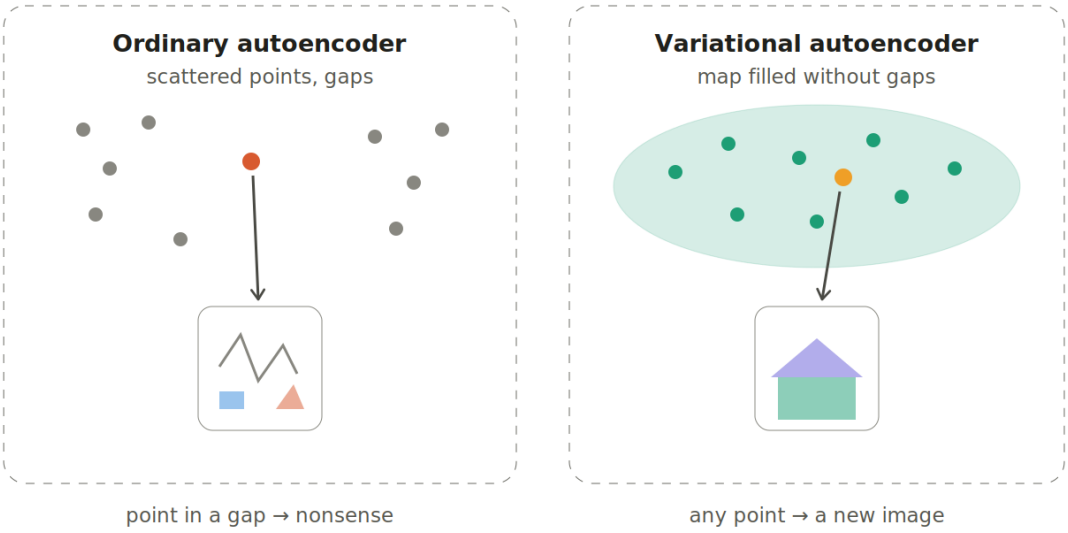


Figure 4: Code maps compared: scattered points with gaps (ordinary autoencoder) versus the gap-free map of the VAE.

The smooth map has another pleasant side effect: you can *walk* between two images. Take the code point of face A and step across the map toward the point of face B, and the decoder produces a fluid transition along the way, with every intermediate image a plausible face. Often specific directions on the map even correspond to properties you can name, such as “more smile” or “head turned further.” In fairness, classic VAEs also have a well-known weakness: their images tend to look slightly blurred, precisely because the decoder has to deliver good results “on average” for whole regions. Newer methods such as diffusion models have overcome that, but conceptually they build on exactly this idea of an ordered, continuous code space.

Key takeaway: A VAE is an autoencoder that maps each image not to a point but to a small random region, and tidies its code map in the process so that every point on it yields a valid, new image. Compression has turned into creation.

3. The road to generative AI: GAN, diffusion, latent diffusion

The road from the VAE to today's image generators is a story in three stages, and at the end the VAE reappears in a surprising way. As a reminder of where we stand: the VAE supplied the decisive idea, an ordered code map on which every point yields a valid image. Its blemish is that the images look blurred.

Stage 1: the GAN – forger versus inspector

The first stage on the road to genuinely sharp images was a completely different approach, the **GAN** (generative adversarial network), best explained as a duel between an art forger and an art inspector. The *generator* (a relative of our decoder) receives only a random point, as in the VAE, and is supposed to forge an image from it. A second network, the *inspector*, is shown real training images and forgeries in turn and has to decide: real or fake? Its verdict is the learning signal for both – the inspector grows more suspicious, the forger more refined.

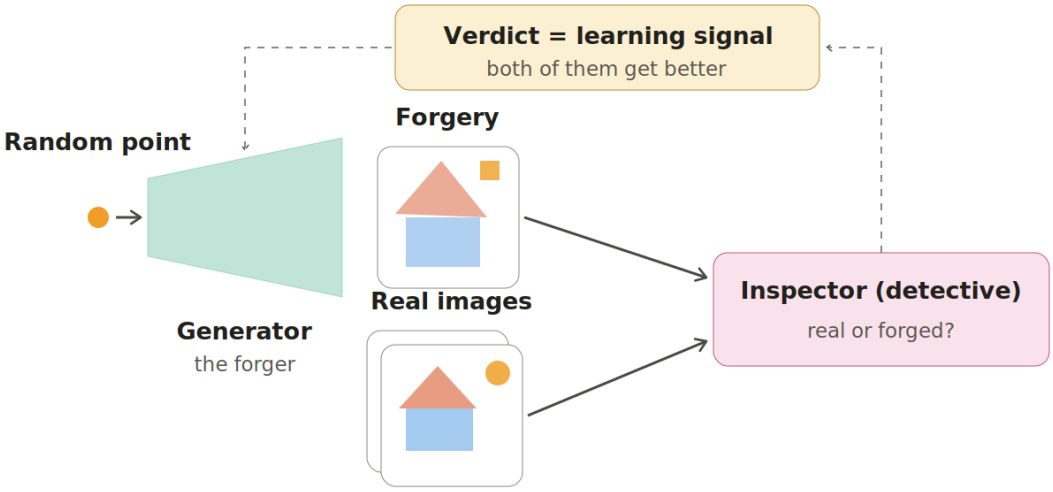


Figure 5: The GAN principle: the generator forges an image from a random point, the inspector compares it against real images and its verdict is the learning signal for both.

In the ideal case this arms race ends with the inspector reduced to guessing, because the forgeries can no longer be told from the originals. From around 2018 onward, GANs delivered the first pin-sharp AI faces of people who never existed. But the duel is temperamental: training easily loses its balance, the forger has a habit of settling on a few favorite motifs, and steering the whole thing deliberately is difficult.

Stage 2: diffusion – generating in small steps

The **diffusion models** of the second stage therefore follow an entirely different philosophy: instead of forging an image in one grand gesture, work it out of noise in many small, reliable moves. The training is astonishingly simple. Take a real image and sprinkle noise over it step by step until nothing is left but TV static. What the network learns is exactly one modest skill: to undo *one* small noise step.

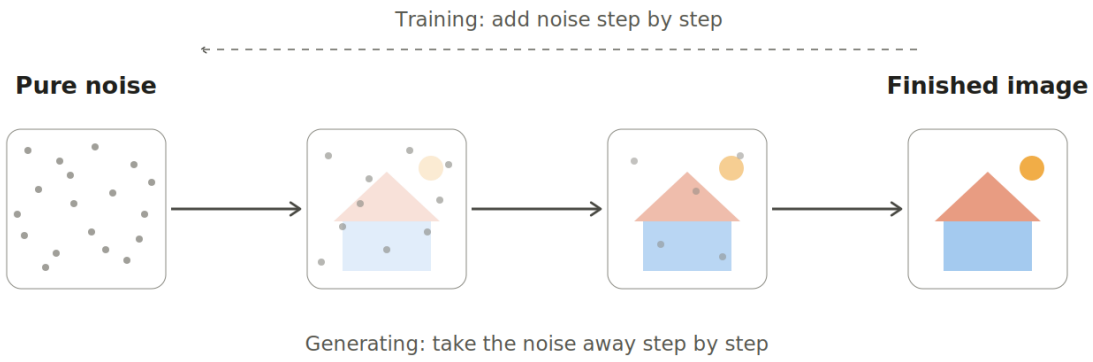


Figure 6: The diffusion principle: during training, noise is added step by step; during generation it is removed step by step.

To generate, you simply turn the process around: start with pure random noise and apply the learned denoising skill a few dozen times in a row, like a Polaroid slowly developing out of the speckle. Because the training objective is so plain, training is robust and the image quality spectacular. The catch is that it is slow and expensive, since each of the many steps computes over millions of pixels.

Stage 3: latent diffusion – the VAE returns

And here, in the third stage, the VAE returns. The idea behind **latent diffusion** (this is how Stable Diffusion works, for example): do not denoise the huge pixel image, denoise a tiny point on the compressed VAE code map, where every step is a thousand times cheaper. Only at the very end does the VAE decoder translate the finished code into the large, sharp image. The steering comes from the prompt: a text encoder turns the words into a signal that nudges each individual denoising step toward the matching region of the map.

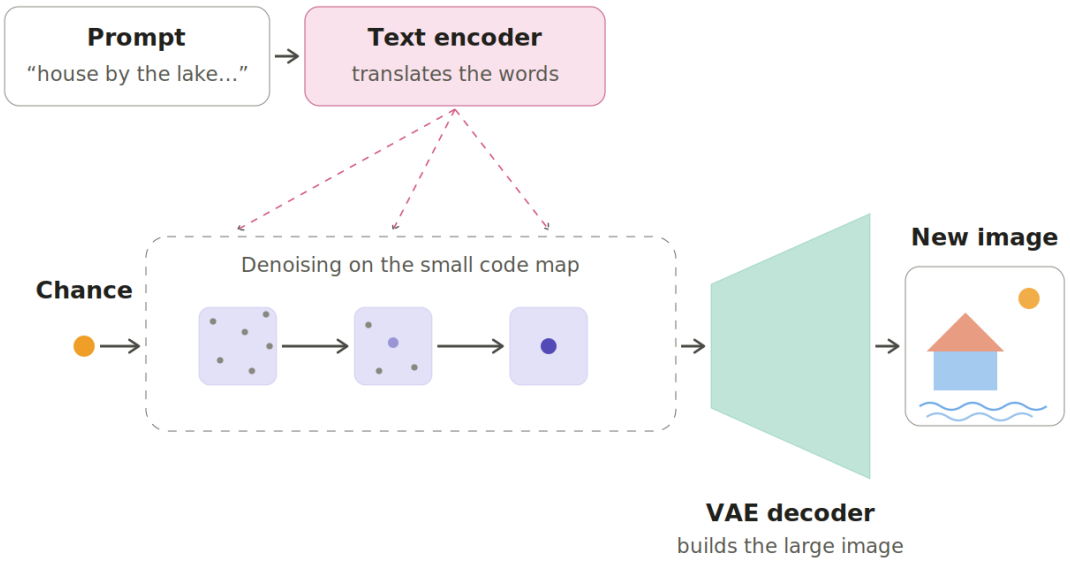


Figure 7: Latent diffusion: denoising on the small VAE code map, guided by the text encoder – the VAE decoder then builds the large image from it.

That closes the circle elegantly: in practically every modern image generator a VAE sits literally at the entrance and the exit. Diffusion has not taken its place, it works *on its map*. For completeness: text AI such as ChatGPT takes a different route, where a transformer (the “T” in ChatGPT) predicts the most plausible next piece word by word. The underlying idea is the same, though. A network internalizes the structure of its training data so thoroughly that it can create something new from it that it has never seen.

Key takeaway: The VAE supplied the ordered map, the GAN proved in the forger’s duel that pin-sharp creations are possible, diffusion turned generating into a reliable craft of small steps – and today’s generators such as Stable Diffusion combine all of it: guided by the words of the prompt, they denoise a point on the VAE code map.

4. A closer look: diffusion in brief, and the difference between decoder and generator

Diffusion in brief: practice with an answer key

Diffusion and the comparison between decoder and generator belong closely together, so let us take diffusion in brief first; the comparison then almost falls out of it. The core of diffusion is a single modest skill: *remove a little bit of noise*. And the training for it is so unbeatably robust because you can manufacture the exercises yourself. Take a real image and sprinkle noise over it with your own hand. That way every exercise comes with an answer key attached: the network makes its denoising attempt, and the comparison with the original shows immediately how good it was. No duel as in the GAN, no compromise between two objectives as in the VAE, just millions of simple exercises with the solution enclosed.

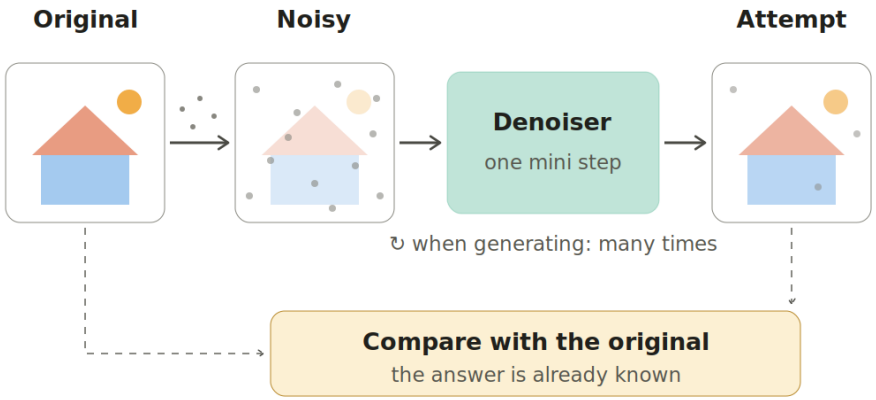


Figure 8: Training the denoiser: for every noisy exercise, the answer key (the original) is automatically available.

For generation, this one practiced skill is simply wired in series: start with pure random noise and apply the denoiser a few dozen times in a row. Each step only has to make the image a little bit better, and many reliable baby steps add up to a considerable feat.

Decoder and generator: same blueprint, different teacher

At first glance, **decoder** and **generator** are twins. Both are the "painting half" of their system: they receive a point from a code space and turn it into an image. The difference is not in the blueprint but in the *teacher*. The decoder paints from a model: its encoder partner produced the point from a specific original, so you can hold the copy up against it and mark every deviation. That makes it faithful, but also risk-averse – where several sharp images would fit equally well, it prefers to paint their soft average. The generator, by contrast, has no original at all: for its raw random point there is no "right" answer to place beside it. Its only teacher is the inspector's verdict, *does this look real?* That forces it to commit boldly to details, since blurring would be spotted at once, but it makes training temperamental.

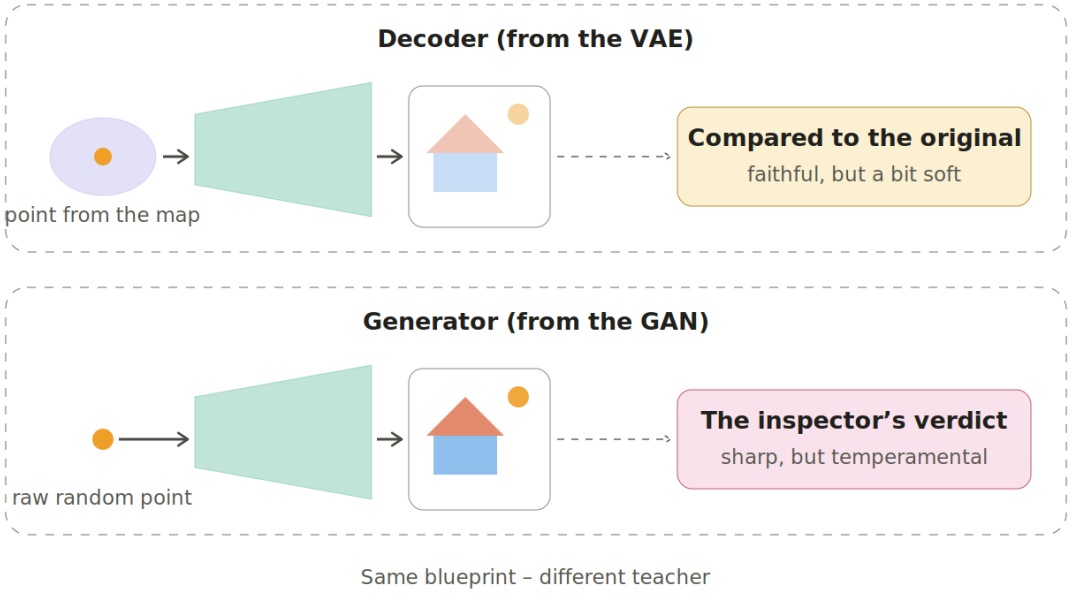


Figure 9: Decoder and generator compared: same blueprint, different teacher – comparison against a model versus the inspector's verdict.

And now you can also see why diffusion performs so well: its denoiser combines the best of both worlds. It learns like a decoder, always with the answer key enclosed, so it is stable and free of moods. But it creates like a generator, starting from pure randomness with no model to copy. The trick that makes this straddle possible is the decomposition into many small steps: for a mini-step there is always a unique correct answer, for a whole image out of nothing there is not.

Key takeaway: Decoder and generator are identically built painters with different teachers – the decoder learns *fidelity* by comparison with the model, the generator learns *persuasiveness* through a critic's verdict, and the diffusion denoiser sidesteps the either-or by breaking creation into many small steps, each of which has a known answer key.

5. Where this is heading: faster, more controllable, smarter (status: July 2026)

Work on improving AI output is currently running on three fronts at once – *faster*, *more controllable*, *smarter* – and on the horizon sits a leap that goes beyond better pictures: worlds you can walk into.

Front 1: faster – the shortcut across the map

Classic diffusion needs dozens of baby steps, and that is precisely its bottleneck. Until now, image generation has mostly been a batch process: send the request, wait seconds to minutes, check the result. The next frontier is generation in real time, where you can adjust a scene while it is still forming. Two ideas make that possible. The first is *distillation*, another teacher-student arrangement: a fully trained model runs its 50 baby steps, and a student network learns to *jump* straight to where the teacher ends up. The second is *flow matching*: you teach the network the straightest possible paths from noise to image in the first place, instead of the zigzag that grew historically. Together the two squeeze 50 steps down to somewhere between 1 and 4 – and suddenly images appear while you are still typing.

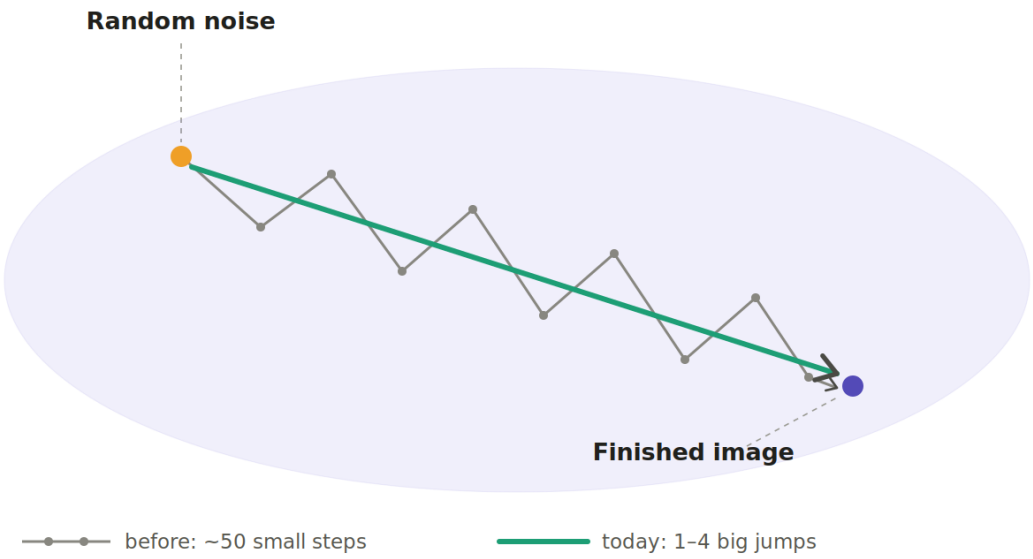


Figure 10: The shortcut across the code map: many small steps (classic diffusion) versus a few large jumps (distilled models).

Front 2: more controllable – from dice roll to tool

The second big improvement is less about image quality than about *predictability*. Users increasingly want control over their source material: on video platforms the “image to video” variant, in which your own starting frame is animated, already accounts for roughly a third of all jobs, because it delivers more predictable results than text prompts alone. Add to that editing by instruction (“make the sky cloudy, leave the house alone”) instead of rolling the dice again, and above all *consistency*. Temporal consistency – characters and environments staying stable across every video frame – was long treated as the holy grail of video generation and has by now matured to the point where AI content is often indistinguishable from filmed footage. Recognizable characters across several scenes and real-time image computation (rendering) are among the defining trends, and structurally the individual specialist tools are growing together into integrated suites that combine image, video and editing in one product.

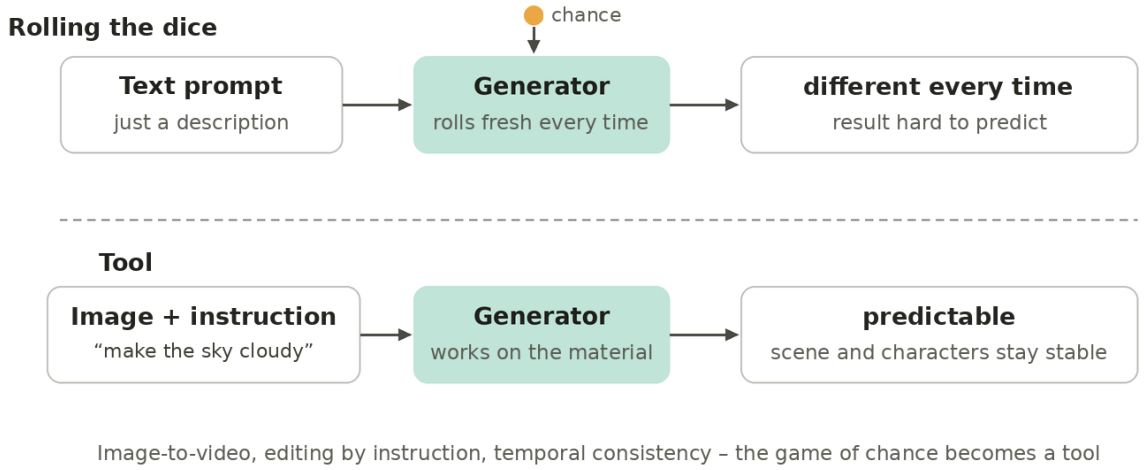


Figure 11: From dice roll to tool: a starting frame, instructions and enforced consistency make generation predictable.

Front 3: smarter – thinking time instead of just model size

This is perhaps the most important new lever, and it applies to all AI output, text above all. For years the recipe read “bigger model, more training data.” Because that kind of training is running into compute and data limits, a new paradigm has established itself: more compute at the moment of answering. The model thinks longer, tries several candidate solutions, checks and improves itself before it commits to a final answer. This comes in two variants: sequential, where the model reflects on its own draft and revises it, or parallel, where it produces several candidates at once and an inspecting model picks the best. The pattern is familiar. The *inspector* from the GAN returns, only this time it is not an opponent in a duel but the model’s own internal quality control.

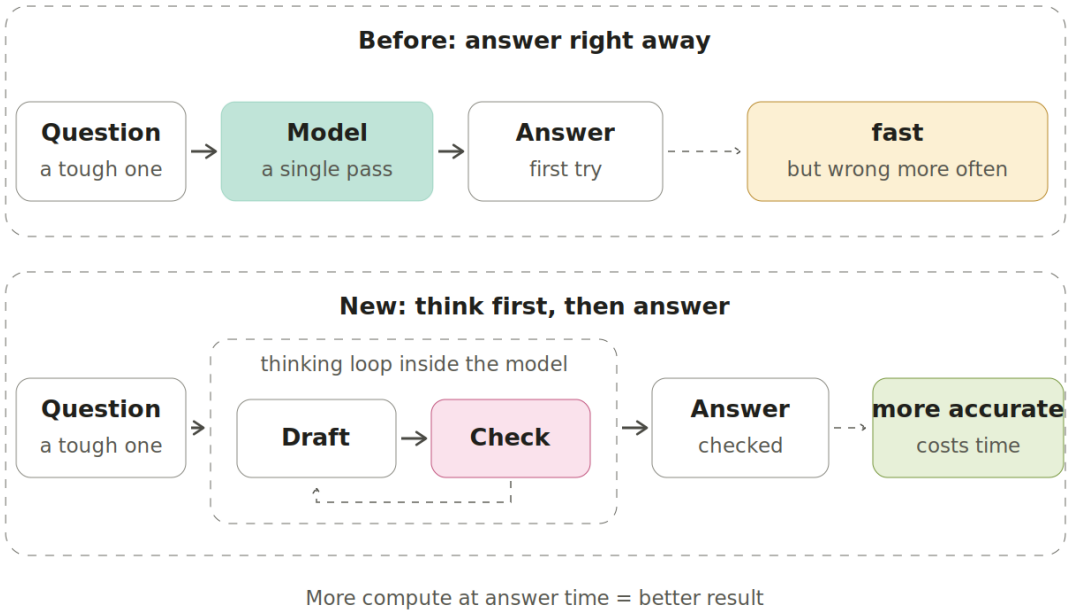


Figure 12: Answer immediately or think first: the thinking loop of draft and inspection delivers more accurate answers, at the price of more compute time.

This thinking, incidentally, is trained with an old acquaintance: the answer key. You let the model practice on tasks whose result can be checked objectively – mathematics, program code – and reward the lines of reasoning that lead to the right answer. How far the weight is shifting shows in the infrastructure: the compute demand for *answering* is forecast to exceed the demand for training by more than a hundredfold, because thinking models consume orders of magnitude more compute steps per answer.

Outlook: from finished video to a world you can walk into

The most interesting expected leap is the one from *output* to *simulation*. World models no longer produce a finished video; they produce only the next frame, autoregressively, in response to the user's ongoing input. The best-known example is Genie 3 from Google DeepMind: a text prompt turns into an explorable world that reacts to user input in real time. Since late January 2026 this has been publicly accessible as "Project Genie" (initially for subscribers in the U.S., worldwide since May 2026), at 720p and 24 frames per second. The minutes-long consistency of the world is a capability of the underlying Genie 3 model; the public prototype currently caps each generated world at 60 seconds. You describe your surroundings, choose whether you walk, ride, fly or drive, and then stroll in. This is more than a toy: generated worlds like these increasingly serve as training environments for AI agents, and in early 2026 Waymo built a specialized world model on the Genie 3 basis for simulating autonomous driving.

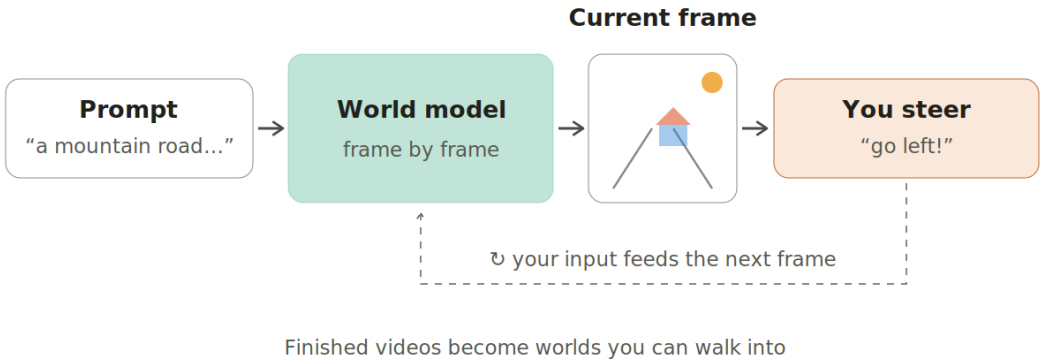


Figure 13: The world model principle: every user input feeds into the next generated frame – videos become worlds you can walk into.

What else is expected follows almost inevitably from the three fronts: generation in real time rather than in waiting mode, so results can be redirected while they form; models that understand and generate text, image, sound and video together; and, as the social flip side of that quality, proof of origin. Because AI content is increasingly indistinguishable from the real thing, regulators in the U.S. and the EU have now established clearer requirements for AI-generated content, and watermarks and labeling obligations are likely to become as unremarkable as the copyright line at the foot of a website today.

Key takeaway: AI output is getting better along three roads at once – *faster* through a few large jumps instead of many baby steps, *more controllable* through steering and consistency instead of luck with the dice, and *smarter* through built-in thinking time with an inspector of its own – and the next big step turns the picture you look at into a world that reacts to you.

6. The building blocks at work: physical AI in the automated vehicle (SAE L3–L5)

With this chapter, AI leaves the screen. Generated images become perceived scenes, outputs become actions – the step the industry is currently christening "physical AI." What has been explained so far and the automated vehicle (SAE Level 3 to 5: from conditional automation, where a human takes over on request, through to driverless operation; the levels defined by the U.S. engineering body SAE are the industry standard) mesh more closely than it first appears. The building blocks from the earlier chapters reappear there in four places: as the *workhorse* (the encoder), as the *sentinel* (the autoencoder trick), as the *fan of futures* (the VAE's region idea) and as the *toolbox for safety assurance* (the generative side). And the question of where object and situation recognition attach has a precise answer: not to raw pixels, but **to the code**.

Where object and situation recognition attach

The perception chain is, at its core, the first half of our autoencoder. The backbone network of modern perception *is* an encoder: it compresses the millions of pixels from several cameras into a compact feature representation, today typically fused into a shared bird's-eye view, which is literally a *code map of the scene*. **Object recognition** attaches exactly there, in the form of small additional networks that read out the code ("what is where" – vehicle, pedestrian, traffic light state, free space; semantic segmentation, or "painting by numbers," where every pixel is assigned a meaning class, even uses literal encoder-decoder hourglasses). **Situation recognition** attaches one level higher, to the *temporal sequence* of these scene codes – tracking over time, interaction, intent estimation – and feeds the planner. And alongside all of this, deliberately *outside* the functional chain, sits the safety mechanism: the sentinel.

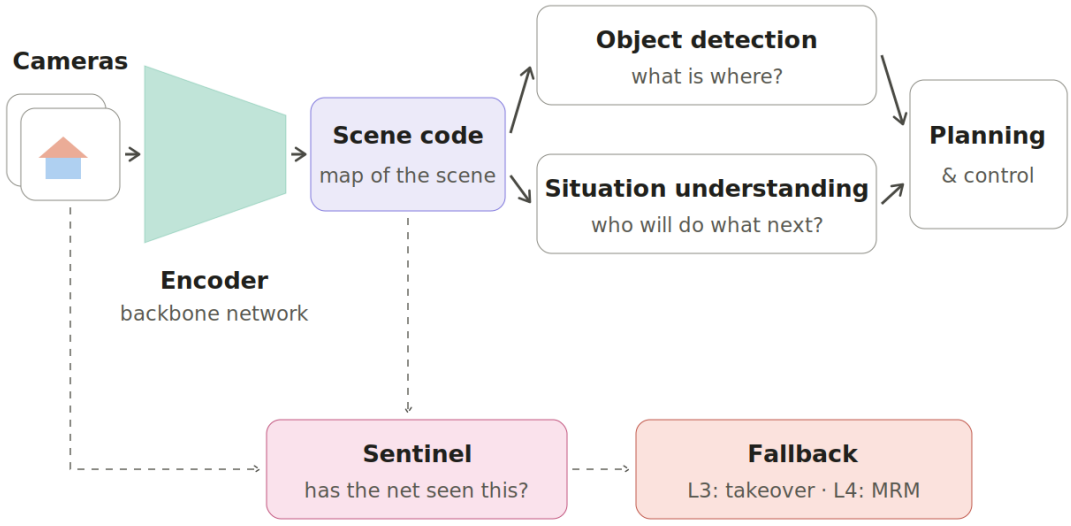


Figure 13: Camera perception in an automated vehicle: object detection works on the scene code, situation understanding on its temporal sequence – the sentinel monitors in parallel.

The sentinel: SOTIF and the standards landscape

The sentinel is the Chapter 1 trick in safety-critical purity: an autoencoder (or its VAE variant) trained exclusively on what is normal within the operational design domain (ODD), which raises a flag when the reconstruction error runs high. That is how it recognizes, at runtime, inputs the network does not know from training ("out-of-distribution"). Normatively this is precisely the seam. Functional safety (*ISO 26262 – Functional safety*) addresses malfunctions of the E/E implementation (electrical and electronic: platform, computer, classic software), while the *functional insufficiencies of the trained function* (missed objects, "false negatives," and misclassifications) together with the triggering conditions that provoke them sit in SOTIF (*ISO 21448 – Safety of the intended functionality*), and safety with artificial intelligence (*ISO/PAS 8800 – Safety and artificial intelligence*) supplies the AI-specific building blocks: data completeness across the operational design domain (ODD), robustness, uncertainty estimation, runtime monitoring, drift. The sentinel turns *unknown-unsafe* into *known-unsafe*, which is what makes it treatable in the first place. The dependence on the level is the decisive difference. At L3 the alarm leads to a transition demand to the driver, and if the driver does not take over, the L3 system too has to come to a minimal-risk stop by itself; at L4/L5 the system has to reach the minimal risk condition on its own from the outset, by way of a minimal risk maneuver (MRM). The "I don't know this" signal thereby becomes a safety-relevant capability in its own right, with its own requirements on miss and false-alarm rates (false-negative and false-positive rates), flanked by classic mechanisms around the AI black box, above all diverse redundancy: plausibility checks using dissimilar sensors, radar and lidar (laser distance measurement).

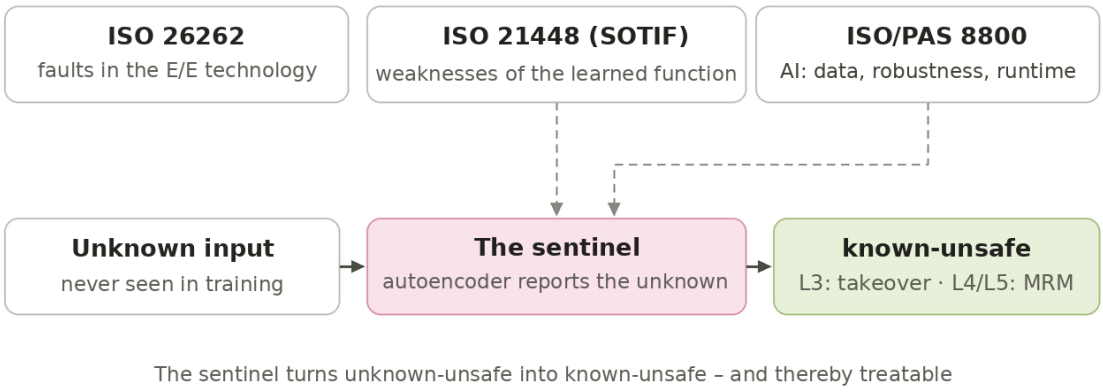


Figure 15: The standards landscape and the sentinel: ISO 26262, SOTIF and ISO/PAS 8800 divide the responsibilities – the autoencoder reports the unknown at runtime.

Prediction is generation: region instead of point

Situation recognition leads straight to the VAE punchline of the series, because it has to predict *futures*, and that is a generative task. The pedestrian at the curb has several plausible futures: cross or wait. A deterministic single-point forecast averages these modes – exactly the "soft decoder" failure from Chapters 2 and 4 – and lands unphysically in the middle, where neither future lies. This is why CVAE-based and by now diffusion-based trajectory predictors are state of the art. A CVAE (conditional VAE) is a VAE with a cheat sheet: the scene context enters encoder and decoder as a condition, and the random region carries only the one open question – which future? Every point in the region stands for exactly one future; evaluate several and you get the fan of weighted futures that the planner can

use for conservative calculation. The diffusion-based relatives transfer the same principle to the craft from Chapter 3: guided by the scene context, they denoise a random path step by step into a plausible trajectory, with each pass delivering one future and several passes filling the fan again.

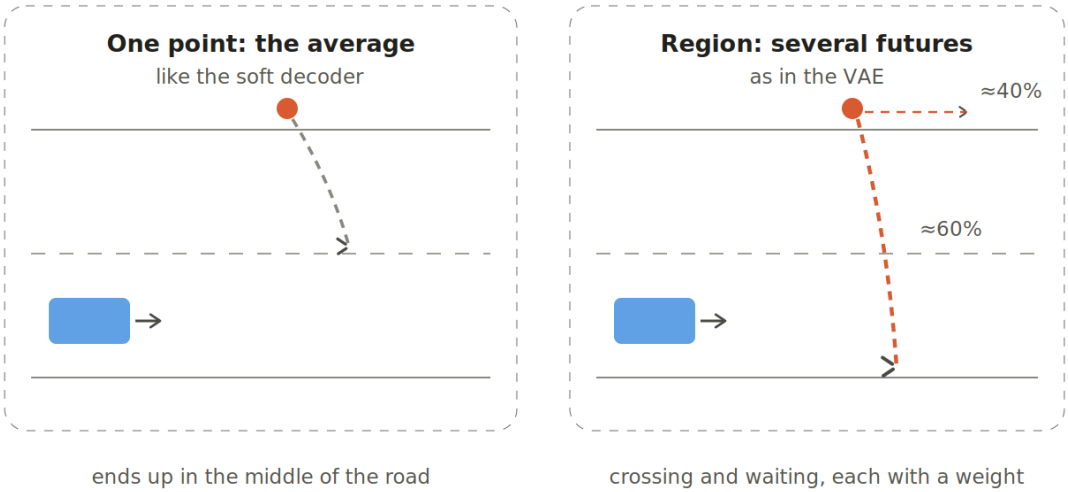


Figure 16: Trajectory prediction: the single-point forecast averages the futures unphysically – the VAE's region idea delivers several weighted futures.

Generative AI and planning: three roles

So where does generative AI stand overall in relation to perception as the supplier to planning? The clearest way to see it is along three roles.

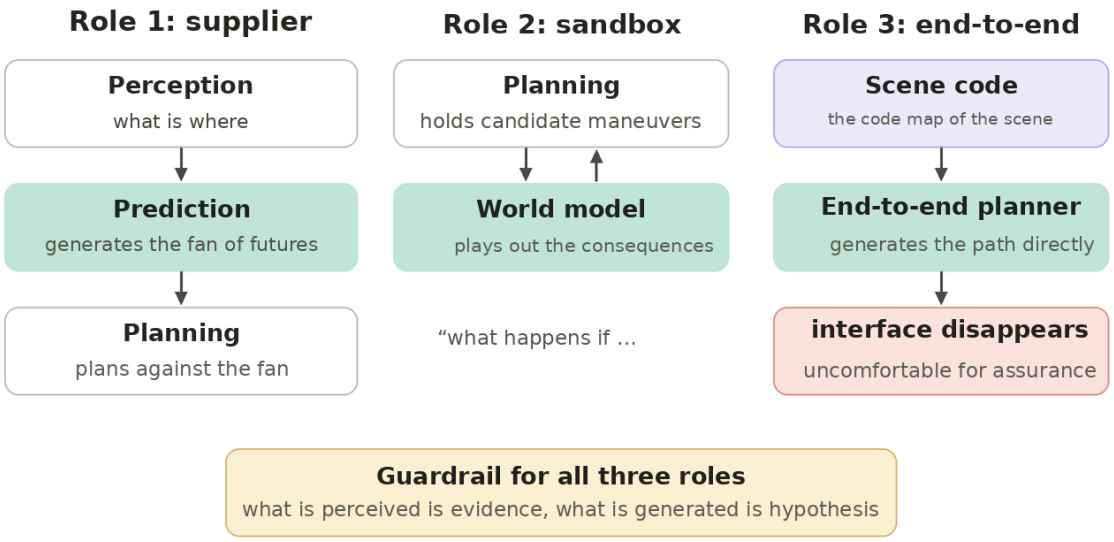


Figure 17: Three roles of generative AI around planning: supplier of the fan, sandbox alongside planning, end-to-end in place of the interface.

First, *in the supply itself*: classically, perception delivers the planner a picture of the state ("what is where"). The generative part of that today is the last link, prediction. It produces the fan of possible futures, so the planner no longer receives only "what is" but additionally a generated distribution over "what might be in a moment," against which it plans instead of against a point estimate.

Second, *alongside planning*: with world models (Chapter 5), generative AI moves from supplier to sandbox – the planner can play through candidate maneuvers internally ("what happens if I brake now instead of swerving?") and pick the best one; perception supplies the initial state, the generative model supplies the consequences.

Third, *in place of the interface*: in end-to-end architectures the clean handover via object list blurs – the planner works directly on the scene code, and in some designs diffusion planners even generate the ego trajectory (the path of one's own vehicle) themselves. This is powerful, but for safety assurance it is the most uncomfortable variant, because the inspectable interface between perceiving and deciding disappears.

The same guardrail applies to all three roles: **what is perceived is evidence, what is generated is hypothesis.** Generative AI may supply futures, consequences and scenarios – but it must never overwrite the sensed actual state.

Generative safety assurance: turning unknown into known

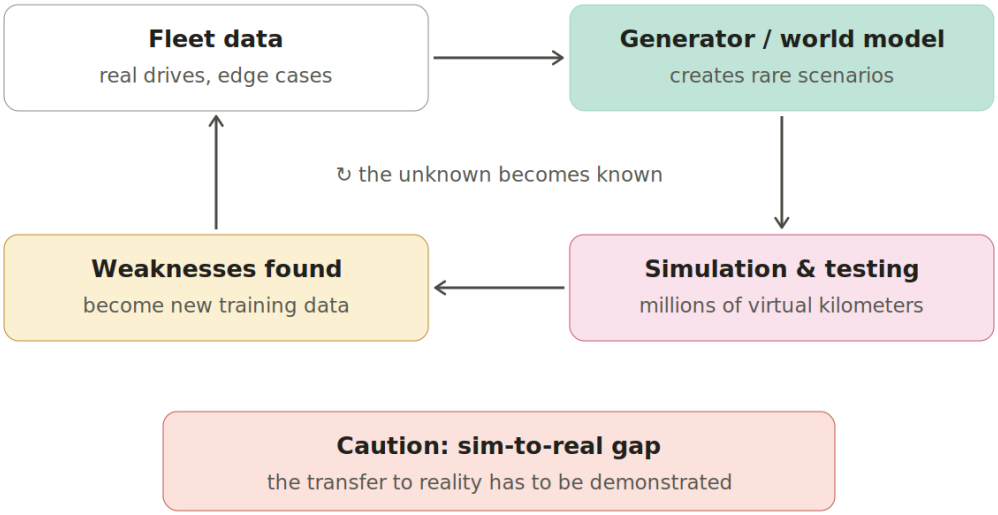


Figure 18: Generative AI in the safety assurance loop – with a sim-to-real gap that has to be demonstrated.

Chapters 3 and 5 finally supply the toolbox for systematically shrinking the SOTIF cloud of unknowns. Diffusion and world models generate and vary precisely those rare scenarios that fleet data is poor in: the child behind the ball, glare, heavy rain, exotic objects and lost cargo. They resimulate recorded drives with what-if variants and deliver millions of virtual kilometers for scenario-based testing (methodologically, for instance, within the framework of ISO 34502 – scenario-based safety evaluation), which is exactly the track on which Waymo uses its Genie 3-based world model for simulating autonomous driving. For the safety argument, though, the flip side belongs in the safety case (the documented safety argumentation): the sim-to-real gap – the distance between simulation

and reality – has to be demonstrated, and generated data inherits the biases of its generator. Test a learned system predominantly against a *related* learned distribution and you invite correlated blind spots. ISO/PAS 8800 addresses this through its requirements on data quality and representativeness; the argument for why synthetic evidence counts has to be made explicitly.

Key takeaway – perception and safety assurance: In L3–L5 camera perception the *encoder* is the workhorse – object recognition attaches to its scene code, situation recognition to the temporal sequence of those codes. The *autoencoder* is the sentinel that reports the unknown and thereby converts unknown-unsafe into known-unsafe, the *VAE's region idea* is the key to honest, multimodal prediction, and *generative AI* is the toolbox with which unknown-unsafe becomes, step by step, known-and-assured.

Behind planning: behavior on two time scales

At the output of the planner, the situation reverses compared with perception. A camera image can hardly be checked independently for correctness at runtime, which is why the learning sentinel was needed there. A planned *trajectory*, by contrast, is a concrete object you can recompute: positions, speeds, accelerations for the next few seconds. Here, for the first time, checking against explicit rules is possible – and that is exactly what happens, on two time scales: *in the loop* (prevent before anything happens) and *across the drive* (detect, record, learn).

In the loop: propose and inspect

In the loop means the pattern **propose and inspect.** The complex, increasingly learned planner proposes a trajectory – but a deliberately *simple, rule-based* inspector decides whether it gets driven.

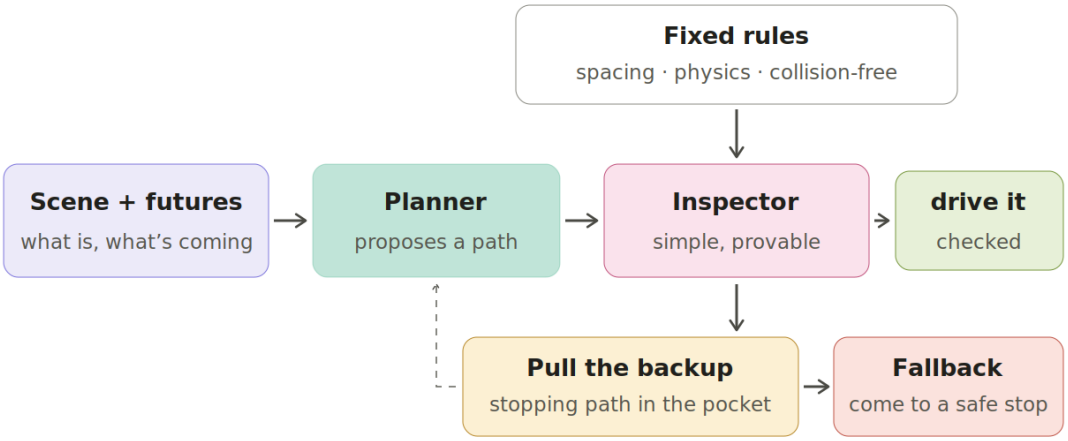


Figure 19: Propose and inspect: the complex planner proposes a path, the simple inspector decides – with a stopping path in the back pocket and a fallback level behind it.

The check runs against the entire prediction fan (collision-free against *all* plausible futures, not just the most likely one), against vehicle physics (friction coefficient, lateral acceleration), and against formalized distance rules. Frameworks such as RSS (*responsibility-sensitive safety*, loosely: the duties of care in road traffic, cast into formulas) turn "too close" and "braked too late" into exact formulas with minimum distances and a prescribed reaction. Typical shortcomings thereby become checkable conditions *before* they end up on the road. If the check fails, the cascade takes over: the planner has to improve its proposal, and for the serious case a pre-checked **stopping trajectory sits in the back pocket** of every cycle – the prepared minimal risk maneuver. If even that is no longer sufficient, the last level is an in-lane emergency stop. At L3 the transition demand is issued as soon as the system limit is reached, giving the driver a time budget; if the driver does not take over, or if the time is not there, the L3 system executes the minimal risk maneuver itself. The architectural punchline for the safety argument: the inspector is deliberately *not* learned but classic, provable software – the narrow, highly assured monitor above the complex channel, entirely in the spirit of ISO 26262 decomposition. So the inspector from Chapters 3 and 5 makes its third appearance here, this time as trajectory sentinel.

Across the drive: measure, record, learn

The second time scale is *the drive itself*. Here a continuous **driving-quality measurement** runs along: time gaps and distances to others, *time to collision* (the calculated time remaining until impact), the frequency of hard braking and steering interventions, lane-keeping quality.

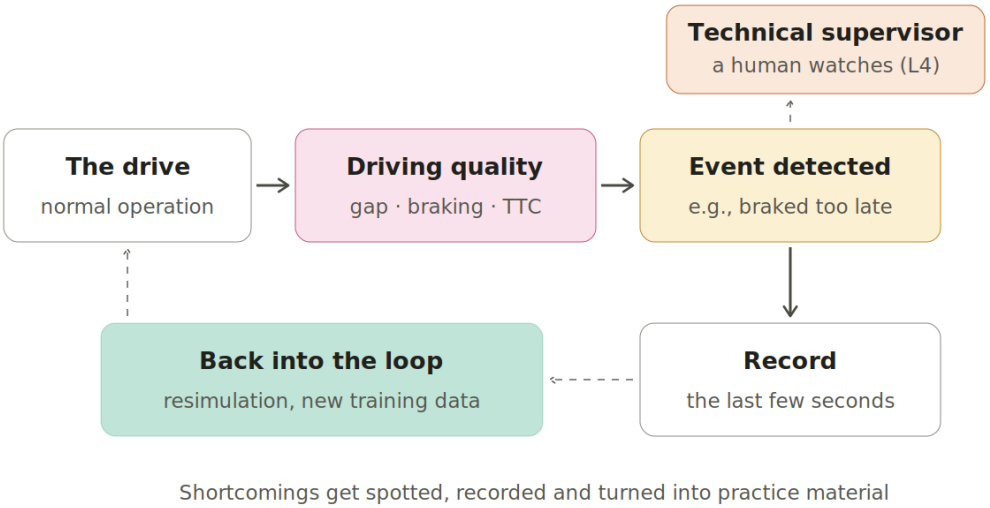


Figure 20: During the drive: driving-quality metrics trigger events that are recorded, reported and turned into practice material by the safety assurance loop.

When something crosses a threshold (braked too late, followed too closely), it becomes an *event*: the last few seconds of all sensor and system data are secured, on the manufacturer side for fleet analysis, while on the type-approval side at least the status log is mandatory (the DSSAD event recorder, *data storage system for automated driving*: who was driving when, system or human). Depending on severity, the system also responds in graded fashion: take off speed, restrict the operating domain, and in case of doubt the minimal risk maneuver. At L4 in Germany a human runtime authority is added that exists nowhere else in this legally anchored form: the **Technische Aufsicht** (technical supervisor) under § 1f StVG (the German Road Traffic Act), who can be brought into the loop, approve maneuvers and deactivate the system. And with that the circle back to the previous section closes: every recorded event is raw material for the safety assurance loop from Figure 18 – resimulation ("would it have gone well without the intervention?"), root cause analysis, new training and test data. Field monitoring, which SOTIF and the AFGBV (the German regulation on the approval and operation of autonomous vehicles) demand in any case, thereby turns from a compulsory exercise into the engine of improvement.

Key takeaway – behavior: The vehicle's behavior is handled on two time scales. In the loop, a simple, provable inspector checks every proposed trajectory against the fan, vehicle physics and formal distance rules, with a stopping trajectory in the back pocket. Across the drive, shortcomings are measured, recorded and turned into practice material by the safety assurance loop: prevent in the moment, learn across the fleet.

7. Around the driving function: the transformer rides along, the fleet learns (status: July 2026)

Chapter 6 walked the length of the driving function – perceive, predict, plan, behave – and its safety assurance with it. That does not exhaust the story of AI in the automated vehicle. Three arenas are missing: the vehicle itself, into which a new family member is currently moving; the data factory behind it, where the fleet learns; and the development process, where AI lends the engineers a hand. The good news for anyone who has read from the front: almost everywhere, the familiar building blocks are the ones doing the work here too, in new roles. Only one appearance is genuinely new, and it belongs to the relative that Chapter 3 set aside in a single sentence: the transformer.

The transformer rides along: language models at the wheel

In Chapter 3 the transformer – the "T" in ChatGPT – was left standing at the roadside: text AI takes a different route. That route has since turned into the vehicle. **Vision-language-action models** (VLA; loosely: a model that sees, thinks in language, and derives driving actions from it) combine the perception of Chapter 6 with the world knowledge and the thinking time of Chapter 5. The gain shows up exactly where systems learned purely from driving scenes have traditionally been weak: at the long tail of rare cases. A network that has only ever seen traffic does not know the sofa on the roadway; a model additionally trained on the knowledge of the world can make sense of it, and of the police officer's hand signal or the traffic light that has failed at the intersection.

The second gain is at least as interesting for safety assurance: these models think in intermediate steps you can read back. In early 2026 NVIDIA presented Alpamayo 1, an open 10-billion-parameter model that produces not only a trajectory from camera images but the **reasoning trace** along with it – a chain of intermediate steps explaining why this particular path was chosen. Since the first quarter of 2026 the approach has been in series production in the Mercedes-Benz CLA; for robotaxis, Alpamayo 2 Super provides a 32-billion-parameter model; and in March 2026 Li Auto showed MindVLA, its own driving foundation model. The thinking loop from Chapter 5 is now literally sitting in the car.

For the safety argument this cuts both ways. The reasoning trace is a new kind of evidence artifact – for the first time you can watch a learned planner think and read back its decision after an event. At the same time the vehicle inherits the well-known weaknesses of language models: a fluent justification can be wrong (*hallucination*), thinking time costs reaction time, and whether the justification is really the cause of the action or merely a well-told retelling of it is an open research question. The architectural answer from Chapter 6 therefore stands unchanged: even the cleverest VLA model is the complex channel, and above it watches the simple, provable inspector.

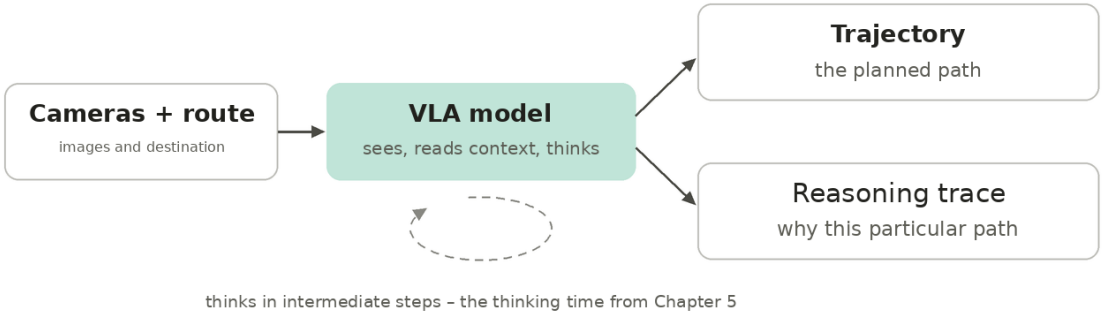


Figure 21: The VLA model in the vehicle: images and context produce a trajectory and a reasoning trace – the thinking time from Chapter 5 rides along.

How tightly the language model is coupled in is itself an architectural decision, and in early 2026 the market shows two answers. Against the driving VLA model, Mobileye sets an advisory variant with **VLSA** (vision-language-semantic-action; loosely: the model supplies meaning rather than a path). A slow-thinking language model gives only structured semantic hints, like a driving instructor riding along – "careful, the officer is directing traffic" – while the trajectory and the safety-critical control remain in the fast, formally assured driving stack (the layered software package of perceiving, planning and controlling). The generative model never enters the safety loop in the first place. The price is a new interface that has to be specified with care: the advice has to be correct, timely and correctly understood, and the fast path has to stay safe without it.

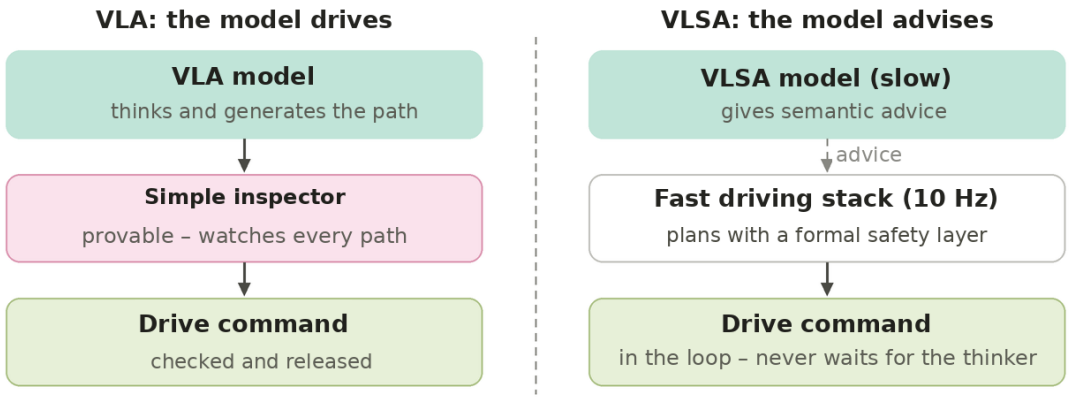


Figure 22: Two ways of coupling the transformer: the VLA model generates the path itself, with the inspector watching; the VLSA model only advises, and the path is produced in the fast, formally assured stack.

Three quiet helpers on board

Besides the prominent newcomer, three less conspicuous AI roles are at work in the vehicle, all three built from blocks out of the early chapters. First, the view inward: at L3 the entire fallback logic hangs on the transition demand, and that presupposes continuous checking of whether the driver is in a position to take over at all – the relevant regulation for such systems (UN R157) explicitly requires driver availability recognition. An interior camera with learned gaze and drowsiness detection does exactly that: perception as in Chapter 6, only turned through 180 degrees. Second, the sentinel in the vehicle network: the same Chapter 1 trick that reports unknown camera images also detects unknown traffic on the vehicle's data buses, from the classic CAN bus to the Ethernet backbone, as intrusion detection in the sense of the cybersecurity regulations (UN R155 and *ISO/SAE 21434 – Road vehicles, Cybersecurity engineering*). From the autoencoder's point of view, an attack is simply an anomaly. Third, the denoiser in the literal sense: the craft from Chapters 3 and 4 computes rain, spray and driving snow out of camera images and lidar point clouds before perception gets to see them – and the anomaly detection reports in passing when a sensor is dirty or has drifted out of calibration.

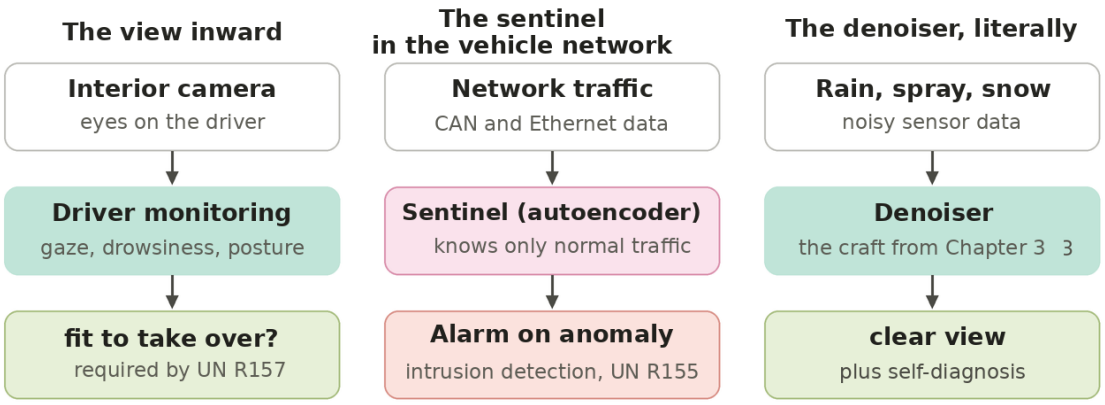


Figure 23: Three quiet helpers on board: driver monitoring, the vehicle-network sentinel and denoising in the literal sense – familiar building blocks in supporting roles.

The data factory: teacher, student, and the sentinel as curator

Chapter 6 ended with the safety assurance loop: events become practice material. The training side of that loop is a factory of its own, and inside it the teacher-student principle from Chapters 4 and 5 rules at industrial scale. At the start there is a selection problem: millions of fleet kilometers are almost entirely uneventful. The sentinel trick solves it – a novelty detector keeps exactly those scenes the system does not yet know; everything else would be labeling effort without learning effect. Then the teacher takes over: a large model outside the vehicle (offboard), which does not have to run in real time, labels the selected scenes automatically (**auto-labeling**). The answer keys that armies of human labelers used to supply are today produced by one model for the next. Finally comes *distillation* from Chapter 5: the large teacher is boiled down into a small, fast student that fits into the control unit. NVIDIA names exactly this chain – fine-tuning, distillation, auto-labeling – as the intended use of its open driving foundation model.

Two additions round out the factory. Basic training of the perception networks today runs as standard without any labeling at all, using **masked autoencoders** (loosely: autoencoders that fill in deliberately covered parts of an image), the Chapter 1 apparatus in a new variant, filling gaps instead of removing noise. And the driving policy itself is increasingly retrained with **reinforcement learning** (reward instead of an answer key), closed-loop in simulation. The world-model sandbox from Chapter 6 thereby turns from a test space into a training space, in which the system plays through millions of situations, is rewarded for good maneuvers, and gets to unlearn bad ones without consequences.

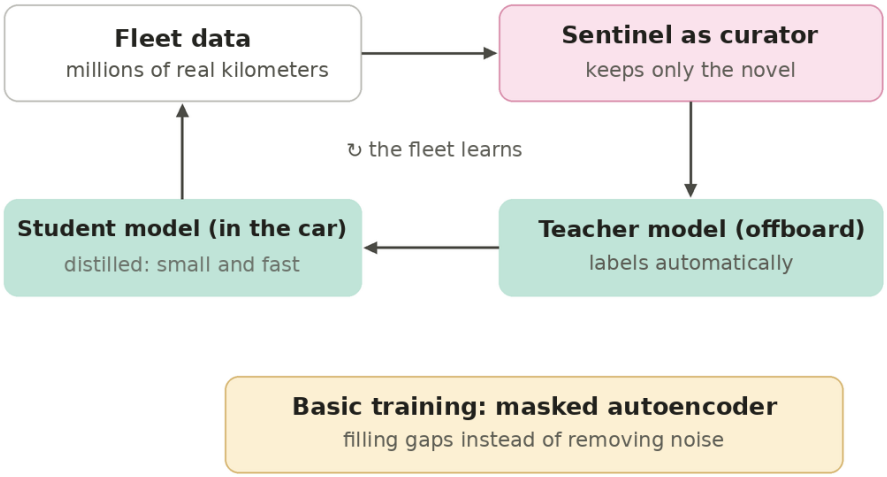
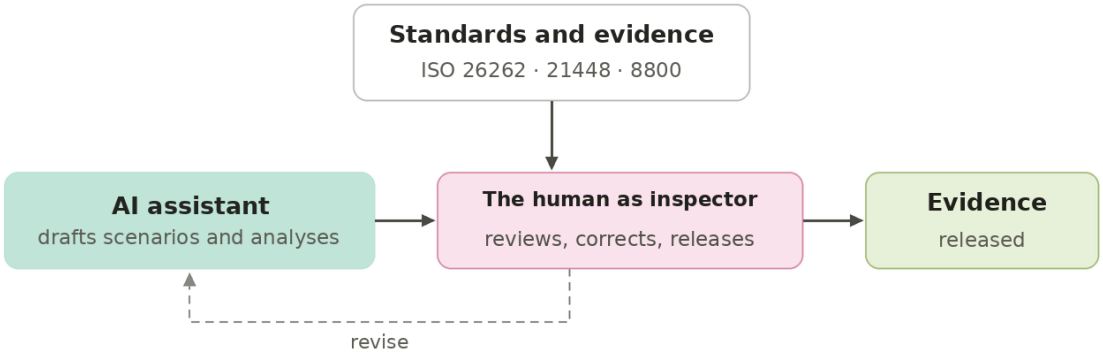


Figure 24: The data factory: the sentinel curates the fleet data, the teacher labels automatically, distillation delivers the student for the vehicle – and basic training is handled by the masked autoencoder.

AI in the development process: assistance, not evidence

The third arena lies before the first meter is driven: in development itself. Language models draft test scenarios from a description in everyday language (“a child steps out from between parked cars with the sun low”) and translate them into formal scenario languages such as OpenSCENARIO; they supply first drafts for hazard analyses, check requirement lists for gaps and contradictions, and summarize test campaigns. That is real relief – and it needs the same guardrail that Chapter 6 drew for the vehicle: *what is generated is hypothesis*. An AI draft becomes evidence only through review and release by a human being; responsibility – and, in the sense of ISO 26262, confidence in the tool chain as well – stays with the engineer. So the inspector from Chapters 3, 5 and 6 makes its fourth appearance, and it is the finest one: this time it is a human.



The inspector from Chapters 3, 5 and 6 – its fourth appearance, this time as a human

Figure 25: Assistance, not evidence: AI drafts, the human reviews, corrects and takes responsibility – the fourth appearance of the inspector.

Key takeaway: Around the driving chain the same family of building blocks carries on working. The transformer moves into the vehicle as a thinking, self-explaining model – driving as a VLA or advising as a VLSA; the sentinel monitors the vehicle network and the sensors on the side, while a perception turned inward keeps an eye on the driver; in the data factory the sentinel curates the fleet data, the teacher labels the answer keys, the student is distilled – and in the development process the old guardrail holds: AI drafts, the human reviews and takes responsibility.

Sources

Chapters 1 to 4, as well as the methodological foundations in Chapters 6 and 7, explain established knowledge of machine learning and of the relevant standards and regulations (ISO 26262, ISO 21448, ISO/PAS 8800, ISO 34502, ISO/SAE 21434, UN R155/R157, StVG/AFGBV).

The statements on current developments in Chapters 5 to 7 (among them Waymo/Genie 3 and NVIDIA Alpamayo) draw on the following sources, accessed in July 2026:

Google / Google DeepMind: “Project Genie: AI world model now available for Ultra users in U.S.”, blog.google, January 29, 2026.

TechBriefly: “Google launches Project Genie globally for adult AI Ultra users”, techbriefly.com, May 20, 2026.

Wikipedia: “Genie (world model)”, en.wikipedia.org (accessed July 2026).

Waymo: “The Waymo World Model: A New Frontier for Autonomous Driving Simulation”, waymo.com/blog, February 6, 2026.

WaveSpeed Blog: “Google DeepMind Genie 3: The World Model That Creates Interactive Environments”, January 30, 2026.

Forbes (A. Husain): “Why World Models Like Google’s Project Genie Work”, January 29, 2026.

Renderforest: “AI video generation trends in 2026: what’s actually changing”, May 2026.

digen.ai: “AI Video Generation Market Trends: 2026 Industry Forecast” and “AI Video Generator Trends 2026”, May 2026.

X. Zhou (Medium): “The State of AI Video Generation in 2026: 5 Shifts That Actually Matter”, February 2026.

tooldirectory.ai: “The state of AI image and video generation (2026)”, June 2026.

Introl Blog: “Inference-Time Scaling”, December 2025, plus survey papers on test-time compute scaling (arXiv, 2024–2026).

NVIDIA Newsroom: “NVIDIA Announces Alpamayo Family of Open-Source AI Models and Tools to Accelerate Safe, Reasoning-Based Autonomous Vehicle Development”, nvidianews.nvidia.com, January 5, 2026.

Electrek: “Nvidia unveils open-source AI for autonomous driving, ships in Mercedes-Benz CLA in Q1 2026”, electrek.co, January 5, 2026.

NVIDIA: “Alpamayo – Open AI for Robotaxis and Autonomous Vehicles” (including Alpamayo 2 Super and AlpaGym), nvidia.com (accessed July 2026).

Li Auto Inc.: “March 2026 Delivery Update” – presentation of the MindVLA driving foundation model at NVIDIA GTC 2026, April 1, 2026.

Mobileye Blog: “Prof. Amnon Shashua at CES 2026: Robotaxi updates, breakthroughs in AI, and robotics” (vision-language-semantic-action, fast-and-slow architecture), mobileye.com, January 2026.

Survey papers on vision-language-action models for automated driving (arXiv/CVPR/AAAI, 2025–2026).