Phil Beisel's articles on FSD and AI

1. Part 7. phil beisel on X: "The Magic of Tesla FSD, part 7 (v14 edition)" / X

The Magic of Tesla FSD, part 7 (v14 edition)

How does Tesla train its vehicle autonomy system, FSD, to drive better than a human? Version 13 of FSD amazes many. It navigates countless driving situations with surprising fluency, though it isn't perfect. Version 14, however, is different. It's not only a bigger model; it's a better-trained model. In this article—part 7 of my FSD series—we'll explore Tesla's end-to-end training pipeline: from the massive driving datasets collected across the fleet to the specialized techniques used in fine-tuning.

And we'll arrive at the forth-coming **version 14**, the realization of autonomy.

If you haven't yet, I encourage you to read

parts 1 through 6

of the series for background. Each article builds on the last, revealing more about FSD (as I learn it myself). This one is no different—it's additive, not final.

Let's dig in.

Let's Learn

Architecturally, FSD is a neural network. Conceptually, it is the driving function—it takes in sensor data (e.g., cameras) and outputs controls (accelerator, braking, steering angle).

A neural network is a type of AI that mimics the human brain, at least in spirit. It consists of neurons grouped into layers. Each neuron is simply a connection point, linked to downstream neurons. Every connection has a numeric weight, and each neuron has a bias; together, these form the parameters of the network.

Setting these parameters happens during training. For FSD, the bulk of training is **imitation learning**: it uses driving data from humans— the raw sensor inputs and the decisions that led to successful driving— and learns to imitate them.

But here's the key question: *if Tesla is just imitating human drivers, even carefully selecting only the best driving data, how does FSD ever become better than a human?*

This is where **reinforcement learning** comes in.

After a model is trained using imitation learning, it can be tested and deployed. If, during training, the model was "cut" (finalized) at the point where the loss was minimized, it can generally be expected to perform well.

In imitation learning, the training process is guided by a loss function. This function measures how close the model's predicted output is to the actual result (the ground truth). During backpropagation, the model adjusts its weights to minimize this loss— effectively forcing the model to become more accurate at the task.

However, the model is only as good as the input data. If key cases are missing or labels are incorrect, the model will perform sub-optimally.

Take handwriting recognition as an example. If a dataset of handwritten samples (for each letter or number) is large enough and correctly labeled, then the model will generalize well to new, unseen handwriting. But if the dataset contains mislabeled samples or fails to capture necessary variations, the model will make poor predictions.

In FSD, it is no different. If the dataset misses certain manifestations of stop signs (e.g., those carried by a crossing guard and held upside down) then the model cannot reliably recognize them. Or, if training samples of pedestrian stops include indecisive or inconsistent behavior, the model will learn that inconsistency and produce erratic outputs ("bad driving" behavior).

In effect, the model performs only as well as the data it is trained on. While FSD benefits from extremely high-quality driving samples and one of the most complete driving datasets ever built, there are still gaps—and the "perfect" representation of driving may not be fully captured in the data.

Call in the Reinforcements Reinforcement learning helps narrow that gap.

Once the initial imitation learning phase is complete, reinforcement learning provides a second stage of refinement, pushing the model toward greater accuracy, consistency, and fluidity.

Tesla uses its simulation harness to create a wide variety of driving scenarios. The trained model—acting as the "agent"—is run through these simulations, producing many possible driving trajectories. Each trajectory is evaluated with a scoring system, often called a reward function, that assigns penalties or rewards depending on the outcome. Over many iterations, the agent learns to favor the behaviors that maximize reward (or minimize penalty). Suppose the simulator sets up a scenario where a pedestrian emerges from behind a parked car, hidden from the vehicle's view, just as the car approaches. Possible outcomes might be scored as follows:

- –50 for hitting the pedestrian

- –25 for braking so abruptly that it causes a rear-end collision
- –5 for swerving into the next lane unsafely
- Higher reward (least penalty) for combining braking and steering to safely avoid both the pedestrian and other hazards

The agent runs through many trajectories. Some hit the pedestrian, some brake too sharply, others swerve dangerously, and a few combine braking and steering in a safe, balanced way. Each trajectory is scored, and the higher-reward patterns are reinforced in the model. Over time, the model internalizes these patterns, improving its ability to handle unexpected real-world events with better judgment.

In imitation learning, model updates are straightforward: the loss function measures error against the ground truth, and backpropagation adjusts weights and biases directly to reduce that error. Reinforcement learning is different because there is no single "correct label" for each situation. Instead, the model learns by trial and reward.

The network outputs probabilities for different actions—brake, swerve, continue, etc.—and one is sampled. After the outcome is observed, the action is scored by the reward function. Actions that lead to higher rewards are nudged to become more probable in the future; actions that lead to penalties are made less probable. Under the hood, this is still gradient descent and backpropagation, but the error signal comes from the reward function rather than a ground truth label.

In other words, supervised learning makes the model competent by teaching it to copy examples, while reinforcement learning makes it skilled by teaching it which actions are worth repeating.

Training Compute

Training FSD requires an enormous amount of compute.

Tesla's "Cortex" training center in Austin, TX— occupying 500,000 square feet at the head of Gigafactory Texas— is essentially a supercomputer. It consists of a training cluster with more than 100,000 GPUs when fully built out.

The first stage of building the FSD model is **imitation learning**. Here, training proceeds in **batches** across multiple **epochs**, with progress bounded by the number of driving datasets available and the size of each batch. As the model grows in parameter count, compute requirements scale roughly in proportion—about 10× more parameters requires about 10× more compute. Training typically continues until improvements in the **loss function** level off, meaning the model is no longer learning much from additional epochs.

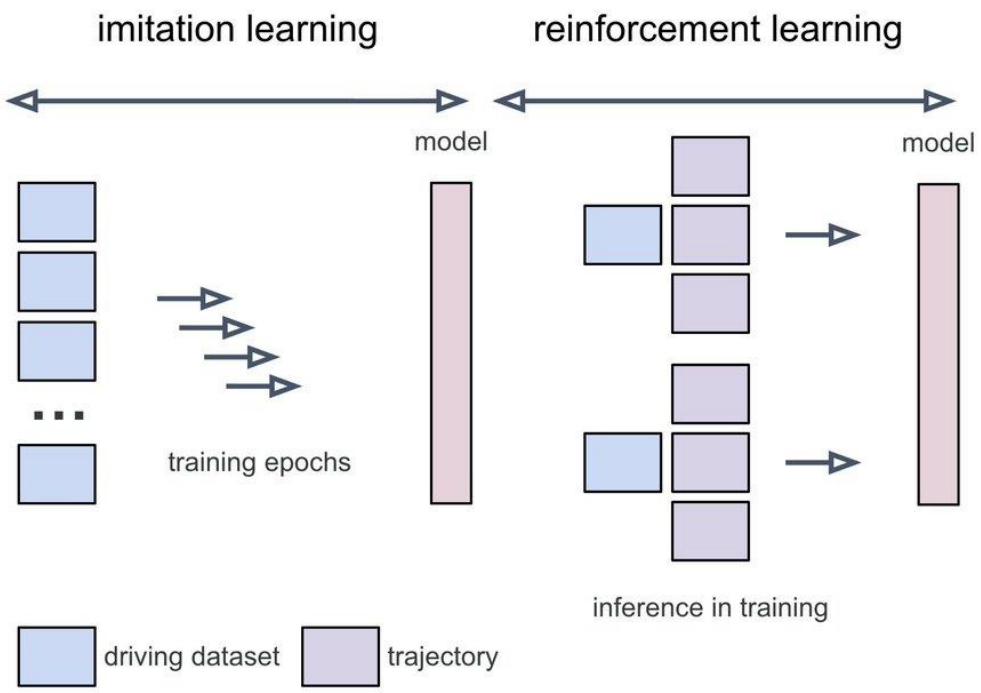


Figure 1: imitation learning + reinforcement learning

But imitation learning is only half the story (see Figure 1). The other half is **reinforcement learning**, which is far more compute-intensive. Reinforcement learning requires repeated **inference runs** (trajectories) in simulation, with the outcomes feeding back into the training loop. In practice, the model drives through simulated scenarios, receives feedback in the form of rewards or penalties, and uses that feedback to improve its driving policy. Thousands of distinct reinforcement scenarios must be explored—each with its own set of trajectory simulations. This is where the real compute burden lies, likely exceeding the imitation learning phase by a wide margin.

Version 14

Version 14 is the forthcoming FSD release, and it represents a leap beyond anything Tesla has shipped before. Architecturally, it remains an end-to-end neural network, but the scale, sophistication, and training philosophy have evolved dramatically.

Bigger, Smarter, More Nuanced At its core, version 14 is a much larger model— 10× the number of parameters of version 13. This increase is not arbitrary; more parameters mean more nuance, more “decision points” in the network, and more capacity to model the subtle complexities of real-world driving. A bigger model also demands more training and inference compute, which in turn drives innovation in how the network is deployed on Tesla vehicles.

Tesla addresses this with clever architecture choices. Version 14 uses a Mixture of Experts (MoE) setup, as described in

Part 5

of this series. Only the relevant "expert" modules are activated for each inference cycle, allowing the model to scale dramatically without overwhelming the vehicle's onboard hardware (HW4). Additional optimizations—ranging from activation sparsity to early-exit routing—ensure that FSD v14 remains efficient enough for real-time driving.

Reinforcement Learning Comes Into Its Own Version 14 is not just bigger, it is better trained. While version 13 introduced reinforcement learning as part of the pipeline, version 14 fully leverages RL as a core mechanism for achieving human-surpassing performance.

Imitation learning gives the model competence: it teaches FSD how to drive safely by copying high-quality human data. Reinforcement learning, however, gives it skill: it allows the network to explore, experiment, and internalize behaviors that even the best human driver might not consistently execute.

In simulation, version 14 repeatedly encounters challenging scenarios—pedestrians darting out from behind obstacles, complex merges in dense traffic, subtle yield situations at intersections. Each trajectory is scored, and over millions of iterations, the network learns to favor safe, efficient, and fluid solutions. This RL phase effectively closes the gap left by the limitations of real-world training data. The result is a model that can handle rare or unusual situations with judgment and consistency beyond human capability.

Fluid, Confident, Human-Like Elon Musk has described version 14 as feeling "sentient," and this is not hyperbole. It's the network's combination of scale, RL-driven judgment, and MoE efficiency that produces driving that feels both safe and fluid—smooth lane changes, subtle adjustments to speed, and anticipatory braking in complex environments. It doesn't just mimic human behavior; it elevates it, blending safety and comfort in a way that feels natural to passengers and other road users alike. **Unsupervised and Robotaxi-Ready** Version 14 is the first fully end-to-end unsupervised FSD. It can also be operated in supervised mode for customer deployments where required. Importantly, it is the version that enables **Robotaxi** to operate without onboard safety drivers or observers.

Better Than a Human

Version 14 doesn't just aim to match human driving; it sets a new standard for autonomy. By combining massive scale, a Mixture of Experts architecture and reinforcement learning refinement, FSD v14 demonstrates judgment, consistency, and fluidity that surpass most human drivers across a wide range of conditions.

It processes every input from cameras and sensors simultaneously, anticipating hazards, optimizing speed and trajectory, and adjusting in real time, all while maintaining smooth, natural motion. In complex scenarios, it can make safer, more calculated decisions than any individual human might.

In short, version 14 marks the transition from remarkable human imitation to driver-level performance that can surpass humans, bringing **true autonomy** within reach. At the same time, it's just the beginning: future versions 15, 16 and beyond will continue to push the envelope, each improving on scale, training, and capability. Eventually, human driving will be out of the loop entirely; it will simply be about moving people and goods efficiently and safely.

2. Part 6. [phil beisel on X: "The Magic of Tesla FSD, part 6 (Robotaxi edition)" / X]

The Magic of Tesla FSD, part 6 (Robotaxi edition)

Just over a month ago, on June 22, 2025, Tesla's Robotaxi rideshare platform officially went live. Since then, a fleet of about a dozen Model Ys has completed hundreds of rides daily within a geofenced area of Austin. On July 15, that geofence was expanded to include more of the city.

Every ride so far has included a safety observer seated in the passenger seat, tasked with monitoring each trip for safety and rider satisfaction. Most notably, the Robotaxi launch introduced the first *unsupervised* version of FSD, reportedly version 13.3, which requires no direct supervision or driver monitoring.

In a related milestone, a Model Y was delivered directly to an Austin customer from the factory and autonomously drove itself approximately 15 miles to the customer's home.

This article is part 6 of an ongoing series[5], and importantly, it's the first one written since Robotaxi went live. Earlier articles explored many of the technical aspects of FSD, but always in the context of a supervised system. While not drastically different from a technical standpoint, this moment marks a clear shift— FSD is now positioned to begin accumulating autonomous miles at an exponential rate.

In this part, I'll discuss some high-level technical factors as well as specific elements related to training and scaling the Robotaxi platform. In the months ahead, Tesla is expected to scale the service: first by removing the safety observer, then by expanding the geofence and increasing the fleet size in Austin, and eventually by entering new cities.

Tesla has cautiously but confidently bet on its vision-based approach as the path to scale and safety— so let's explore why that is.

Background

Vehicle autonomy is one of the most difficult technological challenges ever undertaken. Despite a car having only three basic outputs— acceleration, braking, and steering— the environment it operates in is chaotic and unpredictable. Driving involves an endless array of edge cases, and unlike other domains of robotics, autonomous vehicles must coexist safely with human drivers. Roads were designed for people, not machines— so autonomy must adapt to a world built around human behavior.

Autonomous driving systems are typically defined along two key dimensions: algorithms and sensors.

Algorithms refer to the decision-making process— how the vehicle interprets sensor data and decides what to do. This can be traditional code, AI, or a hybrid of the two. Tesla has pursued a full end-to-end AI approach, while Waymo uses AI mainly for perception and relies on coded logic for planning and control.

Sensors are how vehicles perceive their environment. Tesla relies solely on vision, using eight cameras to provide a 360-degree view. Waymo combines cameras with LiDAR for depth and redundancy.

Tesla's approach is both clean and ambitious— attempting to solve autonomy entirely through vision and AI, with minimal reliance on hardcoded rules.

Vision Only

Tesla has decided on a vision-only approach. There is no LiDAR in the production FSD system.

Photon Counting First we need to understand what vision means to Tesla. It's not what your eyes see— it's what the camera sees, and that's something very different. A traditional digital camera processes raw input through a series of steps to produce a human-viewable image. This process discards massive amounts of data, compresses dynamic range, and biases the result toward human color perception— especially green.

Tesla flips this on its head. It feeds raw 12-bit Bayer mosaic data directly into its system and biases the sensor toward red, where road-relevant signal lies. Tesla calls this "photon counting," and the name fits— it's raw photon measurements, unfiltered.

That raw data goes straight into both training and inference. The images you or I might see? They're never even produced. The neural network sees the world not through a processed image, but through the sensor's raw output—photon-level fidelity.

Here's the key point: Tesla doesn't let the camera process the image. It lets the **FSD neural network** process the image. The network itself becomes the filter, deciding what matters—not

based on what looks good to a human eye, but based on what leads to safe, confident driving. It's outcome-driven perception. That's a radical shift. And it changes everything.

Let's dig into a few details here for those who might be curious what's going on:

A standard digital camera uses a **CMOS sensor** to convert light into electrical signals. Each pixel on the sensor has a color filter— either red, green, or blue— and records a 12-bit value (2^{12} = 4096 levels of intensity).

The filters are arranged in a **red-green-green-blue (RGGB)** pattern, known as a **Bayer mosaic**[1]. There are two greens because the human eye is more sensitive to green wavelengths; doubling the green pixels improves perceived sharpness and luminance detail. While the red and blue filters capture their respective spectral bands, the two greens help span the middle range with greater resolution.

This RGGB pattern repeats across the sensor according to its pixel resolution. Once the raw mosaic data is captured, it's passed to the **Image Signal Processor (ISP)**, which converts it into a human-viewable image (like a JPEG). The ISP performs several steps, including demosaicing[2], white balancing, color correction, noise reduction, sharpening, tone mapping, dynamic range compression, lens correction, and finally compression into a standard image format (e.g., JPEG).

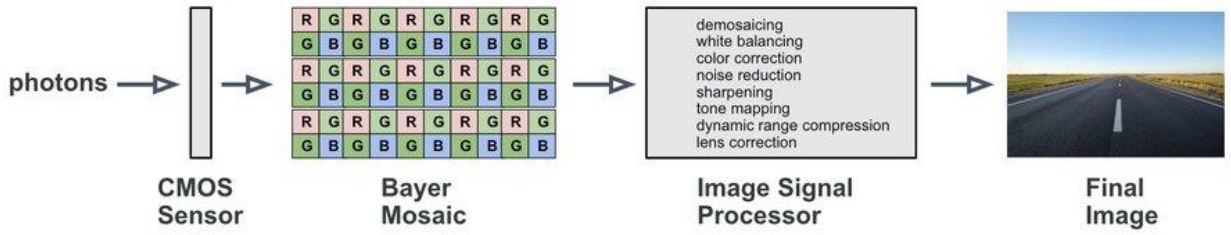


Figure 1: typical image processing pipeline

Tesla, however, makes two fundamental changes.

First, it replaces the traditional red-green-green-blue (RGGB) mosaic with a red-clear-clear-clear (RCCC) configuration (see Figure 2). The red channel is tuned for driving-relevant features like brake lights and stop signals. The "clear" pixels are essentially unfiltered— they record raw photon counts across the full visible spectrum, giving the system maximum sensitivity.

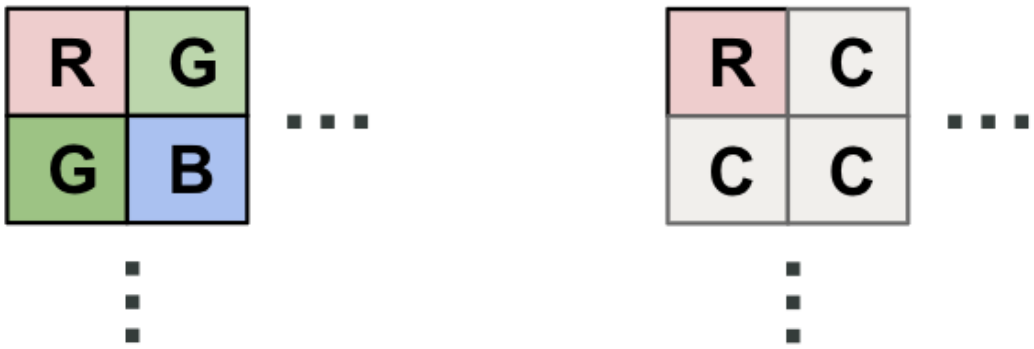


Figure 2: 12-bit Bayer Mosaic: typical camera RGGB vs Tesla RCCC

Second, Tesla eliminates the ISP entirely. The raw 12-bit RCCC data is fed directly into the FSD training and inference stack—no demosaicing, no color correction, no dynamic range compression.

Instead, Tesla lets the neural network decide what matters in the image. And this makes sense: the FSD network is trained by minimizing real-world error, like collisions or awkward maneuvers, while maximizing smooth, safe driving. *In effect, the neural network becomes a learned image signal processor, tuned not for visual aesthetics, but for driving outcomes.*

No LiDAR Tesla does not use LiDAR is its production system. However LiDAR is part of the Tesla vision solution.

Tesla uses LiDAR during development as a ground truth system to train and validate its monocular camera-based vision system[3]. While the production vehicles rely solely on cameras, LiDAR provides precise depth data in training to calibrate the neural network's distance and size estimates.

Consider the following scenario: a set of boxes of varying sizes are placed at different distances from a moving reference object. Suppose one box is positioned 20 meters away, and another at 15 meters. The reference object is equipped with a forward-facing monocular camera and is moving toward the boxes.

As the camera captures a sequence of frames, a neural network processes these images to estimate the boxes' sizes and distances (location). By analyzing pixel changes across multiple frames— leveraging motion parallax and perspective shift— the network infers 3D geometry from what is essentially 2D image data. For instance, it may estimate a box's dimensions by analyzing its bounding box in the image and combining that with inferred depth, though this

process is complicated by perspective distortion and the absence of absolute scale in a single image.

To resolve this, Tesla uses LiDAR during the training phase. The LiDAR system generates a dense point cloud with accurate depth readings for all visible surfaces, providing the precise 3D structure of the scene. This depth data serves as ground truth to train the neural network, teaching it how image sequences correspond to real-world geometry. If the vision system estimates a box's location at, say, 19–21 meters, LiDAR's exact 20.05 meter measurement can be used to adjust the prediction during training—minimizing the error between the vision-based estimate and actual distance.

Once trained, the neural network can accurately estimate object size and location using only camera inputs, without the need for LiDAR in production (LiDAR is essentially factored out).

Though simplified here, this example illustrates how Tesla uses LiDAR-based ground truth during development to supervise and calibrate its camera-only perception system. This approach, as described in

US20200265247A1

, enables reliable 3D understanding from monocular camera sequences through learned depth inference and temporal consistency.

Robotaxi Ready

As noted earlier, Tesla uses LiDAR not in production, but as a ground-truth reference during training and validation. This continues in the real-world Robotaxi rollout, where a specialized fleet of validation vehicles— equipped with solid-state LiDAR and other sensors— is deployed to support geofence expansion (see Figure 3). As Tesla begins operating its Robotaxi service in its first launch city—Austin, TX—many have noticed this fleet of so-called "validation" vehicles circulating throughout the area, especially in zones outside the active geofence. So what exactly are they doing?

Figure 3: Robotaxi validation vehicle

The validation vehicles are indeed validating new road segments in order to expand the geofence.

First, Tesla defines a geometry representing the proposed expansion. Next, it generates a road traversal plan that covers all streets within this area. The test fleet is then loaded with route data to ensure that, in aggregate, the vehicles traverse every road in the new region (or every one they choose). As they drive, the vehicles collect both camera and LiDAR data (see Figure 4), which is uploaded to Tesla's FSD data center for processing.

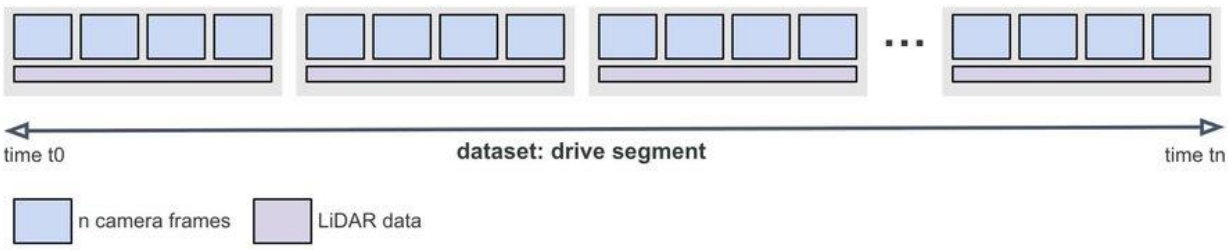


Figure 4: dataset, a drive segment

That’s where the real magic happens.

The camera sensor data is run through a FSD test harness capable of replaying it against any version of the FSD stack— such as the current production version used by Robotaxi (13.3) or a newer one under evaluation. The input data remains in its original photon-counted form (see “Photon Counting” above)— the raw data— and is time-synchronized with a LiDAR point cloud.

As the data flows through the FSD inference engine— identical to the one used in the vehicles— the test harness scores its performance. If any anomalies are detected, such as a potential collision, the relevant data segments are automatically flagged for review.

The LiDAR data serves as a secondary, ground-truth measurement of object distances and spatial layout. FSD outputs not only control commands (the planning side of FSD), but also object positions and estimated distances (the perception side of FSD). If those estimates diverge from the LiDAR-derived ground truth, the data is again flagged for review.

After all test vehicle data is processed, the flagged segments are auto-labeled to reflect corrected spatial or behavioral annotations. These may also be reviewed manually by a Tesla engineer.

Let’s walk through an example.

Imagine a test segment where the vehicle makes a right-hand turn. The test harness detects that it's too close to an oncoming car in the opposing lane—outside its defined safety margin. This is first picked up from the vision data, then verified by the LiDAR’s depth measurements. The segment is flagged, the necessary correction (e.g., increased offset) is auto-labeled in the data, and a Tesla engineer confirms the adjustment. The corrected sequence is then queued for model retraining or fine-tuning.

Simulation But here’s where it gets interesting.

While the validation vehicles collect extensive road-traversal data— sometimes covering the same segment at different times of day (e.g., morning and night)— they obviously can’t capture every possible scenario. The environment is constantly changing. A simple intersection may be empty one moment, and packed with pedestrians the next.

That’s where Tesla’s test harness becomes more than just a playback system— it’s also a full data simulator. It can augment recorded scenes with new objects and conditions. For instance, it can generate alternate versions of a data segment featuring heavy rain, blinding sunlight, or different lighting altogether. It can add pedestrians, other vehicles, or even simulate edge

cases— like a pedestrian entering an intersection, pausing, then abruptly reversing direction after the light changes.

Tesla's simulator is a highly sophisticated software system. It works from the original sensor data, augmenting it with physically accurate, realistic changes—so realistic, in fact, that a human observer would not be able to distinguish the simulated version from an original recording. It's a generative AI engine purpose-built for Tesla's autonomous driving validation.

While the simulator could theoretically create an endless number of scenarios, Tesla likely narrows its focus to realistic conditions tied to the candidate area for geofence expansion. This allows it to generate corner cases and long-tail edge conditions that might otherwise take months or years to naturally occur in the real world.

This is a powerful tool!

Retrain After both the recorded and simulated data are processed through the FSD test harness, Tesla is left with a set of corrected data segments. These are auto-labeled with updated annotations—either for perception (e.g., object classification, spatial location, distance from the vehicle) or planning (accelerator, braking, and steering controls).

For example, if a vehicle underestimated its lateral offset from a parked car, the corrected data reflects the proper buffer. If it initiated a turn too tightly, the planning output is adjusted to maintain an appropriate turning path. These corrections are embedded into the training data (labels) to reinforce desired behavior.

This dataset is then used to fine-tune the current Robotaxi production model— such as version 13.3— across several new training epochs. The aim is to incorporate the improved behavior and perception without degrading prior performance.

Once the model is retrained, it is validated against Tesla's validation dataset to ensure that no regressions have been introduced. If the results are satisfactory, the updated model is promoted to a release candidate— such as 13.3.1— targeted for deployment in the Robotaxi fleet.

Retraining is a critical part of the FSD feedback loop. It reinforces learning by using corrected data to refine the model's behavior. This learning occurs through backpropagation during the training phase— where the model's internal parameters (weights and biases) are adjusted to minimize prediction error and improve overall performance. (Many of the details of training, including backpropagation are covered in the previous parts of this article series[5].)

Bigger Brain

FSD is like a brain— one that improves exponentially as it ingests more data.

On customer vehicles, version 13.2.9 is the latest public release (as of this writing). But Tesla's Robotaxi fleet is running a newer internal build, rumored to be version 13.3. It's unclear how many Robotaxi-only builds exist, and Tesla likely advances them cautiously. Still, it's evident that valuable data is flowing in— data that leads to retrained fixes and model improvements (see "Retrain" above).

According to Elon Musk, these improvements will eventually flow back into the customer FSD release:

Elon Musk @elonmusk·Jul 20

Your Tesla self-driving capability will see a step change improvement as we integrate upgrades for the Austin robotaxi build into the general production release

But Tesla isn't just tuning the current model— it's building a *bigger* one. Version 14 is coming[4].

As covered earlier in this series[5], FSD is powered by a neural network composed of parameters— its internal weights and biases. Version 13 reportedly increased the parameter count by 5x compared to version 12. Version 14 is expected to take another major leap, with roughly 4x more parameters than version 13.

While raw parameter count doesn't determine performance alone—architecture changes matter too, such as deeper networks with more layers—it remains a major factor. Larger models can capture more complex relationships, make finer distinctions, and respond more smoothly to edge cases.

Of course, bigger models require more— more training time in Tesla's Cortex data center, and more compute on the car for inference (the real-time decision-making loop). There's always a tradeoff between model size, onboard hardware limits, and training scalability.

Still, Tesla appears to be pushing ahead. Version 14 is likely being trained not only on the extensive datasets flagged during Robotaxi validation, but also on a growing volume of customer fleet data and other simulator-augmented scenarios.

A bigger brain doesn't just mean more processing— it means safer, smoother driving. And version 14 will represent Tesla's next step change toward that goal.

The Brain Is the Bet

Every engineering project is guided by a core solution philosophy.

Examples are everywhere. Take TCP/IP, the foundational networking technology of the Internet. It was designed with the assumption that networks are inherently unreliable. That led to the development of two key protocols: UDP, which tolerates packet loss for applications that don't

require perfect transmission, and TCP, which ensures reliable, byte-ordered delivery through packet retransmission (and is magically adaptable to fluctuating latency and bandwidth).

Starlink is another example. Its philosophy is to reduce latency by using low-Earth orbit satellites. At just 350 miles overhead, these satellites eliminate most of the speed-of-light delay seen in traditional satellite systems, which rely on satellites orbiting 20,000 miles above the Earth.

FSD is no different. Its core philosophy is to compensate for sensor limitations with a powerful AI brain. This brain—a large neural network—handles both perception and decision-making. It's what enables Tesla to pursue a vision-only approach. Tesla recognized that no sensor, or combination of sensors, would be fast or reliable enough on their own to solve autonomy (and would only increase cost and complexity). Rather than layering on more sensors for redundancy and assuming that one would be correct at a critical moment, Tesla chose to build an AI system capable of interpreting the world accurately from vision alone.

FSD leans heavily on data and a fast feedback loop— constant retraining— to improve over time. The data becomes the brain. And it's this brain that allows Tesla to eliminate not only multiple sensor modalities, but even the camera's image signal processors (ISPs), which introduce photon-to-control delays and degrade image fidelity.

All engineering solutions involve tradeoffs. But often, the winning approaches are the ones that embrace simplicity.

Tesla's approach mirrors how humans drive: our senses are imperfect, yet our brains allow us to operate vehicles effectively (or reasonably so). That's the bet Tesla is making too—not that autonomy is simple, but that its complexity is best concentrated in the brain. Strip away the hardware redundancy. Strip away the sensor fusion logic. Let vision be the input, and intelligence do the work. Simple where it can be. Advanced where it must be.

NOTES

[1] The Bayer mosaic named after Bryce Bayer of Kodak.

https://en.wikipedia.org/wiki/Bryce_Bayer

[2] Demosaicing is the process of reconstructing a full-color image from the incomplete color samples captured by a sensor using a color filter array like the Bayer mosaic. Since each pixel records only one color (red, green, or blue), the algorithm estimates the missing colors by interpolating values from neighboring pixels. The quality of demosaicing affects image sharpness, color accuracy, and artifact levels.

https://en.wikipedia.org/wiki/Demosaicing

[3] Tesla's reliance on LiDAR is shrinking daily, however. The neutral network has largely learned to measure distances and locations of objects in pixel frames. It is however being used in Robotaxi validation as discussed in this article.

[4] Version 14 is also expected to be Tesla's first multi-mode release—capable of operating in both supervised and unsupervised modes. Supervised mode, which requires driver oversight, is currently mandatory in all customer deployments. But eventually, in certain regions, unsupervised operation will be permitted. Robotaxi, of course, will run in unsupervised mode.

[5] Previous articles in this series include:

The Magic of FSD, part 5

The Magic of FSD, part 4

The Magic of FSD, part 3

The Magic of FSD, part 2

The Magic of FSD, part 1

3. Part 5 phil beisel on X: "The Magic of Tesla FSD, part 5" / X

The Magic of Tesla FSD, part 5

Tesla's Full Self-Driving (FSD) is on the verge of its intended milestone: unsupervised operation. This next phase will launch in June in Austin, Texas, as part of the Robotaxi trial.

It marks a defining moment for this AI system, years in the making. FSD is one of two embodied AI's Tesla is developing (the other being the operating system of Optimus), signaling the dawn of true autonomy.

This is part 5 in my series exploring how FSD works. Each part has unpacked technical aspects of the system— often evolving as new details emerge.

In this chapter, we'll dive into recent comments from Elon Musk and FSD VP Ashok Elluswamy during Tesla's Q1 earnings call.

Time to call in the experts.

A Mixture of Experts

During Tesla's Q1 earnings call Elon Musk and Ashok Elluswamy spoke of the concept of FSD adapting to local environments. This confused many, conjuring up ideas of locally mapped and geofenced environments like Waymo. But this is not at all what they meant.

"It’s increasingly obvious that there’s some value to having a localized set of parameters for different ... regions and localities." —**Elon Musk**

"Speaking to the location specific models, we still have a generalized approach, and you can see that from ... the deployment of FSD supervised in China where, with this very minimal data that’s China specific, the models generalize quite well to completely different driving styles. ... So the generalized solution that we are pursuing is the right one that's going to scale well. And you can think of this like location-specific parameters like a mixture of experts." —**Ashok Elluswamy**

Ashok hinted at a **Mixture of Experts** (MoE) architecture.

MoE is an AI model that uses multiple specialized sub-models, each trained for a specific task or context. This approach has been applied in AI assistants like ChatGPT or Grok. For example, if a question is about biology, a sub-model trained on biology texts might be activated, while math and physics sub-models are either inactive or deemphasized.

In the context of Full Self-Driving (FSD), think of it as using sub-models trained for specific driving scenarios— such as highway driving versus navigating parking lots.

Let's explore how a MoE architecture fits with FSD.

FSD Architectures

Before diving into a Mixture of Experts (MoE) architecture for FSD, it's useful to briefly review how neural networks function within the FSD context.

FSD is an end-to-end AI system implemented as a neural network.

Part 3

of this series covers the structure of Tesla’s FSD neural network in more detail, but here's a quick recap.

At its core, a neural network is a set of parameters— **weights** and **biases** —organized into layers. There’s an input layer, one or more hidden layers, and an output layer. Think of it as a *function*: the hidden layers form the body of the function (see Figure 1).

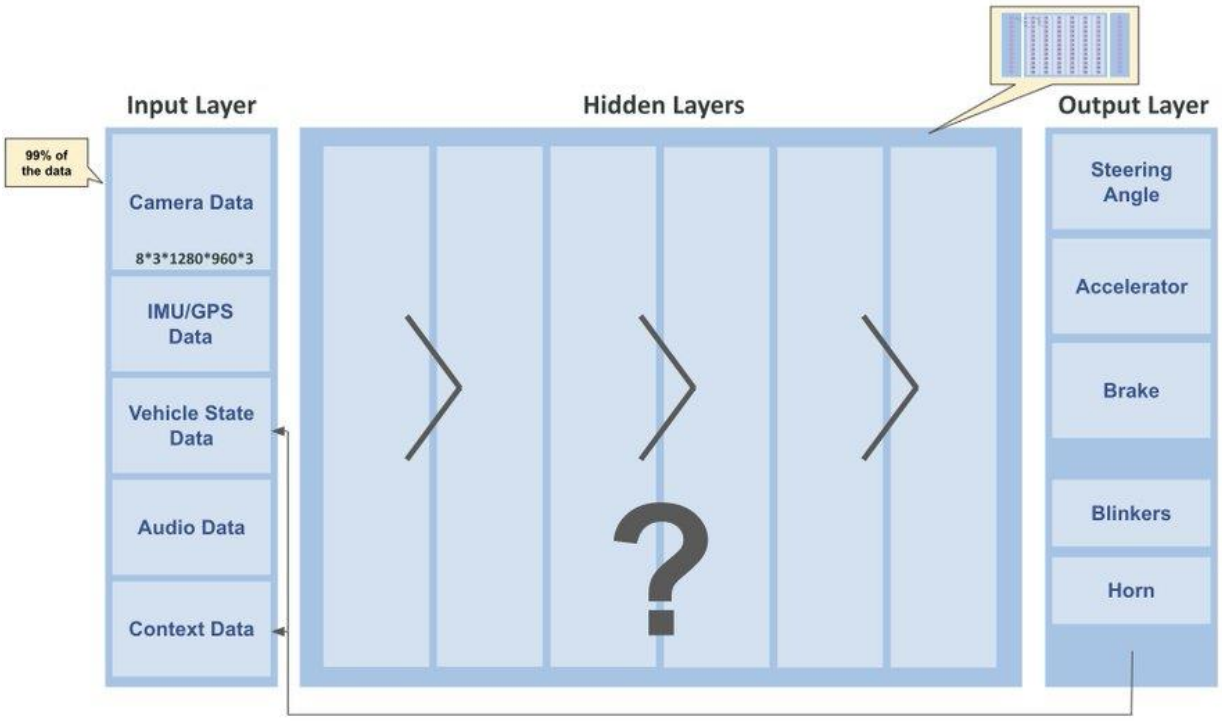


Figure 1

Within the hidden layers are nodes (also called neurons). Each node is connected to some or all nodes in the next layer via **weights**. Each node also has an associated value, called a **bias**.

In the case of FSD, the purpose of the function is to take sensor input and output control values that dictate how the vehicle should drive.

The input is primarily short video sequences from the vehicle's eight external cameras. The output consists of values that determine steering, acceleration, and braking.

Model 1: Single Neural Network Conceptually, FSD can be represented as a single neural network: input data flows through hidden layers and produces outputs (see Figure 2).

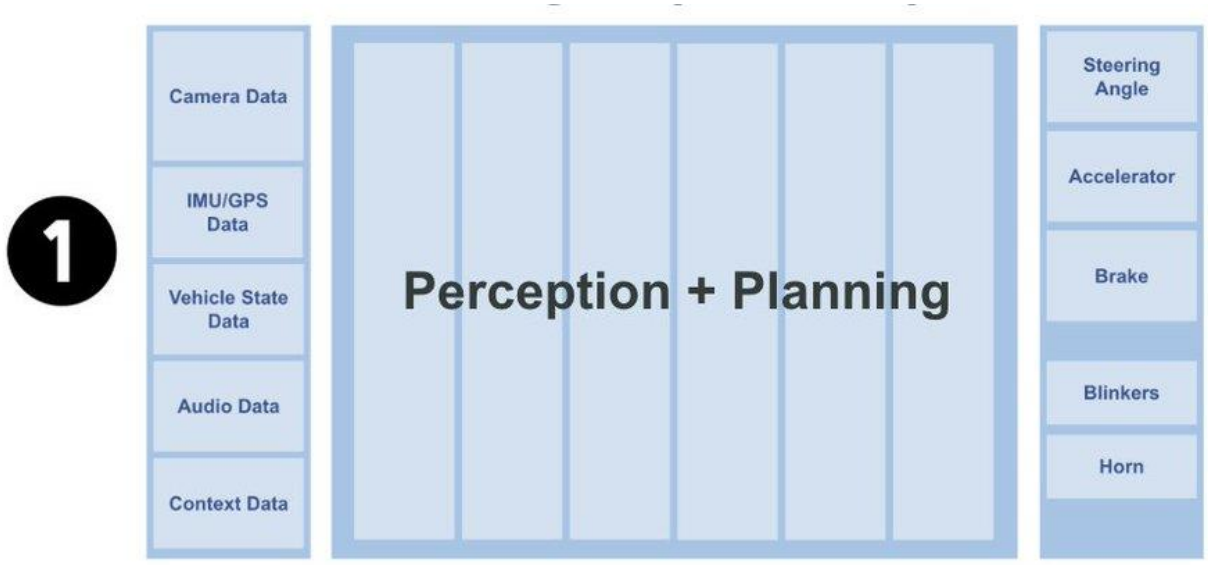


Figure 2

Although FSD is often described this way, it is more accurate to break it down into two logical components: **perception** and **planning** (see

Part 1

):

- **Perception**: Processes camera video to detect and label objects in the environment—such as road markings, traffic signs, and moving agents (e.g., vehicles, pedestrians)—assigning attributes like position, velocity, intent, and confidence.
- **Planning**: Takes the perception data and, in combination with the navigation goal, determines the next vehicle actions. It generates control signals (steering, throttle, brake) 15 to 30 times per second.

In Model 1, perception and planning are fused into a single network. While elegant, this approach is ultimately too simplistic.

Model 2: Duel Neural Networks Model 2 separates the perception and planning functions into two distinct neural networks. The output of the perception network becomes the input for the planning network (see Figure 3).

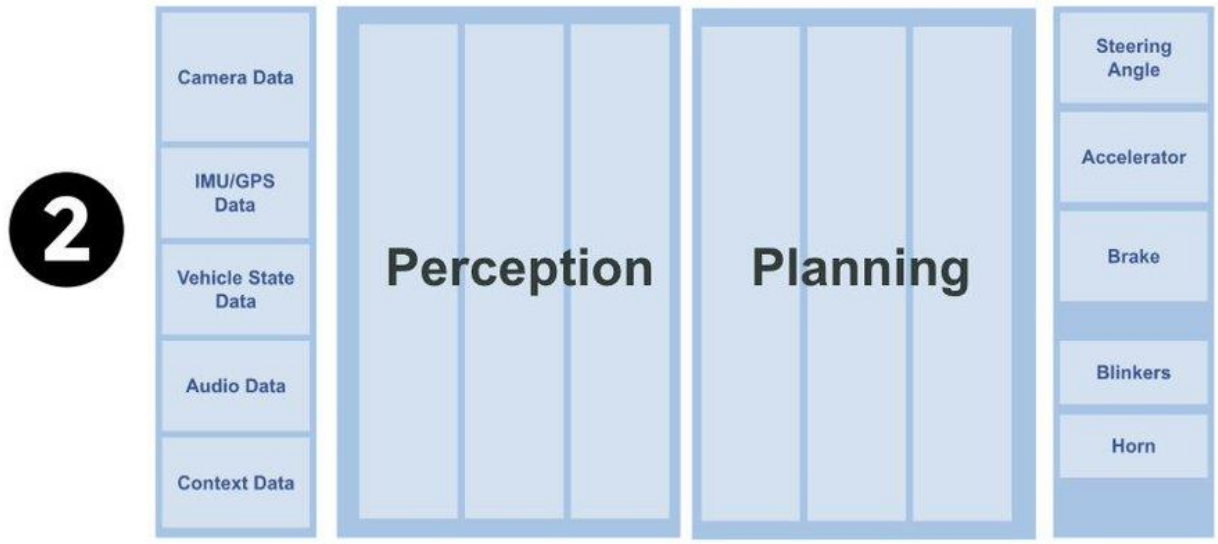


Figure 3

This separation enables greater specialization and scalability, with each network optimized for its specific role.

Model 3: The Experts Model 3 builds upon Model 2 by introducing specialized *expert* sub-layers—this is the architecture we'll explore in detail below.

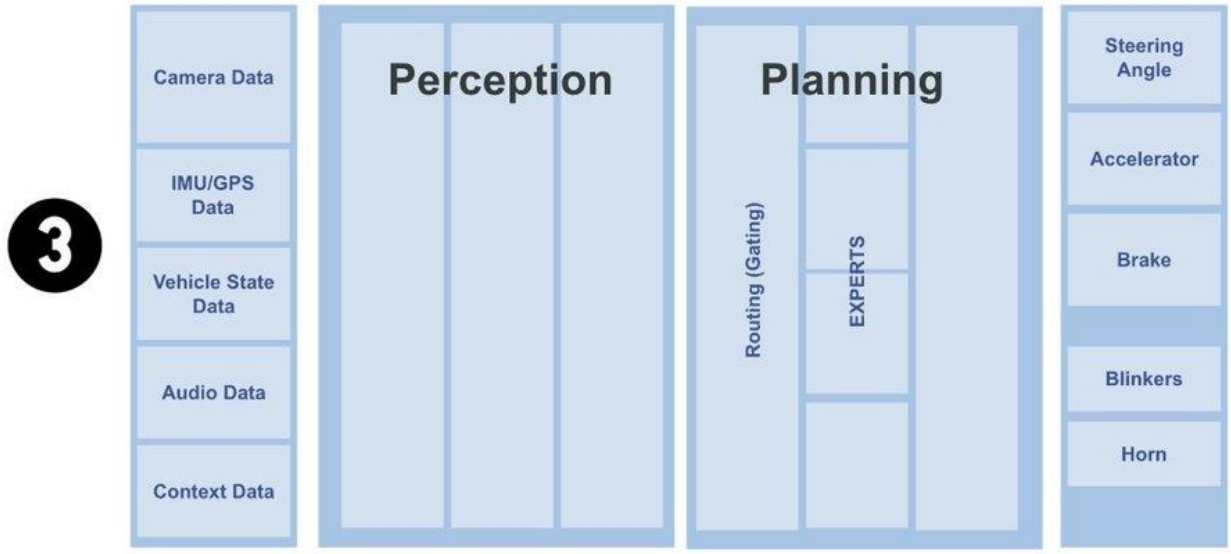


Figure 4

FSD Mixture of Experts Architecture

Model 3 (Figure 4, above) introduces the concept of sub-layer *experts*. Each expert is trained on data specific to its area of specialization.

Driving is not a uniform task, it encompasses a wide range of scenarios that require distinct skills. For instance, highway driving differs significantly from navigating dense urban environments. Some skills are highly localized, such as driving in snow. Drivers in snowy regions often develop techniques for handling slippery roads, while those from warmer climates may have no experience with such conditions.

Figure 5 illustrates the expert sub-layers and outlines the data flow within the system.

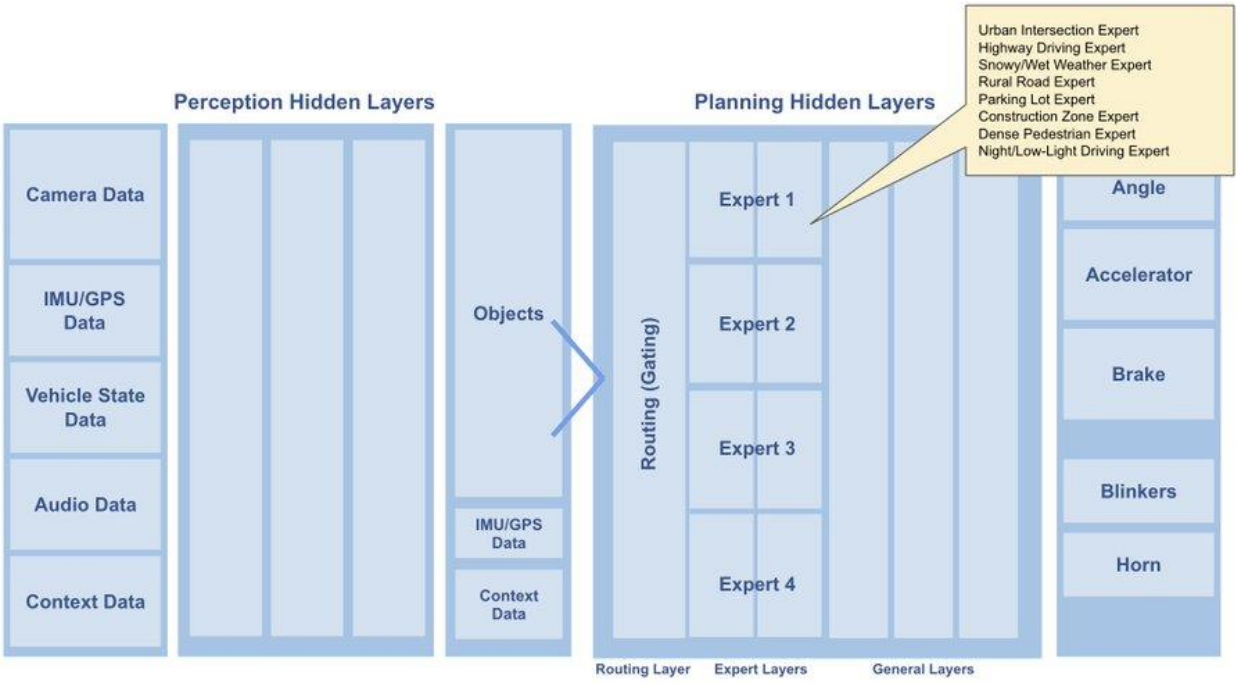


Figure 5

The raw input data (e.g., camera feeds) is processed by the perception neural network, which outputs a set of *objects*— data elements describing the scene. These, along with IMU/GPS inputs and contextual data, are passed to the planning neural network.

At this stage, the context— initially informed by perception— is further enriched with details relevant to expert selection. Existing context data may include factors like precipitation level (e.g., raining), precipitation type (e.g., snow, sleet, rain), time of day, and ambient light. New context is also added, such as:

- Operating on a highway

- Approaching or entering an intersection
- Navigating within a parking lot
- Detecting construction zone elements

A *routing layer* (or *gating layer*) processes these inputs and determines which experts to activate and how much influence each should have on the output.

Examples of potential sub-layer experts include:

- **Urban Intersection Expert** Handles complex intersections (e.g., traffic lights, pedestrian crossings, multi-lane turns). Optimized for predicting vehicle and pedestrian movement and managing right-of-way.
- **Highway Driving Expert** Focuses on high-speed lane-keeping, merging, and overtaking. Tuned for long-range planning and smooth speed transitions.
- **Snowy/Wet Weather Expert** Adapts decisions to low-traction conditions. Prioritizes braking, cautious acceleration, and visibility-aware navigation.
- **Rural Road Expert** Navigates narrow, unmarked, or winding roads with potential obstacles (e.g., animals, debris). Emphasizes adaptive speed and obstacle avoidance.
- **Parking Lot Expert** Specialized for low-speed maneuvers in tight or crowded spaces. Focuses on pedestrian detection and identifying parking spots.
- **Construction Zone Expert** Recognizes temporary changes such as cones, detours, and road workers. Adjusts path planning dynamically.
- **Dense Pedestrian Area Expert** Designed for areas with heavy foot traffic (e.g., school zones, downtown). Prioritizes pedestrian tracking and conservative driving behavior.
- **Night/Low-Light Expert** Handles poor visibility scenarios (e.g., fog, nighttime). Enhances reliance on sensor fusion and contrast detection.

Figure 6 depicts a single inference cycle in which a subset of experts is activated. For example, both the *Urban Intersection Expert* and *Snow/Wet Weather Expert* are selected, with weights of 0.79 and 0.21 respectively. These weights influence the model's output.

Consider the example of determining the accelerator pedal value. If the Urban Intersection Expert suggests accelerating to 45 mph (e.g., green light, open intersection), but the Weather Expert suggests 25 mph due to light rain, the resulting output would be a weighted combination— approximately 40 mph.

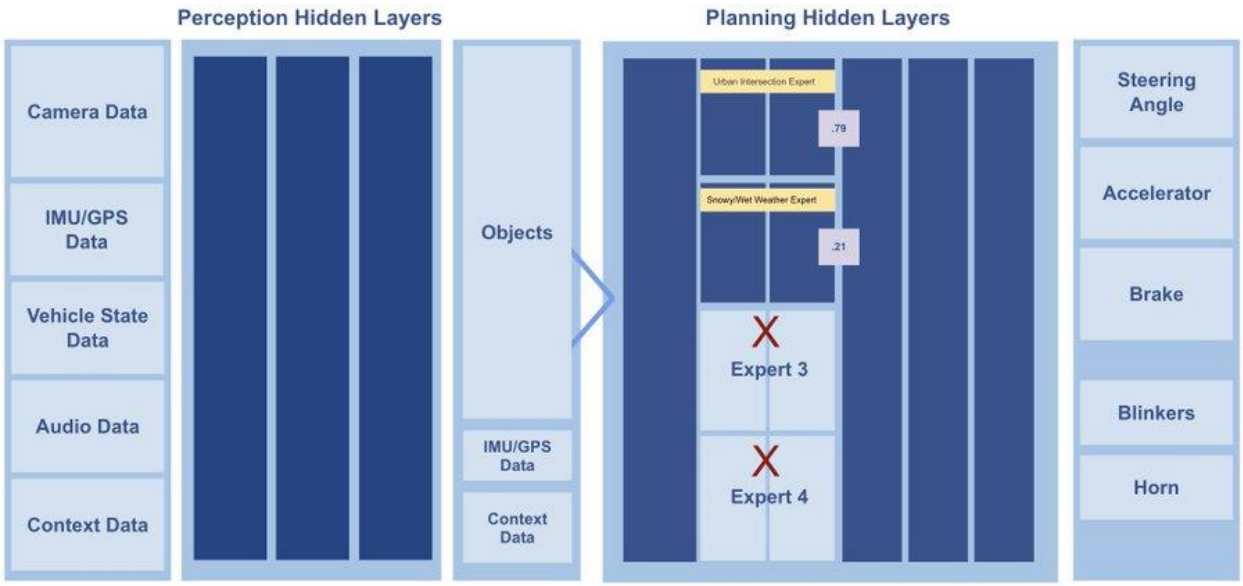


Figure 6

In each inference cycle, not all experts are used. Only selected experts (e.g., Experts 1 and 2) are activated, while others (e.g., Experts 3 and 4) are inactive.

This selective activation not only improves decision quality by leveraging relevant expertise but also reduces computational overhead. For instance, while the full model may have 3 billion parameters, activating only a subset of experts may engage just 1.6 billion parameters— saving some inference compute time.

Finally, the expert configuration can change dynamically (in each inference cycle). As the vehicle exits the intersection, the Urban Intersection Expert is deactivated, and others may be engaged depending on the new driving context.

Expert Advantages

There are many advantages to using a Mixture of Experts (MoE) model in Tesla's self-driving system.

MoE enables Tesla to develop a general-purpose autonomous driving system that can still adapt to local variations in laws, road conditions, and driving culture. This combination of broad capability and local specialization is extremely powerful. Experts trained for specific regions or scenarios can deliver a smoother and safer driving experience by handling the nuances that differ from place to place.

Another advantage of MoE is the flexibility it brings to training. Instead of retraining the entire model from scratch, Tesla can focus on updating individual expert sub-networks— either

refining existing ones or introducing new ones as needed. This modular approach makes the system more scalable and efficient, allowing it to improve over time without full retraining cycles.

On the vehicle itself, MoE improves runtime efficiency. Only a subset of experts is activated for any given inference cycle, reducing computational load. This allows Tesla to optimize for ultra-low "photon to control" latency—meaning more visual observations per second and faster response times, which directly enhances driving safety.

The Road Ahead

As Tesla prepares to launch unsupervised FSD trials in Austin, the Mixture of Experts architecture emerges not just as a technical evolution, but as a philosophical one: a system that can learn not only how to drive, but *where* it's driving— contextual, adaptive, and truly autonomous.

What's being tested on the streets of Austin isn't just software. It's a theory of intelligence: one that says a machine can become situationally aware, regionally fluent, and globally scalable.

If it works, FSD won't just generalize. It will *contextualize*— turning learned experience into local fluency, one expert at a time.

And that's the real milestone ahead.

4. Part 4. [(1) phil beisel on X: "The Magic of Tesla FSD, part 4" / X]

The Magic of Tesla FSD, part 4

Most people still don't fully grasp what Tesla has accomplished with Full Self-Driving (FSD) — and that's okay. If you're reading this, you're already ahead of the curve. You're curious about how it works, and more importantly, you recognize the scale of what's coming.

We're now about two months out from the launch of the Robotaxi trial in Austin, TX. At the heart of it all is unsupervised FSD— the real deal.

As Robotaxi begins to scale and Cybercabs show up everywhere, people will, for the first time, experience being driven by a robot. Some will ask how it works. Others will just accept the magic and move on.

But for the curious— let's zoom out. This article gives you a cocktail-party-level explanation of what's under the hood. Enough to impress your friends, or just help them understand the seismic shift steering us into the future.

I suggest that you read Part I, II, and III of this series to gain a more complete understanding and perspective.

Part I

sets the stage,

Part II

goes deep, and

Part III

ties up some loose end and discusses FSD's future.

The Driving Function

When we think about a human driving a car, we usually picture the input controls— the accelerator, brakes, and steering wheel. Since the invention of the automobile, it's been a human in the driver's seat: observing the world, forming a plan, and executing it through those controls. That "plan" includes both the destination and every tiny decision along the way.

But if you ask,*"how exactly does a human decide what to do?"* — most people haven't really thought about it. Driving feels automatic. You learn the rules, you practice, and eventually, you just do it.

Now flip that around. The idea that a car might drive itself forces a different kind of scrutiny. Suddenly, *how* it works becomes a real question.

While most people have never written a line of code, they can still wrap their heads around the basic logic of driving. There's an intuitive sense that decisions are made based on inputs— "if the light turns red, then stop" or "if a car slows down ahead, then brake." That kind of *if-condition-then-action* thinking feels natural. It's how we imagine a self-driving car must work.

But that's not how FSD works. Not even close.

Functionally Speaking We can think of autonomous driving as a simple function: from a set of input data the *function* decides what to do and returns those instructions as output.

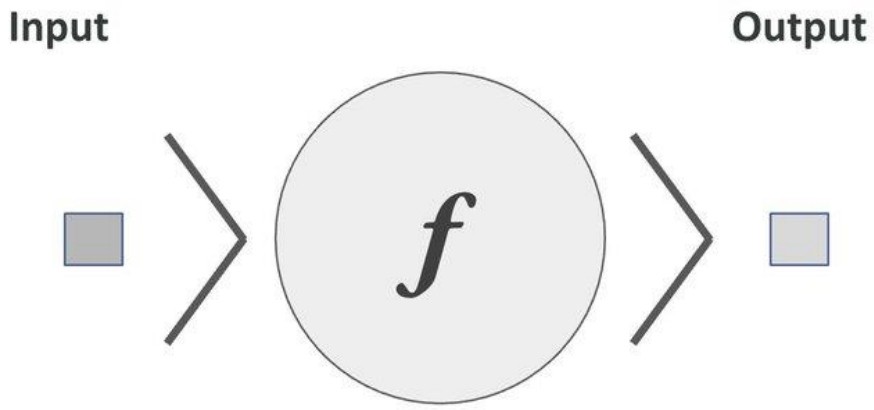


Figure 1: the driving function

In FSD, the input is primarily camera data— each vehicle is equipped with 8 high-resolution cameras. Much like a human uses eyes as the input sensor, FSD too is a vision-based system. The output is simple a set of values that determine the control states of accelerator, brakes, and steering.

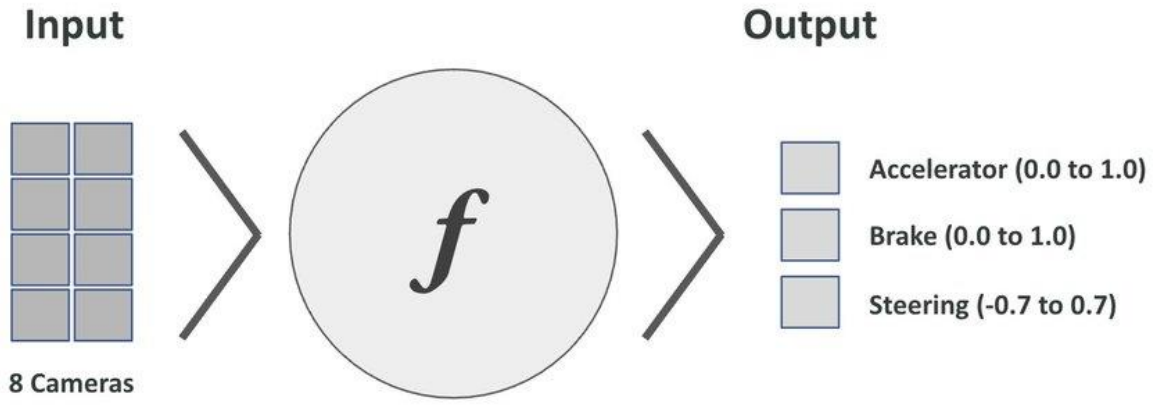


Figure 2

What might that function do?

The Coded Function This function could be written in code— a set of instructions in a programming language like C++. These instructions take input data and decide what to do— in other words, they output values to control the vehicle's accelerator, braking, and steering.

As discussed above, it's logical to think of a coded approach, because at its core, the behavior could be expressed as a (very complex) series of *if-condition-then-action* statements. This is often referred to as a rules-based approach.

One common technique for building a rules-based solution is a **finite state machine** (FSM). In this model, the vehicle is always in one of a predefined set of states, and based on input data (conditions), it either remains in the current state or transitions to a new one— producing corresponding outputs along the way.

Before version 12, the Full Self-Driving (FSD) system used a coded approach for its planning component. FSD can be logically divided into two main parts: **perception** and **planning**. Perception is the process by which the vehicle interprets the world using camera input, identifying and categorizing elements in the scene such as vehicles, pedestrians, traffic signs, etc. Planning, on the other hand, is the function responsible for deciding the vehicle's next action based on the perceived scene.

Let's focus on planning and develop an example finite state machine coded-solution (see Figure 3).

```
// output control signals
struct ControlCommand {
    float acceleration      // 0.0 to 1.0
    float brake             // 0.0 to 1.0
    float steering_angle    // -0.7 to +0.7 radians
}

// define planner states
enum State {
    DRIVE,
    DRIVE_FOLLOW,
    STOP,
    TURN,
    EMERGENCY
}

// start in DRIVE
state = DRIVE

// main control loop
while (true) {

    // input from perception system
    objects = get_perception_objects()

    // default control values
    command = ControlCommand(0.2, 0.0, 0.0)  // moderate acceleration, no brake, straight steering

    switch (state) {
        case DRIVE:
            if (objects.closest_obstacle.distance < safe_distance) {
                state = EMERGENCY
            } else if (objects.lead_vehicle.distance < follow_distance) {
                state = DRIVE_FOLLOW
            } else if (objects.traffic_light.state == RED && objects.traffic_light.distance < 25) {
                state = STOP
            } else if (objects.nav_path.turn_required) {
                state = TURN
            } else {
                command.acceleration = 0.5
                command.brake = 0.0
                command.steering_angle = compute_steering(objects.lane_center)
            }
            break

        case DRIVE_FOLLOW:
            command.acceleration = compute_follow_accel(objects.lead_vehicle)
            command.steering_angle = compute_steering(objects.lane_center)
            if (objects.lead_vehicle.distance > resume_distance) {
                state = DRIVE
            }
            break

        case STOP:
            command.acceleration = 0.0
            command.brake = 1.0
            command.steering_angle = 0.0
            if (objects.traffic_light.state == GREEN) {
                state = DRIVE
            }
            break

        ...
    }

    send_commands(command)
}
```

Figure 3: Example "planning" finite state machine

The planner includes a fixed set of states: DRIVE, DRIVE_FOLLOW, STOP, TURN, and EMERGENCY.

Structurally, it operates as a control loop. In each cycle of the loop, it gathers input from the perception module and, based on the current state and those inputs, determines whether to remain in the same state or transition to another. For each cycle, it outputs a command—

acceleration, braking, and steering angle— and sends that command to the vehicle for execution.

This code is purely illustrative— at best, it might crash your car. But imagine developing it into a real planner. While the number of states might not grow dramatically, the logic behind each state would quickly become exponentially more complex. (It's been said that FSD version 11's planner was 300,000 lines of C++ code.) With the sheer variety of possible driving situations, calling it a finite state machine might start to feel ironic.

What can we conclude?

The coded approach is *brittle*. For example, suppose we have a rule that says: *if the stoplight ahead is red and the car is within 25 meters, switch to the STOP state.* But what about 24.8 meters? Or 25.5? In some scenarios, stopping slightly earlier or later may be more appropriate. Hardcoded thresholds like these are inflexible and prone to edge-case failures.

The coded approach is *complex*. While the number of states may remain fixed, the number of conditionals and branching logic paths grows rapidly. Some of these logic flows even loop back on themselves. The states may be finite— but the logic feels anything but. There's always one more exception, one more nuance that forces you to add more code.

The coded approach, however, is *debuggable*. When the system behaves incorrectly, we can trace the problem back to a specific rule, a bad input, or a flawed assumption. As we'll see with the AI-based approach (discussed below), that level of traceability largely disappears— all we are left with is behaviors.

For all the reasons states above, we can understand why starting in version 12 of FSD this approach was discarded.

The AI Function If we set aside the coded approach, how else can we develop the logic required to autonomously control a vehicle? The answer lies in the data itself. By training a system on high-quality driving data, it can learn to behave appropriately— not by following explicit rules, but by learning from experience.

In this data-driven approach, the function isn't defined by hand-written logic conditions. It's not programmed line by line by a team of developers. Instead, the behavior is learned from examples.

Before going deeper, let's keep things intuitive. To understand how Full Self-Driving (FSD) works, it helps to build a mental model— even without a technical background. Earlier, we discussed how even non-programmers can grasp the rules-based idea: *"If the light turns red, then stop."* That kind of logic is simple and intuitive. You don't need to understand coding to understand the model.

But while that logic is easy to grasp, it doesn't hold up well in practice.

Now imagine we collect driving data from thousands of real-world situations where vehicles approach a traffic light. Sometimes the light is green, sometimes amber, sometimes red. As it shifts from amber to red, driver behavior changes. Some drivers come to a stop. Others— if they're close to the intersection— accelerate to make it through. And occasionally, someone runs the red light, whether intentionally or not.

Among all these scenarios, some behaviors stand out as clearly better than others— safe, efficient, and compliant with traffic laws, without being overly cautious or erratic.

Now imagine we could tell a system: *"Behave like this."* Learn from the good examples. That is FSD.

The coded function is replaced by a "behave like this" function— one that implicitly encodes those learned behaviors through selected data.

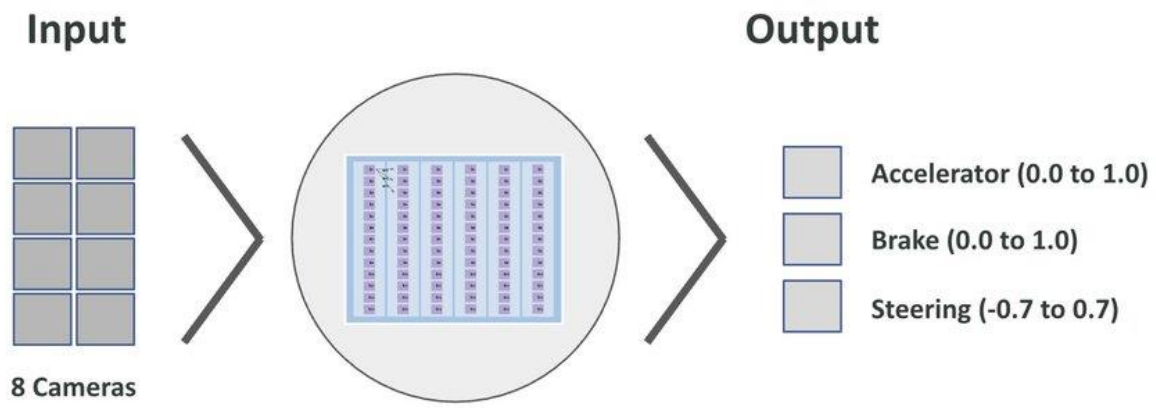


Figure 4: the AI function

Part 2

explains how this process works in detail. Here, we'll just scratch the surface— to give you a rough sense of what the AI is actually doing.

The AI function, as depicted above in Figure 4, contains a vast set of numbers, called **parameters**, that capture these good driving behaviors. These parameters are set during a process called training that happens exclusively in Tesla's data center.

These parameters form a layered structure called the **model**. The model is transferred to the vehicle and, through a process called **inference**, input data (the camera data) is fed into the model, processed, and control outputs are produced.

Through inference, raw sensor data is transformed into driving actions that *behave like* the examples the system was trained on— safe, efficient, and human-like.

This is our AI function.

Fixing the Function How do we debug problems in the coded approach and the AI approach?

By examining this question, it will further draw out the distinction between these very different approaches.

Let's imagine a scenario where a Tesla, with Full Self-Driving (FSD) engaged, fails to execute a lane change. The vehicle ahead (Vehicle A) begins to slow down, and the Tesla responds by applying the brakes to maintain a safe following distance. As Vehicle A continues to decelerate— now traveling below the speed limit— the Tesla maintains its position behind it. Meanwhile, in the adjacent left lane, another vehicle (Vehicle B) is traveling faster than Vehicle A. Despite the opportunity to change lanes and resume a higher speed, the Tesla fails to initiate the maneuver. This frustrates the supervising driver, who intervenes manually and reports the issue.

Let's start with the coded approach.

Tesla engineering begins debugging the issue. Let's assume this is FSD version 11, where perception is handled by AI, but planning relies on conventional code. When investigating a problem, all potential causes are considered. It could be a perception issue— such as misclassifying Vehicle B or failing to estimate its forward speed accurately. Or it could stem from the planning logic.

To narrow it down, engineers load a simulator with representative data that reproduces the scenario. In this case, Vehicle B maintains a position close to Vehicle A for about 2 seconds before accelerating. Upon reviewing the planning code, a logic error is identified— specifically, a flaw in the system's evaluation window: the system only uses a 1-second time frame to assess lane change opportunities. During that brief period, it sees Vehicle B traveling at the same speed as Vehicle A and concludes the lane change is unproductive.

The code is updated to use a 2-second evaluation window. When the scenario is re-run in the simulator, the Tesla correctly identifies the opportunity and executes the lane change.

While this code change works, it is brittle. Is 2 seconds the correct value, maybe it should be 2.5. You get the idea.

Now let’s examine what it might look like to debug and fix this problem using the AI approach.

As with the coded approach, Tesla begins by loading representative data into a simulator and confirming the undesired behavior. The production model fails to perform the lane change.

Next, engineers examine several training data samples where the in-lane vehicle slows down. They observe that while many of these samples include a faster-moving adjacent vehicle (like Vehicle B), none capture the specific case where that vehicle maintains a matching speed briefly before accelerating— a subtle but important nuance.

To address this, Tesla engineers collect several hundred new training samples that reflect a broader range of adjacent-vehicle behavior, particularly variations in speed over short time windows.

The production model is then re-trained using *only* these new samples, leading to adjustments in its internal parameters. After training, the same scenario is re-run in the simulator. This time, the AI model correctly identifies the lane change opportunity and executes the maneuver.

This process is known as **fine-tuning**. Only the new data samples are used during this step.

Which parameters are adjusted? Many of them.

At this point, two important questions must be answered:

1. Did the identified problem get fixed?
2. Did fixing the problem break any existing behaviors?

In the AI approach, there is no code logic to inspect directly. There is no way to verify correctness except through testing. This is why a validation dataset is critical. Before releasing the newly trained model, it must be evaluated against the validation set to confirm that it performs with equal or better loss compared to the previous version.

Data Drives

FSD is data driven, that is data itself builds the *function*. There is no code in the decision framework for autonomous driving.

In a coded system, the performance depends on the skills of the programmer or development team. In an AI-based system, it depends on the quality of the training data, which reflects the skills of the drivers. But to make this data-driven approach work, you need high-quality data—data where good driving behavior is chosen and poor behavior is discarded.

Data selection is a critical part of the training process (covered in

Part 3

). It's a complex task, and Tesla has built a highly-skilled AI team to automate both data selection and labeling.

The process centers around two key goals: capturing data for all conceivable driving scenarios, and ensuring that data comes from the best drivers.

Ultimately, FSD can only drive as well as the data it's trained on— in fact, in a sense, it can drive no worse.

The Mechanics Behind the Magic

What Tesla has accomplished with FSD isn't just a step forward in autonomy— it's a defining moment. A technological achievement that, for the first time, delivers on the promise of a car that can truly drive itself. And increasingly, the technologies reshaping our world are starting to feel more like magic than machines.

When the printing press emerged, it was revolutionary— but not mysterious. People could see how it worked: type, ink, paper, gears. It was understandable. Even if they didn't need to understand it to benefit from it, they *could*. That gave the technology a kind of transparency— its impact was massive, but its workings were visible.

FSD exists on the opposite end of that spectrum. The experience feels simple—even magical. You sit. The car drives. But the underlying mechanism is anything but obvious. No one watches a neural network "think" the way they could watch a press operate. There are no gears turning, no plates inking paper— just behavior learned from vast oceans of data encoded into mathematical functions.

This is the new kind of magic: useful, invisible, and increasingly hard to explain. And maybe that's fine. Most people won't need to understand FSD to use it, just like they didn't need to understand the printing press to read a book.

But for the curious— for those who want to see what's behind the curtain— understanding offers something valuable. Not just insight, but a kind of *orientation*. A way to stay grounded as we move into a world increasingly shaped by systems most people can't build and few truly understand.

Because when technology starts to feel like magic, it's worth asking: is the magic in the machine— or in our ability to understand it?

5. Part 3. [(1) phil beisel on X: "The Magic of Tesla FSD, part 3" / X]

The Magic of Tesla FSD, part 3

Robotaxi is coming— starting with a trial in Austin, TX this June. And with it we'll likely see the release of FSD v14, the first **unsupervised** version.

Over the past six months, I've covered FSD in two major articles. Now, it's time for part three—to go deeper, tie up loose ends, and preview what's next.

I encourage you to read the first two articles before diving in, in particular part 2 (many of the topics discussed assume you've take a read of part 2).

phil beisel

@pbeisel

.

Sep 4, 2024

The Magic of Tesla FSD

In this article I will attempt to explain how Tesla's Full Self-Driving (FSD) works (at a high level) and why it's like magic in a box! In Tesla's latest release of FSD, autonomy is on the path to...

phil beisel

@pbeisel

.

Mar 18

The Magic of Tesla FSD, part 2

Tesla is on the brink of revolutionizing the world with autonomy— robots on four wheels (self-driving vehicles) and two legs (Optimus humanoid bot). Its Full Self-Driving (FSD) technology is already...

Lots of Supervision

Normally, when you hear the word 'supervised' in the context of Full Self-Driving (FSD), the immediate thought is that the human driver must remain alert and ready to intervene, as the system isn't fully autonomous and depends on human oversight to manage challenging scenarios.

However, 'supervised' also carries a more technical meaning in this domain. The AI powering FSD is trained using **supervised learning**, a process where an algorithm learns to make predictions or decisions from a **labeled** dataset. This dataset pairs input data (**features**) with correct outputs (**labels**), serving as a 'supervisor' to steer the model's training.

This is touched upon in Part 2, but here we'll go a bit deeper.

Assume we have a batch of 100,000 data samples showing a vehicle ahead in a lane slowing. We need to label the outputs— steering angle, accelerator, and brake— so our vehicle can safely adjust speed, maintain a proper following distance, or come to a stop if necessary.

In a small dataset, labeling can be done manually. For example, in handwriting recognition, a human can review each sample and assign the correct label. However, for large-scale driving datasets, manual labeling is impractical.

Each data sample consists of input features (as discussed in Part 2) and corresponding output labels— the desired vehicle controls.

Figure 1: A data element for training, the feature with its label

Tesla likely employs an advanced labeling system within an overall driving simulator framework. This system processes each data sample and calculates appropriate control values (e.g., steering = 0.00, accelerator = 0.00, brake = 0.25 for 25% braking). A human oversees the process, refining labels as needed, and the system likely flags uncertain cases for manual review.

The labeling tool likely uses a baseline physics engine to simulate vehicle dynamics and interactions with other objects, generating control labels that follow safe-driving policies. It

likely leverages Tesla's own perception AI systems to identify objects, lane lines, and dynamic actors in the scene, using that output as part of the labeling process. This combination of physical modeling and AI-driven perception enables fast, scalable, and contextually accurate label generation with minimal human input.

Data Selection

Training a model like version 13 of FSD requires hundreds of billions of data elements. But which ones?

Most of the data comes from real-world driving, collected by Tesla vehicles. Additional data is generated through simulation.

The core idea behind data selection follows the "Goldilocks Principle"—finding a balanced middle ground. Data from poor drivers is excluded, such as those who are overly aggressive, excessively cautious, or simply inattentive.

FSD collects data whether it's actively engaged or not. When not engaged, it runs in "shadow mode", silently comparing its own decisions to those made by the human driver.

The first round of data selection happens on the vehicle itself, since raw driving data is large and costly to upload and store. When FSD is engaged, data is likely prioritized around interventions or specific driving scenarios Tesla wants to capture. In shadow mode, the system flags moments where its decisions differ from the human driver's, as these may be useful for training.

Simulated data also plays a crucial role. Tesla has developed an advanced driving simulator that can create complex scenarios or modify real-world driving data to change the "scene." For example, they can load a dataset showing a vehicle navigating a specific city intersection, then alter it by adding or removing dynamic and static objects—or by modifying existing ones, such as partially occluding a stop sign or introducing rain, snow, or fog.

Simulated data is particularly valuable because it represents scenarios that cannot be easily gathered from real-world driving. The idea is that this data is a variation of real-world data, but different in how it can be controlled and customized. For example, scenes can be augmented with weather conditions like heavy snow, rain, or fog. It can even simulate situations where sensors are occluded, such as a camera being blocked by sunlight. Ultimately, data generated by the simulator is indistinguishable from real-world data.

Recently, Tesla launched Full Self-Driving (FSD) in China, a particularly challenging feat due to the restrictions on uploading data from Chinese drivers to Tesla's U.S.-based data center. To overcome this, Tesla gathered a variety of driving scenarios from publicly available sources, such as YouTube videos, and used this data to configure their simulator to replicate these driving

environments. This approach allowed Tesla to simulate real-world driving conditions in China and improve the FSD system, even without direct access to local data.

There are two key takeaways from this section:

First, Tesla's data selection process is a significant competitive advantage. While they have access to the most real-world data, the challenge lies in selecting only the data that represents the best driving behavior. This selective process is crucial for training a highly effective system.

Second, understanding how FSD is trained on high-quality data reveals why it can't drive poorly. Since it is trained on human driving decisions, FSD will make human-like choices. It won't just be safe; it will also be smooth, fluid, and natural, mimicking the decision-making process of experienced drivers. It really can't do otherwise.

Modeling FSD's AI

A simplistic view of Tesla's FSD AI is that it's powered by a single neural network. As discussed earlier, the purpose of training is to build a **model**— essentially a function. On the vehicle, during inference, input data flows through the model to produce the necessary outputs—namely, vehicle control commands.

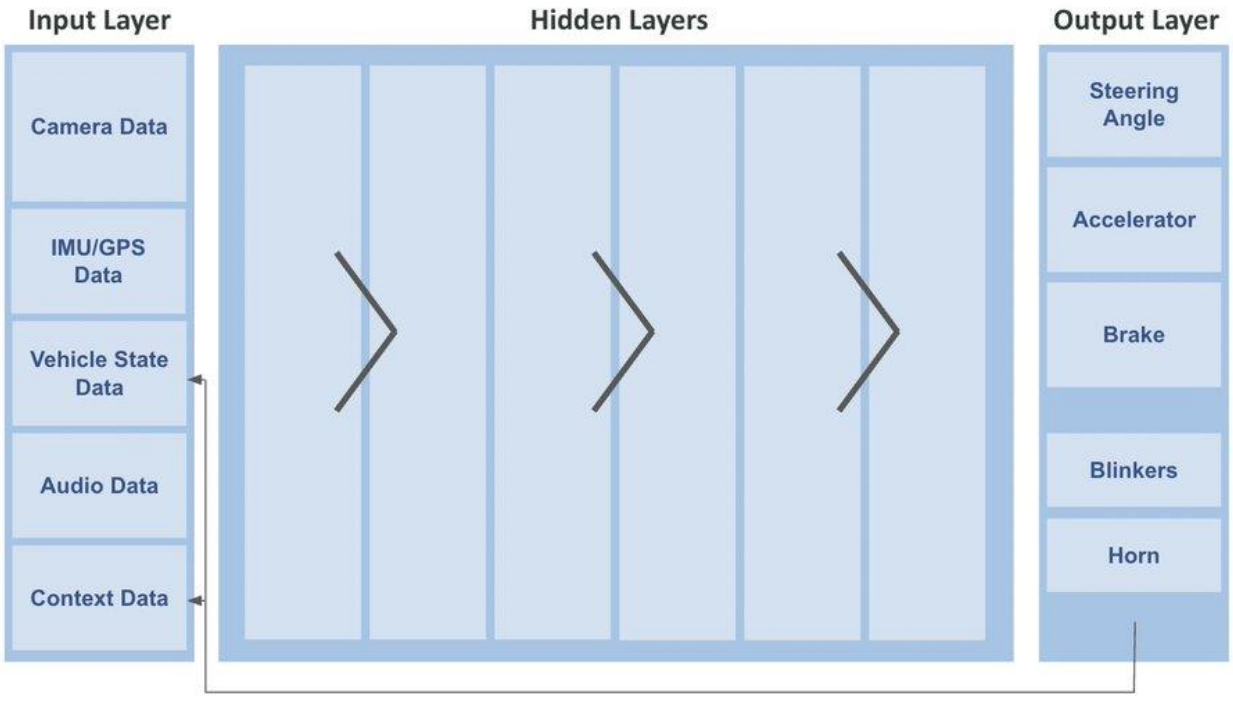


Figure 2: FSD architecture, 1 model

It’s conceivable that FSD runs as one large, unified model— in fact, that would be ideal. It would be elegant, streamlined, and high-performance.

But reality may call for something more layered. In Part 1, there’s considerable discussion about perception and planning— two formal components in self-driving systems. Perception is how the vehicle interprets its environment; planning is how it decides what to do next. These are logically distinct, and it’s possible Tesla maintains that separation internally.

So how might that look in FSD?

In the model description above, camera data is shown as a direct input to the model. That is, for each inference pass— typically running 15 to 30 times per second— a few frames from each of the eight cameras are captured and pushed through the model. Part 2 dives deeper into the structure and sizing of this data.

But now consider an alternative architecture: instead of sending raw camera data into the planning model, the system could first convert it into an object map— a higher-level interpretation of the scene (see Figure 3). The same idea could apply to audio: instead of raw waveform data, the model would consume processed interpretations.

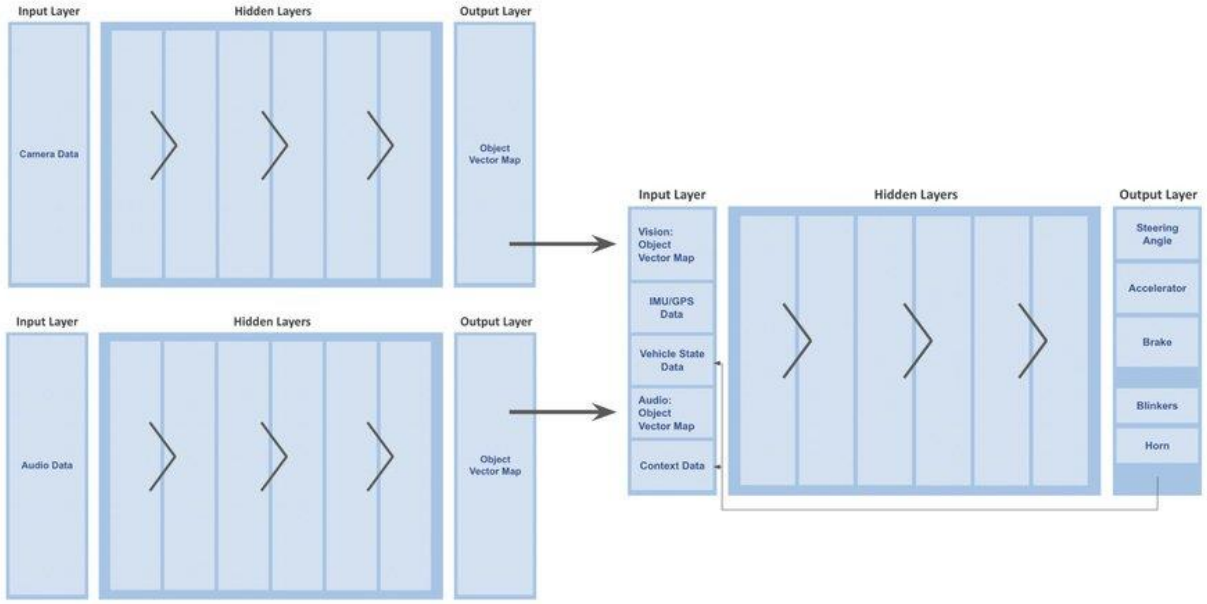


Figure 3: FSD architecture, perception and planning models

This effectively separates perception from planning, at least for the primary input types. Camera data is first processed by a perception model, whose output is a set of objects— each representing a static or dynamic object in the scene.

Each object is defined by its type, position, direction, speed, and size (see Figure 4 below). Additionally, it may include 'intent data'— indicating a future state— such as a 'pedestrian' object having an intent like 'entering crosswalk'.

Object Vector	Example: stop sign	Example: vehicle
Object Type	stop_sign	vehicle
Object ID	3DE-122F4-D45	2DE-543F4-F98
Timestamp	1746441600.500	1746441600.500
Direction	0 degrees	10 degrees
Speed	0 m/s	29.06 m/s
Position	37.7749° N, 122.4194° W, 10m alt)	37.7744° N, 122.4184° W, 11m alt)
Size	2.7m x 1.1m x 0.6m	4.7m x 1.9m x 1.6m

Figure 4: Object

One benefit of this multi-layer architecture— using multiple neural networks— is the availability of intermediate outputs. For example, the Tesla user interface shows a virtual view of the environment, essentially visualizing what the system "sees" (see Figure 5).

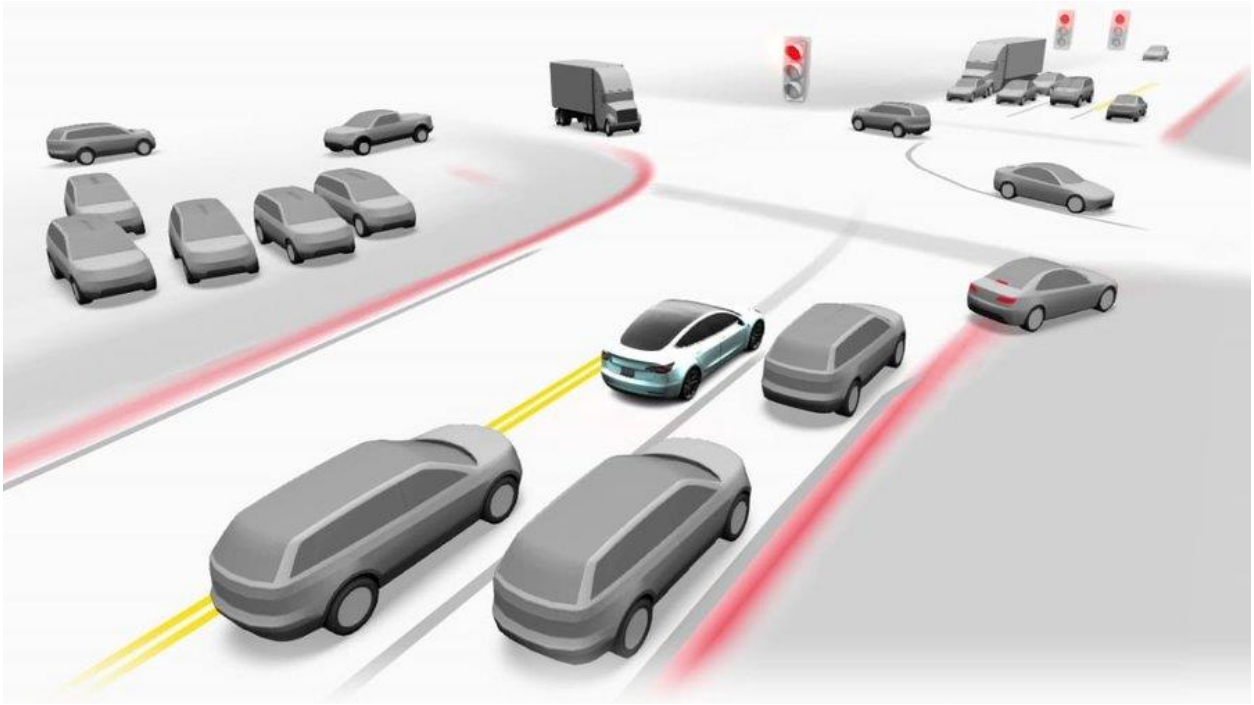


Figure 5: Tesla UI, visual depiction of objects

There are other ways to obtain this data, but accessing object outputs directly from a dedicated perception system is a clean and efficient approach.

So is perception inference actually separate from planning? Are multiple models running? Truthfully, we don't know. We can only look for clues. And that virtual scene depiction is a pretty strong one.

Photon to Control Latency

Driving is essentially a photon-to-control processing task for both humans and Tesla's FSD. Each captures photons through their respective sensors, processes that input, and then actuates controls to execute the appropriate action.

For humans, this process comes with a delay of about 100 to 300 milliseconds between detecting an obstacle and reacting to it. The driving environment— from traffic light timings to speed limits— has evolved to align with this natural human response time.

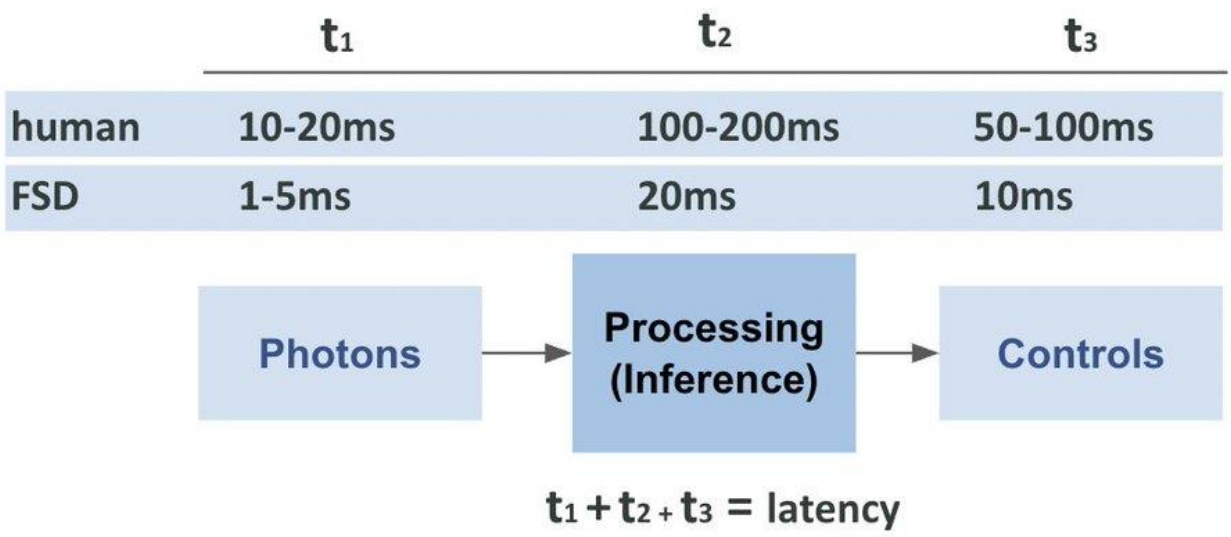


Figure 6: Photo-to-Control Latency

FSD aims for faster reaction times— lower latency from photons to controls. Reduced latency means a safer system, all else equal. The faster the system can respond to changing road conditions, the better its chances of avoiding a collision.

But reducing latency is a tough challenge. Each new version adds more parameters and potentially more input data, both of which increase processing time. And once deployed, a hardware system has fixed processing power— it must operate within those limits. That leaves only a few options: constrain the model, reduce the input data, or optimize inference performance.

Tesla has developed a proprietary neural processing chip (NPU) for its vehicles. Currently, two versions are in the fleet: HW3 and HW4. Soon, Tesla's purpose-built Robotaxi— Cybercab— and future vehicles will be equipped with HW5 (also known as AI5). HW3 is rated at around 72 TOPS (trillion operations per second), while HW4 delivers 150–200 TOPS. HW5 is expected to be 10 times more powerful than HW4, potentially reaching 1,500–2,000 TOPS or more.

As discussed in Part 2, FSD version 13 contains roughly three times the number of parameters as version 12— driving up inference time by about 3x. In fact, version 13 can no longer run on HW3-equipped vehicles. And of course, Tesla is already pushing toward version 14, which will increase the parameter count even further.

Tesla's engineers relentlessly work to optimize performance. One method is **quantization**[1]— scaling floating-point parameters to integers, enabling swift INT8 (8-bit integer) inference over

slower FP32 (32-bit floating point). They also enhance compute efficiency with techniques like streamlining data pipelines, ensuring processors stay fully utilized without delays.

While larger NPUs, like the forthcoming AI5, help address many performance challenges, legacy systems must either be accommodated, upgraded, or end-of-lifed, and of course, the software functionality scales in proportion to the hardware capabilities over time.

While not the only factor, lower latency remains one of the clearest paths to greater safety.

Factoring out LiDAR

There's an ongoing debate about whether LiDAR is truly essential for building a safe autonomous system. Some argue that it offers capabilities vision-based systems lack, while others see it as just another layer of redundancy—adding more data to help complete the perception pipeline.

First, what is LiDAR? LiDAR stands for **Light Detection and Ranging**. It's similar in principle to radar, but instead of using radio waves, it emits rapid pulses of laser light, typically in the near-infrared range, between 850 nm and 1550 nm. By measuring the time it takes for these pulses to reflect off surrounding objects and return to the sensor, a LiDAR system constructs a detailed 3D map of its environment.

On its own, however, LiDAR doesn't "understand" what it's seeing— no more than a raw camera image does. Interpretation requires a perception system, usually powered by machine learning or other AI algorithms, to identify and classify objects like cars, pedestrians, road signs, curbs, and more.

As for limitations, LiDAR doesn't perform well in rain, snow, or fog. Because it relies on light, any airborne particles can scatter the signal and degrade the quality of the returns. Direct sunlight can also lead to sensor saturation or introduce noise, although higher-end systems often use narrowband optical filters and timing-based discrimination to mitigate this.

One common application of LiDAR is "map matching". In this approach, the vehicle uses a pre-built high-definition map and aligns real-time LiDAR data to known features in the map—effectively localizing itself within a known road segment. This is a core part of how systems like Waymo operate. However, this model has scalability issues and introduces safety concerns: it assumes the map is always up to date and accurately reflects the real world, which is not always the case.

LiDAR systems are typically expensive and often have a bulky form factor, making them difficult to integrate seamlessly into vehicle designs. In some cases, their prominent appearance can be off-putting to potential customers—some perceive them as intimidating or even assume they might be harmful, similar to how people sometimes react to visible radar equipment.

Tesla, notably, does not use LiDAR in its autonomous driving solution. Instead, it relies primarily on a **camera-based vision system**, supported by onboard AI running at high frequency. Tesla's system processes raw pixel data to interpret the surrounding environment and estimate spatial relationships between the vehicle and nearby objects. This approach treats vision as the primary input— arguing that, with enough training and compute, it can match or surpass the spatial awareness traditionally associated with LiDAR.

Tesla has effectively "factored out" the advantages traditionally attributed to LiDAR by replicating its spatial awareness capabilities using a vision-only system.

Instead of relying on a dedicated depth sensor, Tesla leans on a high-frequency AI model that processes video frames and learns from massive volumes of driving data. The AI is trained not just on ideal conditions, but also on edge cases and degraded scenarios— such as glare, obstructions, or partial sensor failure— allowing it to generalize and infer spatial relationships even under compromised conditions.

In this model, the neural network itself becomes the critical component that duplicates the functional role of LiDAR, but in a more scalable, cost-effective package. The key differentiator isn't just the sensors— it's the software stack and the volume of real-world driving data used to train it.

This vision-first philosophy is rooted in a simple premise: if humans can drive safely using only vision, a machine trained on enough examples should be able to do the same. It stands in sharp contrast to companies like Waymo, which treat LiDAR as a fundamental requirement for safe autonomy, using it as a ground-truth anchor rather than a redundant aid.

The beauty of Tesla's vision-based approach lies in its ability to offer higher-availability autonomy in a "drive anywhere" package. Tesla's autonomous system can operate anywhere, and its ubiquity across all Tesla vehicles ensures widespread access to these capabilities.

Futures

Today, FSD is in version 13, marking the second major update since the shift to an end-to-end AI architecture. This version has demonstrated the ability to perform fully unsupervised driving— the "holy grail" of vehicle autonomy. June 2025 will debut the first Robotaxi trial, likely alongside version 14, which is expected to be the first fully unsupervised version.

However, it's important to note that FSD is still a work in progress. There's much more to be done before it's truly complete.

For example, FSD must handle parking in all its forms, including complex parking garages and for-pay parking systems. It also needs to integrate user preferences, like preferred routes and restricted areas, into its navigation. The system must be able to navigate through gated

communities, manage High Occupancy Vehicle (HOV) lanes, and adapt to toll roads automatically. It will also need to geofence areas or road segments that are temporarily or permanently unsafe to drive through.

One of the most exciting next steps will be the integration of a voice interface, which will allow for immediate and natural control of the vehicle, offering a seamless user experience.

As we approach unsupervised driving, the challenge lies not just in perfecting the technology but in ensuring it can navigate the complexity of real-world driving scenarios. The ultimate goal is to build a system that not only drives itself but also adapts to the needs and preferences of its users while staying safe, efficient, and compliant with road regulations.

Voice Voice interaction is critical. Whether it's a Robotaxi or a customer-owned vehicle, the user should be able to command and query the vehicle naturally—and it should respond in kind. Voice enables fast, intuitive control, which is essential. Touchscreen interfaces, while state of the art for Tesla vehicles, are slow and cumbersome. Many interactions are time-sensitive.

You should be able to give instant commands: "Don't take a right here, go straight." Or ask: "Why are you heading this way? This area doesn't look safe." Voice allows complex commands or queries to be interpreted and acted on with minimal latency— short of direct vehicle control. In Cybercab, the purpose-built Robotaxi, there are no physical controls; voice will be the primary interface.

User Preferences FSD already includes the notion of user preferences, though the current implementation is rather limited. For example, the Full Self-Driving (Supervised) Profile allows users to choose between "Chill", "Standard", or "Hurry".

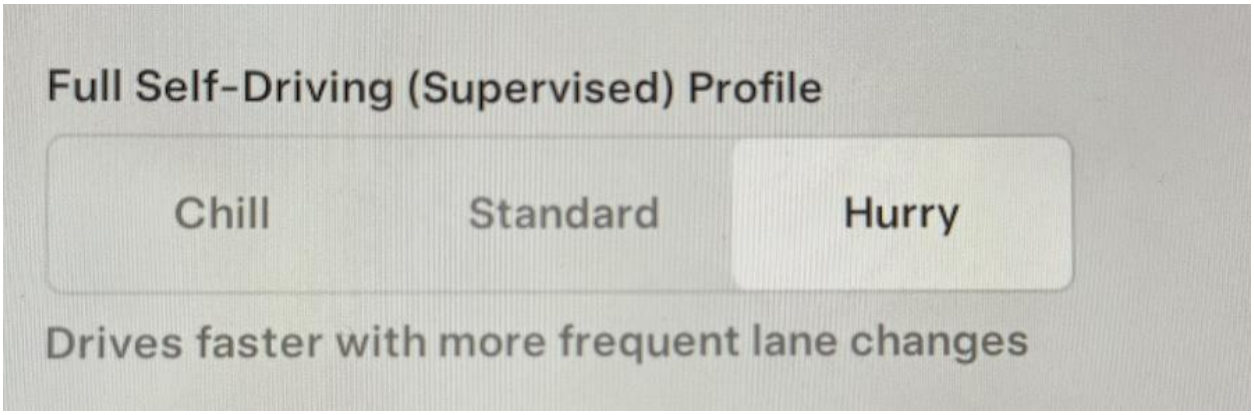


Figure 7: FSD user preferences

So how does this work exactly? In Figure 2 above, the last bucket of input data is *Context Data*. This acts as a kind of memory for the neural network. During inference, a series of data elements are added to the *context window*. Readers might be familiar with this concept from LLMs like Grok or ChatGPT. The context window holds a short-term memory— effectively, the "conversation." When you interact with your favorite AI assistant, a summary of the previous conversation is included in the current prompt.

The context window is limited, but quite large in modern LLMs. For example, Grok's context window can hold about 100,000 words— roughly 200 pages of text. The larger the context window, the more inference compute is required, since the entire context must be replayed with each new interaction.

For the FSD profile, there's likely a single byte that stores the value for "Chill", "Standard", or "Hurry". This preference is stored in the vehicle's flash (permanent) memory but loaded into the context data when FSD is engaged.

Obviously, more context data can be added. Take, for example, a geofenced area— say the user indicates that a certain part of the city is off-limits for FSD. A set of spatial coordinates (GPS data points) can be recorded to describe this area (as a polygon) and pushed through the model during each inference cycle as part of the context data. To reduce context size, only geofence data that's nearby or intersects with the current route would be included.

How is the geofence set up in the first place? That's simply defined via some in-vehicle user interaction.

Another example might be a gated community. In this scenario, there are two lanes: a resident gate to the right (where a license plate reader grants entry) and a guest lane to the left, where the driver must interact with a security guard. When the vehicle running FSD approaches the gates for the first time, it has no idea which lane is appropriate. The user could say, via voice, "Please take the right lane and stop at the gate—it will open automatically." The vehicle could respond, "OK. Do you wish me to take the right lane for future visits?" If the answer is yes, the vehicle records this piece of preference data and adds it to the context data.

With the right context data, a wide range of preferences can be stored and acted upon.

This section is called "user preferences" rather than "vehicle preferences" because, in most cases, the preferences are tied to the individual, not the car. This is especially important in the context of Robotaxi, where users hop from vehicle to vehicle. The key idea is that preferences travel with the user and are applied each time they take a ride.

To summarize:

- User preferences and experience are stored with the user's profile.

- They’re injected into the model’s context window (as metadata, flags, or environment-specific cues) during inference.
- The model itself remains unchanged (read-only), but behavior appears personalized— because the context steers the inference.

Beyond core autonomy, the next generation of FSD will likely emphasize adaptation: to environments, to users, and to the surrounding ecosystem. This means real-time learning pipelines, intelligent fallback behavior based on context, multi-vehicle negotiation, and even emotional intelligence within the cabin. Just as importantly, these systems will need to operate fluidly across different regions and use cases— respecting local norms, adjusting to changing rules, and aligning with the user’s identity over time. The road ahead isn’t just technical— it’s deeply contextual.

Driving Like a Human

Modern roads were engineered for human drivers— not just in terms of signage and signals, but in their physical layout and tolerances. Elements like curve radius, lane widths, and stopping distances are built around human capabilities and limitations. A core assumption in traffic engineering is the average human reaction time of approximately 250 milliseconds all in— the time it takes for a person to visually detect a stimulus, process it cognitively, and initiate a motor response. This constraint shapes everything from signal timing to speed limits. We don’t drive 200 mph not because cars can’t, but because humans can’t safely react fast enough.

Tesla’s Full Self-Driving (FSD) system operates in this human-centered environment. It must interpret ambiguous signage, inconsistent lane markings, and unpredictable human behavior— all in real time, with no special infrastructure or external guidance. FSD isn’t operating in a system designed for machines; it's operating in a system built around human imperfection. Its challenge isn’t just perception or planning, but mastering the nuanced dynamics of a road network that was never meant for autonomous vehicles.

Driving isn’t just physics and geometry— it’s instinct, anticipation, judgment. It’s a dance of imperfect machines operated by imperfect minds. And yet, somehow, it works.

The marvel of Tesla’s approach isn’t that it drives like a robot with perfect math, but that it drives like us— imperfect, adaptive, intuitive. In a world designed for human reflexes and human error, the highest compliment for any AI behind the wheel is that it doesn’t stand out. It blends in. It flows.

Because driving like a human is the immediate goal, but the real ambition is to drive better— creating a system that is safer, more efficient, and far exceeds the limitations of human drivers.

NOTES

[1] In neural network inference, **quantization** reduces the precision of weights and activations—for example, converting 32-bit floats to 8-bit integers—to make models smaller, faster, and more energy-efficient. It maps continuous values to a discrete set, trading some accuracy for improved performance, which is especially useful for deploying AI on resource-constrained devices. Techniques like post-training quantization and quantization-aware training help minimize accuracy loss.

6. Part 2. [(1) phil beisel on X: "The Magic of Tesla FSD, part 2" / X]

The Magic of Tesla FSD, part 2

Tesla is on the brink of revolutionizing the world with autonomy— robots on four wheels (self-driving vehicles) and two legs (Optimus humanoid bot). Its Full Self-Driving (FSD) technology is already in vehicles today[1] and will soon power the upcoming Robotaxi ridesharing platform[2].

The future of transportation is autonomous. The era of manually driven cars is fading, even if it may seem far-fetched. While some doubt self-driving cars are imminent and others fear sharing the road with— or being driven by— robots, Tesla's autonomy aims to surpass even the best human drivers— and fast.

So, how does it work? When it comes to door-to-door autonomy, Tesla isn't just one of the most advanced— it's the only serious contender delivering this magic technology to everyday people right now. While the idea of trusting AI with your safety may seem daunting, it's easier to understand than you might think.

My goal is to break down how FSD works and why you can trust it, with a particular focus on FSD AI (artificial intelligence).

Let's dive in.

Part 2

Back in September 2024, I wrote an in-depth article on Tesla's Full Self-Driving (FSD). It was well-received— with over a million views— and I still consider it one of the best explainers on the topic:

phil beisel

@pbeisel

.

Sep 4, 2024

The Magic of Tesla FSD

In this article I will attempt to explain how Tesla's Full Self-Driving (FSD) works (at a high level) and why it's like magic in a box! In Tesla's latest release of FSD, autonomy is on the path to...

Now, I want to take a different approach. I'll still break down FSD at a high level, but this time, I'll dive deeper into the AI itself. Many see it as "magic"—and in some ways, it is— but let's demystify it.

And I promise, by the end, you'll not only understand it but also feel like you've just learned how the card trick is done!

Overview

"Photons in, controls out".

FSD (Full Self-Driving) is Tesla's autonomous driving system, primarily relying on camera-based vision. Additional inputs, such as audio (for detecting emergency sirens) and vehicle telemetry, complement the camera data. No LiDAR or radar systems are in use.

The sensor data is processed by Tesla's AI-driven FSD system, which determines and executes vehicle controls— including throttle, braking, and steering— to navigate the vehicle along its intended path.

The FSD AI

Artificial intelligence (AI) represents a transformative computing paradigm, with its principles— long in development— now driving an explosive revolution. At its core is **machine learning**, which allows systems to learn from *data* without explicit programming. A type of machine learning, **neural networks** are modeled after the human brain. These networks, which drive breakthroughs like image recognition and speech processing, are the foundation of self-driving systems such as FSD, enabling them to continuously improve by learning from vast datasets and real-world feedback.

Neural Network A neural network is composed of **nodes** (or neurons) that process information. These nodes are organized into layers: the **input layer**, where data enters; the **hidden layers**, where the network processes and learns from the data; and the **output layer**, which generates the final result or prediction.

Each connection between nodes has a **weight** that determines the influence one node has on another, and each node has a **computed value** and a **bias** that adjusts its output. The weights and biases are collectively known as the **parameters** of the network. During training, the

network adjusts these parameters to learn patterns and improve its predictions as data moves through the layers.

FSD (Full Self-Driving) is an example of a neural network. It likely uses multiple input layers, each feeding into separate branches with their own hidden layers, which then converge into a final shared hidden layer before producing an output layer.

The **input layers** likely consist of:

- **Camera data**: Each vehicle has 8 cameras. Assuming 3 frames per "cycle," with each frame having a height of 1280 pixels, a width of 960 pixels, and 3 channels (red, green, and blue), this results in 88,473,600 values in total.
- **IMU/GPS data**: Values for latitude, longitude, altitude, yaw, pitch, roll, speed, and acceleration— 8 values in total.
- **Vehicle state data**: Values for speed, steering angle, throttle, brake pressure, and gear— 5 values in total.
- **Audio data**: Values encoding microphone data at some sampling rate per "cycle." Perhaps 250,000 values in total.
- **Contextual data**: Values for time of day (e.g., hour, minute) and weather (individual values for rain, sun, fog, etc.). Less than 10 values in total.

To simplify, let's focus on the camera data. This data forms a multi-dimensional array of values, known as a **tensor**, with the following shape:

[8 cameras, 3 frames per camera, 1280 pixels in height, 960 pixels in width, 3 RGB channels per pixel]

This is a **5-dimensional tensor** with **88,473,600 values** (8 × 3 × 1280 × 960 × 3). While it's easy to visualize 1D, 2D, or 3D data structures, it becomes more abstract when working with higher-dimensional tensors like this one.

The **output layer** defines what the FSD system *does*. For autonomous driving, it typically predicts control actions: steering angle, throttle, brake pressure (and maybe blinkers and horn).

- **Steering**: 1 value (e.g., -1 to 1 radians)
- **Throttle**: 1 value (e.g., 0 to 1)
- **Brake**: 1 value (e.g., 0 to 1)
- **Blinker**: 0=none, 1=left, 2=right.

- **Horn**: 0=no sound, 1=horn sound

This would be a 1-dimensional tensor with 5 values, for example:

[-.2, .15, 0, 0, 0]

e.g., turning slightly left with a throttle of 15%; no brake; no blinkers; and no horn.

Now the **hidden layers**. There are many and perhaps a few per branch, ultimately resulting in a fusion layer and some dense layer, that is a "compressed" representation of all inputs.

What are the sizes and dimensions of these hidden layers? Let's just say "large" without further color, but certainly not bigger than the input layers. There are many hidden layers, perhaps 10 or more.

Each **node** in a layer produces a value (its activation) and is connected downstream to every node in the next layer. These connections will eventually have values associated with them called **weights**.

Consider the above our definition of the FSD neural network— it's a simplified design, and we skipped many details.

Let's simply further to the neural network depicted in Figure 1:

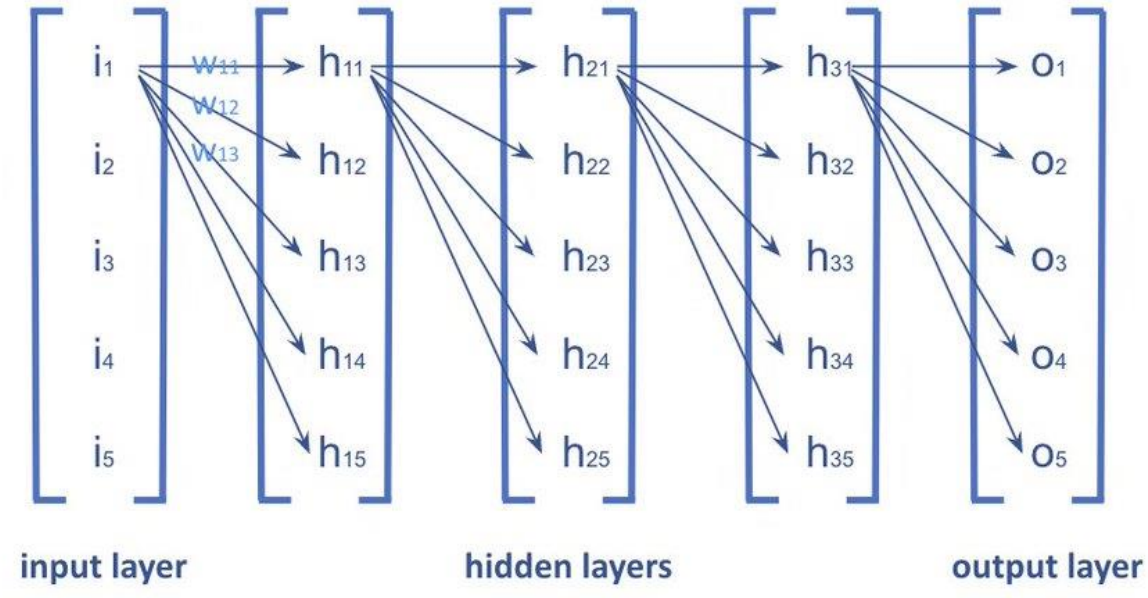


Figure 1: Example Neural Network

This neural network has an input layer (with 5 nodes), 3 hidden layers (also with 5 nodes each), and an output layer (with 5 nodes).

Each node in the input layer is connected to each node in the adjacent hidden layer (these connections are labeled w11, w12, w13, etc.). They are the **weights**. We are only showing the connection lines from the top nodes of each layer, but do understand that *every* node is connected to every downstream node!

Each node (represented as h11, h12, etc.) has *two* associated values: the computed value or activation (a11) and the bias (b11). In other words, h11 has a11 and b11; h12 has a12 and b12, etc.

Let's pretend our input data, the camera data, can fit in this input layer. Of course, it cannot, there are only 5 input values and for our paired down input of *just* camera data we would need 88,473,600 values.

So Figure 1 represents the *architecture* of the FSD neural network. At present there is no data in it.

Training Neural networks are trained on input data— that's how they learn. The hidden layers' weights and biases are initialized to random values, then training kicks off.

For FSD, each training sample is camera frames, vehicle state data, etc. (as discussed above), labeled with the correct output by a human. This is supervised learning[3]— the answers are known! Millions upon millions of data samples are used. Each sample is passed through the network, and the weights and biases get modified.

Each hidden layer node processes the data using a matrix multiplication operation (combining inputs with weights and adding biases), followed by an 'activation' function to determine its output (the specific details don't matter here). Then, **backpropagation**[4] steps in: it calculates the **loss** for that sample— how far off the output is from the known answer— and adjusts ("nudges") the weights and biases to reduce that loss.

There's a set of data samples held aside, called the **validation set**. During or after training, these samples are run through to calculate the loss. No modifications are made to the weights and biases here— it's just to check we're converging on good answers.

The whole point is that the neural network, having been trained on the data, will become *highly predictive* of the correct outcome— the right answer, if you will. The data builds the *function*, and the purpose of that function is to take any data sample and predict the proper output. That's the process called **inference**, which we'll discuss below.

The final output is called the **model**— the model is the "filled in" weights and biases. Once the model is produced and validated, it is read-only and ready to use for inference (see Figure 2).

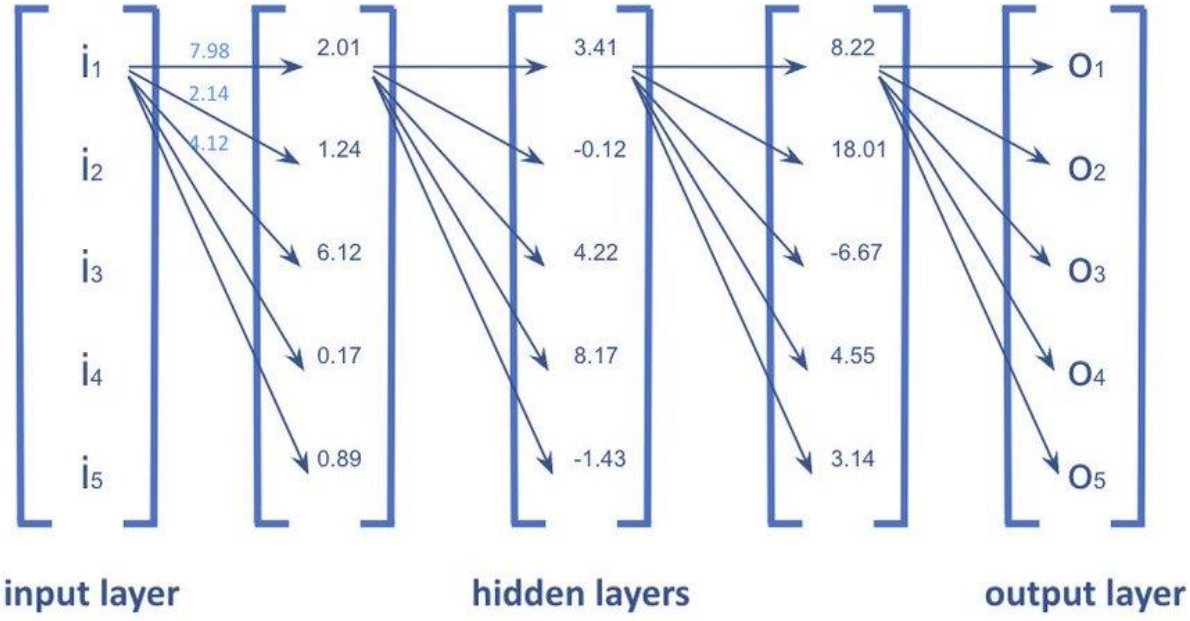


Figure 2: Neural Network Model with weights and biases

The model can, of course, change in a subsequent version (e.g., 13.2.8 → 13.2.9).

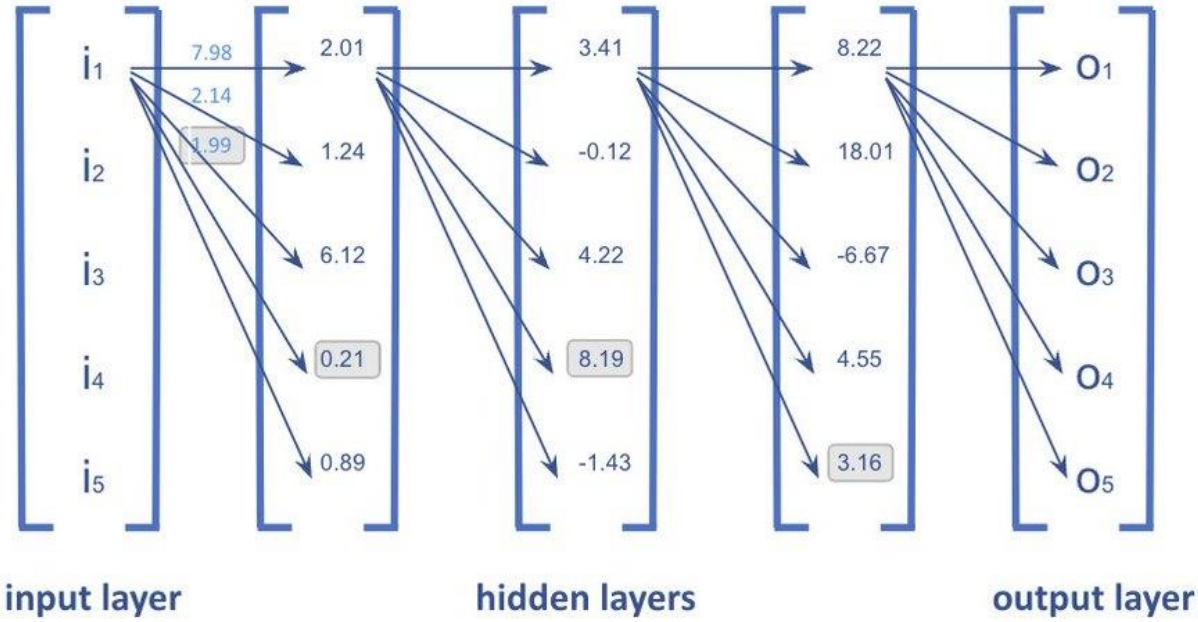


Figure 3: Changes to the model

Figure 3 demonstrates how a small number of weights and biases are modified following the processing of new training data (although in practice, most are altered, Figure 3 provides a simplified view for illustration). In this case, the model is not undergoing continuous training but rather **fine-tuning** to correct a specific issue in its output.

For example, imagine that in version 13.2.8, FSD is misjudging a curb—steering too close to it, leading to a driver intervention. Tesla identifies this issue, gathers new training data, and pushes it through the training process. This fine-tuning "nudges" certain weights and biases, resulting in a corrected model where the over-steering behavior is eliminated.

It's not necessary to know exactly which weights or biases were modified— in fact, predicting these changes is nearly impossible. What matters is that the updated model now functions correctly. Tesla verifies that the intended fix (eliminating the over-steering) is successful while also ensuring that the parameter changes do not introduce regressions (unintended failures in previously working behavior).

The training process may take several months and requires an enormous amount of computing power. It is performed using hundreds of thousands of graphics processing unit (GPU) processors. The process relies heavily on parallelism[4]: each GPU accelerates training by parallelizing common mathematical operations, while the massive cluster of GPUs further distributes the workload across the system.

Inference Once a model is trained, it is ready for use. Tesla versions the model (e.g., 13.2.8) and includes it in an over-the-air (OTA) update for the customer vehicle fleet.

Inference is the process of *using* the model. While running FSD, input data (as described above) is fed into the model to produce outputs. These outputs control the vehicle— throttle, braking, and steering.

Inference happens repeatedly at high frequency, typically 20 to 50 Hz (20 to 50 times per second). Each output modifies the vehicle's behavior; for example, if another vehicle begins to move into its lane, the system may decide to decelerate by braking or reducing throttle.

Inference is less computationally intensive than training but still requires significant processing power. Unlike training, inference does not modify the model— it simply reads from it.

Tesla vehicles use a specialized neural network processor, designed in-house, to efficiently run inference on the vehicle's onboard Full Self-Driving (FSD) computer, known as Hardware 4 (HW4) or Hardware 5 (HW5, also called AI5) in newer models. This processor functions similarly

to a GPU but is optimized for the inference tasks specific to processing the neural network model. While HW4 is currently in use, AI5 is forthcoming (likely to debut on the production Cybercab and all future Tesla vehicles).

1 or 2 Neural Networks?

The discussion above describes FSD as a single neural network, but this may not be accurate. In my previous article, I discussed two distinct neural networks: one for **perception** and one for **planning**.

Perception is the process by which the vehicle identifies all static and dynamic objects in its field of view, including people, other vehicles, signs, and road markings. Planning, on the other hand, determines how the vehicle should respond— controlling the brakes, throttle, and steering accordingly.

There is evidence suggesting Tesla employs two distinct neural networks (one feeding the other). Prior to FSD version 12, AI was used only for perception, while planning was handled algorithmically (through 300,000 lines of C++ code). Version 12 introduced "end-to-end" AI for FSD, making it possible that Tesla added a planning neural network and linked the two together.

Additionally, the Tesla UI visually represents a virtual view of the roadway ahead, which implies that the neural network must include the fundamental perception objects in its outputs.

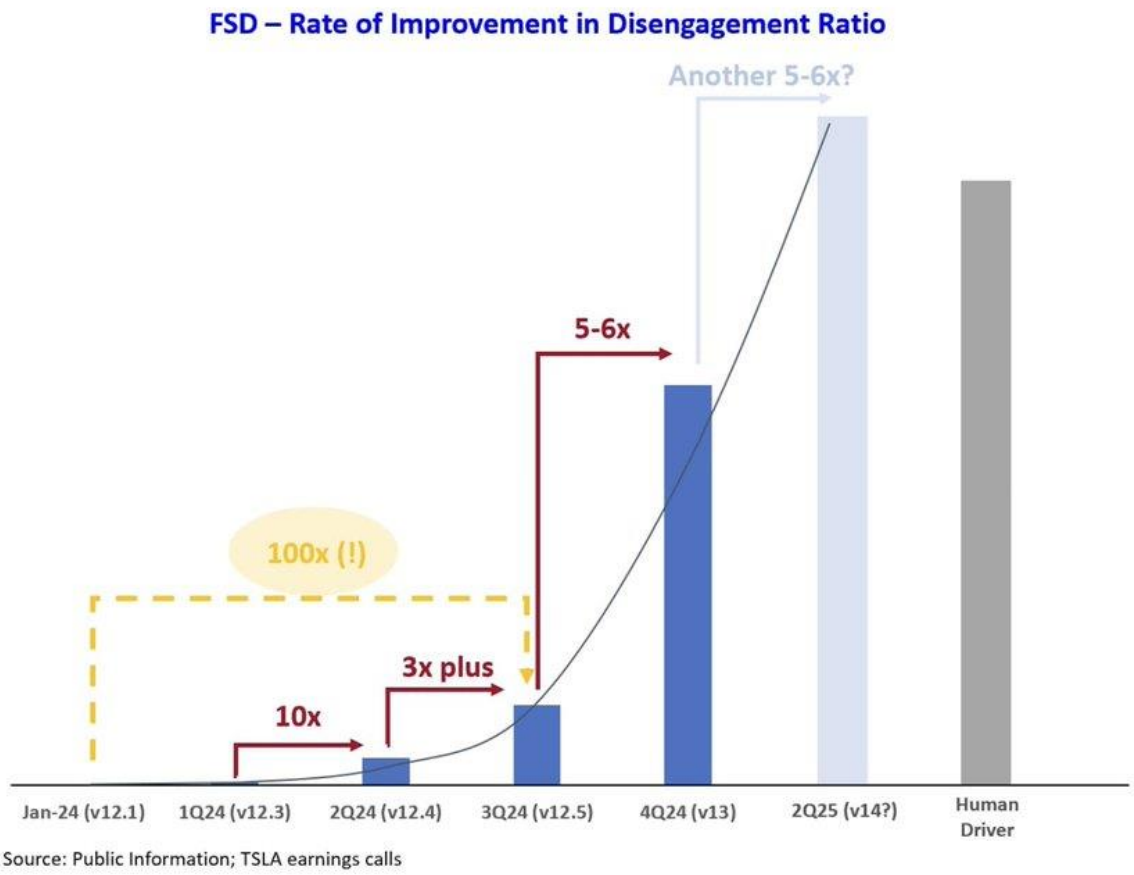
Regardless of whether these are distinct neural networks or combined into one, the discussion is sufficient to describe Tesla's FSD AI at a high level. Please read the first article to learn more about the perception and planning aspects, as both are most assuredly taking place.

Convergence

FSD is on a path of convergence— *autonomy is being solved***.** Tesla has signaled this through both safety data and accelerated investment.

Each major AI model release (e.g., v12, v13, and soon v14) yields an order of magnitude improvement in safety— a process we might call the "March of 999's," referring to the steady increase in reliability toward near-perfect safety (e.g., 99.9%, 99.99%, etc.).

Version 14 is expected to surpass human driver safety, likely serving as the foundation for Tesla's initial Cybercab trial.



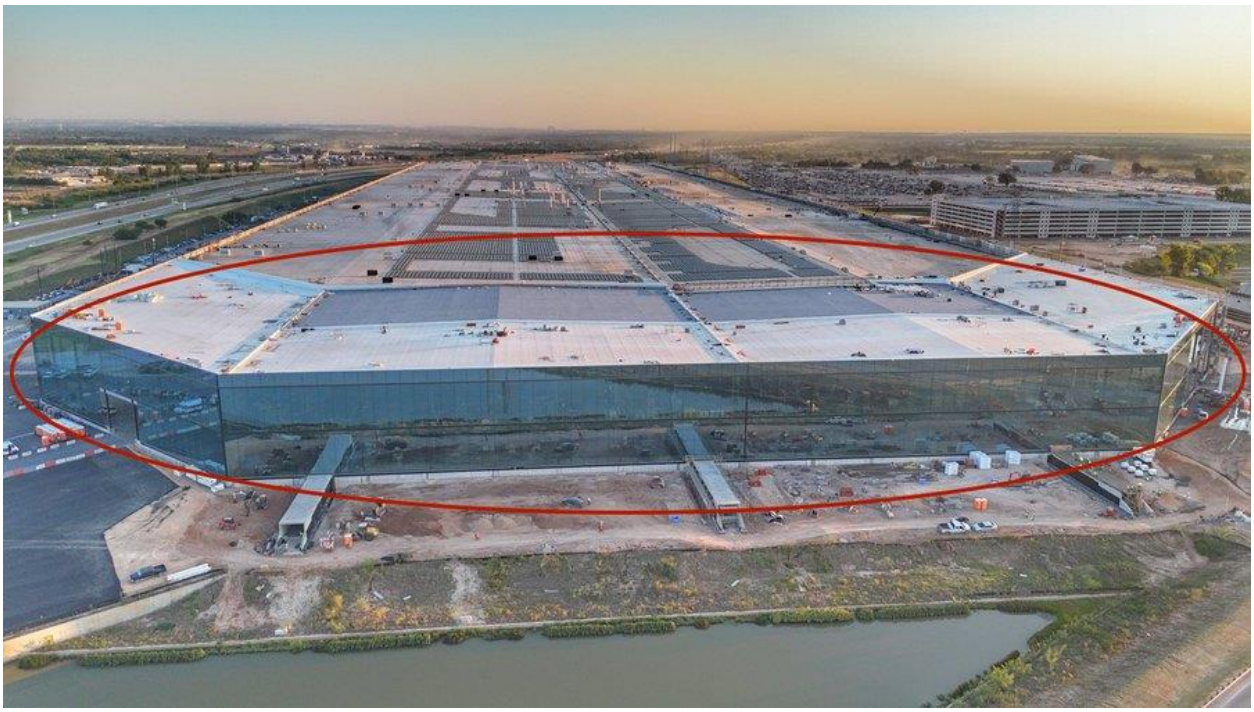
At this stage, achieving autonomy is merely a **compute-bound problem**, meaning the challenge is primarily limited by the available computational power, both in training and inference.

Each new model is significantly larger— more parameters and more training data— leading to longer training times.

It was reported that version 13 is 3× larger (3× more parameters) than version 12**,** implying (without optimizations) that training requires 3× more compute— not even accounting for the increase in training data.

This training is happening at Tesla's Cortex AI supercomputer, located at the Gigafactory in Austin, Texas. Cortex's growth has been rapid, with its initial 50,000 Nvidia H100 GPUs powering on in Q4 2024. Plans call for expansion of the GPU cluster, initially by 3×, to meet the surging compute demands of ever-larger models and increased training data.

Cortex expansion, Tesla Gigafactory Texas



A glimpse of Cortex: one GPU-packed hallway in the supercluster.

Magic Box of Numbers

When you think about FSD, it's conceptually reasonable to imagine it as a magic box of numbers— a whole lot of numbers. Estimates suggest that version 13 contains between 500 million to 1 billion weights and 5 million to 25 million biases.

But it's not magic.

In fact, the numbers in this neural network are relatively straightforward to compute. However, there is a significant amount of "secret sauce" in how Tesla has structured the network: the specifics of the hidden layers, the format and content of the input data, the exact nature of the output, etc.

What's important to understand is that these numbers replace all human-written code[6]— the algorithmic approach. It's convenient and somewhat intuitive to think there's a piece of code interpreting the camera data and deciding how the car should respond to obstacles, etc. *But that's not what's happening at all.* This box of magic numbers contains the learnings of all possible driving scenarios (at least those in the training set). Any scenario not included can simply be added to a future version.

Imagine "fine tuning" an existing FSD model to fix a problem. The result of the new training is just a set of changed values— a small tweak might nudge 100 million weights and 5 million biases to different values. It's often said that AI is nothing more than the *function* being derived from the data. In the case of FSD, "the function" is to take the input data and determine how to steer, brake, and accelerate safely.

And while it's more intuitive to think of the coded approach, in the end, both just result in numbers. The code converts to numbers that the CPU can interpret as logic and data manipulation. And of course, the neural network is just numbers, connected in a very specific way.

Now you're armed with the knowledge of what's in this magic box of numbers!

Zero to One

Peter Thiel's *Zero to One*[7] explores the concept of building successful startups and driving innovation. Thiel argues that true innovation comes from creating something entirely new—moving from zero to one—rather than simply improving upon existing ideas.

Tesla's Full Self-Driving (FSD) technology embodies this "zero to one" approach at scale.

Tesla redefined autonomous driving by prioritizing cameras as the primary input, abandoning the traditional reliance on sensor fusion. This not only simplified the technology but also reduced both complexity and costs.

Tesla committed to the bold idea that an "end-to-end" AI solution would ultimately provide the key to achieving full autonomy. This revolutionary approach has the potential to transform the world, paving the way for a new era of transportation-as-a-service and fundamentally reshaping the automotive and transportation industries.

I hope this article (and the previous one) equips you with a clear understanding of how FSD works— the underlying AI— so that as this technology reshapes your world, you'll feel empowered by this "under the hood" knowledge.

NOTES

[1] Full Self-Driving (FSD) is an optional feature available for purchase on compatible Tesla vehicles. As of March 18, 2025, the latest FSD version is **13.2.8.** FSD, including version 13.2.8, currently requires driver supervision, meaning the driver must stay attentive and prepared to intervene at all times. Version 13, including 13.2.8, is designed to operate on Tesla vehicles equipped with Hardware 4 (HW4), while version 12 remains compatible with Hardware 3 (HW3) vehicles, though HW3 support for newer versions may vary based on Tesla's ongoing development and announcements. Compatibility and feature availability can depend on the specific vehicle model, hardware, and regional regulations.

[2] Robotaxi is Tesla's upcoming autonomous ridesharing service, where self-driving Teslas will operate as driverless taxis. Robotaxi will begin trials this summer starting in Austin TX. A recent article I published is a fun read: **phil beisel** @pbeisel · Mar 11

Robotaxi Drive-by

Robotaxi is set to revolutionize transportation with a seismic shift that goes far beyond ride-hailing competition. Its scale and cost dynamics will dismantle personal car ownership, public transit...

(1) phil beisel on X: "Robotaxi Drive-by" / X

[3] Supervised learning uses data that has been manually labeled by humans or predefined systems, meaning each input comes with a known correct output. This labeled data serves as a guide for the model to learn patterns and make accurate predictions on new, unseen data.

[4] Backpropagation is a learning algorithm for neural networks that adjusts weights by propagating the loss (or error) backward from the output layer to the input layer. It calculates how much each neuron (node) contributed to the loss and and updates weights and biases

accordingly to minimize it. It is called backpropagation because this correction flows in *reverse* through the network, allowing earlier layers to learn from mistakes.

[5] Learn more about accelerated computed from my article: **phil beisel** @pbeisel Oct 2, 2024

Accelerated Computing

Starting in late 2023, accelerated computing became part of the mainstream technology lexicon, and for good reason. The world took notice of this pivotal technology trend, which emphasized speed,...

(1) phil beisel on X: "Accelerated Computing" / X

[6] FSD still relies on human-written code for many aspects of its architecture and operation, primarily in the inference engine. This code is responsible for feeding input data, organizing AI computations on the neural processor, and managing the outputs. However, the critical point is that the logic driving the vehicle's behavior is determined by the AI— the trained neural network that make real-time decisions based on the sensor data.

[7] Peter Theil, Zero to One

6. Part 1. (1) phil beisel on X: "The Magic of Tesla FSD" / X

The Magic of Tesla FSD

In this article I will attempt to explain how Tesla's Full Self-Driving (FSD) works (at a high level) and why it's like magic in a box!

In Tesla's latest release of FSD, autonomy is on the path to being solved. Introduced in January 2024, version 12, with its end-to-end artificial intelligence approach, represents a bold leap forward from its predecessors.

Vehicle autonomy will profoundly transform transportation-- enabling transportation-as-a-service at scale, eliminating the monotony of challenging commutes, and making our roads safer.

The Problem

For the past 40 years, computer scientists have been struggling with the vexing problem of autonomous driving of vehicles. It's a *very* hard problem-- driver scenarios are complex, highly dynamic, and infinite in their presentation.

The first part of the problem is understanding the road configuration and its state-- this is called **perception**. As humans we do this primarily with our eyes (and to some extent our ears). A

vehicle can do this with cameras and other sensors like radar and lidar. But gathering data from these sensors does not imply understanding-- what are the cameras seeing? In any given segment of roadway there can be hundreds of different objects-- some static, others dynamic (in motion). And from one encounter to the next, this road segment can change.

Take for example an intersection of two roads. There are road markings-- lane lines, crosswalks, lane markings like turn arrows, etc. There can be stop or yield signs (or signs that have other complex meanings e.g., right turn on red except from 8am to 5pm M-F). There can be traffic lights, functioning or in some error state (flashing or no power). There are moving objects-- other vehicles, pedestrians, bicycles. There can be transient elements (e.g., rain, standing water, potholes, etc.); there can be construction (e.g, cones or blocked lanes).

The second part of the problem is the decision process-- the vehicle must decide what to do; this is called **planning**. It must decide what controls (e.g, accelerator, brakes, steering) to apply to navigate the immediate road segment and also how to proceed to reach the stated goal (the eventual destination). Part of this is collision avoidance-- the vehicle must not come in contact with any static or dynamic object. And part of this is directional control-- what lane at what speed.

Early Attempts

Before the emergence of artificial intelligence, the approach to both the perception and planning problems was *algorithmic*. The programmer would write code to both perceive the environment and plan the vehicle's movement.

For example, if we assume the input data is only camera data, a programmer would have to write very complex code to find and categorize objects within digital images captured by the camera(s).

Images are a series of pixels arranged in rows (Y-axis) and columns (X-axis) and colors (Z-axis-- a 32 bit integer representing the RGB value). So a stop sign is just a series of pixels in various shades of red with white pixels in the interior spelling the word STOP. Various algorithmic approaches (e.g., Canny Edge Detector, Sobel Operator, etc.) can find a stop sign within an image-- but it's very complex and overly brittle. Stop signs that appear partially occluded might be missed.

Writing mountains of code, even with advanced algorithms, to find objects in images is error prone and ultimately an exercise in futility.

An algorithmic approach to planning is equally complex and error prone. The programmer must write code for every conceivable condition-- a huge decision tree (*if* this happens, *then* do that).

The long tail of driver scenarios makes it a nearly impossible task. *Version 11 of FSD employed an algorithmic approach to planning.*

A New Model: Software 2.0

The algorithmic approach discussed above is all about traditional software development. In this model, called *Software 1.0*, programmers write code to solve the problem. The focus is on creating deterministic systems where inputs are processed to produce predictable outputs. Unfortunately this model has limited flexibility and scalability, requires significant effort to handle edge cases (it is *brittle*), and can be complex to maintain as the system grows.

For vehicle automation, one of the most complex software problems, the Software 1.0 approach simply does not work.

But luckily a powerful new model has emerged that we call *Software 2.0.* Software 2.0[1] represents a shift towards ***data-driven development***, where the software learns from data rather than being explicitly programmed. Instead of writing rules, developers create *models* that learn patterns from large datasets. These models[2] can generalize and make predictions based on new inputs.

With enough data and training compute power, the Software 2.0 model can be *highly predictive*. Such a model provides greater flexibility, adaptability, and potential for handling complex, non-deterministic tasks. *The system improves as more data is provided.*

The Software 2.0 approach to achieving vehicle autonomy is based on artificial intelligence techniques, specifically machine learning. Machine learning, a subset of artificial intelligence (AI), involves creating models that learn from large datasets and make decisions or predictions based on that data.

Machine learning starts with data. In vehicle autonomy this is largely video clips (and possibly radar and lidar data). The quality and quantity of data directly influence the performance of machine learning models. Instead of being explicitly programmed to perform a task, machine learning algorithms use statistical techniques to learn from examples. This means they identify patterns or rules from the data provided during the training phase.

A specific call out is *supervised learning*. The model is trained on labeled data sets, where the desired output is known. It learns to map inputs to outputs, and can then predict outcomes for new, unseen data.

Tesla FSD: Version 12

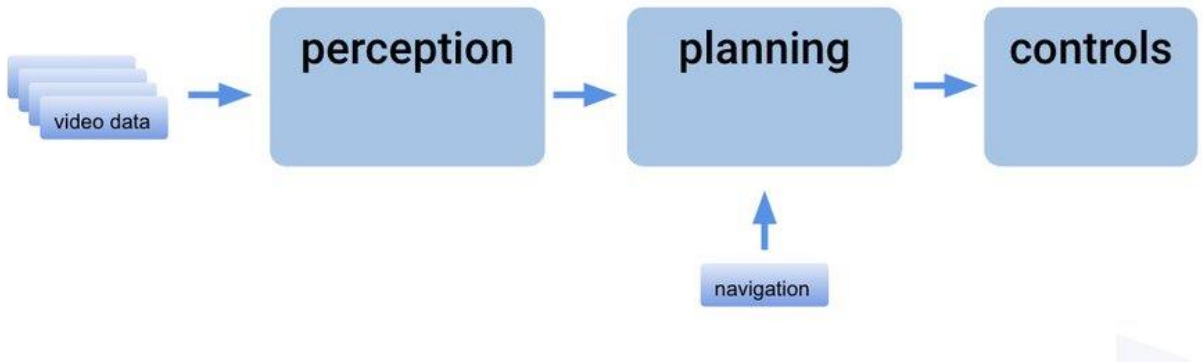
Tesla's FSD is a *Software 2.0* architecture-- it is a data-driven approach. FSD uses only video data; no other sensors are used for the *key* elements of the solution. As Elon Musk is fond of saying "photons in, controls out".

Described below is an approximate and high-level description of the architecture and inner workings of FSD. Tesla has created a proprietary solution and shares few details outside its walls.

As mentioned at the outset, version 12 is a great departure from Tesla's previous versions. Other than some orchestrating code of the framework, *there is no code*-- it is all AI inference (discussed below).

Below is a block diagram of the execution body of FSD on the vehicle. At a very high-level:

1. Video data from 8 cameras is fed to a ***perception*** process.
2. The perception process determines *what* it is seeing.
3. The ***planning*** process takes the perception data, along with the overall navigation goal (the vehicle's route to its destination) and determines *how* it should move forward.
4. The plan is fed to a ***controls*** module that combines steering, acceleration, and brakes to drive the vehicle.



Block diagram of FSD on-vehicle architecture

Training vs. Inference Tesla uses a machine learning technique called a *neural network*[2] to implement Full Self-Driving (FSD). Machine learning involves an *asymmetric* architecture with two distinct processes: training and inference. Training, which occurs in a data center (the cloud) once (per version), is the most compute-intensive part of the process. During training, data is used to build a *model*. Once the model is created, it is transferred to the vehicle (part of the FSD software release). The second process, *inference,* is then executed (by running data through the model) in the vehicle *repeatedly* to drive the car.

Training FSD training builds a model for perception and a model for planning[3]. Both models are constructed using a vast amount of video data from the fleet (customer's vehicles). Every (capable) Tesla vehicle contributes data to the training process in the form of small video clips from the vehicle's camera sensors (these are likely H.265 formatted video clips, each a few seconds in length) along with IMU (inertial measurement unit) and GPS data. The data is uploaded to Tesla's cloud using either cellular or (preferably) Wi-Fi. There is no need for the data to arrive in real-time, it can be batched and sent at a later time.

There is a lot of video data generated from customers' vehicles (there are about ~5 million capable vehicles in the fleet). Obviously Tesla must employ some on-vehicle logic to determine *what* data to select and *when*. It is important to note the FSD inference process is likely running at all times (even when the customer does not have it explicitly engaged). In this *shadow mode*, it can aid in the selection of critical video data; for example, it may select video data for just the segments of the drive when the human driver deviates from the plan (course) that the FSD process would select.

When the training process completes, the models are validated. The validation likely involved simulation-- where the inference process can be run against known validation datasets in the cloud. Having passed this type of validation, the models are likely further validated by employees using actual vehicles on fixed and ad-hoc routes. From one version to the next, Tesla is looking for improvements and no regressions[4]. If the models are validated, they are rolled out to the customer fleet.

Inference The inference processes (described below) take place in the vehicle. All Tesla vehicles, starting with Hardware 3 (HW3)[5] have Tesla custom silicon that includes *neural processing units*. These NPUs have custom operations (e.g, ReLU, Sigmoid, Tanh, etc.), similar to GPUs but more specialized, for common AI inference operations.

Perception About half of the FSD problem is *perception*-- understanding the vehicle's environment.

Tesla FSD identifies every object in each camera's field of view. The objects are labeled (with one or more attributes), given a size, location and vector representation-- a magnitude (speed) and direction. Fixed objects (like signs, etc.) have 0 magnitude, of course, but some objects are in motion (e.g, a human might be walking at 2 mph across the road). The road itself is a type of object, known by its size and foreseeable length, and any lane markings (e.g., divided roadway, shoulders, etc.). It's important to note that with Tesla FSD only cameras are used; there is no other primary sensor input.

This process of identification is pure inference-- video clips are evaluated by the perception inference engine, using the model (constructed in the training process). The output is object

identification. Each object is given a probability of confidence-- for example, an object may be identified as a stop sign with 92% confidence.

But what if the object is scored at a very low probability of confidence, say 10%? This is the magic of machine learning-- specifically *supervised* machine learning. Supervised learning means a human is involved in the process[6]. Images that have objects with a low probability of confidence can be sent back to Tesla's cloud for human inspection-- and a human then can label the object as either a stop sign, something else, or discard it as a valid object. The labeled representations can then be fed back into the next iteration of the training process so that objects like this will now score with a much higher probability of confidence.

So imagine how this process unfolds iteratively over time using 'stop sign' detection as an example. At the start the system is trained on thousands of images containing signs, some of which are stop signs. This is data labeled by a human. It attempts to make an educated guess (based on its own internal statistical weights) which are stop signs and which are not. If it guesses wrong it updates these weights to make a better guess next time.

After training on all the available labeled data, a validation step is conducted. Here images containing various signs are provided, and the system again guesses which contain stop signs and which do not. If the guesses are correct the model is valid.

In the field (on vehicle) when images containing stop signs are identified with a low probability of confidence these images are flagged and make their way back to the next training iteration. If a low probability of confidence 'stop sign' is indeed a stop sign it is labeled as such. And training on this stop sign will update the internal weights of the model to better guess this type of stop sign in the future.

The more images containing stop signs (think of all the variations one might encounter in the real world), the better the system becomes.

Planning Planning is the step that determines what the vehicle should do.

Planning is where the magic of FSD lies (and why version 12 is a breakthrough), and not surprising it's the hardest to describe in terms of how it works as the details are a closely guarded secret.

To effectively build a model for planning its clear that good driver scenarios should be selected for training. Driver scenarios where the driver is unsafely or incorrectly proceeding along the route are pruned from the training data. The idea is to select good driver data-- not too aggressive, not too passive, and correct for the situation. Some pruning is easy; for example, a driver operating the vehicle 25 mph over the speed limit.

Ultimately the idea is that a vehicle operating under FSD *mimics* human drivers-- good ones. Essentially this is what Tesla has done by using actual driving data from humans-- it mimics the aggregate behavior of all the good drivers.

As described above (the stop sign perception example), the training process starts with a lot of labeled driving scenarios. The process makes an educated guess as to the planned goal (e.g., move forward at the current speed, start slowing down, begin moving to another lane, etc.) and verifies this against the properly labeled examples. With enough data a proper plan can be derived.

One important aspect to understand, however, is that Tesla does this for *like scenarios* not exact scenarios. It does not train the model for specific intersections but rather for intersections similar to the training data. The process of selecting *like scenarios* is complex-- one can imagine it might use the perception model to determine *like objects* (e.g., where one intersection configuration is similar to another).

Much like perception, once the model has been constructed it is transferred to the vehicle where inference takes place. Inference takes input data from the perception process and outputs "a plan". The plan takes into account the route (the ultimate goal).

It is important to understand "the plan" is nothing more than instructions to the controls block (discussed below). Planning likely takes place 15 times a second or more. The vehicle must adjust its plan often to deal with changing road dynamics. For example, imagine the vehicle is driving at 60 mph in lane 1, it begins to pass another vehicle in lane 2. Just at that moment, the vehicle in lane 2 starts to come into lane 1. The plan must adjust immediately to avoid a collision (and it does). In fact, FSD's attention and reaction time are far greater than the best human drivers (there is no distraction and no 2nd guessing).

Controls The last processing block in Tesla FSD is *controls*.

The *controls block* is responsible for actually driving the car. It actuates four key functions: steering, acceleration, braking, and turn signals. Commands are received from the planning block. It is likely that these commands contain a priority indication to specify whether the function must be employed immediately or should be integrated over time. For example, 'brake to slow' vs. 'brake now'. The controls block manages this smoothing (and may take data in real-time from the IMU to assist[7]).

The control process is likely inference too.

The Tesla FSD Moat

Version 12 of FSD is end-2-end AI. As a Software 2.0 architecture, it is a data driven approach. It is data that builds the functional logic of the perception and planning processes-- billions of video clips. And the more data, the more it improves.

Tesla's moat is widening in two areas, data and data processing.

Tesla adds about 5,000 data providers to its network every day-- namely 5,000 Tesla vehicles. Every vehicle is capable of sending video clips from its 8 cameras-- and, while operating in FSD shadow mode, this is likely happening.

To process this data-- to train new versions-- Tesla is building massive data centers and compute platforms. Two noteworthy build-outs are the Cortex supercomputer at Tesla's Gigafactory Texas in Austin and the Dojo supercomputer in Buffalo, New York.

Cortex is based on Nvidia's H100/H200 GPUs, with plans for around 100,000 units. The power requirements for Cortex range from 80 to 100 MW, that is roughly the power required for 80,000 homes.

Tesla is also developing its own Dojo[9] supercomputer at the Riverbend Gigafactory in Buffalo, with an initial build cost estimated at $500 million. Dojo is a ground-up design by Tesla, centered around a custom silicon GPU (NPU) called the D1. The system features a custom electrical architecture engineered for high performance while minimizing power consumption.

It is estimated that Tesla will spend $10B on AI computer infrastructure in 2024.

That is a wide moat!

Humans vs. Machines

Ultimately why does anyone think a human can operate a vehicle better than a machine? *Humans are terrible drivers.* There are over 40,000 deaths each year on US highways, most of them caused by human driving error. That is the equivalent of one fully loaded 737 crashing and killing all its passengers every other day.

Humans are slow processors and often distracted. Distraction is simply another way of saying the input sensors are blocked periodically or the processor is busy doing something else. Machines don't get distracted and they can process (and decide) much faster than humans-- its likely the planning portion of Tesla's FSD is operating at 15 hertz (15 times a second) or better.

Beyond inherent instincts that help us avoid obvious dangers, humans learn to drive primarily through experience. The more we drive the better we get. Although new drivers often have quick reflexes due to their age, they are not yet safe drivers because they lack experience in real-world driving situations. Data driven autonomous systems like FSD get better with age too-- but at a *much* faster rate. And they learn from the *collective experiences* of the entire driving

fleet. It's like plugging in the experience of *all* drivers into the brain of your 16 year old on day one (and then updating it often).

The Future

Tesla's FSD will usher in the age of vehicle autonomy—vehicles as robots. This will have a profound impact on transportation and society.

Tesla began offering early access to beta versions of FSD in October 2020 (known as "FSD Beta"). Version 12, the breakthrough version described in this article, was not released until January 2024.

With the release of Version 12, progress will accelerate rapidly.

In a few weeks (October 10th, 2024), Tesla will unveil its Robotaxi service. Robotaxi is Tesla's TaaS (transportation as a service) platform based on FSD. It includes a new type of vehicle designed explicitly by Tesla for this purpose (no driver controls, etc.) and a service that allows existing Tesla owners to 'loan out' their vehicles for rideshare. At first glance, it may seem like an Uber competitor—and it is—but it is much more. At scale, it will change the way we travel; many will never own a car, simply choosing to move around fluidly in the Robotaxi ecosystem.

Technology revolutions start slowly and then happen all at once. Many pieces must fall into place before we reach a tipping point.

For Tesla, the breakthrough moment didn't happen overnight—Elon Musk and his team have been working on this for the better part of a decade. As you have read, data is essential to the end-to-end AI approach—that alone took years to build out, one vehicle sale at a time. Compute at scale (i.e., accelerated computing) was also necessary. Along the way, various methods to achieve vehicle autonomy were prototyped and failed, but these failures led to success.

Consider Nvidia—what was once a "boring" video graphics company has turned into an AI powerhouse, with a ~$3 trillion market cap. Nvidia sold its first AI system to OpenAI (maker of ChatGPT) in 2016 (did you notice?). In the fall of 2022, Nvidia's CEO previewed the future of his company—namely the accelerated computing/AI theme (did you notice?). The stock was at a post-pandemic low at the time. In November, the "ChatGPT moment" occurred, and the world woke up to the potential of AI. Nvidia has not looked back[10].

Tesla is at its "fall of 2022" moment now—it is poised to disrupt the world once again. The future takes time, but when it arrives, it happens pretty fast!

Finals Thoughts

"My God, this car is driving itself, and no one knows."

FSD feels very real to me—I own a 2024 Tesla Model 3 Performance. As of September 2, 2024, it is running FSD 12.5.1.3. This is the fourth version installed on my vehicle since I took ownership in late June. I now have 1,900 miles on my new Tesla, with approximately 500 miles driven using FSD. I *routinely* use FSD for various trips. Currently, it is referred to as "supervised FSD" because I must remain attentive, but this requirement is expected to be eliminated in the future.

I've had few interventions (times when I've needed to take control), especially since version 12.5 was installed. The rate of progress is truly astonishing—each version has shown significant improvements. My estimate is that FSD will be capable of fully autonomous driving (with no supervision) within 8 to 10 months.

Last night, while driving with a close friend, I remarked, "My God, this car is driving itself, and no one knows." By that, I meant that drivers of the cars around me are unaware of it, and more broadly, when I talk to people who aren't familiar with the technology, they are completely unaware of how rapidly this revolution is approaching.

As some of you know, I have closely witnessed the rise of vehicle autonomy. I helped form the early tech team at Rivian and managed several teams in areas such as in-vehicle infotainment, cloud services, and mobile applications. Relevant to this discussion, I also worked closely with a progressive autonomy team at Rivian.

NOTES:

[1]

https://x.com/pbeisel/status/1828863063774376223

[2]

https://x.com/pbeisel/status/1830689694008709123

[3] It is possible that the perception and planning models are one model. However, there are two clues suggesting they are distinct. First, historically, perception has been an AI process for some time, whereas planning has not (the version 11 planning module was code-based). Second, the in-vehicle display effectively manages a clean visualization of the roadway and its objects, essentially showing the driver a visual representation of the perception process.

[4] A **regression**, in this context, is a bug where a feature that once worked correctly stops functioning after a change to the AI model. Developers use regression testing to detect these issues, ensuring that new changes haven't adversely affected existing functionalities.

[5] Tesla labels its overall hardware platform in a vehicle with a version number. The current version is Hardware 4 (HW4), and Hardware 5 (HW5) is coming. Hardware 5 will be renamed

AI5, but strictly speaking the hardware platform describes more than the AI components (cameras, etc.).

[6] Supervised learning is a machine learning approach where an algorithm learns from labeled data, meaning each training example is paired with an output label. The data is often labeled by a human, but can also be auto-labeled. The algorithm uses this data to learn the mapping function that predicts the output from given inputs. Its goal is to generalize from the training data to unseen situations with high accuracy.

[7] Within a vehicle, an **IMU** (inertial measurement unit) serves as a critical component for stability control, navigation, and autonomous driving systems. It detects changes in vehicle orientation, velocity, and gravitational forces by measuring acceleration and rotational rates. This data helps in real-time adjustments for vehicle stability, assists in GPS navigation during signal loss, and supports advanced driver-assistance systems (ADAS) by providing precise motion data.

[8] Tesla vehicles come with **8 external cameras** for models equipped with Hardware 3 (HW3), which is standard for most current models like the Model 3 and Model Y. However, newer hardware versions, like Hardware 4 (HW4), might include up to **11 cameras**.

[9]

https://en.wikipedia.org/wiki/Tesla_Dojo

[10] See Cern Basher's (

@CernBasher

) video on Tesla stock performance relative to its peers etc. @

https://www.youtube.com/watch?v=W5gUgGP8lac

. This excellent video details Nvidia's recent rise.