Australian Cobotics Centre

Session 1 — No Code Vision Quality Assurance: Train, Test, Deploy with Cobots

PROGRAM 4

Industrial transformation Training Centre in Collaborative Robotics
funded by the Australian government through the Australian Research Council

Australian Government
Australian Research Council

Introduction to Visual Quality Assurance and the Technology Landscape

By: Dr. Michelle Dunn

Visual Quality Assurance

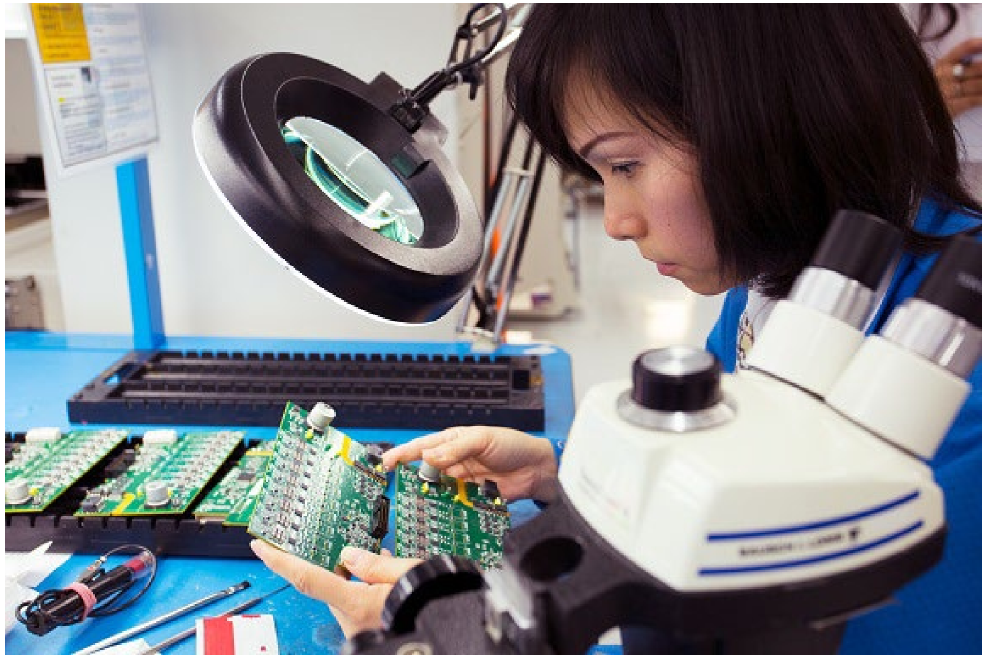
Use of visual inspection (manual or automated) to verify produce appearance, integrity and compliance with standards

- Compliance
  - Meets regulatory and safety standards
- Ensures Product Quality
  - Maintains brand reputation and customer satisfaction
- Reduces Waste and Rework
  - Early detection prevents defective products from reaching later stages
- Cost Efficiency
  - Avoids costly recalls and production downtime

Common Challenges of Manual Inspection

Manual inspection is carried out by a human operator

- Manual Inspection is subject to
  - Human Error
    - Fatigue
    - Inattention
    - Poor training
  - Subjectivity
    - I think it is okay - you think it is problematic
  - Lack of scalability
    - To inspect more parts we need more operators
    - Cycles back to subjectivity



Automated Inspection

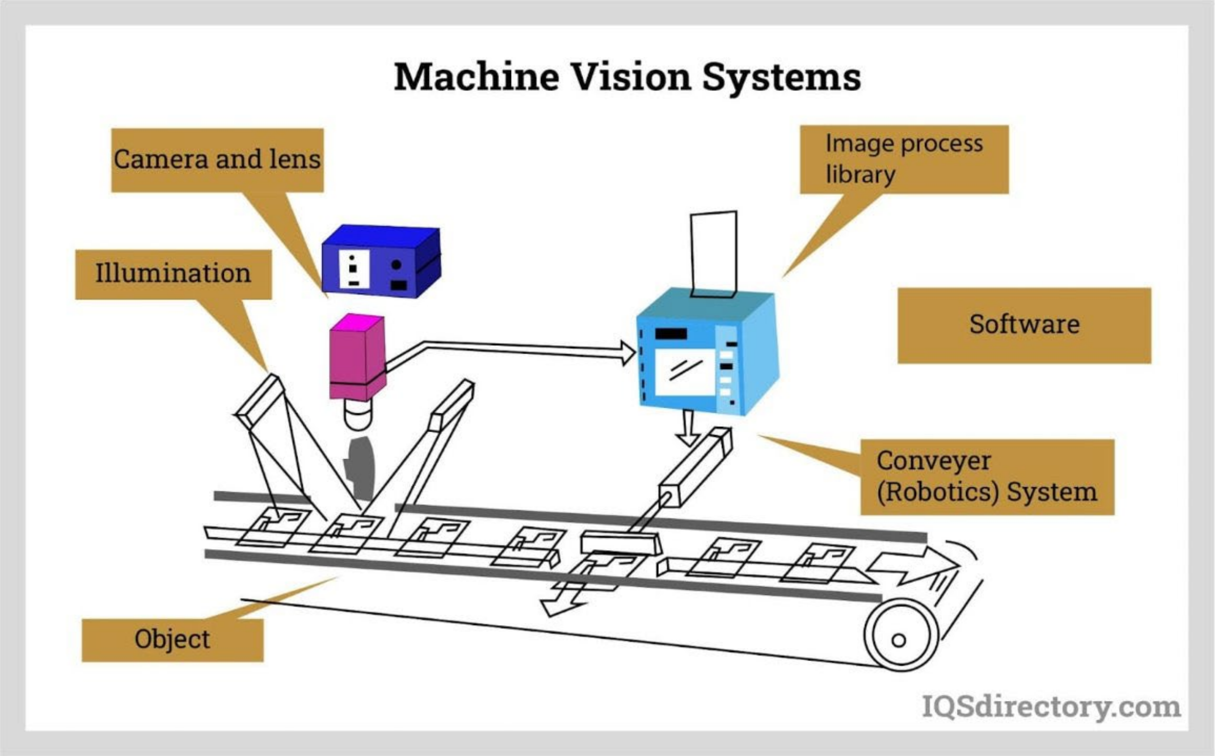
Modern tools can automate inspection

- Machine vision systems (cameras and image processing software)
  - High-resolution cameras capture product images
  - Computer algorithms detect defects
- Different options
  - 2D vision
  - 3D vision
  - Hyperspectral and multispectral imaging
  - Lasers and Structured Lighting
- Artificial Intelligence (AI) and Deep Learning
  - Neural networks for pattern recognition
  - Adaptive inspection

Automated Inspection Limitations

It isn't easy

- Highly complex
  - System design
  - Programming expertise
  - Avoid variabilities
    - Lighting, positioning, etc.
  - Large datasets needed
  - Integration into existing systems
  - Limited accessibility for SMEs



- A no-code approach
  - Modern tools can be used to simplify model creation without coding

Machine Learning Foundation

By: Dr. Mariadas Roshan

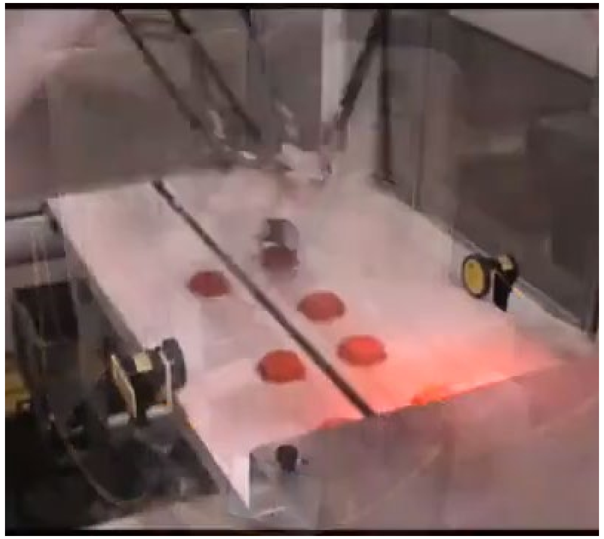
What Is AI?

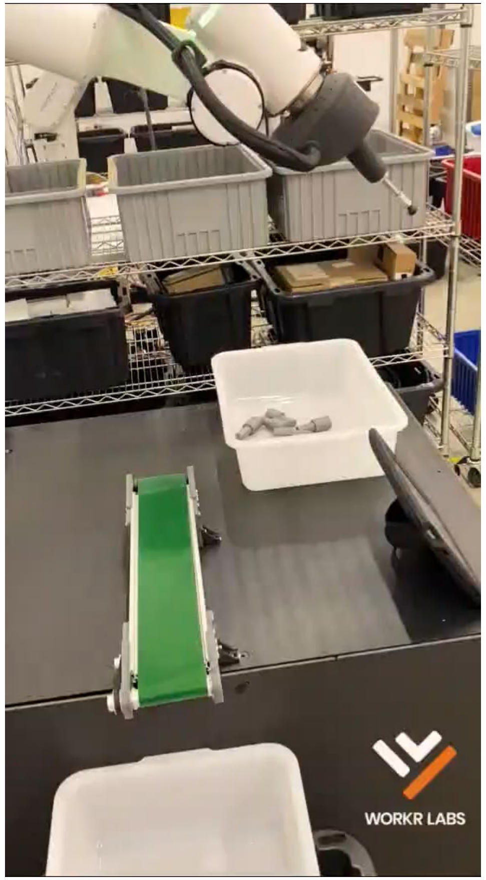
Artificial Intelligence

AI involves techniques that equip computers to emulate human behavior, enabling them to learn, make decisions, recognize patterns, and solve complex problems in a manner akin to human intelligence.

Artificial Intelligence

Example: Bin picking using robot





Applications: Autonomous Vehicles, Mobile phones, cloud services and emails.

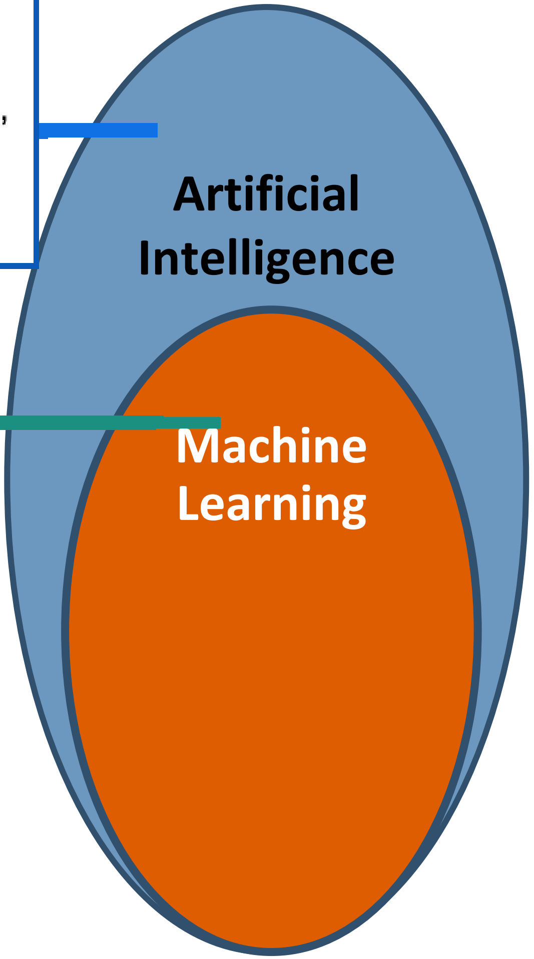
What Is Machine Learning?

Artificial Intelligence

AI involves techniques that equip computers to emulate human behavior, enabling them to learn, make decisions, recognize patterns, and solve complex problems in a manner akin to human intelligence.

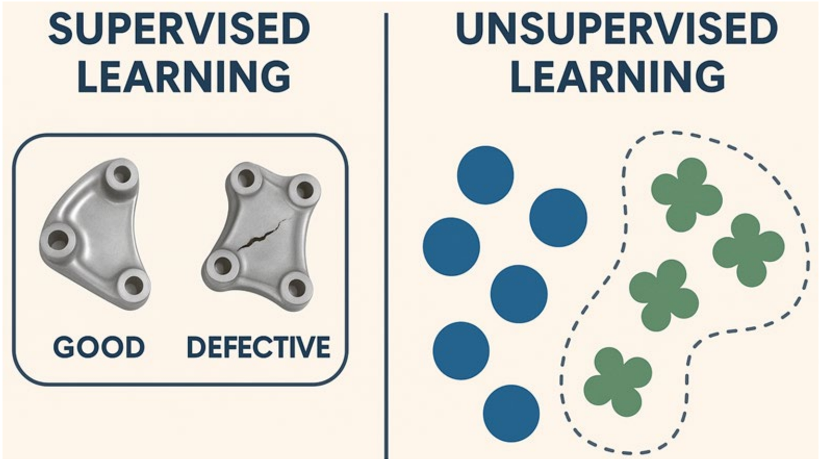
Machine Learning

ML is a subset of AI, uses advanced algorithms to detect patterns in large data sets, allowing machines to learn and adapt. ML algorithms use supervised or unsupervised learning methods.



Example: Bin picking using robot

Supervised Learning	Unsupervised Learning
Uses labelled data	Uses unlabelled data
Learns exact categories	Finds patterns/groups
Example: Good vs Bad parts	Example: Grouping similar defects
You know the correct answer	Model discovers structure



What Is Deep Learning?

Artificial Intelligence

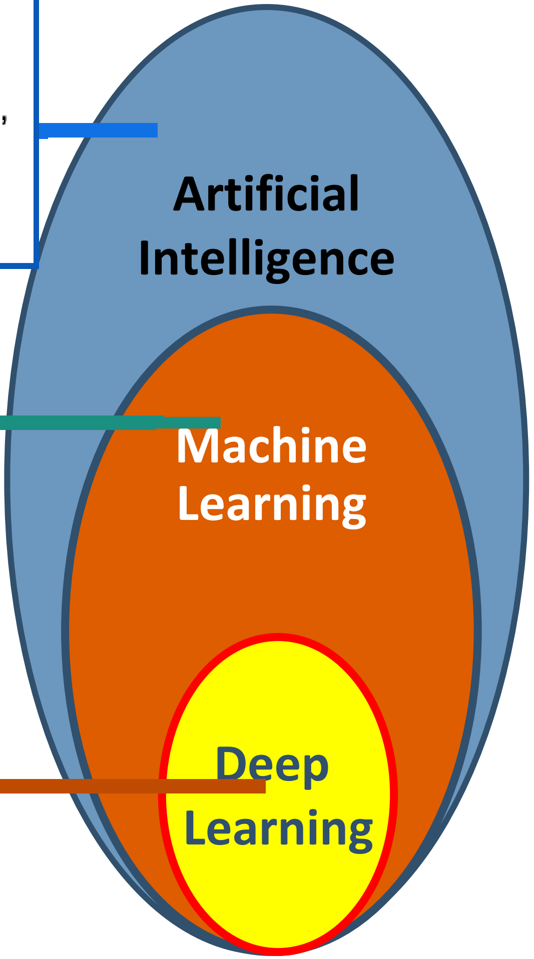
AI involves techniques that equip computers to emulate human behavior, enabling them to learn, make decisions, recognize patterns, and solve complex problems in a manner akin to human intelligence.

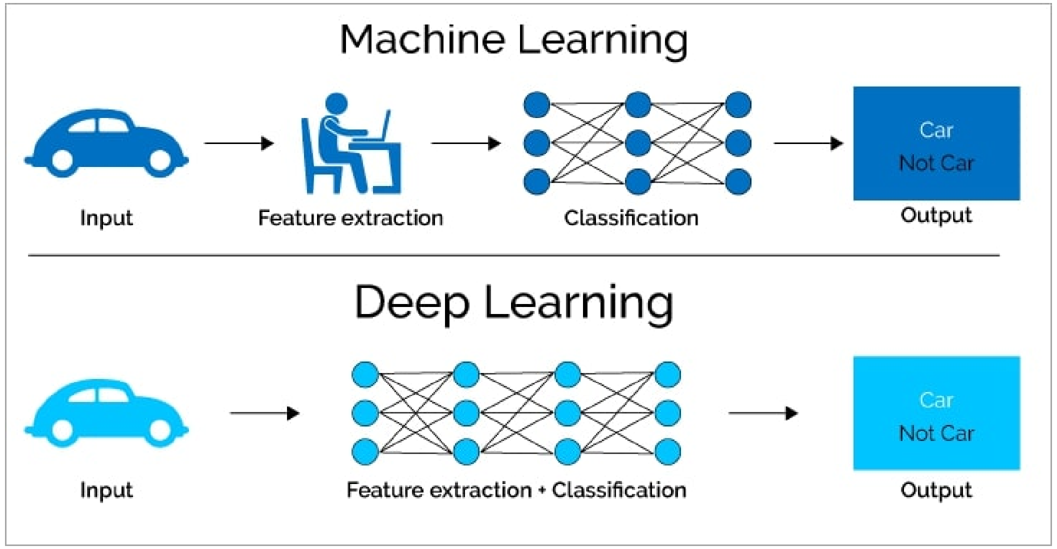
Machine Learning

ML is a subset of AI, uses advanced algorithms to detect patterns in large data sets, allowing machines to learn and adapt. ML algorithms use supervised or unsupervised learning methods.

Deep Learning

DL is a subset of ML which uses neural networks for in-depth data processing and analytical tasks. DL leverages multiple layers of artificial neural networks to extract high-level features from raw input data, simulating the way human brains perceive and understand the world.





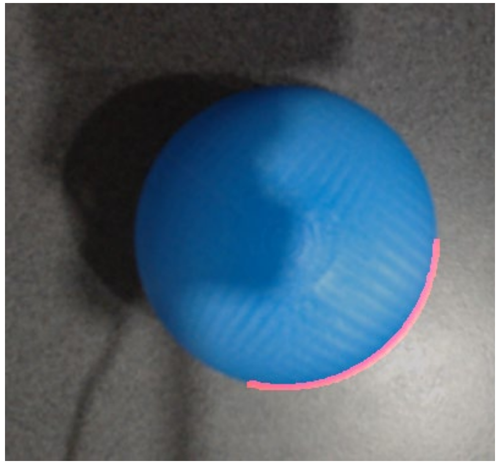
https://www.atomcamp.com/difference-machine-learning-deep-learning/

Analogy:

- Machine Learning = A technician with experience in 1 type of defect.
- Deep Learning = A senior inspector who has seen thousands of samples and can spot even subtle defects automatically, across different lighting, orientations, and shapes.

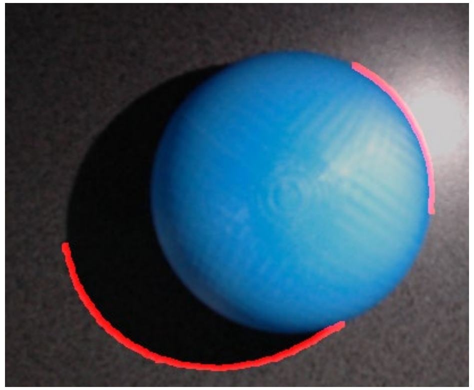
Why Machine Learning for Visual QA?

- Traditional rule-based vision fails under real factory variation
- ML learns complex patterns from examples, not fixed rules
- Detects subtle, irregular defects that rules cannot describe
- Fewer false positives → higher throughput
- Essential for flexible, robot-integrated QA systems.



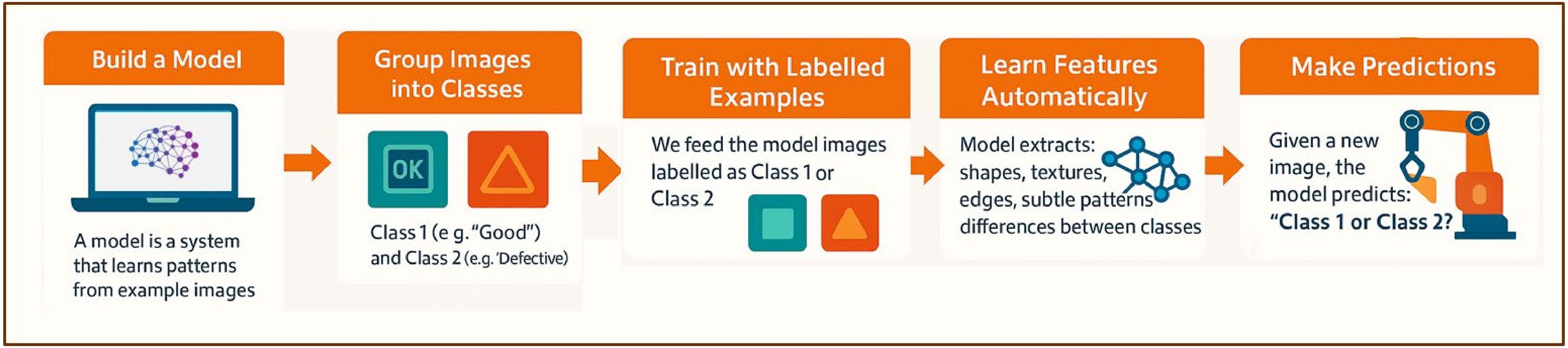
Normal room light

Without light



Focused light

What Does an ML Model Actually Do?



Build a Model → Group images into Classes → Train with labelled examples → Learn Features Automatically → Make Predictions

Machine Learning Parameters and Metrics

Build a Model → Group images into Classes → Train with labelled examples → Learn Features Automatically → Make Predictions

Training Parameters

- Classes
- Learning Rate
- Batch Size
- Epoch

Training Phase Metrics

- Accuracy per epoch
- Loss per epoch

Testing Metrics

- Accuracy per Class
- Confusion Matrix

Real-time Metrics

- Confidence Score

Training Parameters

1. Classes

➢ What is it?

- Groups or categories the model learns to recognise.
- Example in QA: “Good”, “Defective”, “Type A Defect”, “Type Defect”.

➢ Why does it matter?

- Clear, well-separated classes help the model learn correctly.
- If classes overlap (e.g., unclear defect labels), the model become confused.

➢ Impact on prediction:

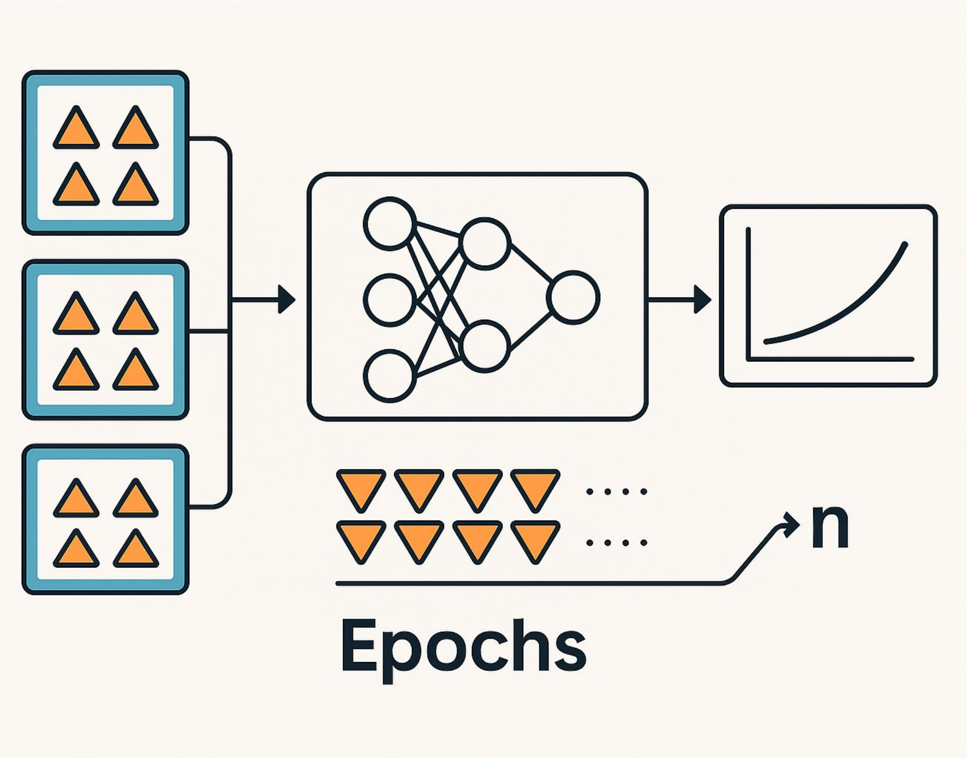
- Better class definitions → higher accuracy.
- Poorly separated classes → misclassification during testing.

Build a Model → Group images into Classes → Train with labelled examples → Learn Features Automatically → Make Predictions

Training Parameters

2. Epochs

- What is it?
  - One complete pass through the entire training dataset.
- Why does it matter?
  - More epochs = more opportunities for the model to learn patterns.
  - Too few epochs → underfitting (model hasn't learned enough).
  - Too many epochs → overfitting (model memorises training data, fails on new images).
- Impact on prediction:
  - Proper number of epochs improves generalisation.
  - Overfitting shows high training accuracy but poor real-world performance.



Build a Model → Group images into Classes → **Train with labelled examples** → Learn Features Automatically → Make Predictions

Training Parameters

3. Batch Size

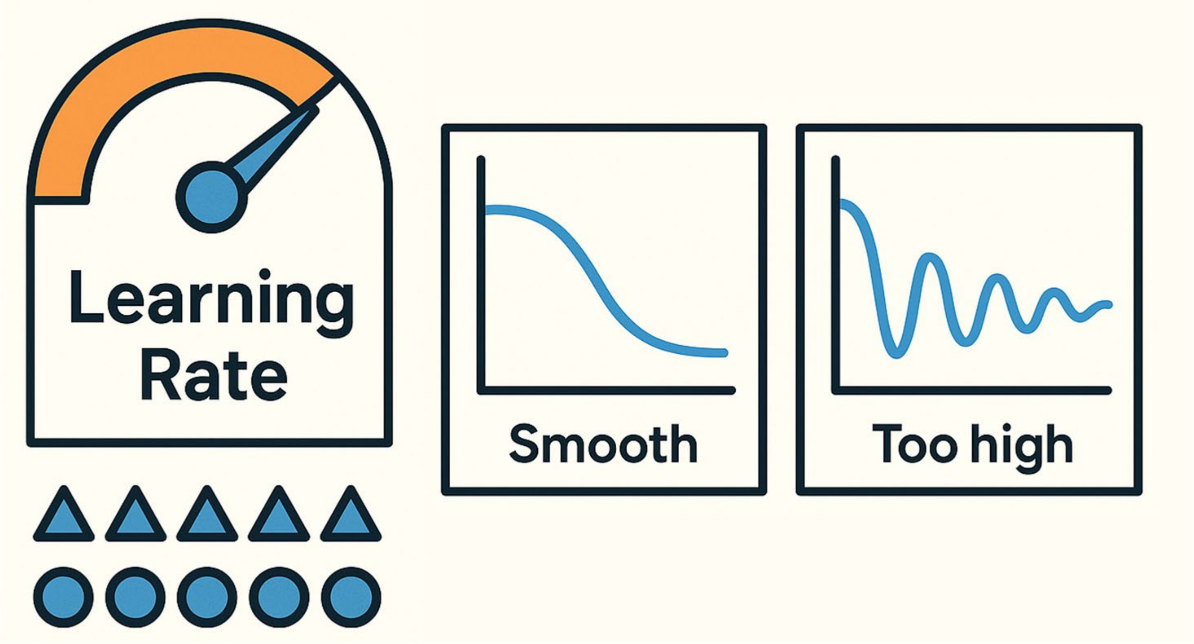
- **What is it?**
  - Number of images the model processes at a time during training.
  - Example: batch size of 16 = learns from 16 images before updating its parameters.
- **Why does it matter?**
  - Small batch = slower but more stable learning.
  - Large batch = faster training but may skip fine details.
- **Impact on prediction:**
  - Wrong batch size can cause unstable learning or poor generalization.
  - Balanced batch size improves consistency and accuracy.

Build a Model → Group images into Classes → Train with labelled examples → Learn Features Automatically → Make Predictions

Training Parameters

4. Learning Rate

- What is it?
  - How big a step the model takes when adjusting itself during training.
  - Think of it as the model's "learning speed".
- Why does it matter?
  - Too high → model jumps around, misses important patterns.
  - Too low → training becomes very slow or stuck.
- Impact on prediction:
  - Correct learning rate leads to stable, efficient learning.
  - Poor learning rate = inaccurate predictions or failure to converge.



Build a Model → Group images into Classes → Train with labelled examples → Learn Features Automatically → Make Predictions

Machine Learning Parameters and Metrics

Build a Model → Group images into Classes → Train with labelled examples → Learn Features Automatically → Make Predictions

Training Parameters

- Classes
- Learning Rate
- Batch Size
- Epoch

Training Phase Metrics

- Accuracy per epoch
- Loss per epoch

Training Phase Metrics

1. Accuracy per Epoch

➢ What is it?

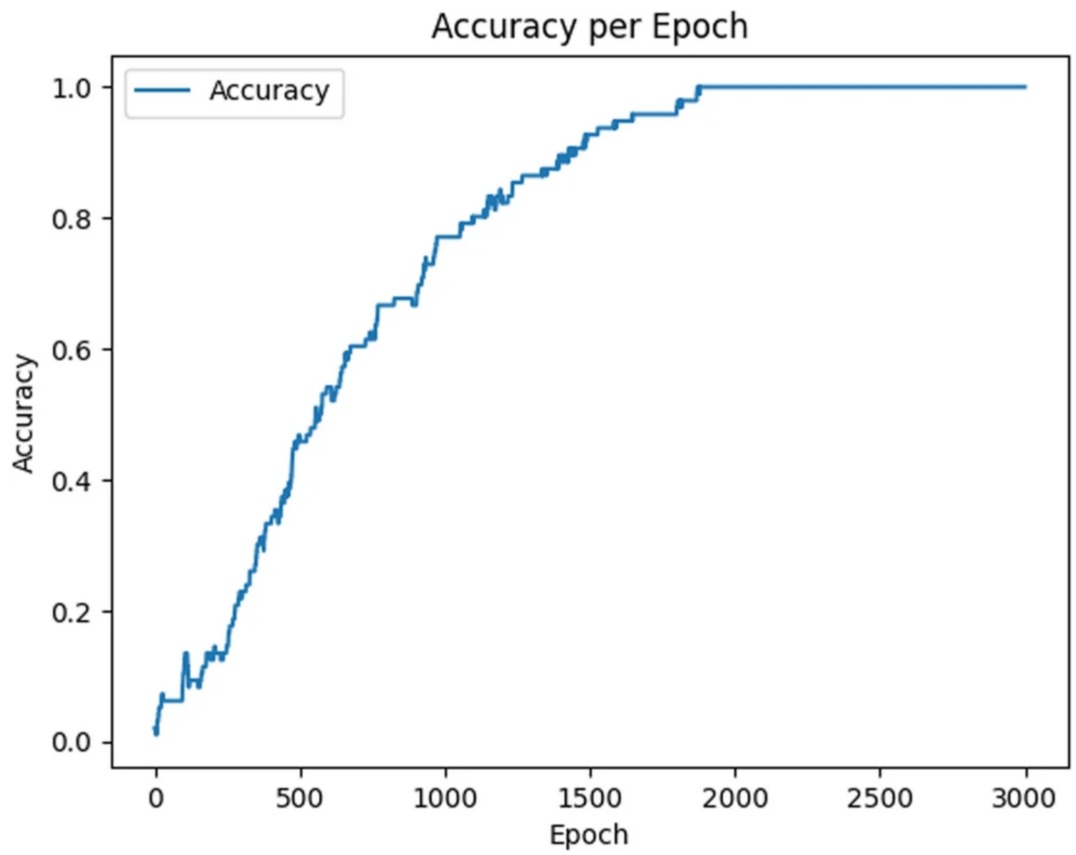
- Shows how the model's accuracy improves after each training cycle (epoch).

➢ Why it matters:

- Helps see if the model is learning properly.
- A steadily increasing accuracy curve = healthy learning.
- Flat or unstable curves indicate issues (bad learning rate, insufficient data, noisy labels).

➢ Industry impact:

- Ensures the final model is trained correctly and not underfitting.

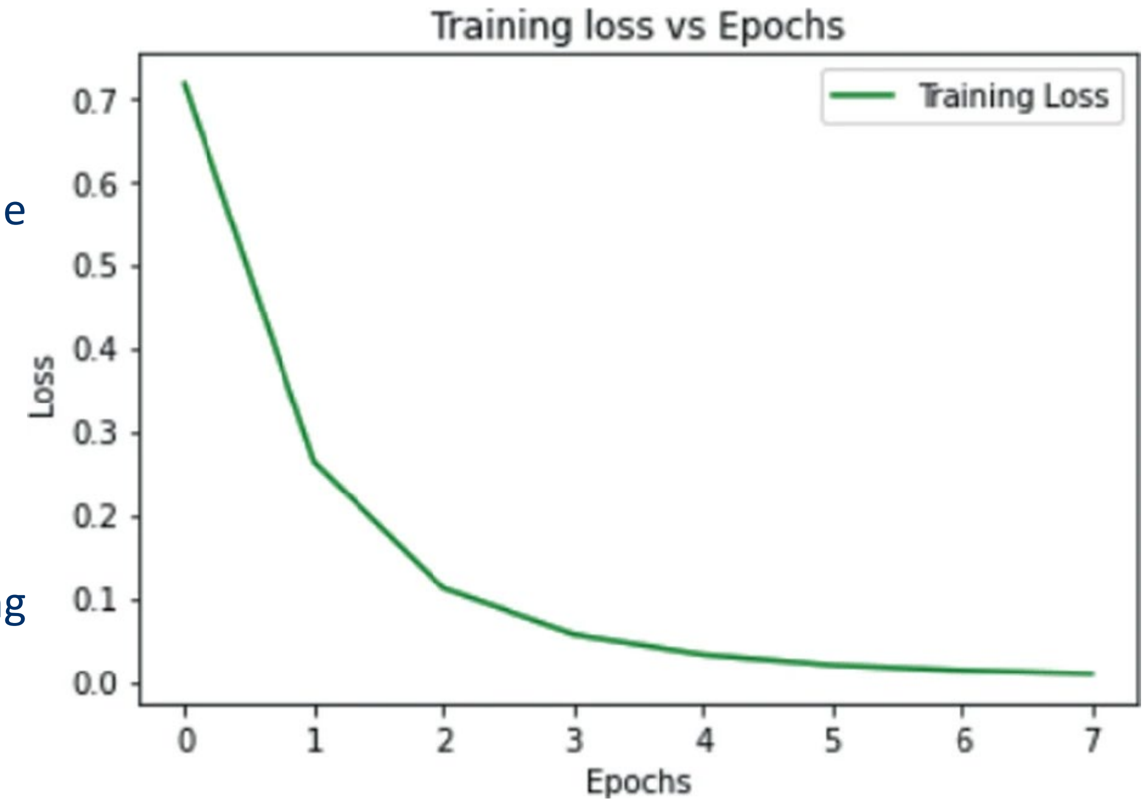


Build a Model → Group images into Classes → Train with labelled examples → Learn Features Automatically → Make Predictions

Training Phase Metrics

2. Loss per Epoch

- What is it?
  - Loss measures how far the model's predictions are from the correct answers.
  - Lower loss → better predictions.
- Why it matters:
  - Loss should go down steadily during training.
  - If loss increases or becomes unstable, the model is not learning well.



- Industry impact:
  - A low, stable loss value means the model will perform better in real QA environments.
  - Helps diagnose problems early, before deploying the model.

Build a Model → Group images into Classes → Train with labelled examples → Learn Features Automatically → Make Predictions

Machine Learning Parameters and Metrics

Build a Model → Group images into Classes → Train with labelled examples → Learn Features Automatically → Make Predictions

Training Parameters

- Classes
- Learning Rate
- Batch Size
- Epoch

Training Phase Metrics

- Accuracy per epoch
- Loss per epoch

Testing Metrics

- Accuracy per Class
- Confusion Matrix

Testing Metrics

1. Accuracy per Class

➢ What is it?

- Measures how well the model predicts each class individually.
- Example:
  - Class “Good”: 95% accurate
  - Class “Defective”: 88% accurate

➢ Why it matters:

- Some classes are easier to identify than others.
- Helps you understand if the model struggles with specific defect types.

➢ Industry impact:

- Ensures consistent-quality inspection across all product categories.
- Helps focus data collection on weaker classes.

Accuracy per class

CLASS	ACCURACY	# SAMPLES
Class 1	1.00	5
Class 2	1.00	6

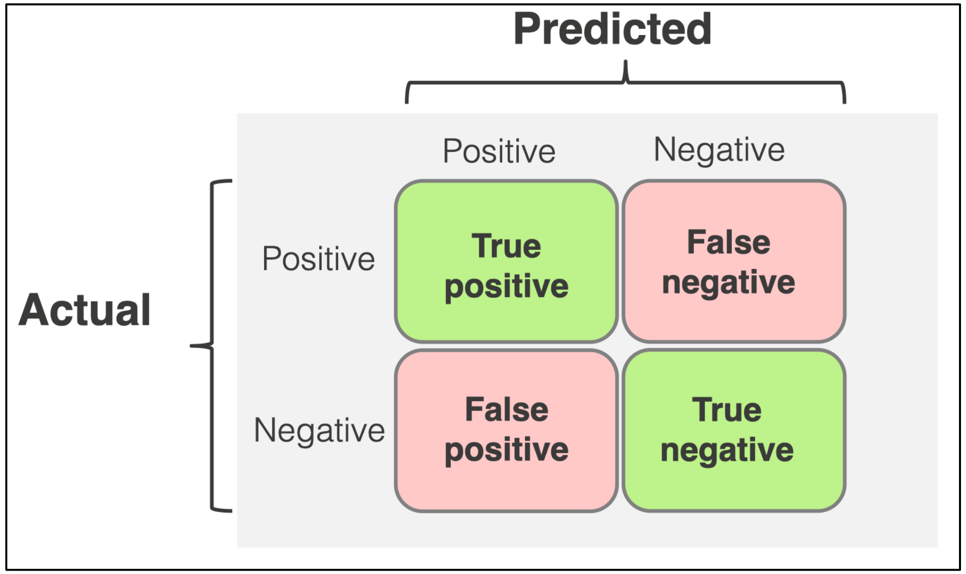
$$\text{Accuracy} = \frac{\text{Correct predictions}}{\text{All predictions}}$$

Build a Model → Group images into Classes → Train with labelled examples → Learn Features Automatically → Make Predictions

Testing Metrics

2. Confusion Matrix

- What is it?
  - A table showing exactly where the model gets things right, and where it makes mistakes.
    - Rows - True labels and Columns- Predicted labels
- Why it matters:
  - Shows if the model confuses two specific defect types.
  - Highlights false positives and false negatives.
  - Helps diagnose issues source.
- Industry impact:
  - Helps avoid costly misclassifications, such as passing a defective part.
  - Allows targeted improvements (e.g., adding more images for a tricky defect).



		Predicted		
		Dog	Cat	
Actual	Dog	24	6	30
	Cat	2	18	20

Build a Model → Group images into Classes → Train with labelled examples → Learn Features Automatically → Make Predictions

Machine Learning Parameters and Metrics

Build a Model → Group images into Classes → Train with labelled examples → Learn Features Automatically → Make Predictions

Training Parameters

- Classes
- Learning Rate
- Batch Size
- Epoch

Training Phase Metrics

- Accuracy per epoch
- Loss per epoch

Testing Metrics

- Accuracy per Class
- Confusion Matrix

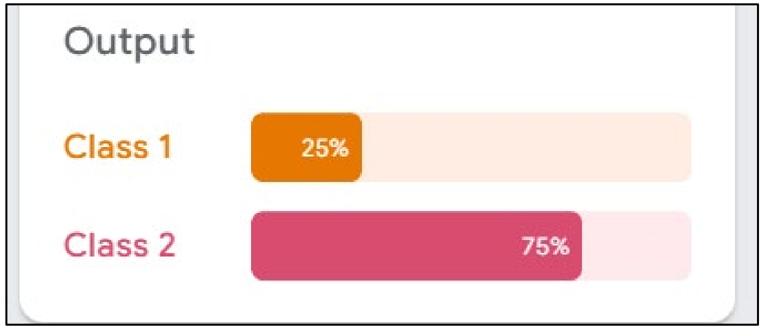
Real-time Metrics

- Confidence Score

Real-Time Metrics

1. Confidence Score

- **What is it?**
  - A number (0–100%) that tells how sure the model is about its prediction.
- **Why it matters:**
  - Higher confidence → model is more certain.
  - Low confidence → model isn't sure, may require review.
- **Industry impact:**
  - Helps decide when to trust the model vs when to double-check.
  - Useful for QA systems that need a threshold (e.g., reject if <80% confidence).



Build a Model → Group images into Classes → Train with labelled examples → Learn Features Automatically → Make Predictions

Machine Learning Parameters and Metrics

Build a Model → Group images into Classes → Train with labelled examples → Learn Features Automatically → Make Predictions

Training Parameters

- Classes
- Learning Rate
- Batch Size
- Epoch

Training Phase Metrics

- Accuracy per epoch
- Loss per epoch

Testing Metrics

- Accuracy per Class
- Confusion Matrix

Real-time Metrics

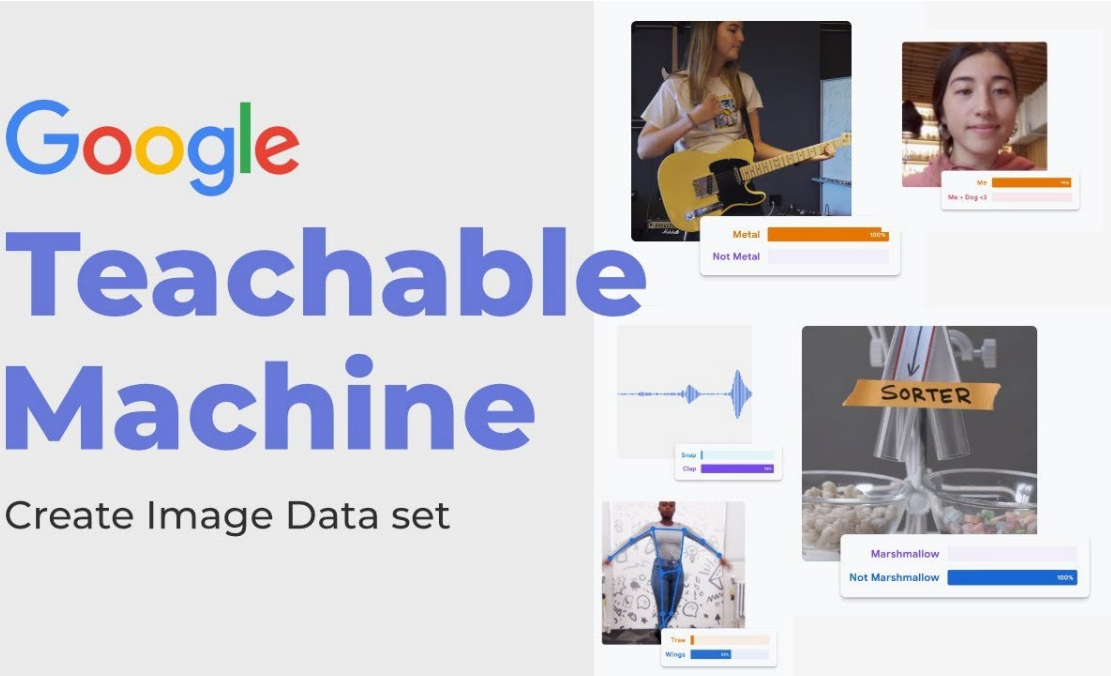
- Confidence Score

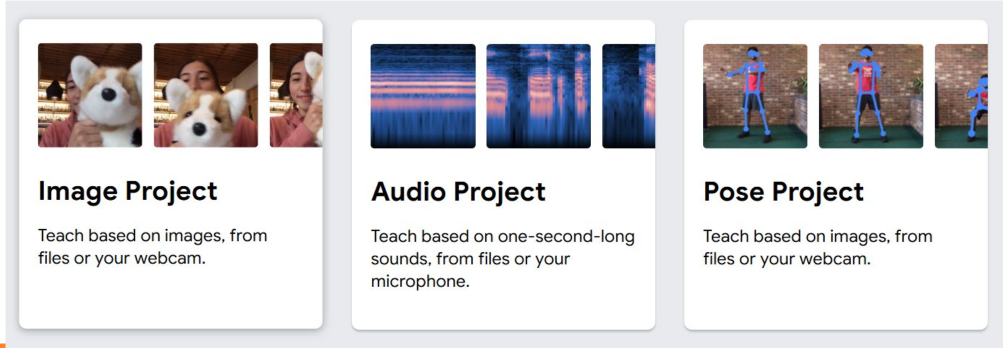
Hands-On Machine Learning Training Session

By: Dr. Mariadas Roshan

Introducing Google Teachable Machine (GTM)

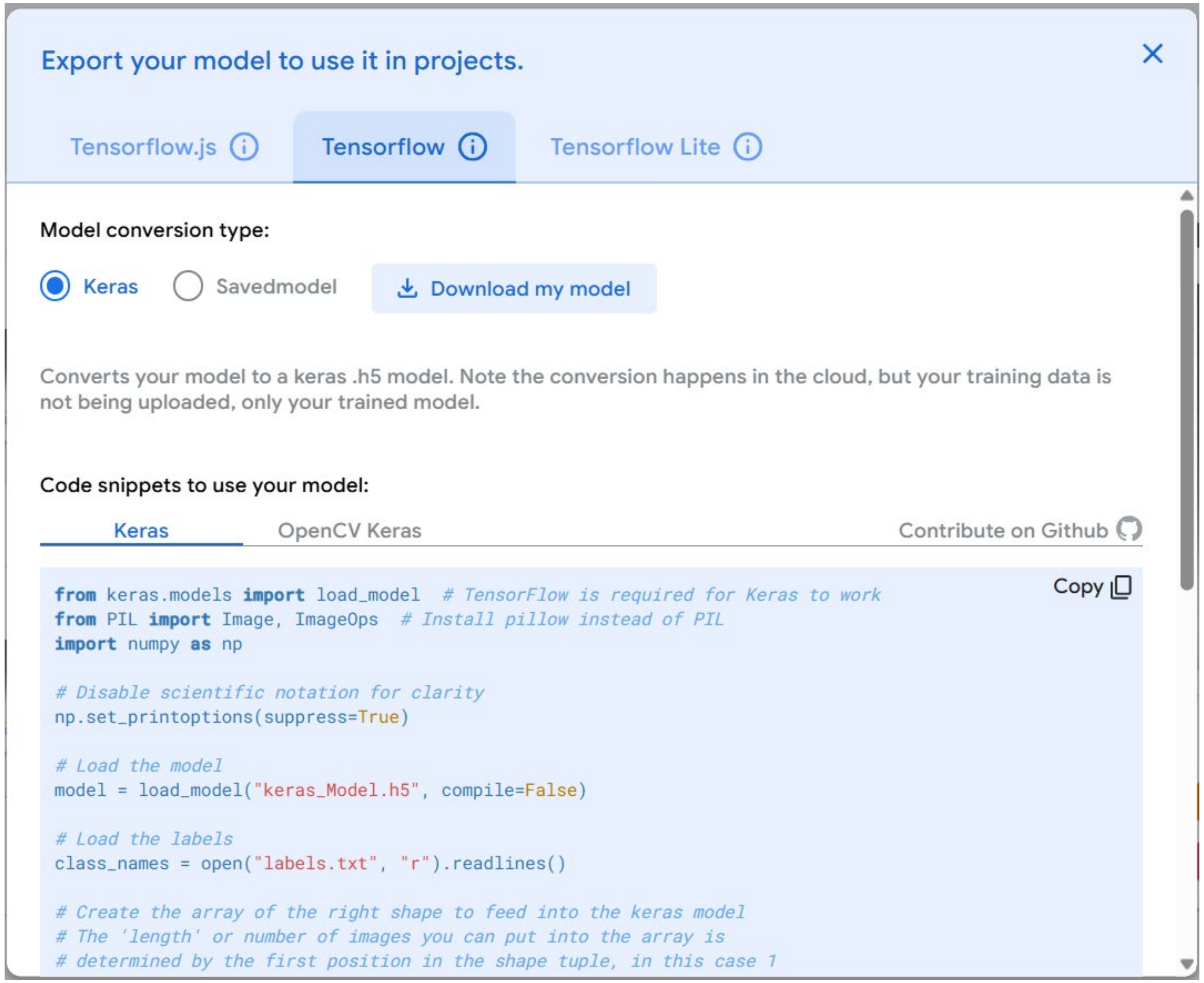
- What is Google Teachable Machine?
  - A simple, no-code machine learning tool from Google.
  - Let's you train image, sound, or pose recognition models without any programming.
  - Perfect for beginners and industry users.
- Limitations
  - Not meant for industrial accuracy or high reliability, however, best used as a teaching and prototyping tool, not production QA.
  - Limited training depth and customization
  - Not suitable for complex defects or large-scale datasets





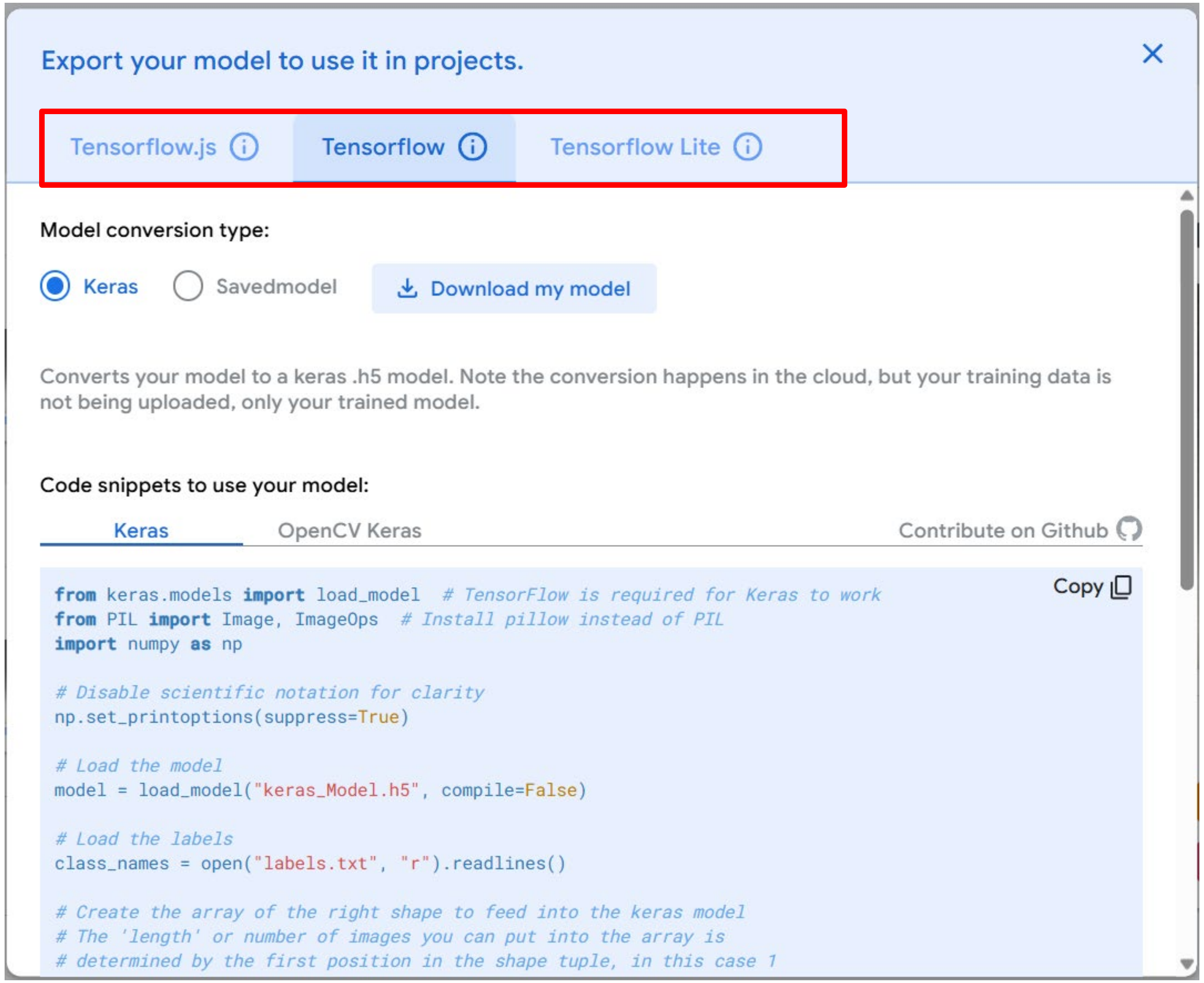
GTM - Step-by-Step Workflow – Exporting

- What Happens After Training?
  - Once the model is trained in Teachable Machine, we can export it for use in other applications.
  - Exporting gives us the actual files the ML system uses to make predictions.



GTM - Step-by-Step Workflow - Exporting

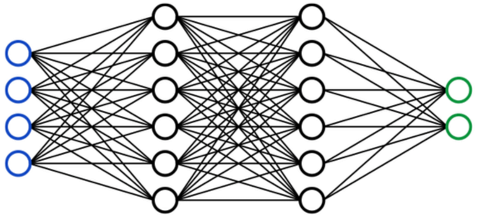
- Where Can the Exported Model Be Used?
  - Robot vision systems (integration with cobots and QA work cells)
  - Edge devices such as Raspberry Pi, NVIDIA Jetson, or microcontrollers
  - Web applications
  - Offline inspection tools
- Export Options in Teachable Machine
  - TensorFlow/Keras (.h5) → For Python, robotics, QA pipelines
  - TensorFlow Lite (.tflite) → For embedded or mobile devices
  - TensorFlow.js → For web-based applications.



GTM - Step-by-Step Workflow - Exporting

➢ What Are the Keras Files?

1. Keras Model (.h5 file)
   - Keras is a widely used machine learning framework built on top of TensorFlow.
   - The .h5 file contains the trained model: The learned patterns, the features extracted, and the final classification logic.
   - This file is what we load into other software or Python scripts to use the model.



2. Labels File (labels.txt)
   - Contains the names of the classes you trained (e.g., Good, Defective).
   - The model uses this file to understand what each prediction corresponds to.
   - Without labels, the model would only output numbers.

Industry Case Study

By: Munia Ahamed

Integrated Quality Control Framework

Combining PAF Model • Technology Readiness • ROI Methodology for Manufacturing Excellence

🎓 Theoretical Foundations

PAF Model: Cost categorization
Juran: Plan→Control→Improve
DMAIC: Systematic improvement
Swiss Cheese: Multi-layer defense
Lean/Six Sigma: Waste & variation
SPC/DOE: Statistical control

⚡ Technology Readiness

✓ Product Complexity Assessment
✓ Process Maturity Evaluation
✓ Human vs. Cobot Matrix
✓ Safety Requirements (ISO)
✓ ROI-based Implementation

PAF Cost Optimization Model

PREVENTION
2-5% of Sales
Training • Poka-Yoke • APQP • Planning

APPRAISAL
2-4% of Sales
Inspection • SPC • Testing

FAILURE
<3% Target
Rework • Scrap • Returns

Optimal Balance
↑ Prevention Investment = ↓ Failure Costs
Target ROI: 200-400%

🚀 Technology Integration

Manual Baseline | Data Analytics | Automation Tools | Cobots System

Implementation Strategy:

- Start with human optimization
- Add statistical process control
- Integrate automated inspection
- Deploy collaborative robots

💰 Savings & Benefits

- 15-30% defect reduction Year 1
- 50-70% rework cost savings
- 90%+ detection rate
- Multi-layer defense system

Industry Target: <4% quality costs
(vs. 15%+ for poor performers)

📊 ROI Calculation Methodology (No Numbers)

1
Current State
Map & Measure

→

2
PAF Analysis
Cost Breakdown

→

3
Theory Savings
DMAIC, Lean, SPC

→

4
ROI Formula
(S-I)/I × 100%

→

5
Validation
Track & Optimize

Bringing It All Together

By: Dr. Mariadas Roshan

From Workshop to Workplace

1. From Concept to Factory Floor

- Today's GTM activity mirrors the same workflow used in industry: Capture → Train → Test → Deploy
- The difference in real factories is mainly:
  - More robust hardware
  - Better lighting control
  - Structured data pipelines
  - The underlying ML logic remains the same

2. Industrial ML Tools & Models

- Low-/No-Code (SME-ready): Roboflow (vision datasets + training), Edge Impulse (embedded ML), Microsoft Custom Vision (fast cloud training), Siemens Industrial Edge (shop-floor integration)
- Advanced Models (for later/partners): YOLO (real-time detection), EfficientDet, FoundationPose, etc.

From Workshop to Workplace

4. Scaling Path for SMEs

- Move from pilot → production gradually

5. Program 4 – Real Industry Collaborations

- QA frameworks for medical manufacturing
- Quality inspection & classification for steel coil production (InfraBuild)
- Robotic adaptive welding (SMENCO)

6. Where can we help you?

- Testing a concept
- Running a pilot
- Evaluating cobot integration
- Exploring a new automation challenge.


Q&A

Final Reflection and Closing Remarks

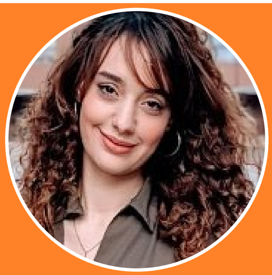
By: Dr. Michelle Dunn

THANK YOU

Dr. Michelle Dunn, Swinburne

Email:

jdunn@swin.edu.au



Dr. Katia Bourahmoune, UTS

Email:

Katia.Bourahmoune@uts.edu.au

A/Prof. Chris McCarthy, Swinburne

Email:

cdmccarthy@swin.edu.au

A/Prof. Gavin Paul, UTS

Email:

Gavin.Paul-1@uts.edu.au

Dr. Mariadas Roshan, Swinburne

Email:

mroshan@swin.edu.au

Ms. Munia Ahamed, UTS

Email:

Munia.Ahamed@student.uts.edu.au

If you have questions, want to explore potential collaboration, or need support with quality-assurance, robotics, or AI projects, please reach out to our team