ADVANCED APPLICATIONS AND IMPLEMENTATION STRATEGIES

Lecture 3 of 3

Dr. Nazarii Mediukh,
Institute for Problems of Materials Science, NASU

Supported by

Co-funded by the European Union

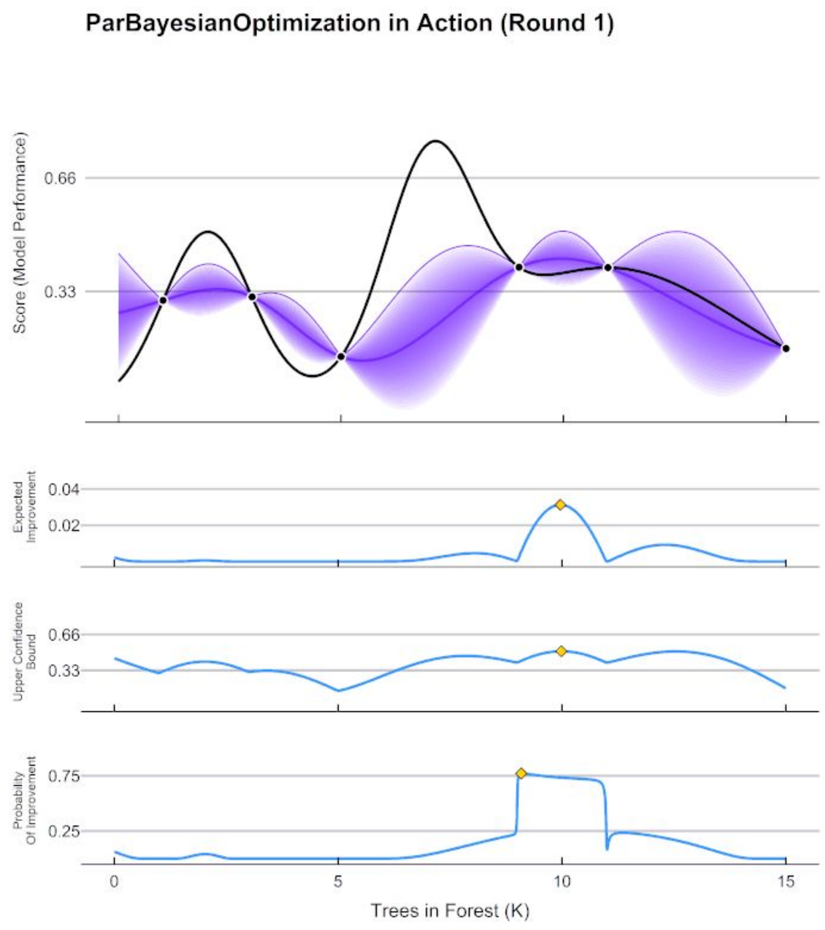
Lecture 3: Advanced applications and implementation strategies

We'll dive deeper into:

- Multi-objective Optimization: Optimize multiple conflicting goals simultaneously (quality vs speed vs cost)
- Deep Learning for AM: Computer vision for defect detection, physics-informed neural networks
- Real-time Process Control: Online monitoring and adaptive parameter adjustment
- Implementation Challenges: Deployment in production, integration with existing systems

Bayesian optimization: Recap

Step 1: Run a few initial experiments (5-10 random samples)
Step 2: Fit a Gaussian Process to the data
 - GP predicts outcome everywhere
 - GP gives uncertainty everywhere
Step 3: Use an acquisition function to decide the next experiment
 - Balance **exploitation** (try near current best)
 - vs **exploration** (try uncertain regions)
Step 4: Run the chosen experiment, add to the dataset
Step 5: Update GP, repeat Steps 3-4 until converged or budget exhausted
Result: Find the optimum in 20-100 experiments instead of thousands!



Conflicting objectives

- **Quality:** Minimize porosity, surface roughness, residual stress
- **Speed:** Maximize build rate, minimize print time
- **Cost:** Minimize material usage, energy consumption, post-processing

Example: Higher laser power
- ✅ Faster printing (good for speed)
- ❌ More porosity (bad for quality)
- ❌ Higher energy cost (bad for cost)

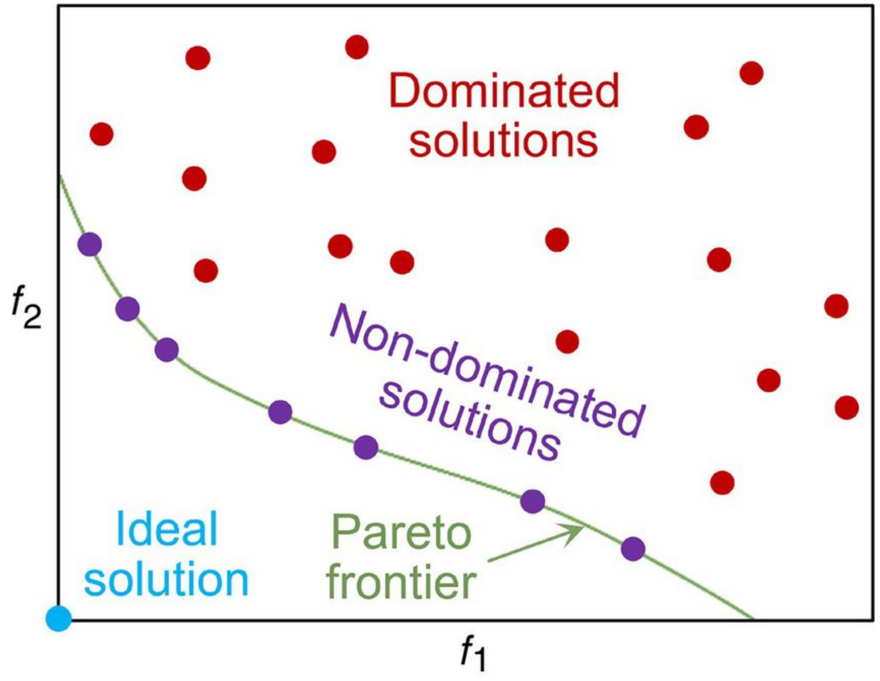
Solution: Pareto Frontier

There is no **single optimal** solution; we will look for a set of suboptimal ones.

Step 1: Run a few initial experiments (5-10 random samples)
Step 2: Fit a Gaussian Process to the data for **all objectives**
- GP predicts outcome everywhere
- GP gives uncertainty everywhere

Step 3: Compute current Pareto Frontier.
Step 4: Use an acquisition function to decide the next experiment that improves the Frontier. Balance **exploitation** vs **exploration**
- Expected Hypervolume Improvement (EHVI)
- Probability of Pareto improvement
- Weighted sum with uncertain weights

Step 5: Run the chosen experiment, and add to the dataset
Step 6: Update GP, repeat Steps 3-4 until converged or budget exhausted
Result: Find Pareto Frontier. Let engineers decide which is the best solution.



Example: We want to **minimise** porosity and **maximize** speed.

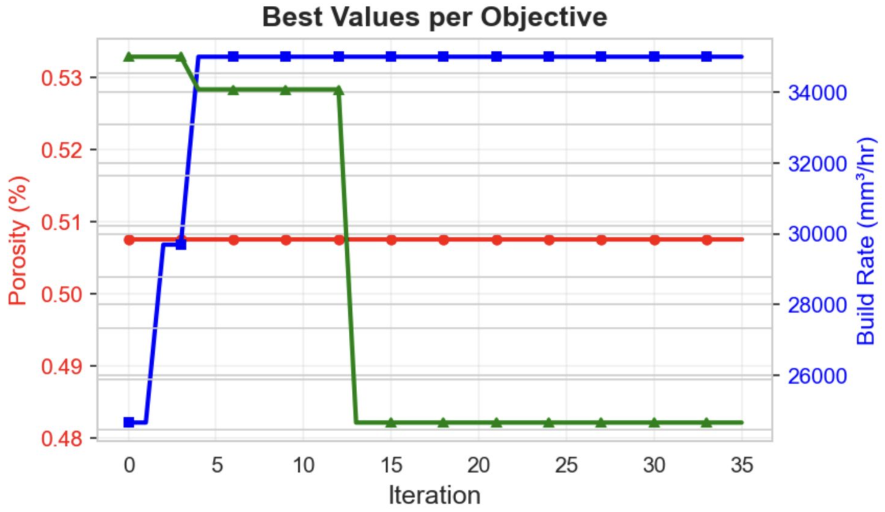
Porosity	Speed	Frontier?
1.5	10	yes
2.5	8	yes
1.0	5	no
2.0	9	no

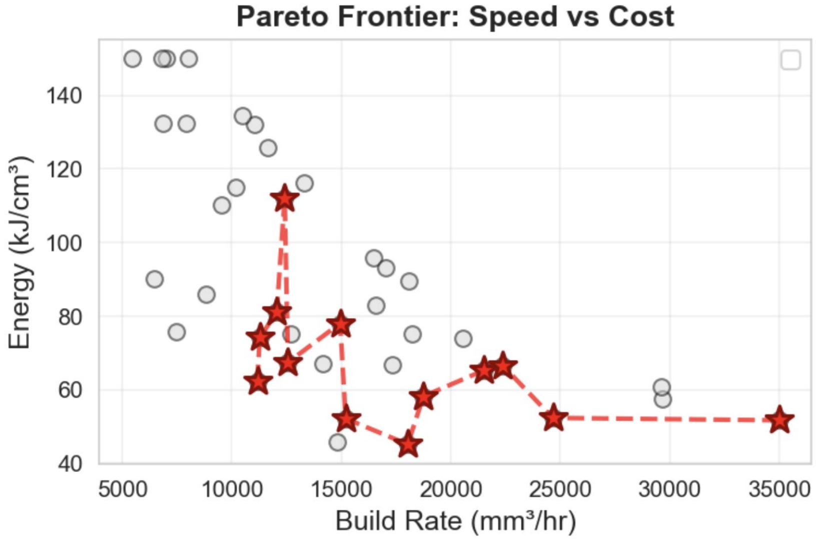
Demo 1: Multi-objective optimization

Three conflicting objectives:
- Minimize porosity (%) — quality
- Maximize build rate (mm³/hr) — speed
- Minimize energy consumption (kJ/cm³) — cost

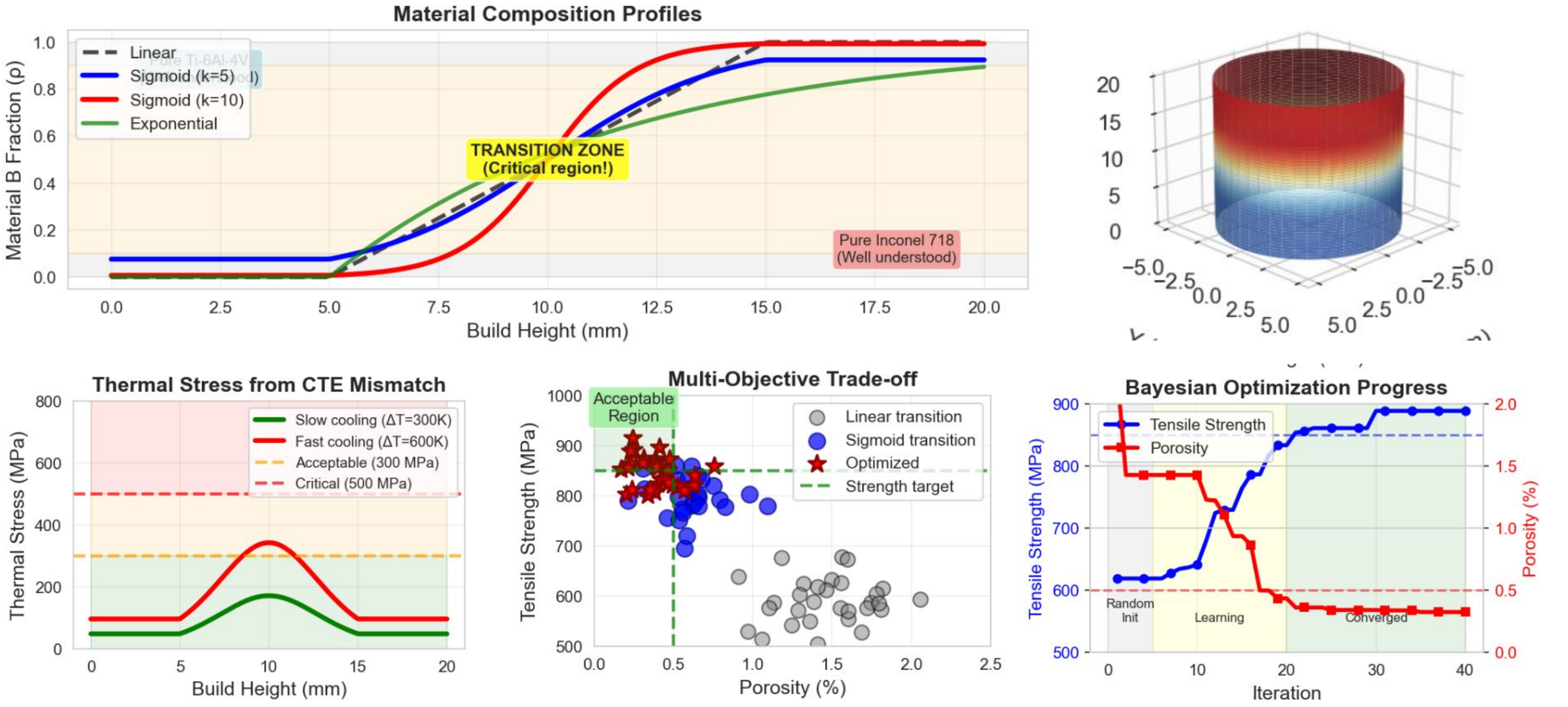
Parameters:
- Laser power (150-400 W)
- Scan speed (600-1400 mm/s)
- Hatch spacing (0.08-0.15 mm)
- Layer thickness (25-50 µm)

Acquisition function:
- **Random scalarization:** f_weighted = a1·porosity + a2·(-build_rate) + a3·energy
- **Expected improvement**: EI(x) = E[max(f(x) - f(x_best), 0)]

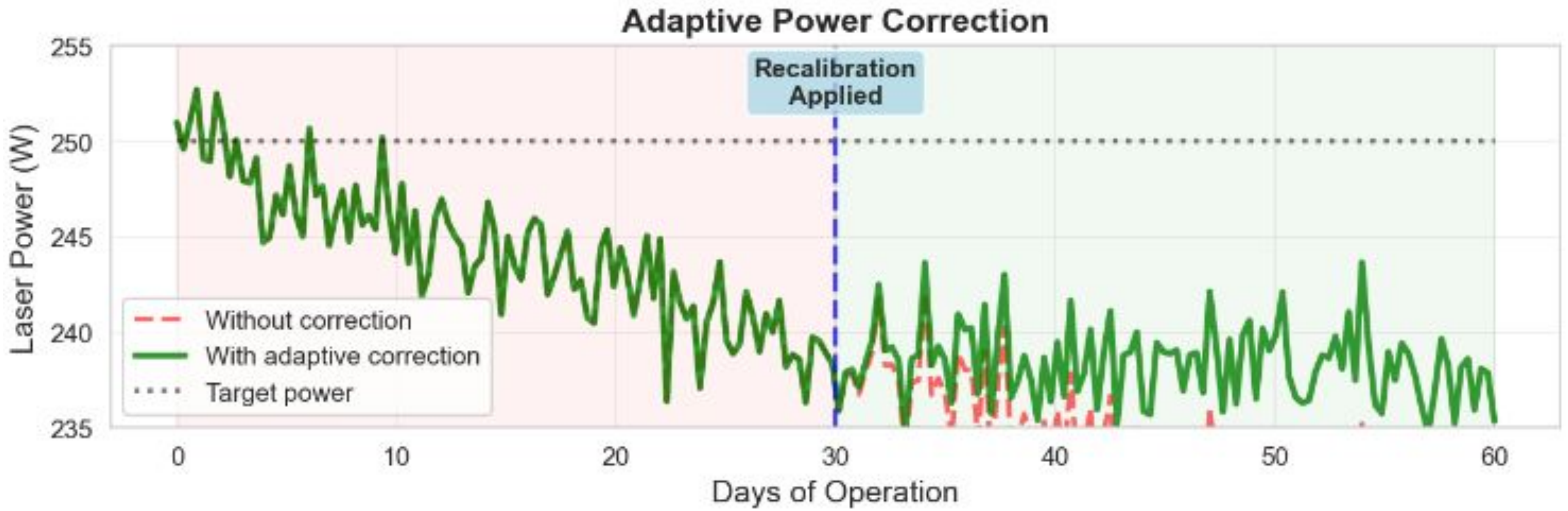


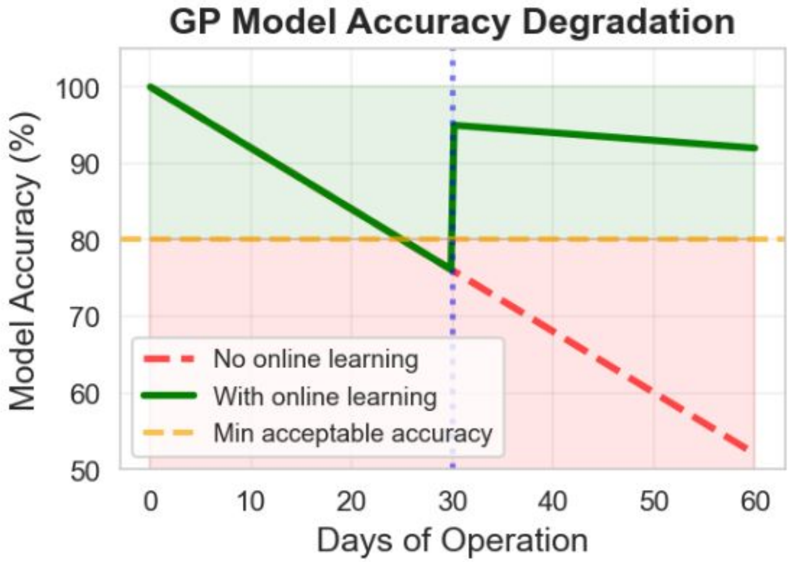


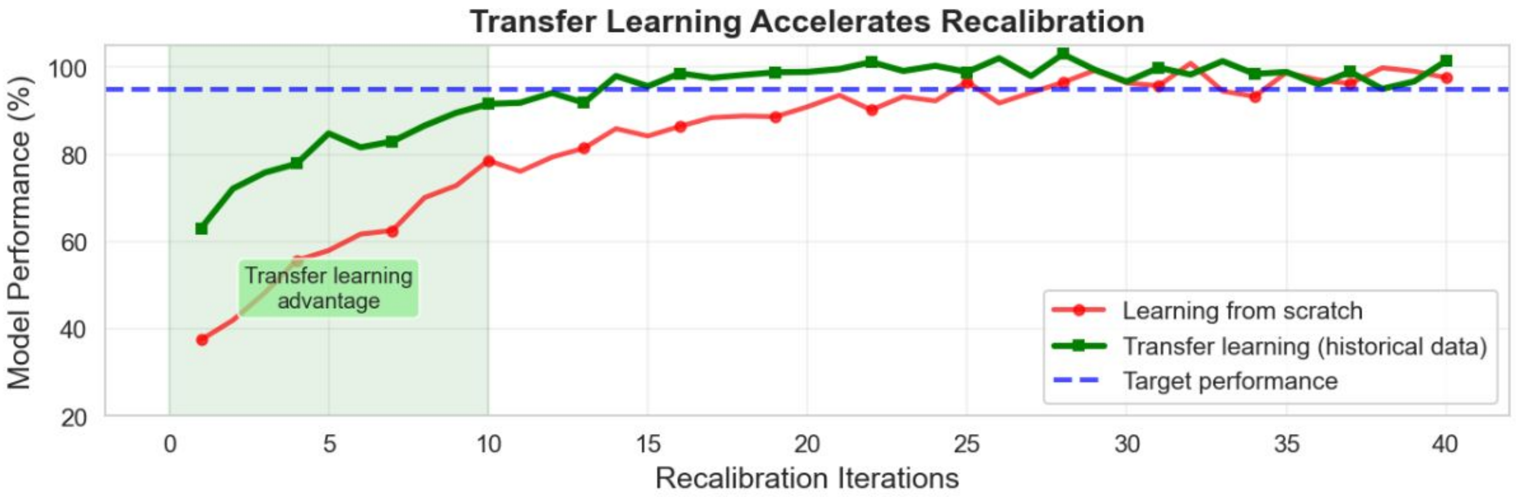
Multi-Material optimization



Process drift/Equipment variation







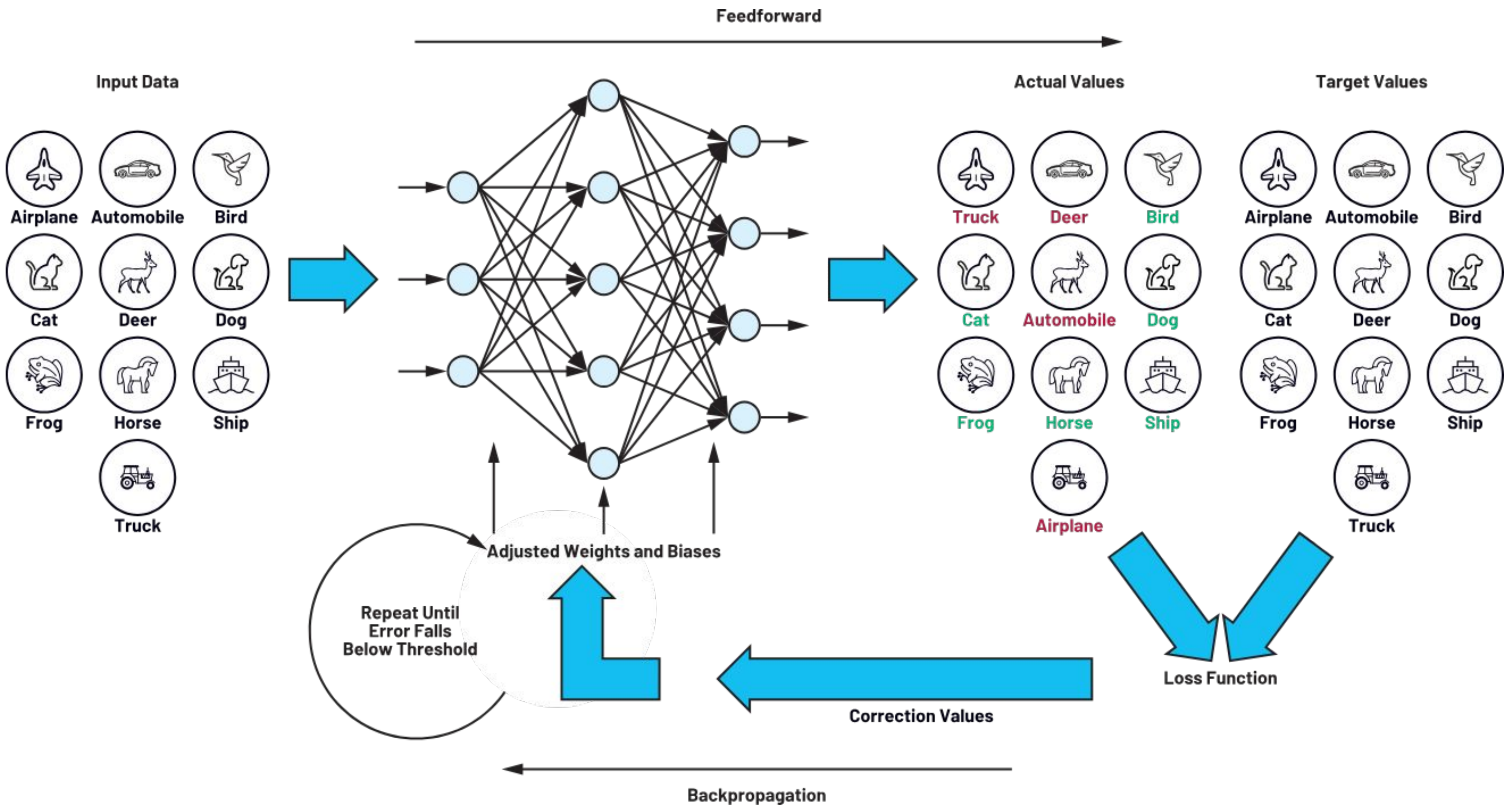
Why deep learning for ML?

Traditional ML works great for:
- Tabular data (process parameters → properties)
- Small to medium datasets (100-1000 samples)
- Uncertainty quantification

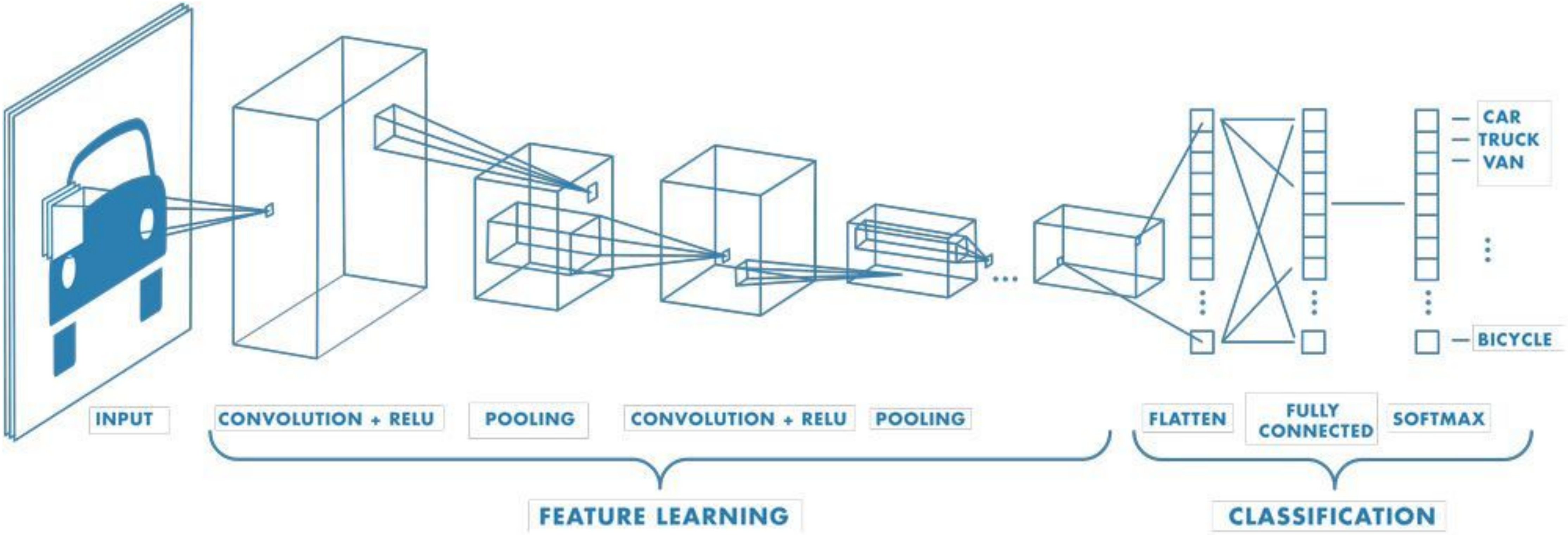
Deep learning great for:
- High-dimensional data: Images, videos, sensor streams
- Complex patterns: Nonlinear relationships, hierarchical features
- Large datasets: 10,000+ samples
- Real-time inference: Fast predictions on edge devices

Application	Input	Output	Architecture
Defect Detection	Layer images	Defect class/location	CNN
Process Monitoring	Thermal camera	Temperature field	U-Net
Quality Prediction	Melt pool video	Final properties	CNN-LSTM
Path Planning	Part geometry	Optimal toolpath	Graph NN
Microstructure	Process history	Grain structure	ResNet

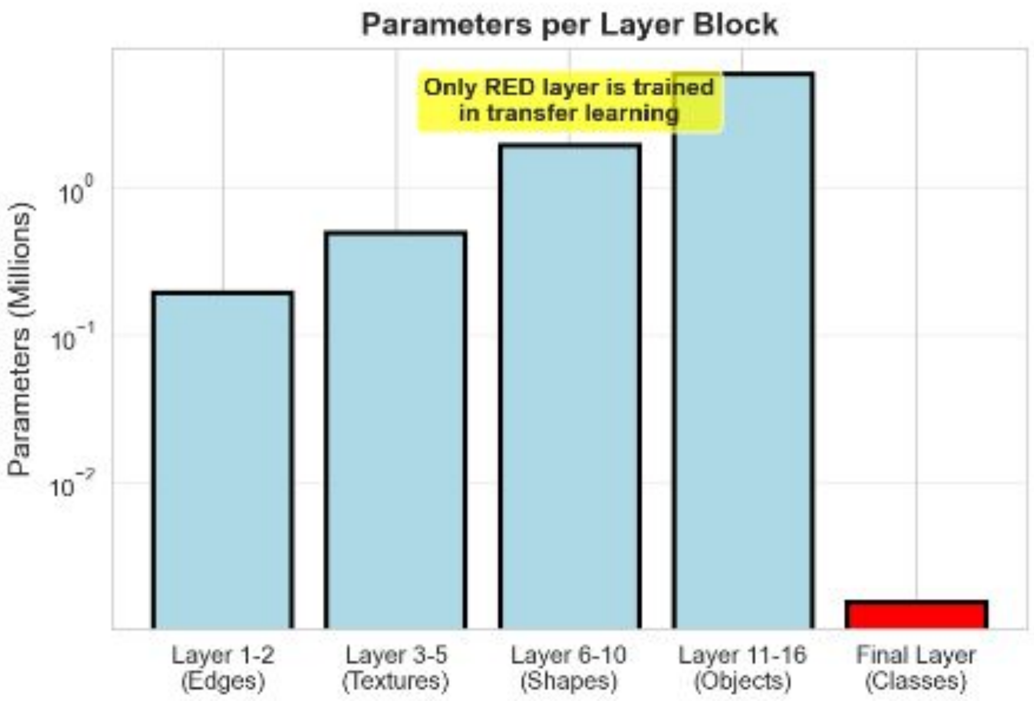
Neural Networks: Recap

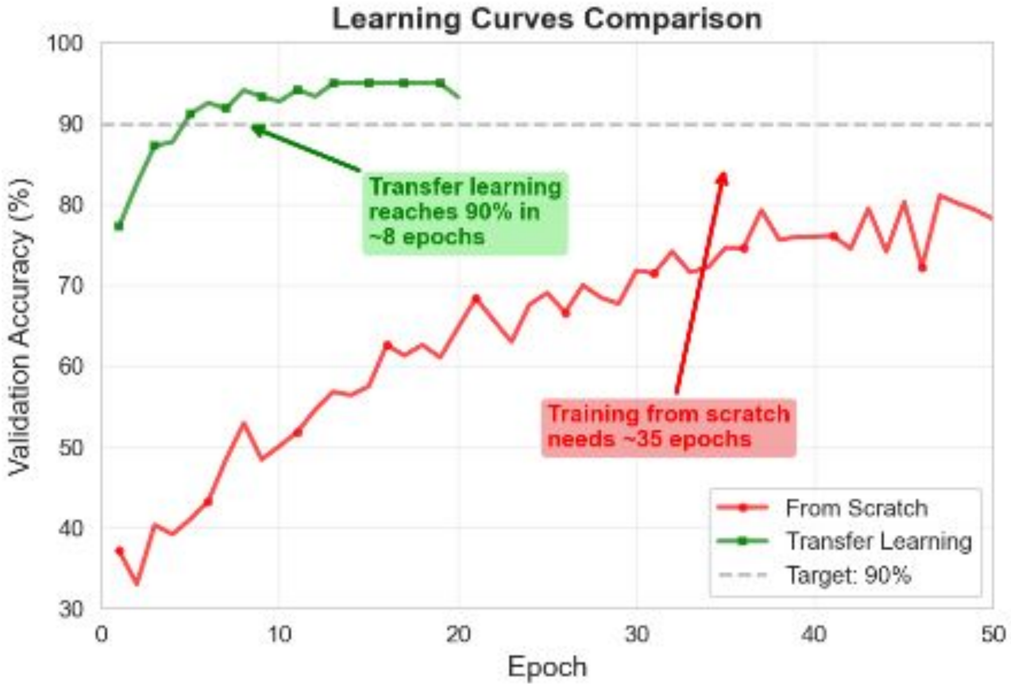


Neural Networks: CNN

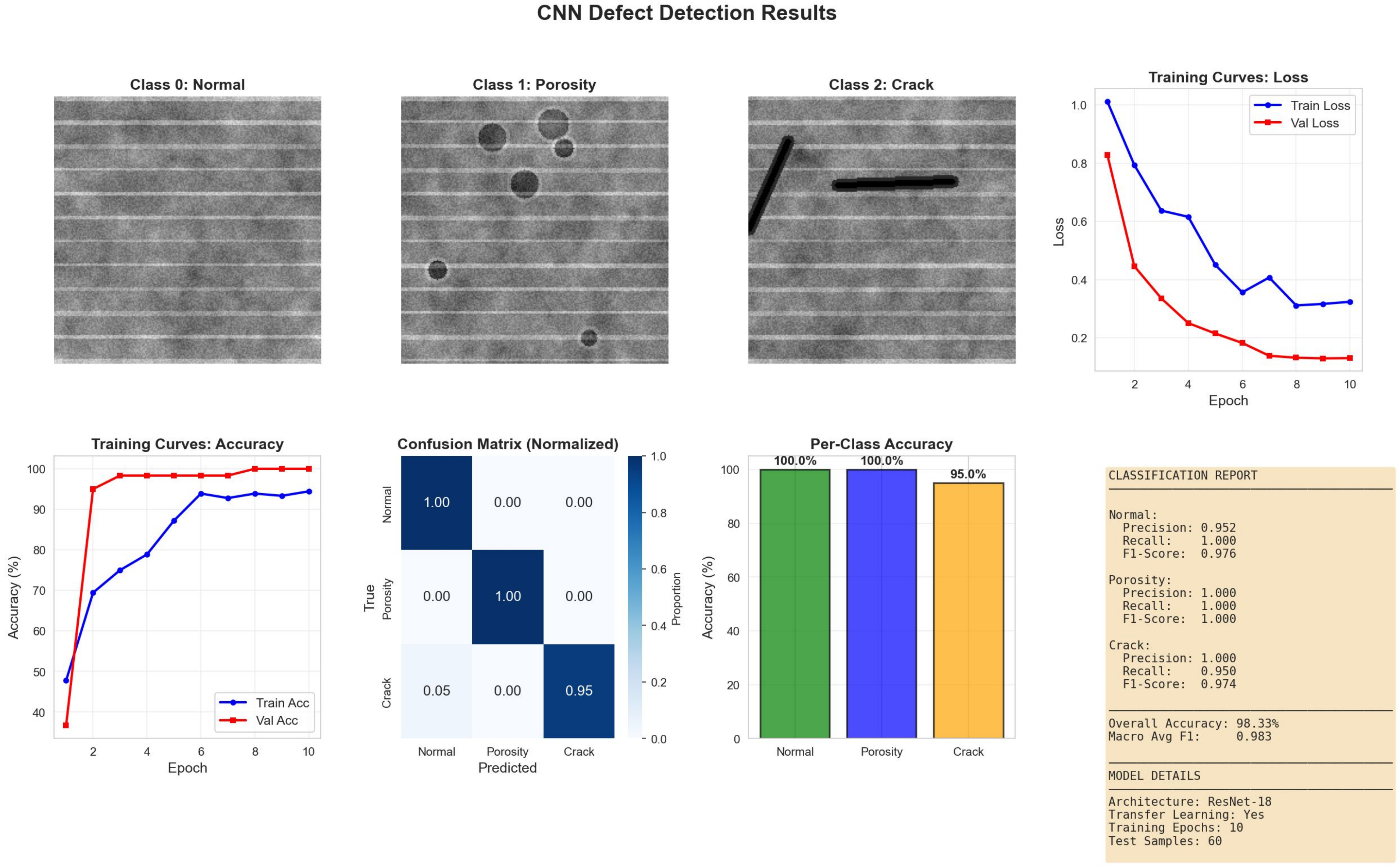


Pre-trained models: transfer learning



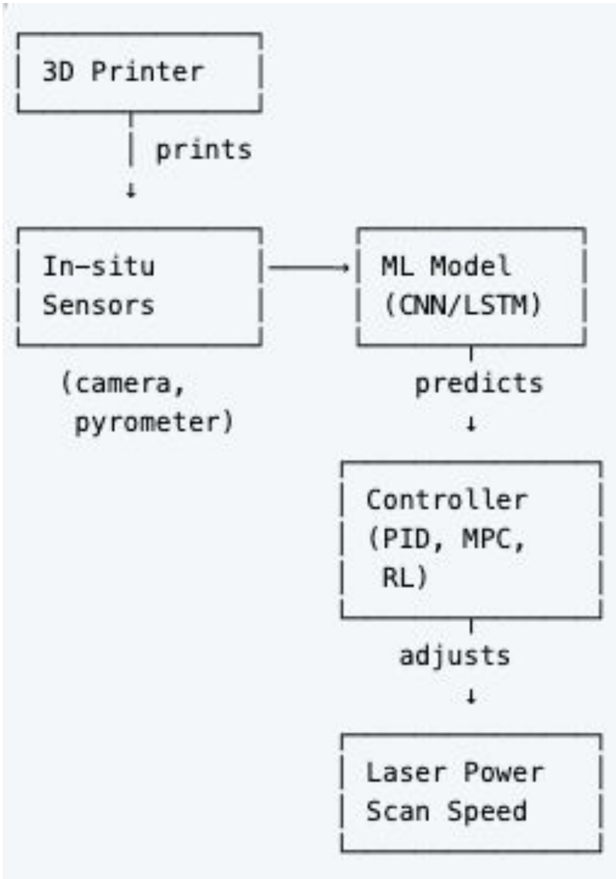


Demo 2: CNN for defetct detction



Adaptive manufacturing

Traditional AM: fixed parameters throughout the build
Adaptive AM: Adjust parameters in real-time based on feedback



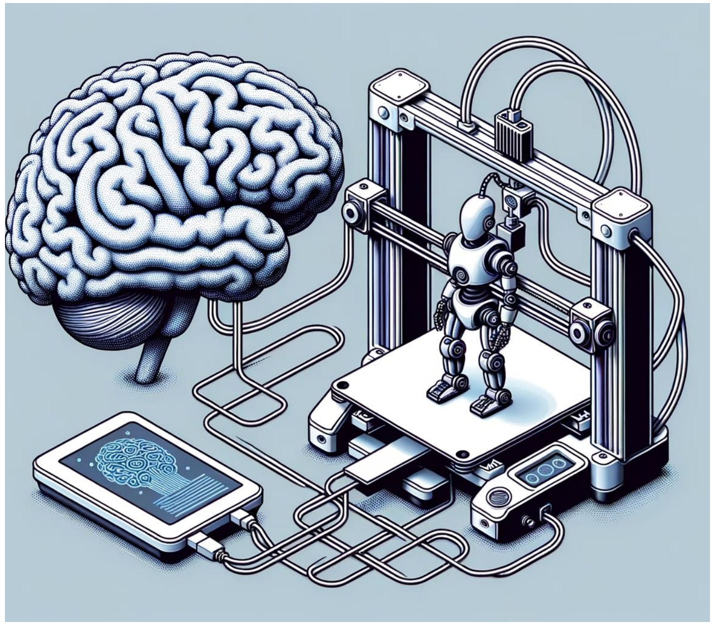
Sensor	Measures	Sampling Rate	Use Case
Thermal Camera	Temperature field	100-1000 Hz	Melt pool monitoring
High-speed Camera	Melt pool geometry	1000-10000 Hz	Defect detection
Pyrometer	Single-point temp	10 kHz	Process stability
Acoustic Emission	Sound waves	100 kHz	Crack detection
Photodiode	Light emission	1 MHz	Spatter detection
OCT	Layer topology	100 Hz	Surface roughness

Control Strategies

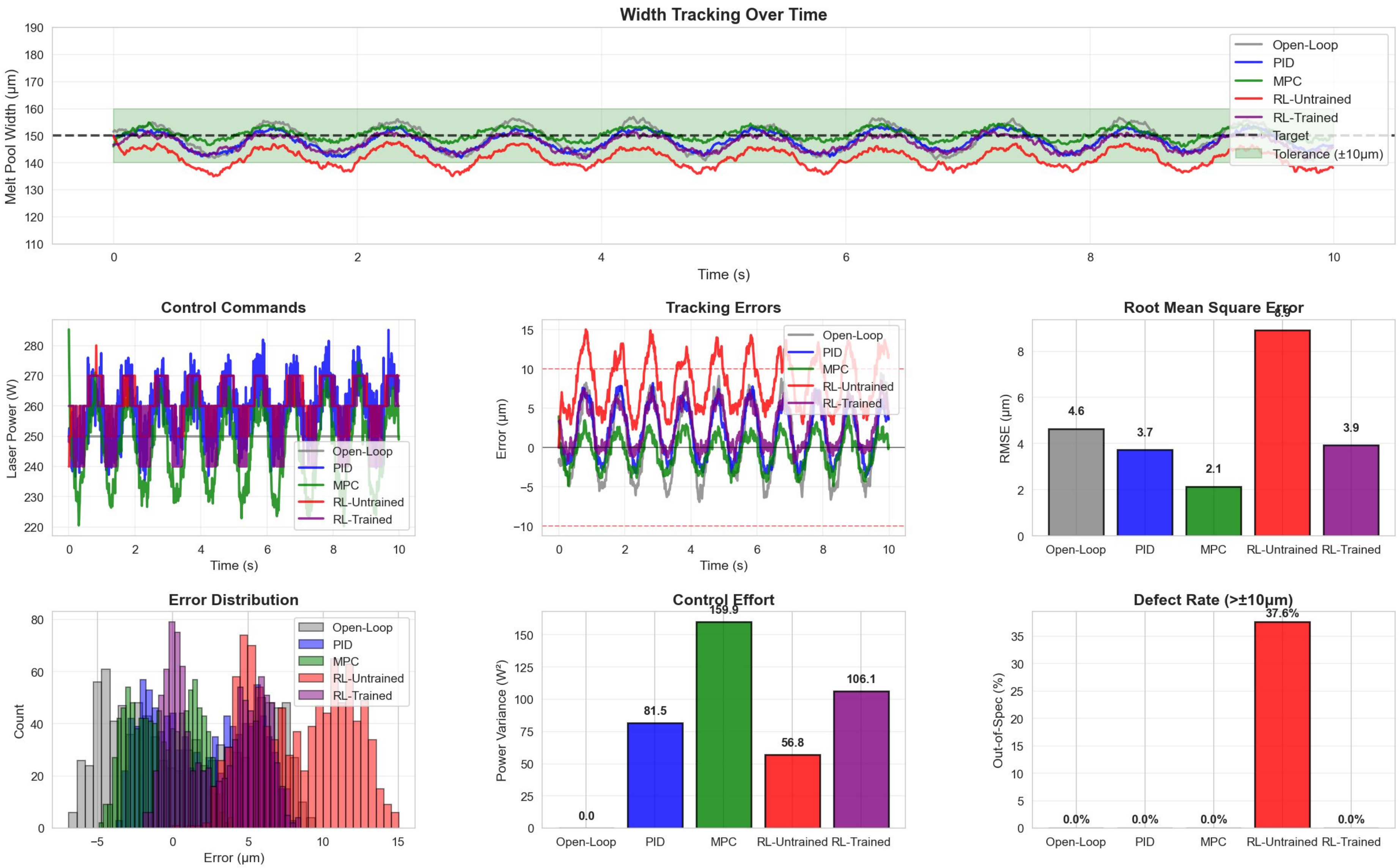
PID Control: Simple, well-understood, fast
- $e(t)$ = setpoint - measurement
- $u(t) = Kp \cdot e(t) + Ki \cdot \int e(t)dt + Kd \cdot de/dt$

Model Predictive Control (MPC): Needs an accurate model, predictive, and expensive:
- predict the next N steps using the model
- optimize control to minimize cost
- apply the first control action

Reinforcement learning: Needs many data to train, usually quite accurate, can learn as we go:
- State (sensor readings)
- Action (parameter adjustment)
- Reward (improvement of the quality metric)
- What action to take in each state to maximise reward?



Demo 3: Adaptive process control



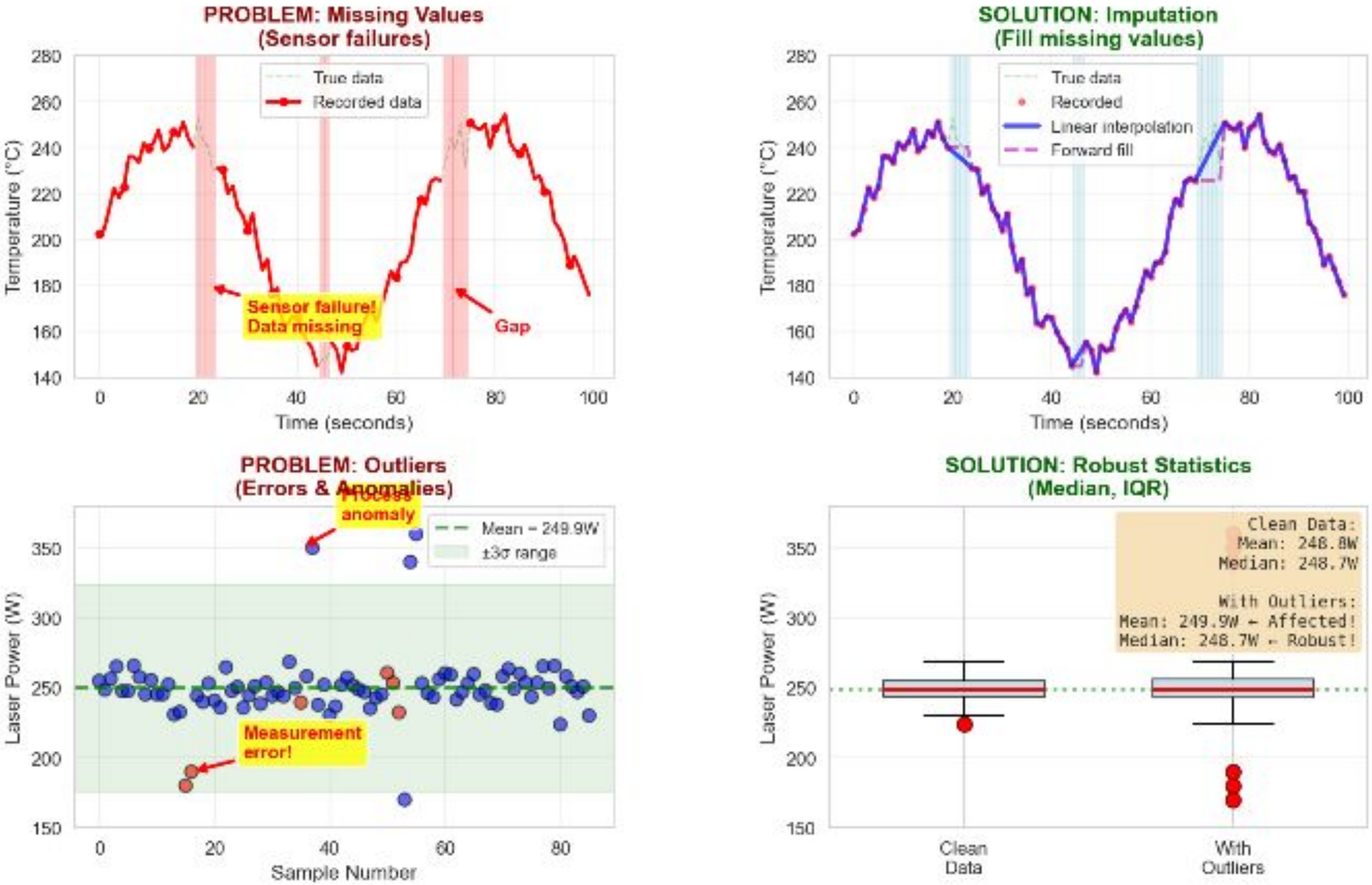
Controller must be fast!

Trade-offs:

- Edge models smaller -> may loose accurace
- Latency is critical
- Cost should be balanced

Device	Inference Time	Power	Cost
Cloud-based ML	50-200ms (mostely latency)	?	?
Raspberry Pi 4	50 ms (ResNet-50)	15W	$50
NVIDIA Jetson Nano	10 ms	10W	$100
Google Coral TPU	5 ms	2W	$60
Intel Movidius	8 ms	1W	$8

Challenges: Data Quality



Challenges: Data Availability

Strategies to overcome:
- Synthetic data from physics simulations
- Data augmentation (for images)
- Transfer learning (reuse models from similar processes)
- Active learning (intelligently select what to measure)
- Collaboration (shared datasets, consortia)

Synthetic data Example:
- Run FEA thermal model (ANSYS, Abaqus)
- Render as a grayscale image
- Add realistic noise
- Generate defect variations

Challenges: Computational requirements

Strategies to overcome:

- pre-trained models
- cloud training, edge inference
- model compression

Model compression techniques:

- Quantization: Convert 32-bit floats to 8-bit int
- Pruning: remove low-magnitude weights (neuron, connections) & fine tune to recover accuracy
- Train big teacher model, then small student based on teacher inputs.

Challenges: Integration with existing systems

Potential problems:

- data formats
- communication protocols
- software ecosystem (ML must fit into workflow)
- organizational (push back to adapt “black box” AI)

Best practices:

- start with a pilot (one machine, limited scope)
- data first (set up data collection infra the sooner the better)
- dive into domain knowledge, combine physics models and ML
- continuous learning: don’t forget that models degrade over time, set up monitoring, retraining, processes for human intervention

Thank you!
Any questions?