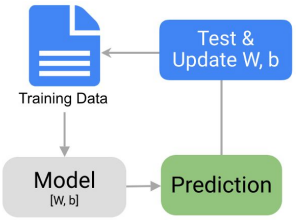
The Path to following

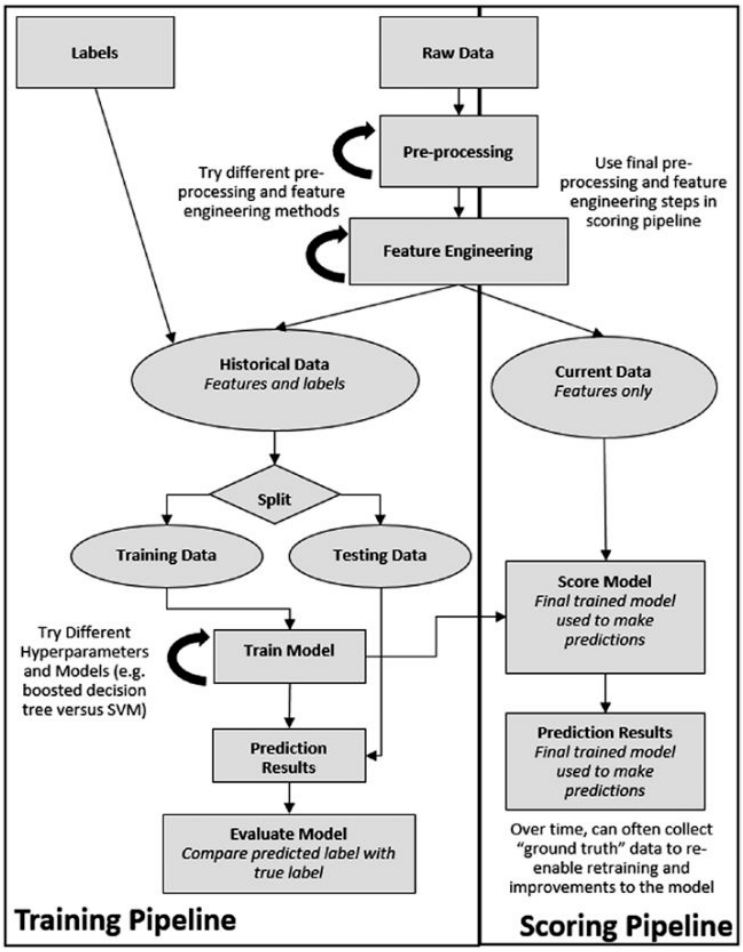
[Frameworks for Approaching the Machine Learning Process]

https://www.kdnuggets.com/2018/05/general-approaches-machine-learning-process.html

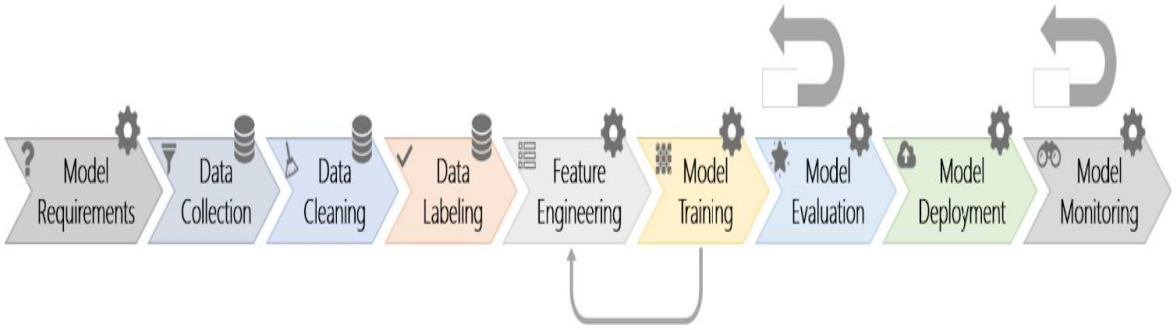


Approach for classical, supervised machine learning solutions

Ebook: Deep Learning with Azure



Paper 2019: Software Engineering for Machine Learning: A Case Study



Template Dataset

- NEU surface defect database:
  http://faculty.neu.edu.cn/yunhyan/NEU_surface_defect_database.html
  (publicise many high score papers)

- Old Solution: (will implement in next time)
  https://software.intel.com/en-us/articles/use-machine-learning-to-detect-defects-on-the-steel-surface
- Severstal: Steel Defect Detection
  https://www.severstal.com/eng/media/news/document29364.phtml
  Steel is one of the most important building materials of modern times. Steel buildings are resistant to natural and man-made wear which has made the material ubiquitous around the world. To help make production of steel more efficient, this competition will help identify defects.
  Severstal is leading the charge in efficient steel mining and production. They believe the future of metallurgy requires development across the economic, ecological, and social aspects of the industry—and they take corporate responsibility seriously. The company recently created the country's largest industrial data lake, with petabytes of data that were previously discarded. Severstal is now looking to machine learning to improve automation, increase efficiency, and maintain high quality in their production.
  The production process of flat sheet steel is especially delicate. From heating and rolling, to drying and cutting, several machines touch flat steel by the time it's ready to ship. Today, Severstal uses images from high frequency cameras to power a defect detection algorithm.
  In this competition, you'll help engineers improve the algorithm by localizing and classifying surface defects on a steel sheet.
  If successful, you'll help keep manufacturing standards for steel high and enable Severstal to continue their innovation, leading to a stronger, more efficient world all around us.

Avalable Solutions for commerce

1. **Dr. Schenk GmbH: Metal Inspection**

https://www.drschenk.com/products/metal-inspection.html

EasyInspect for metal inspection is a fast, reliable, and efficient way for optimizing the production and product quality of metals. Together with EasyMeasure, EasyInspect for metal inspection continuously optimizes the complete metal production process.

EasyInspect easily and seamlessly integrates into any existing or new environment, providing perfect metal inspection and yield monitoring at all stages of production, returning optimum product quality at reduced production costs.

EasyInspect for metal inspection is used for local defect detection, e.g. for:

- Edge fault detection
- Dent detectionAI Solutions
- Scratch detection
- Stain detection
- Seam detection
- Hole detection
- Fold detection
- Slip detection
- Weld detection
- Scale detection
- Roll mark detection

EasyInspect and EasyMeasure for metal inspection offer:

- Quality inspection performance for defect-free, flawless metal surfaces
- Advanced defect classifier for reliable and fast defect classification
- Highest system availability ensures reliable production
- On-line reports and visualization return comprehensive quality information at a glance
- Easy integration minimizes cost & ensures fast implementation, guaranteeing optimum efficiency
- Reliable features and quality for unparalleled user acceptance

EasyInspect & EasyMeasure for metal inspection use Dr. Schenk's unique MIDA Technology to inspect metals with multiple optical channels on a single scan line. One defect generates multiple images to deliver the most comprehensive material analysis and defect classification on the market.

AI Solutions

https://www.drschenk.com/products/ai-solutions.html

2. **ISRA Vision**

https://www.isravision.com/en/ready-to-use/surface-inspection/surface-defects/

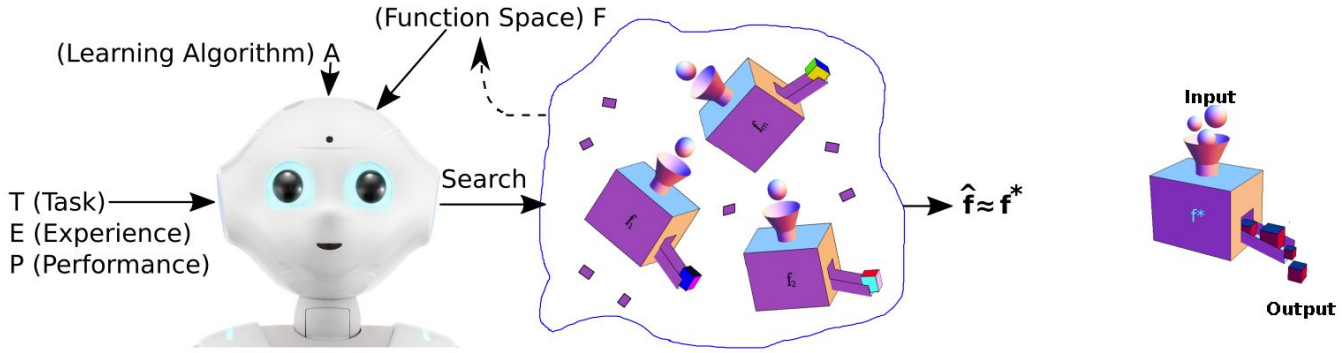
Surface Defects

The optimum solution for inspecting the quality of reflective surfaces: SpecGAGE3D inspection systems monitor glass, aluminum and injection-molded plastic parts in a fast, objective and transparent way.

The highly sensitive deflectometry sensors detect even the slightest deviations from a perfect surface. Highly precise spot measurements at the nanometer level allow reliable detection of faulty parts.

To ensure flawless products, high-resolution modules allow you to check both cosmetic criteria and functional surface properties, such as on touch displays, spotlights or technical mirrors.

Machine Learning : General Framework



improve the detection model
https://github.com/facebookresearch/maskrcnn-benchmark/issues/1064
Seems like the model has not generalized well enough and hence you have false positives, did you try adding few more examples? Also, try training from scratch once and compare the results. They are indeed many ways in which a model MAY improve. Following are such

1. Increase the training set.
2. use data augmentation techniques (Vertical flipping, Horizontal flip, color jittering)
3. Fune tuning by varying the learning rate.
4. Checking or visualizing the activate maps to get an idea about what model is seeing.
5. Changing the SGD momentum optimizer to RMSProp.
6. Increasing the POST_NMS_TEST number.

A lot of experimentation needs to be done.
You can also automate a lot of these work by using Microsoft's Azure cloud called HyperDrive which selects the best hyper params based on some heuristics like Grid Search, Random Search, and Bayesian Optimization, etc.

How to Build A Own Data Set For Your Machine Learning Project

https://towardsdatascience.com/how-to-build-a-data-set-for-your-machine-learning-project-5b3b871881ac

Docs: intelligent defect inspection powered by computer vision and deep learning

https://www.infopulse.com/blog/intelligent-defect-inspection-powered-by-computer-vision-and-deep-learning/

How to train your own dataset

https://github.com/facebookresearch/maskrcnn-benchmark/issues/521

In order to complete and unify all the issues about how to train a model using a custom dataset, you will find here the basic steps to do it. Taking into account that this implementation it's a recent release, this steps can change in the future. I'm requesting feedback, this was the steps that I followed for training models and it works perfectly.

Steps

1) COCO format

The easier way to to use this model is labelling your dataset with the official coco format according with the problem you have. In my case, It's for instance segmentation, so I used the detection format.

2) Creating a Dataset class for your data

Following the example coco.py. Create a new class extending from torchvision.datasets.coco.CocoDetection (you can find another classes in the official docs), this class encapsulates the pycocoapi methods to manage your coco dataset.

3) Adding dataset paths

This class will needs as parameters the paths for the JSON file, it contains the metadata of your dataset in coco format, and for the images, the folder where they are. The engine automatically searches in paths_catalog.py for these parameters, the easier way is including your paths to the dict DATASETS following the format of course, then include an elif statement in the get method.

4) Evaluation file

Here is the importance of use the coco format, if your dataset have the same structure then you can use the same evaluation file used for the class COCODataset, in the __init_.py file just add an if statement like the COCODataset statement.

This evaluation file follows the coco evaluation standard with the pycocoapi evaluation methods. You can create your own evaluation file, and you have to do it if your dataset have another structure.

5) Training script

Here you can find the standard training and testing scripts for the model, add your own arguments, change the output dir (this one is very important), etc.

6) Changing the hyper-parameters

The engine uses yacs config files, in the repo you can find different ways to change the hyper-parameters.

7) Finetuning the model

The issue #15 has all the explanation.

My Task:

1. Pulicise: surface dataset of phone button
2. Publicise paper: localizing and classifying surface defects on phone button