Machine Learning for Architecture
The 12 Principles

Department of Architecture

October 15, 2025

What you'll learn (in plain language)

- A clear mental model of how ML works without math overload.
- The 12 principles that make ML predictable, not mysterious.
- How to use generative tools responsibly for studio work.
- Where ML helps (and where it doesn't) in architectural workflows.

How we'll run this

- Jargon translated to design terms; examples from studio contexts.
- Short "quick activities" to check understanding (no installs).
- Clear do's/don'ts for client-facing work (IP, bias, privacy).
- Everything is reproducible: prompts, seeds, masks, versions.

Class Reflection — Before the ML Primer

Prompt

"Write a one–line answer to each question below before the session."

1. What comes to mind when you hear **machine learning**?
2. Where does ML already touch architecture or design?
3. What makes a **good embodiment** of ML?
4. When is ML **useful** rather than gimmicky?
5. What **worries or excites** you about ML in design?

Instructions

Answer quickly and honestly (1–2 words or a short phrase). No right answers — we'll revisit these at the end.

Principle 1 of 12: ML learns a mapping from examples, not rules

Idea

An ML model is a function $f_\theta(\mathbf{x}) \to y$ whose parameters θ are fit from examples $(\mathbf{x}, y)$.

Why it matters

Expect pattern recognition, not hand-coded logic. The model learns regularities present in the data.

Studio example

(massing image) $\to$ (predicted daylight metric); (facade photo) $\to$ (style label).

Quick activity (2 minutes)

Name three inputs you could give a model in your current studio project. What output would you want?

Principle 2 of 12: Data is the design brief

Idea

The dataset defines what the model can and cannot learn.

Why it matters

Better data beats fancier models; curate for coverage and quality, not just quantity.

Studio example

Facade-style classifier fails if trained only on one region or era.

Quick activity (2 minutes)

List two gaps in a 'facade' dataset that would cause failures on your campus. How would you fill them?

Principle 3 of 12: Representation shapes what can be learned

Idea

How we encode inputs (pixels, meshes, graphs, text) determines what the model can 'see'.

Why it matters

Choosing representation is like selecting drawing type: plan vs section vs perspective.

Studio example

Street photos (pixels) for segmentation vs adjacency graphs for layout optimization.

Quick activity (2 minutes)

For your task, would pixels, vectors, meshes, or text be best—and why?

Principle 4 of 12: Loss = what 'better' means

Idea

The loss function translates 'good' into math; models only improve on what loss rewards.

Why it matters

If the loss ignores safety or legibility, the model will too.

Studio example

Daylight surrogate with MSE learns averages; worst-case glare needs a different objective.

Quick activity (2 minutes)

Write a one-line 'loss' for your project (e.g., minimize peak glare while keeping DA% above target).

Principle 5 of 12: Training = reduce loss (via gradient descent)

Idea

Training iteratively adjusts parameters in the direction that reduces loss.

Why it matters

Explains learning rate, convergence, and why training looks rough before it stabilizes.

Studio example

Don't judge early iterations; monitor the validation curve.

Quick activity (2 minutes)

Sketch how you would know training is 'done' (hint: validation loss/early stopping).

Principle 6 of 12: Generalization is the goal

Idea

We care about performance on new, unseen data—not just the training set.

Why it matters

Overfitting = memorization; good models transfer to the next site/city.

Studio example

Roof-type detector perfect on your curated set, fails on cloudy days.

Quick activity (2 minutes)

Name two differences between training images and deployment images. How will you test generalization?

Principle 7 of 12: Bias–variance: too simple vs too specific

Idea

Underfit models are too simple; overfit models memorize noise.

Why it matters

You control this via model capacity, augmentation, and regularization.

Studio example

Tiny model misses brick vs terracotta; huge model memorizes specific buildings.

Quick activity (2 minutes)

Which knob would you turn first to reduce overfit? (augmentation, regularization, or more data?)

Principle 8 of 12: Uncertainty is real—treat outputs as suggestions

Idea

Predictions are probabilistic; confidence varies by context and input quality.

Why it matters

Sample multiple outputs and critique; avoid single-image 'truth'.

Studio example

Use seeds and A/B comparisons for renders.

Quick activity (2 minutes)

Generate 3 variations (same seed + small prompt tweaks); choose and justify one.

Principle 9 of 12: Distribution shift breaks models

Idea

When deployment data differ from training (climate, culture, camera), expect failure.

Why it matters

Plan pilots on real sites; monitor drift; retrain when needed.

Studio example

Berlin-trained segmenter struggles in Bangkok (materials, colors, sky).

Quick activity (2 minutes)

Write one 'shift' you expect in your project and a way to detect it.

Principle 10 of 12: Generative models = distributions + conditioning

Idea

Diffusion models learn a distribution and sample from it; prompts/masks/refs condition sampling.

Why it matters

Prompts steer; seeds give reproducibility; masks/refs preserve geometry and identity.

Studio example

Keep geometry via masks; change cladding via prompt; reuse the same seed for versions.

Quick activity (2 minutes)

Take a white model view; list what you'd mask vs what you'd let the model change.

Principle 11 of 12: Evaluation must match the task

Idea

Pick metrics that reflect the goal and pair them with human A/B judgment for visuals.

Why it matters

Numbers alone can reward 'pretty wrong' results.

Studio example

For concept imagery, use pairwise preference + rubric (material legibility, camera plausibility).

Quick activity (2 minutes)

Draft two rubric criteria to compare two generated facade images.

Principle 12 of 12: Ethics, IP, and human-in-the-loop

Idea

Datasets carry bias; licenses and privacy matter; designers remain accountable.

Why it matters

You choose sources, cite them, disclose process; keep clients safe.

Studio example

Use open/owned references; document prompts, seeds, masks; avoid identifying people without consent.

Quick activity (2 minutes)

Pick one slide and add a visible credit line for every image source.

The ML pipeline (big picture)

- Collect data $\rightarrow$ split into train/validation/test.
- Pick a model family (trees, CNNs, transformers, diffusion).
- Train to reduce loss; validate; test; deploy; monitor.

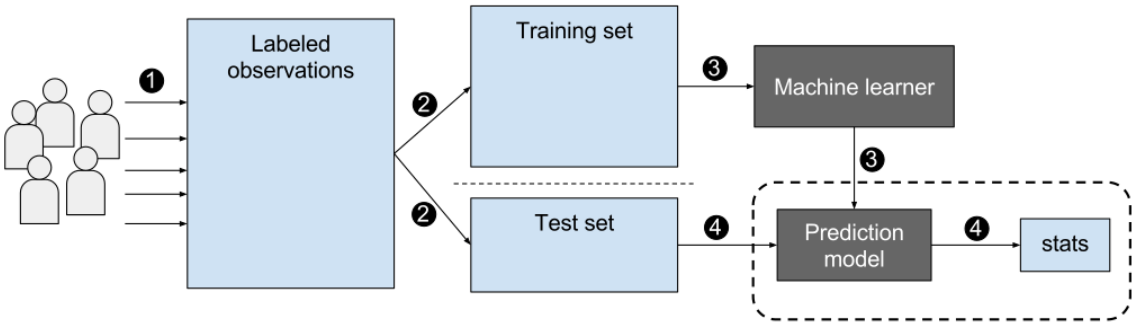

Figure 1: Supervised learning pipeline

Wikimedia Commons diagram (CC BY-SA).

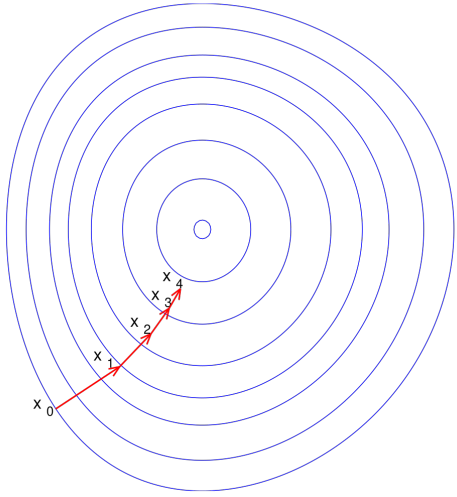

Figure 2: Gradient descent intuition: follow the slope downhill

Gradient descent: Wikimedia image (CC BY-SA).

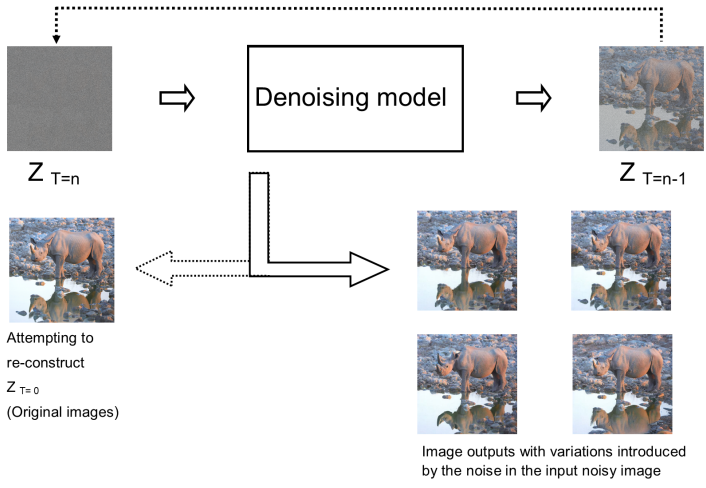

Figure 3: Diffusion: iteratively denoise from noise to image

Wikimedia diffusion example (MrAlanKoh).

2D Convolution animation (CC BY-SA 3.0)

Placeholder image

→ Replace this file with the cited image to show the real figure in the PDF.

File path: convolution2d.png

Source URL:

https://commons.wikimedia.org/wiki/File:2D_Convolution_Animation.gif

Educational use. Add proper credit in captions when replacing.

Figure 4: 2D convolution (intuition for CNNs)

Wikimedia animated ver.

2D convolution (intuition for CNNs)

Follow this animated example for the full animation.

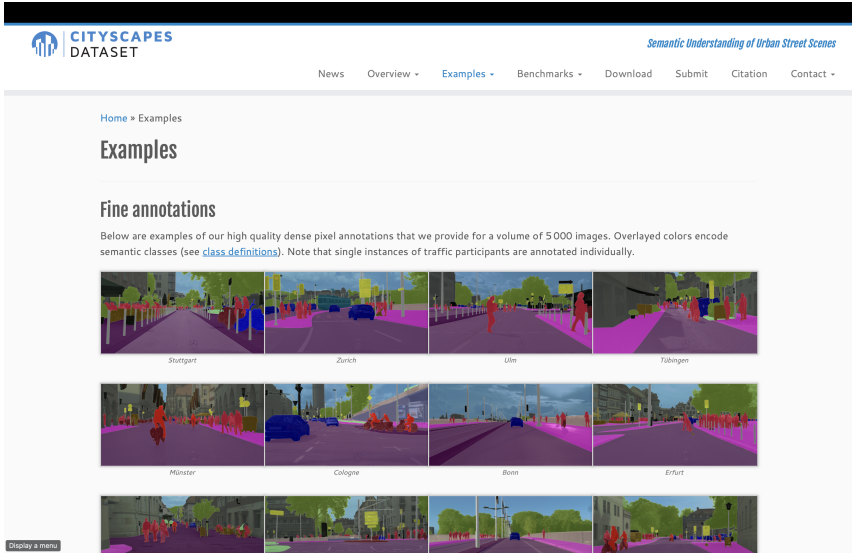

Figure 5: Semantic segmentation for site/streetscape analysis

Sample from Cityscapes dataset.

Why ML now (for architects)

- **Speed** — Turn sketches and massing into rich visuals quickly
- **Exploration** — Search wide design spaces, then iterate
- **Analysis** — Faster proxies for daylight/energy to inform form-finding
- **Communication** — Tell stories to clients with images & video
- **Constraints** — Use ML to respect site/structure/program constraints

Parametric vs. ML vs. Optimization

- **Parametric design** — You specify rules; the computer evaluates
- **Optimization** — Searches parameters to minimize a cost (e.g., solar gain)
- **Machine Learning** — Learns patterns from data to predict or generate
- **Combine them** — Parametric model + optimizer + ML surrogate or generator

Generative AI: two common operations

- Text $\rightarrow$ Image (ideation)
- Image $\rightarrow$ Image edits (style transfer, inpainting, outpainting)
- Control with references, masks, and seeds

Midjourney vs Nano Banana (quick compare)

	Midjourney	**Nano Banana (Gemini 2.5 Flash Image)**
Best at	Artistic ideation, stylization	Precise edits, multi-image fusion, identity consistency
Inputs	Text, refs, style refs	Text, single/multi images, masks, refs
Access	Discord/web; subscription	Google AI Studio / Gemini; free tier
Use in studio	Moodboards, quick concept art	Consistent characters/material tests/3D-esque

Sources: docs.midjourney.com; developers.googleblog.com.

Workflow: concept ideation

- Block massing in Rhino $\rightarrow$ export viewport
- Text-to-image for mood/style $\rightarrow$ iterate and curate
- Use references to keep materials/context coherent

Workflow: massing → render (hybrid)

- Start with white model render
- Image-to-image with prompts (architectural style, lighting)
- Use masks to protect key geometry; iterate seeds; compare side-by-side

Workflow: site analysis via segmentation

- Run semantic segmentation on site photos
- Compute sky/green/road/building ratios; quantify views
- Inform facade porosity, shading, and massing

Surrogate modeling (use carefully)

- Generate dataset: parametric samples + simulation outputs
- Fit regressor (e.g., gradient boosted trees / small NN)
- Cross-validate; publish error bars; only use within domain

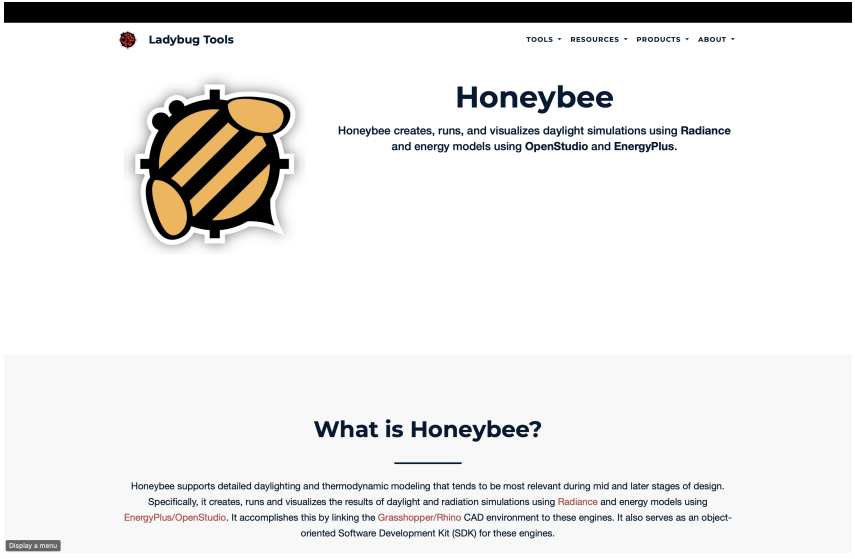

Figure 6: Surrogates: train a model to approximate simulation

Replace with Honeybee/Ladybug resource – use cautiously and validate.

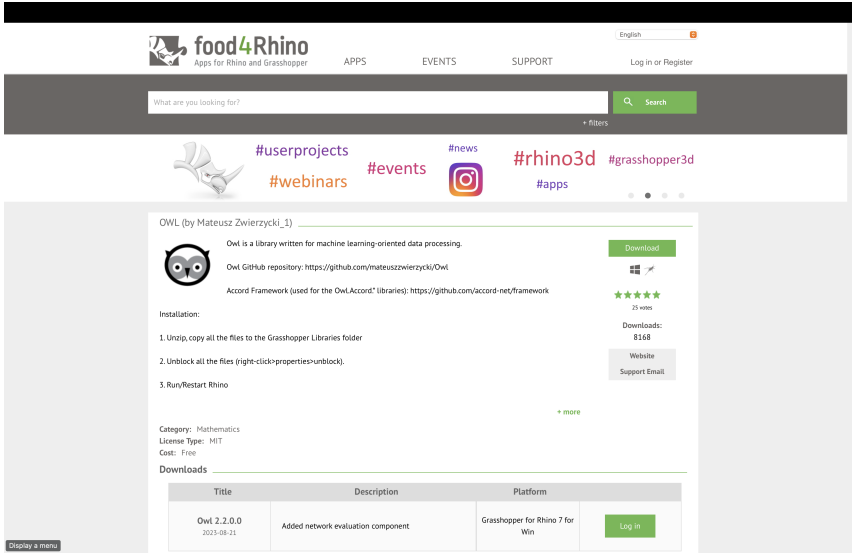

Figure 7: Grasshopper: Owl ML plugin (data workflows)

Replace with plugin screenshot from Food4Rhino page.

Grasshopper + ML options

- Owl (data workflows)
- Pug (Tensorflow.NET in Grasshopper)
- LunchBox ML (classic ML tasks)
- Connect to external notebooks via Python/REST

Promptcraft 101 (for design)

- Subject + medium + materials + lighting + camera + time + mood
- Architectural cues: plan/section language, materials taxonomy, lighting verbs
- Use aspect ratio, stylize/creativity controls, seeds

Prompt patterns: architecture

- "*brutalist concrete library, deep ribs, board-formed texture, overcast sky, 35mm*"
- "*timber diagrid atrium, skylight grid, soft morning haze, product photo lighting*"
- Add refs: material samples, site photo, precedent

Recipe: Midjourney (text $\rightarrow$ image)

- "concept sketch of mixed-use tower with stepped terraces, limestone + glass, golden hour, aerial 3/4 view, –ar 16:9 –stylize 250"
- Iterate: change time-of-day, lens, weather; use style refs

Recipe: Nano Banana (image → edits)

- Upload your viewport render
- Prompt: "replace glazing with vertical timber louvers; keep structure; late afternoon warm light; add pedestrians"
- Mask sky to preserve; iterate

Data: where to start (ethically)

- Wikimedia Commons (check license), personal photos, open datasets (Cityscapes for streets)
- Always cite; avoid scraping private or copyrighted sources without permission
- Respect people's privacy (faces, license plates)

Evaluate design outputs critically

- Check structure/plausibility (stairs/handrails/windows)
- Look for perspective/physics errors
- Use AI output as *references*, not final drawings

IP, copyright, and model terms

- Generated images may still infringe IP (style/characters/logos)
- Use tools per their licenses; keep client permissions in writing
- Prefer references you own or that are openly licensed

Bias, stereotypes, and ethics

- Datasets reflect real-world biases $\rightarrow$ watch for stereotypes
- Diverse precedents and prompts mitigate bias
- Explain to clients how images were made (transparency)

Class Reflection — After the ML Primer

Before

- "Robots, AI art, black box."
- "Generate ready-looking designs."
- "It replaces designers."

After

- "A model learning patterns from data."
- "Transparent, reproducible, human-guided."
- "When it supports iteration and evaluation."
- "It expands design space — still needs judgment."

Discussion Prompt

Which of your five answers changed the most — and why?
Keep your notes and let's revisit this in class next Wednesday.

Entry ticket for next week (1 minute)

- Which single principle will change how you work this week—and how?