Custom Models

Custom Model use cases

Recognize specific structures in satellite images

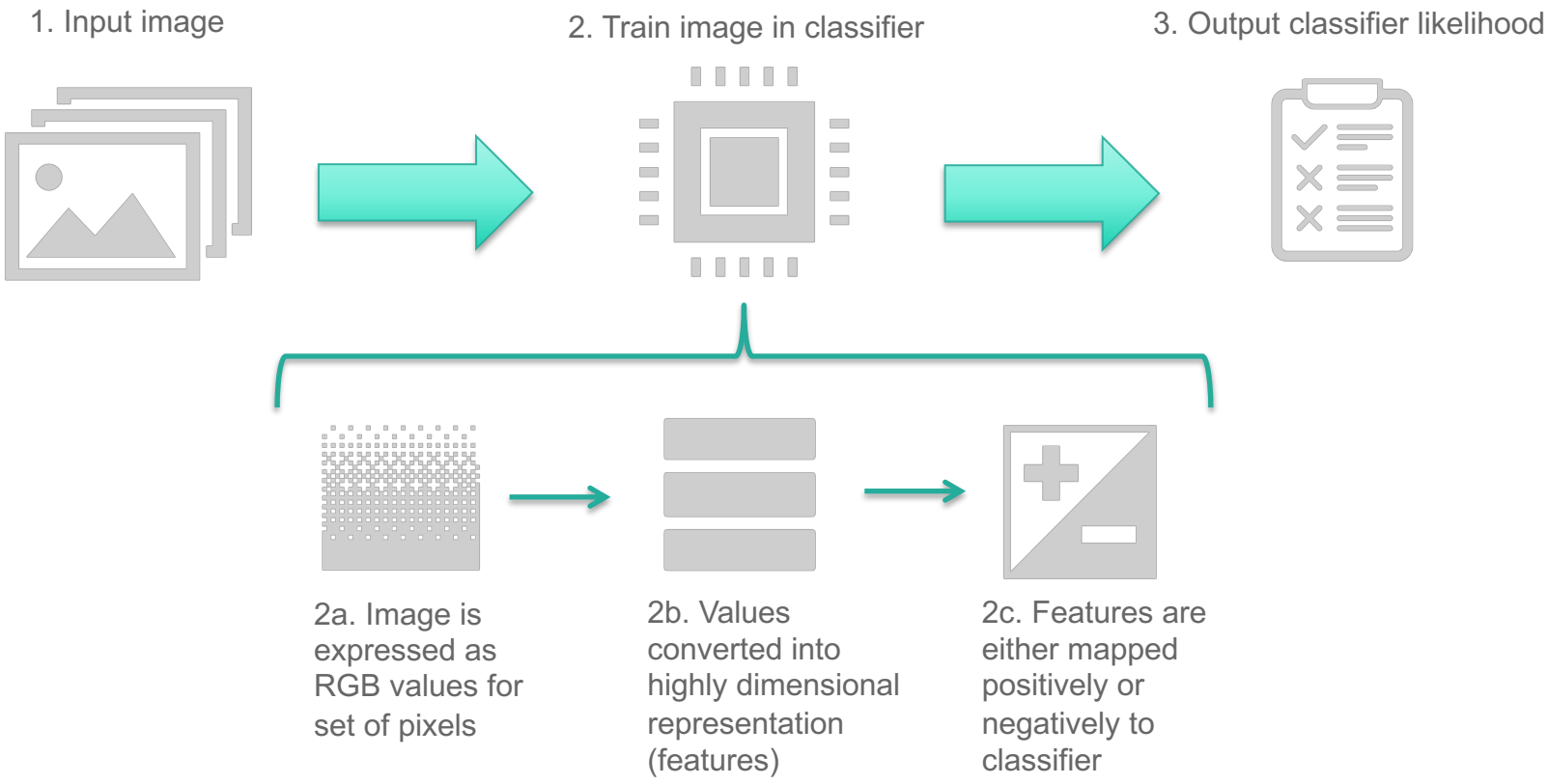
Spot cracks in pipe

Pinpoint rust or corrosion on infrastructure

Flag known defects in a manufacturing process

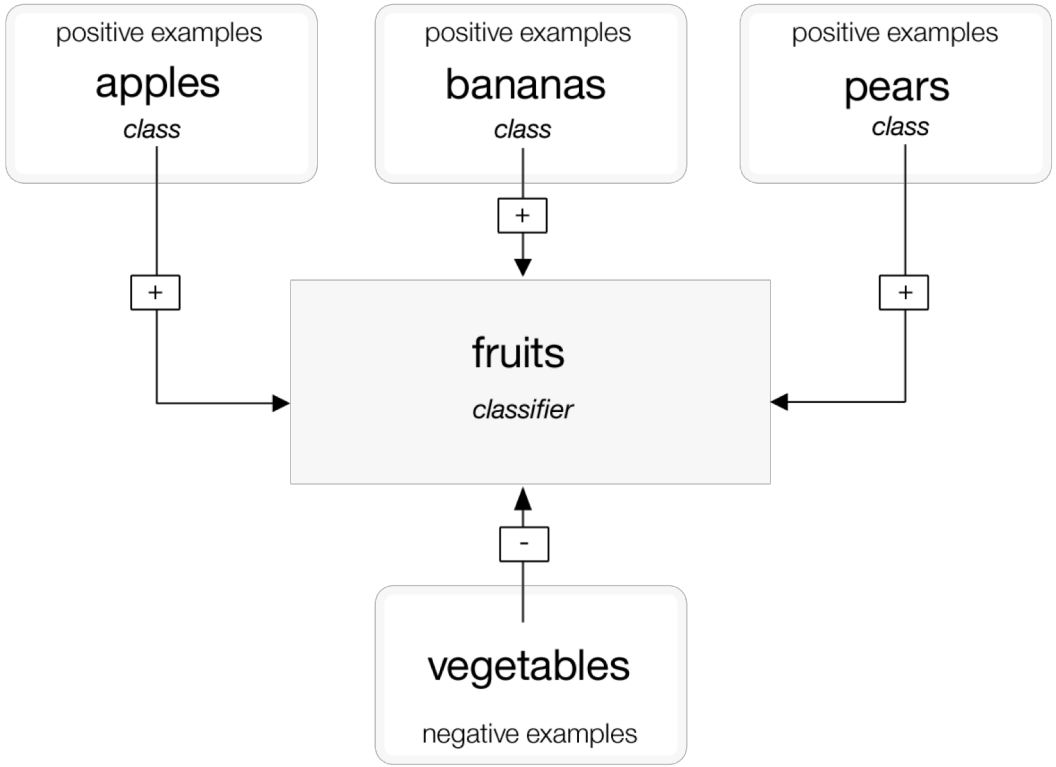
Process for Creating Custom Classifiers

A **custom classifier** is a group of **classes** that are trained against each other.



Process for Creating Custom Classifiers

A **custom classifier** is a group of **classes** that are trained against each other.



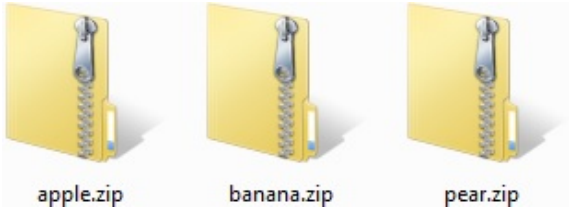
Process for Creating Custom Classifiers

A **custom classifier** is a group of **classes** that are trained against each other.

Before you can upload your images to Watson Studio (or using API), you need to organize your images in **.zip files**

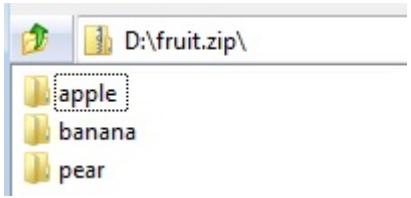
Option 1:

- One .zip file for each class



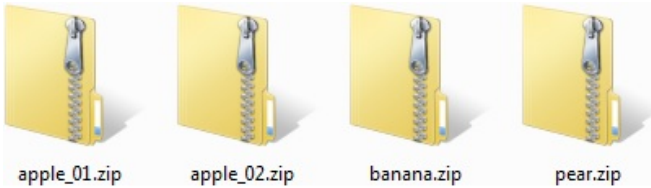
Option 2:

- One .zip file for all images with images separated into one folder for each class.



Option 3:

- Divide images of a class into multiple .zip files



Step 1: Image Preparation

“Clients who closely control their image training processes observed greater than 98% accuracy”

Image collection

- Positive classes
- Negative class (optional)
- At least 10 images per class
- **Recommended** 150 to 200 per Zip
- Max 10’000 images, 100 Mb per Zip
- Consider tools

Image pre-processing

- Subject matter of classifier is 1/3
- Consider for detecting details
- Tile, zoom, crop, resize, scale

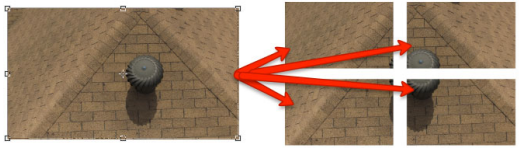


Image quality and limitations

- Light, exposure, angle, focus, color, background
- Min size 32x32, **recommended** 244x244

Training and Testing sets

- Have some similarity in both sets

Training set

Good

Testing set

Bad

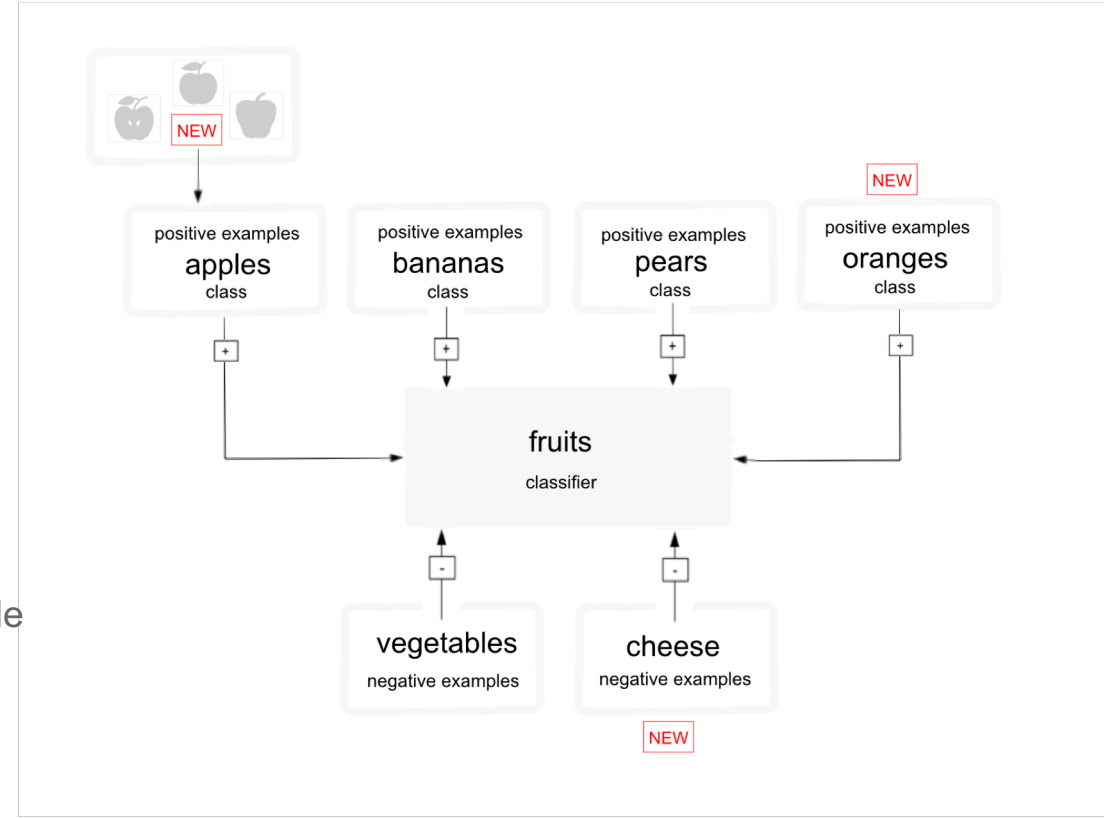
Step 2: Training and retraining

Training insiders

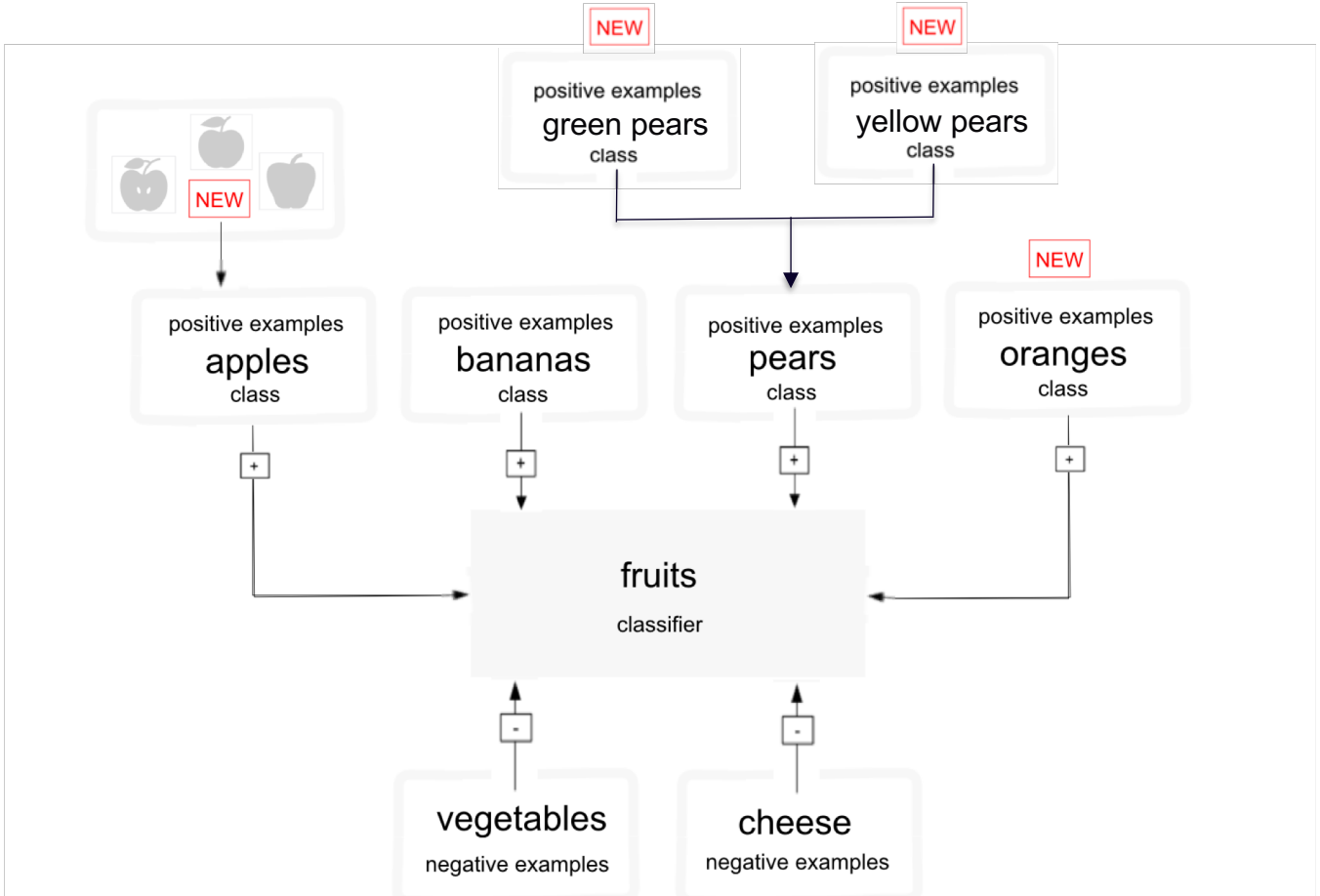
- One classifier = one model
- Positive from other classes are trained as negative
- Service performs Statistical classification

Updates

- By adding new set of positive images to existing classes
- And/or adding new set of positive images as a new class
- Suffix “_positive_examples” is required
- Splitting class definition is possible but be CAUTIOUS



Step 2: Training and retraining



Step 3: Score your classifier

Scores

- Score is the response to a classification.
- Score defines the “true” or “false” positive or negative conditions of an image against a class.
- The `/classify` method produces a score between 0.0 and 1.0 for each image for each class.
- Custom classifiers use binary "one versus the rest" models to train each class against the other classes.

What do the scores mean?

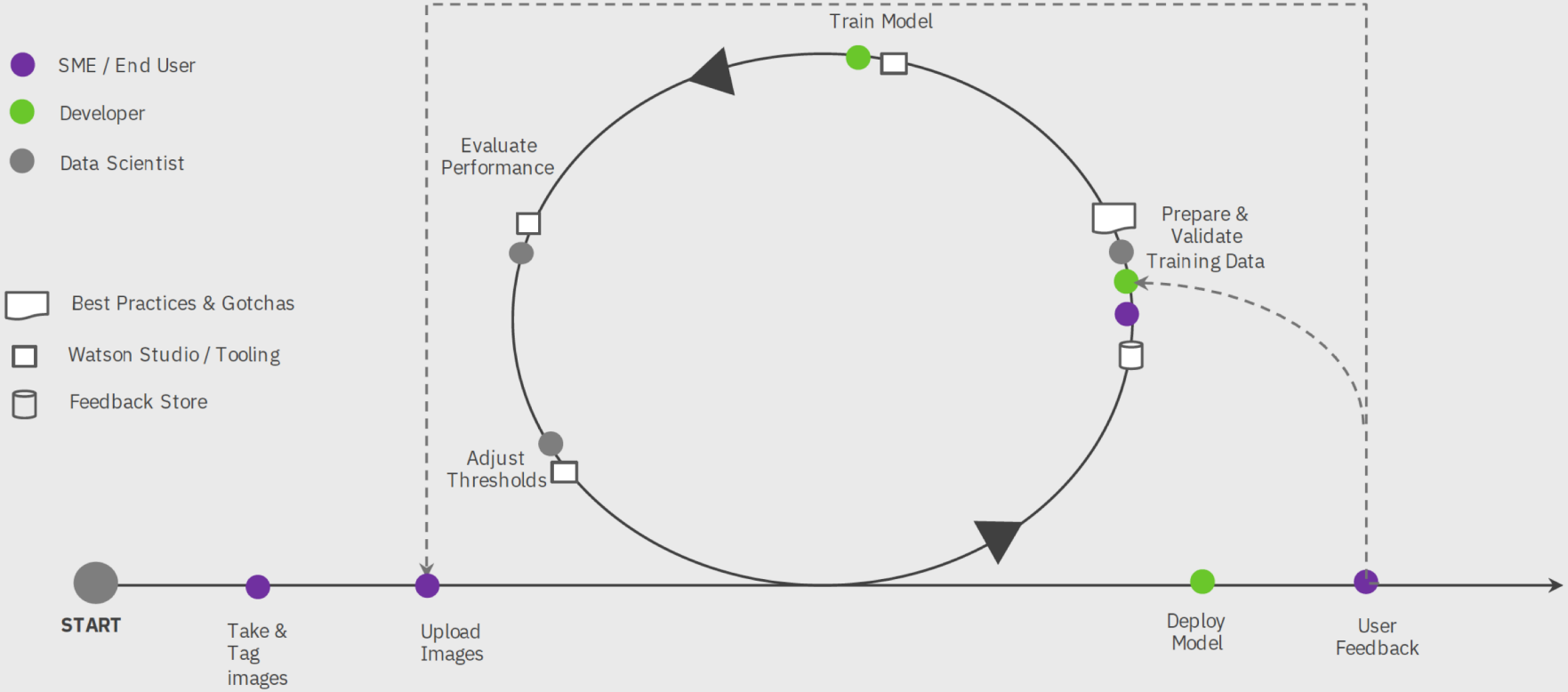
- The scores are comparable indicators, with a range from 0.0 to 1.0.
- The scores for custom classifiers are not comparable to the scores returned by the General classifier (which has classifier_id: "default")

How to evaluate classifier accuracy?

- Assemble a set of labeled images "L" that was not used in training the classifier.
- Split L into two sets, V and T - validation and testing.
- Run V through your classifier and pick a score threshold "R" which optimizes the correctness metric you value, such as top-5 precision, across all of V.
- From T, select a random subset "Q" and classify it using your classifier and "R". Compute the probability of a correct classification on Q. That's one experiment.
- Repeat step 4 with a different subset Q from T, then compute the average % correct across all experiments.

Continuous Learning Approaches

Continuous learning – Lifecycle & Ecosystem



Power Line use case

Objective

1. Identify where on a tower there may be deterioration
2. Classify what level of rust deterioration
3. Determine confidence to take action

Current State

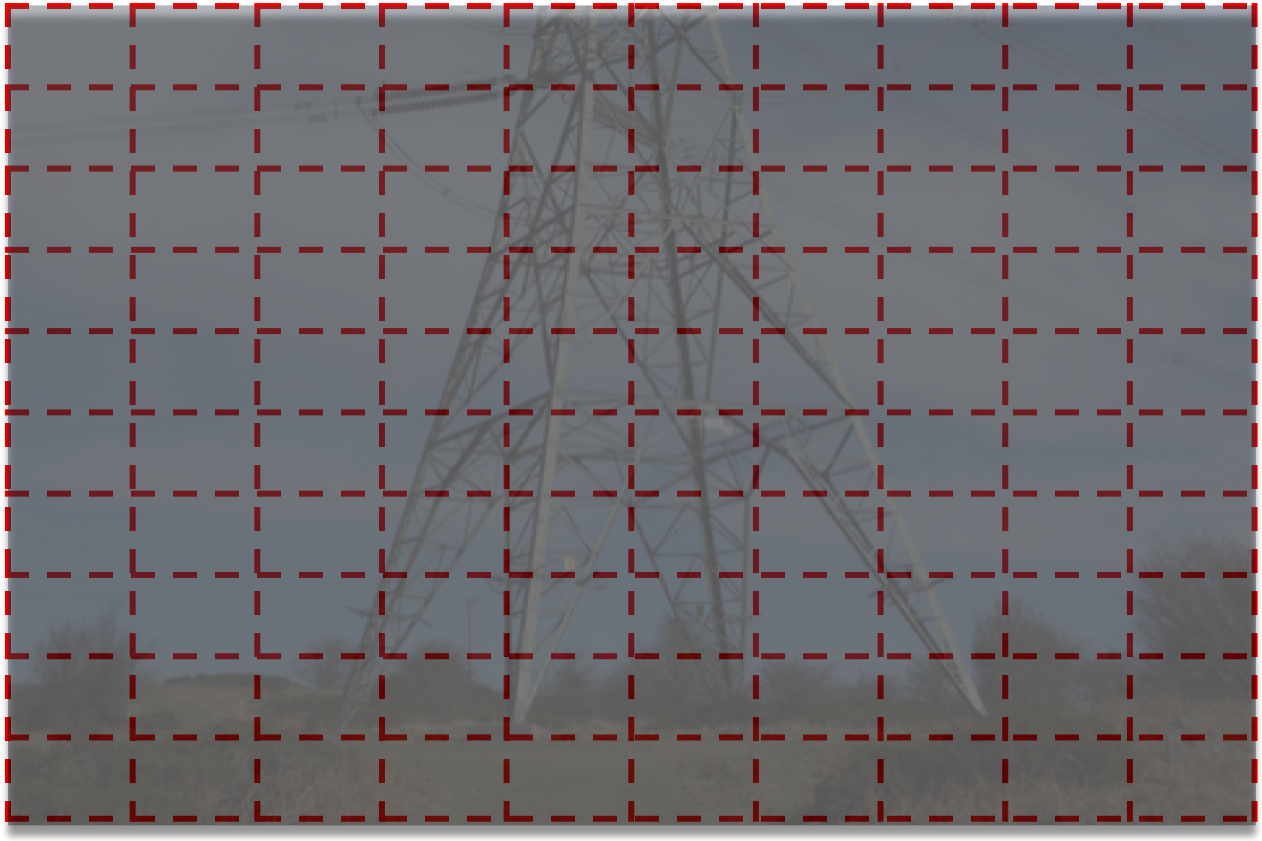
-Helicopters fly around
-Manually “zooming in” on images to determine rust areas
-Manually classification

Challenges

- Costly
- Time intensive, manual
- Fraught with human errors and judgement discrepancies between operators and technicians

Example of raw image

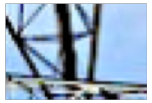
Step 1: Divide raw image into grid to classify where in image to “zoom in”



Step 2: Create high-level “is it metal?” classifier. Train VR what to look for

Train Watson using examples for each level of classification you are looking for mimicking human reasoning.
Guidelines for good training: https://www.ibm.com/watson/developercloud/doc/visual-recognition/customizing.shtml#goodtraining

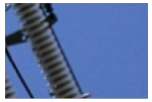
Class 1: Metal Structure

Training Example | Training Example
Training Example | Training Example
Training Example | Training Example
Training Example | Training Example

Class 2: Other Structure

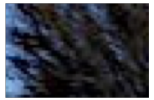


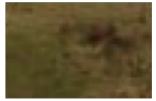



Training Example | Training Example
Training Example | Training Example
Training Example | Training Example
Training Example | Training Example

Negative: Non-Metal)








Training Example | Training Example
Training Example | Training Example
Training Example | Training Example

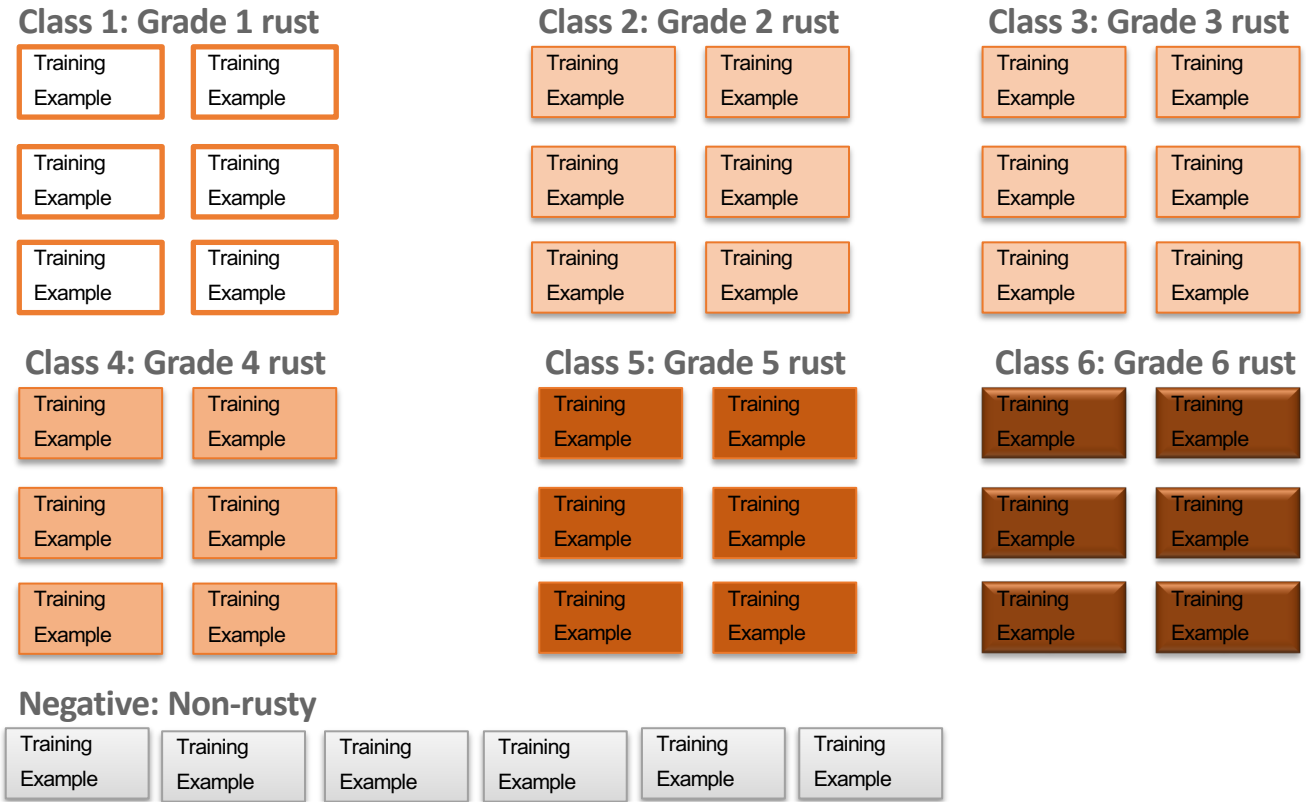
Step 3: Determine confidence if metal or not

Classifier: Metal Structure?

metal_structure .87
other_structure .13

Step 4: Train for rust classification types

Train Watson using examples for each level of classification you are looking for mimicking human reasoning.
Guidelines for good training: https://www.ibm.com/watson/developercloud/doc/visual-recognition/customizing.shtml#goodtraining



Step 5: Determine confidence of rust type

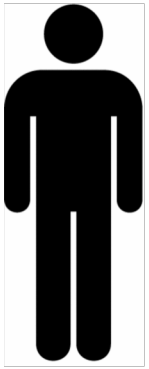
Classifier: Rust Type?

Rust_level_5	.67
Rust_level_6	.41
Rust_level_4	.28
Rust_level_3	.08
Rust_level_2	.04
Rust_level_1	.02

Step 6: Re-train and improve confidence over time

Classifier: Rust Type?

Rust_level_5	.67
Rust_level_6	.41
Rust_level_4	.28
Rust_level_3	.08
Rust_level_2	.04
Rust_level_1	.02



"That **is level 6 rust** for sure! I'll send that back in as a training example to Watson! I want Watson VR to be more confident next time in this classification."

Step 7: Rinse and repeat. Watson VR improves over time

Classifier: Rust Type?

Rust_level_6	.89
Rust_level_5	.41
Rust_level_4	.28
Rust_level_3	.08
Rust_level_2	.04
Rust_level_1	.02