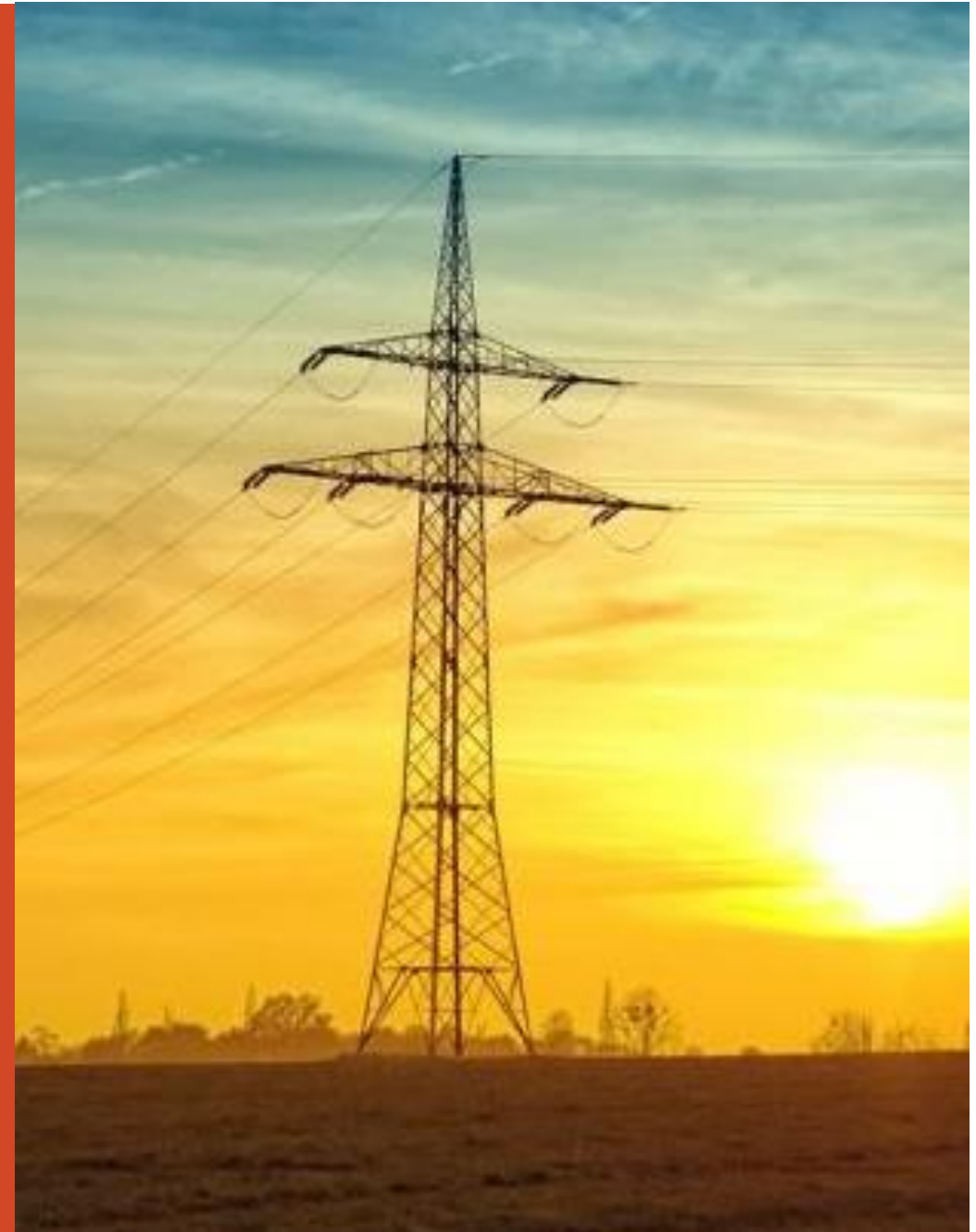


AI-Driven Power Pole Inspections

Case Study

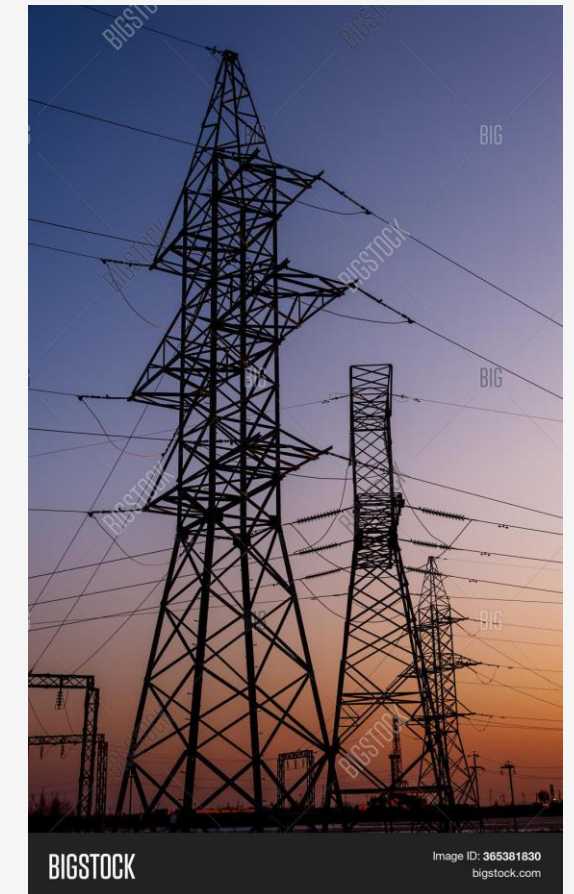


Overview

1. Data
2. AI Pipeline (Focus)
3. Tools
4. Future

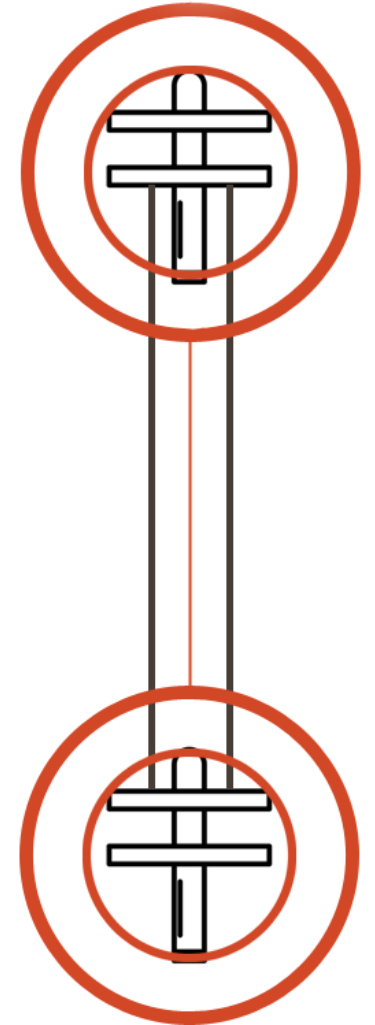
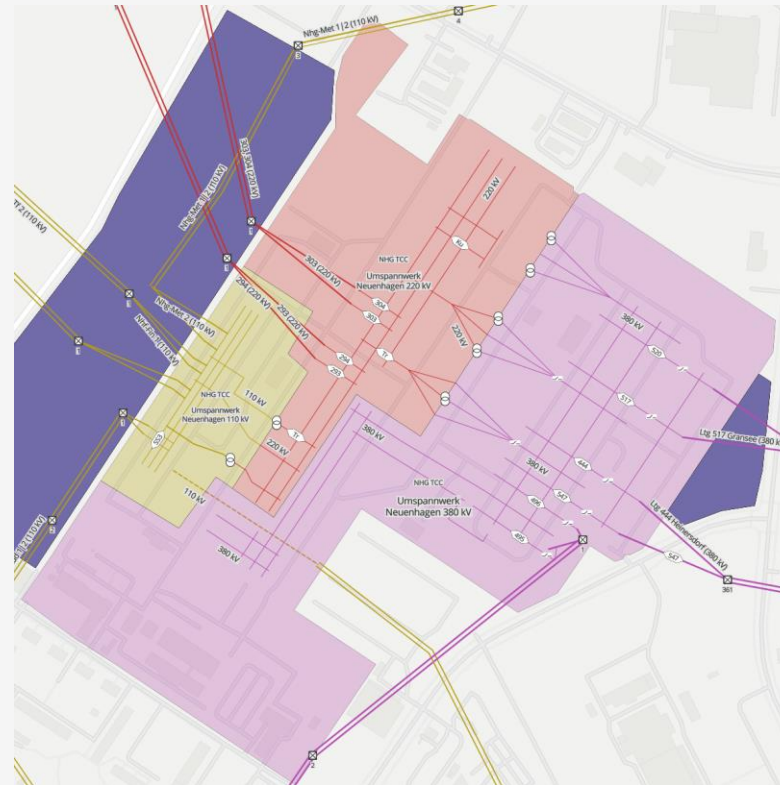
Data

1. Videos or images? Image pose estimation is difficult with repetitive structures (metal power post). Videos are better in this case.
2. Multiple images of the power post from different angles and distances. Close up images of each point of interest? (pole cap, insulator, etc..)
3. IMU (Accelerometer, Gyroscope, Magnetometer) of the drone
4. Infrared images? Maybe some defects (like rust) don't reflect infrared?
5. Do we need 3D? Photogrammetry? 3D Sensors? Attention: very fine structures!
6. Camera intrinsics, camera poses, sparse point cloud with observations (and dense 3d mesh reconstruction). This can be extracted from many photogrammetry systems and should preferably be done by the user (This way we know the camera path was trackable and reconstructable. Detecting a defective recording would be too late on our own server/pipeline. Less computations on our side).
7. Do we have a map of all power posts with their location and maybe meta data (#isolators, material, #attachments, #cables, etc.). Can we get it? Google maps or OSM (OpenStreetMaps)?



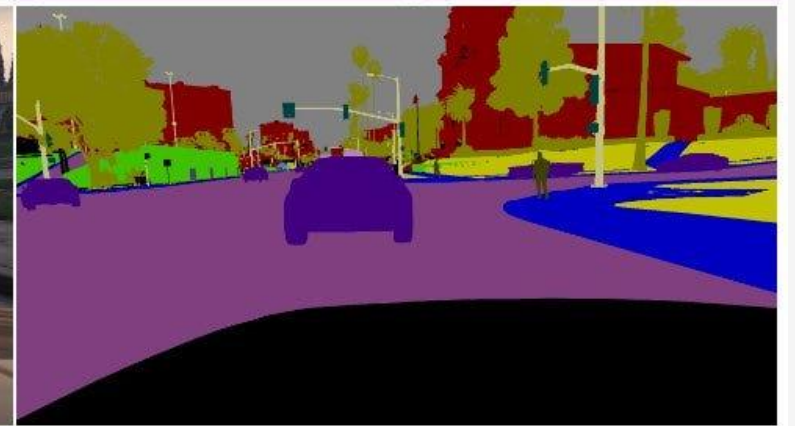
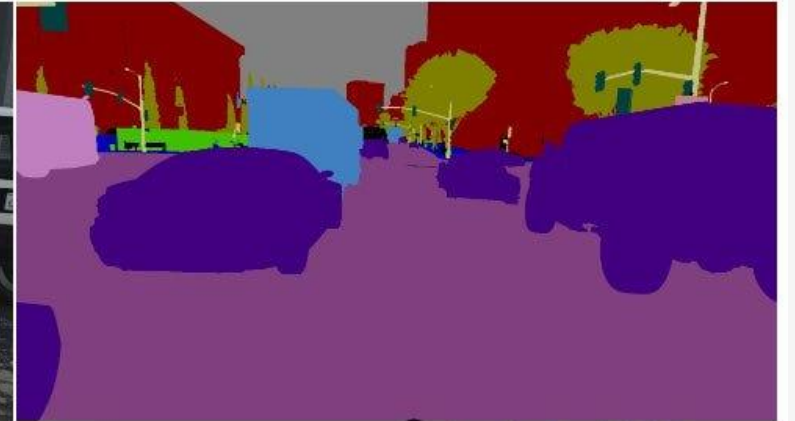
Gathering Data

1. Straight line along the wires (+ 360° for each post at different distance) (+ close up of attachements)
2. Cloudy, no direct sun, no wind, no rain, no snow
3. How close can we get to the post and the attachements?
4. Highest possible resolution
5. Is it possible to make annotations during the drone capture? (e.g. make image classifications by talking into your mobile device which is connected to the drone)
6. Fully automated drone flight and capture? Now? Future?
7. Import power post data from OSM, GoogleMaps



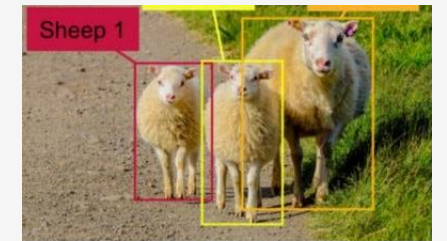
Gathering Data – Synthetic Training Data

1. Just one big main object with a rather empty environment
2. Easy Materials, wood, metal, concrete
3. Adding defects with textures und UV's
4. 100% correct and automated ground truth data
5. Generate as much data as you want > 1000000 power posts in different environments and conditions
6. Very realistic rendering in Unreal Engine

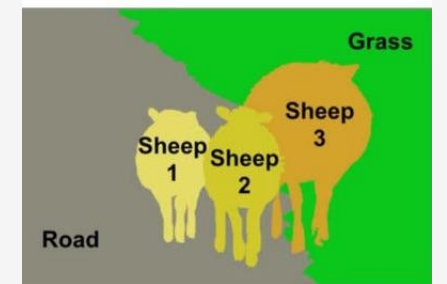


General Problems and Questions

1. All or nothing: Either all required data (defects, etc.) is collected with the required accuracy and reliability or a manual inspection is necessary anyway.
2. What is important and how detailed does the automated inspection have to be:
 1. Yes, anomaly detected. Please plan a manual inspection.
 2. Segmented and classified data in 2D and 3D for each POI with meta data and many images.
3. What difference does that make to the response/reaction?
Comparison: Autonomous Driving and End to End learnin
3. Customer: We want pixel masks! Why? What are you going to do with that Pixel? How important are pixel masks?



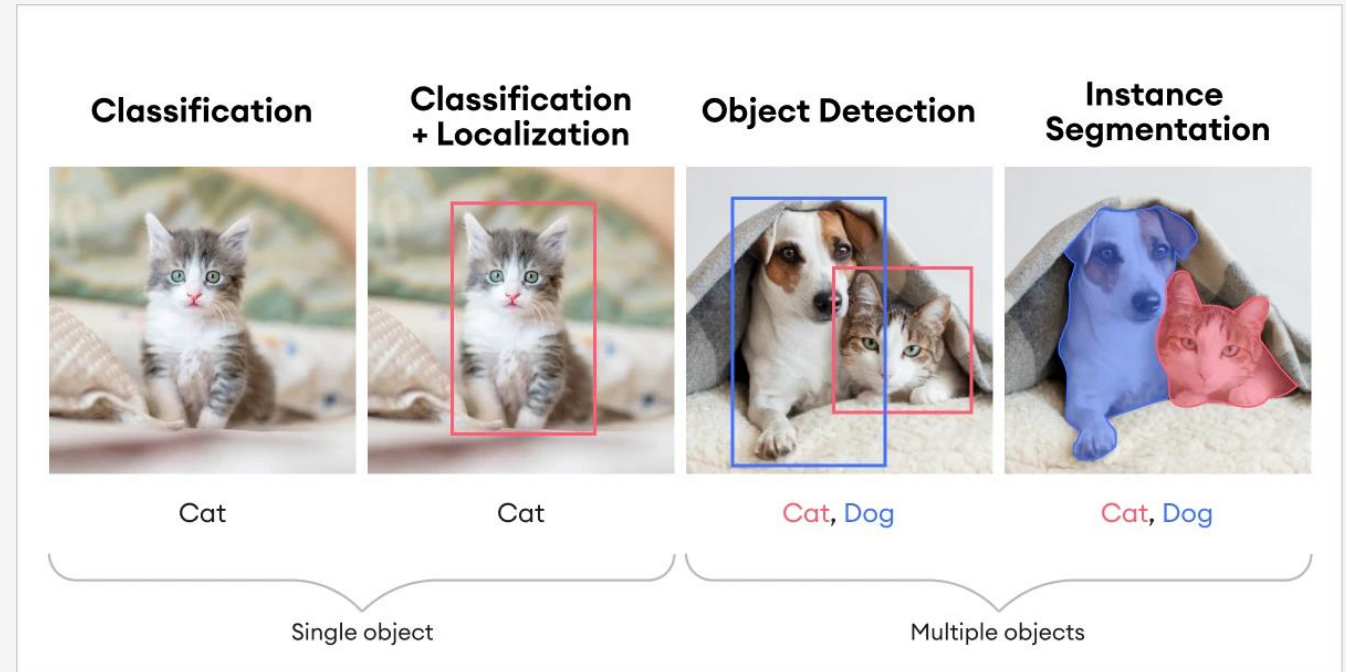
Object Detection



Instance Segmentation

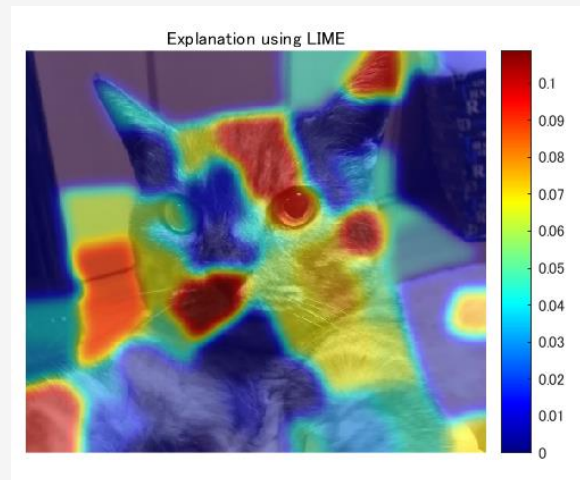
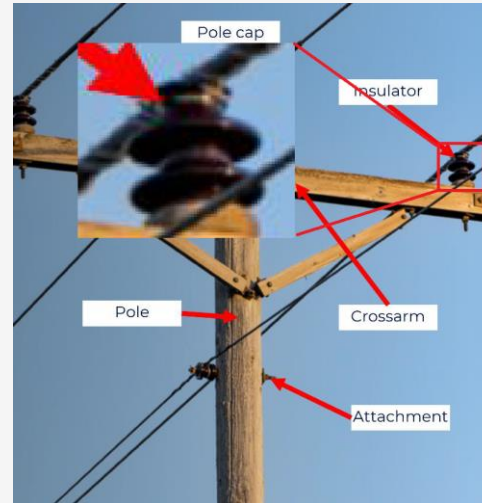
Classification vs Detection vs Segmentation

1. Image classification: CNNs, AlexNet, VGGNet, ResNet
2. Object detection: Faster R-CNN, YOLO (You Only Look Once), and SSD (Single Shot MultiBox Detector), Mask R-CNN
3. Image segmentation, panoptic segmentation (instance + class): Unet, EfficientPS
4. Object detection has problems with complex and overlapping structures which is probably not the case for power poles
5. Neural network complexity (sensitivity, inaccuracy):
Classification < Detection < Segmentation
6. Labeling complexity, effort, costs:
Classification < Detection < Segmentation
7. Bei classification, welche pixel haben am meisten zur classification beigetragen



Classification vs Detection vs Segmentation for Power Poles

1. Close up images -> Classification (Defect classes)
2. Take close up images directly with drone or crop from larger image after object detection (or segmentation)
3. Explainable image classification: It's possible to make inferences about which pixels contributed most to the resulting class.



a. Iron cap corroded insulators.



b. Rust on surface.



c. Pin corroded porcelain insulator.



d. Pin corroded glass insulator.



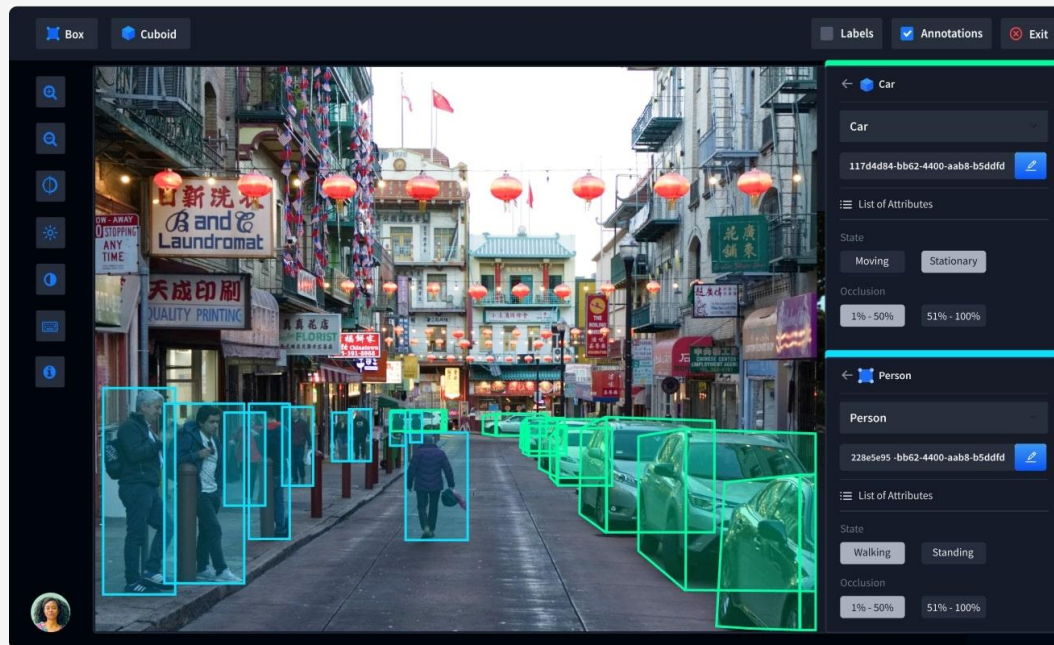
Classification vs Detection vs Segmentation – Annotations and Labels

1. Labeling complexity, effort, costs:
Classification < Detection < Segmentation

Multi-label Classification



- Dog
- Cat
- Horse
- Fish
- Bird
- ...



Object Classes and Defect Classes

Object classes:

- Pole
- Crossarm
- Pole cap
- Insulator
- Cable
- Attachements
- ...

Defect classes:

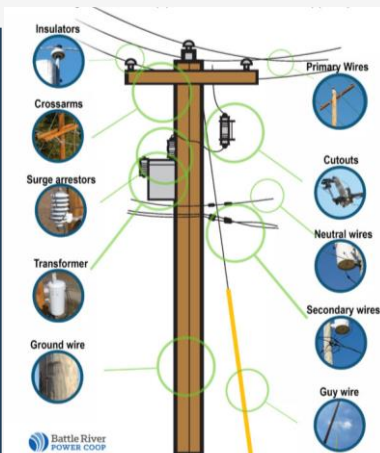
- Cracks
- Chippings
- Burn marks
- Rotting
- Rust
- ...

- 2 models (1 object class model, 1 defect class model)

- Object detection or segmentation

- Can process any image, no matter of angle and distance

- Very general defect models (+/-), lots of data to train (+), doesn't depend on any object type(+), probably less accurate and robust(-)



Object classes (detection or segmentation):

Pole Crossarm Pole cap Insulator ...

Defect classes (multi-class classification)

Pole defect classes: Crossarm defect classes: Pole defect classes: Insulator defect classes:

- | | | | |
|-------------|-------|-------|--------------|
| • Cracks | • ... | • ... | • Chippings |
| • Chippings | | | • Burn marks |
| • Rotting | | | • Rust |

- 1 + #Object classes models:

- 1 object detection or segmentation model for object classes

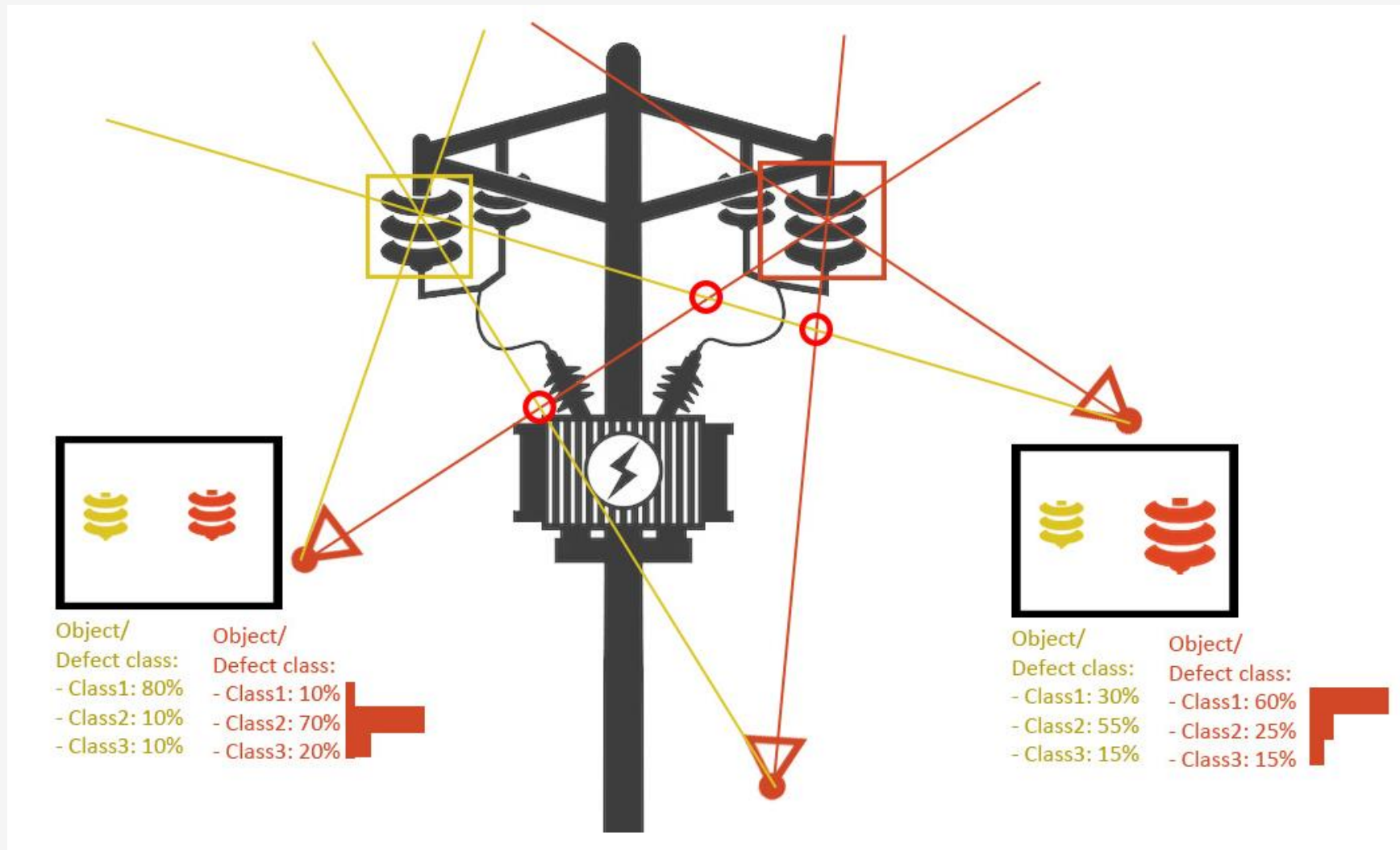
- 1 specific multi-class defect image classification model for each object type. Not every defect can appear on each object (no rust on wood etc).

- Object detection on all images, classification on close ups (from crops or raw)

- Specific defect models (+), less data to train but more precise(-/+), depends on objects type and new objects require new model (-), more accurate and robust (+)

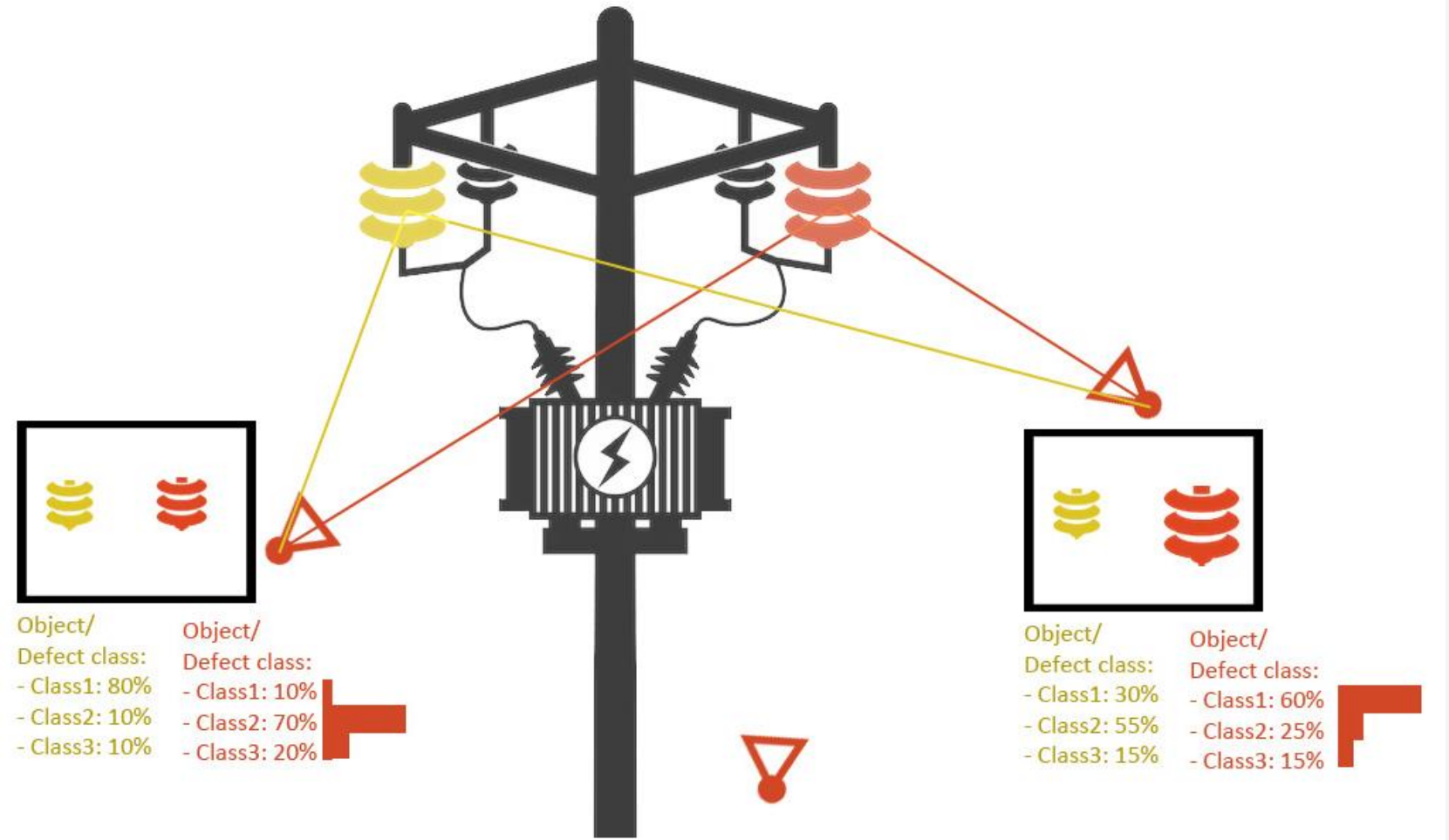
Data Association, Data Fusion and 3D Localization (with camera poses)

1. Input: multiple images from different angles and distances to the power pole with camera poses (and maybe a sparse point cloud with observations)
2. Run models on each image
3. Associate data of the same object from multiple images with computer vision (cluster intersection points)
4. Fuse probability distributions of object and defect classes to increase reliability, robustness, accuracy



Data Association, Data Fusion and 3D Localization (with 3D reconstruction)

1. Input: multiple images from different angles and distances to the power pole with camera poses (and maybe a sparse point cloud with observations)
2. Run models on each image
3. Associate data of the same object from multiple images by backprojecting the pixels into the scene and onto the 3d surface. Then this surface can be projected into other images again to find correspondences.
4. Attention: 3d geometry might be wrong or missing. Especially for small and thin structures.
5. Fuse probability distributions of object and defect classes to increase reliability, robustness, accuracy



Most important questions

1. Close-ups during capture? (maybe with voice labels)
2. No camera poses, camera poses, and mesh?
3. Object classes, defect classes?
4. Classification, detection, segmentation?

Option 1

- No camera poses or 3d reco
- We can only make statements about existence or no existence, but not count or fuse data since we have no associations.
- Example: we have at least one crack in the post and there are no transformers

Option 2

- Camera poses and 3d reco
- 2 segmentation models
- Pixel perfect labeling in 2d and 3d on mesh
- We can fuse data and make robust and detailed predictions

Option 3

- Camera poses but no 3d reco
- 1 object detection model + N defect multi classification models
- We can fuse data and make robust and detailed predictions

Data Preparation / Preprocessing

1. Imbalanced data: defect vs no-defect frequency?
2. Imbalanced data strategies: upsampling, downsampling, class weights
3. Common image normalization and data augmentation techniques
 1. Normalize image data (0-255 -> -1 - +1)
 2. Rotations
 3. Cropping
 4. Scaling
 5. Flipping
 6. Translations
 7. Color space transformations
 8. Noise injection
4. Synthetic defects with poisson image editing?



Source image



Target image



Result

Models

1. We probably need all of them:
 1. (Multiclass) image classification: CNNs, AlexNet, VGGNet, and ResNet
 2. Panoptic segmentation (semantic, instance): U-Net, EfficientPS, UPSNet, VPSNet, EPSNet, FPSNet (<https://arxiv.org/ftp/arxiv/papers/2111/2111.10250.pdf>)
 3. Object detection: Faster R-CNN, YOLO (You Only Look Once), and SSD (Single Shot MultiBox Detector), Mask R-CNN
2. Are there already general networks for rust, corrosion, crack detection?
3. Transfer learning
4. Use machine learning platforms to adapt those model for your purposes (Transfer Learning. (Amazon Rekognition, Amazon Lookout for Vision, Google Visual Inspection AI, Azure AI Vision, Azure AI Custom Vision, HuggingFace)
5. (There are models that fuse 2d with 3d data, but they are very complex in all manners [model complexity, data preparation, labeling, etc.] and not ready to use, especially for our use case with multiple images. Are there NERF like classifications models?)

Metrics


1. Image classification

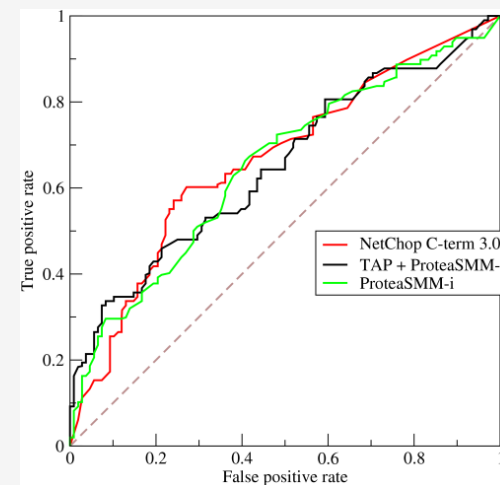
1. AUC of ROC (bad with imbalanced data)
2. PR-Curve (precision recall)
3. Fbeta score, F1 score (precision and recall), prioritize with beta

$$F_{\beta} = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

2. Segmentation, Detection: intersection over union (IoU)

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$





		Actual Class	
		Cancer	No Cancer
Predicted Class	Cancer	TP = 542	FP = 108
	No Cancer	FN = 650	TN=98 700



Tools

1. TF vs PyTorch:

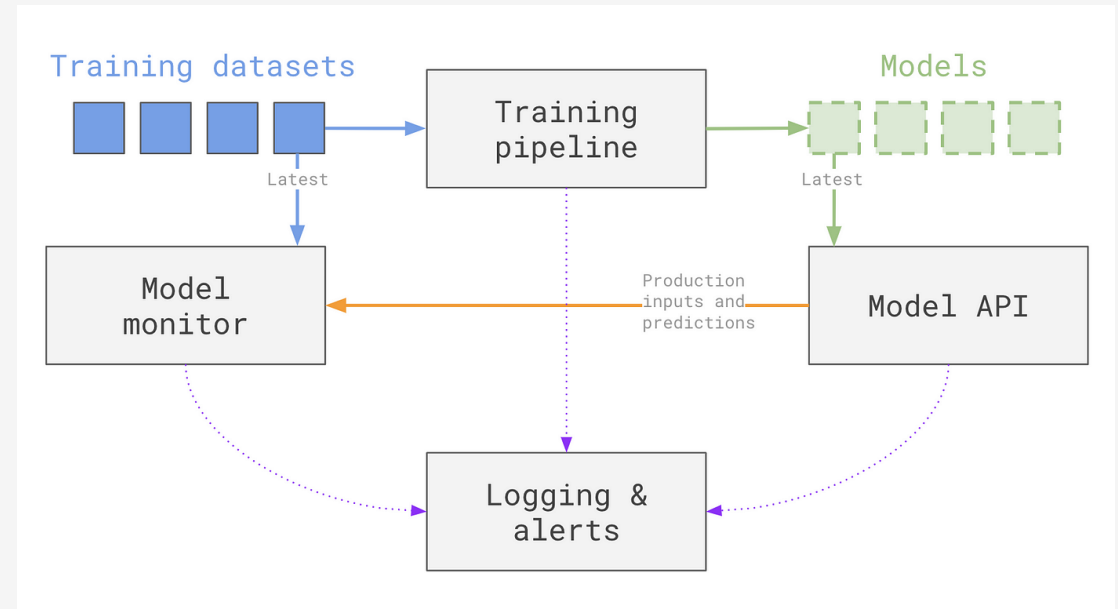
1. Pytorch is much more popular and used more in research and easier to learn and debug, more libraries (3d, language extensions), closer to research and academia, >90% papers with code, slightly less optimized gpu execution
2. TF offers better deployment, distribution, more optimized, static computation graph,
3. Simply use both, whatever is more comfortable in the current situation. Our model input are just images, so nothing complicated.

2. MLOPS (<https://mad.firstmark.com/>)

1. ClearML
 2. MIFlow
 3. Kubeflow (Kubernetes)
 4. Neptune
 5. Weights and Biases
 6. TensorBoard
- ## 3. Just take Amazon SageMaker or Azure and see what's missing?
4. Experiment tracking: no experience, we used git commits + config files
 5. Model serving: maybe no API necessary because models are part of a larger software package

Pipeline

1. Treat model drift:
 1. concept drift, example: new defect or change of how a defect looks like
 2. data drift: change in raw data, seasonal data, different cameras, different drone routes etc..
 3. upstream data drift: change in data preprocessing of raw data
2. Detect model drift
3. Inspect the evaluated data before relearning
4. Retrain with new and validated data and track metrics
5. All data is saved on the server. Image data is assigned to a single power post. So, we can track the condition over time and learn to predict or identify problems earlier.



Future

1. More specialized algorithms for object and defect detection / classification (data and knowledge driven algorithms)
2. Predictive Maintenance: predict and detect defects before they even happen, optimize preventive maintenance