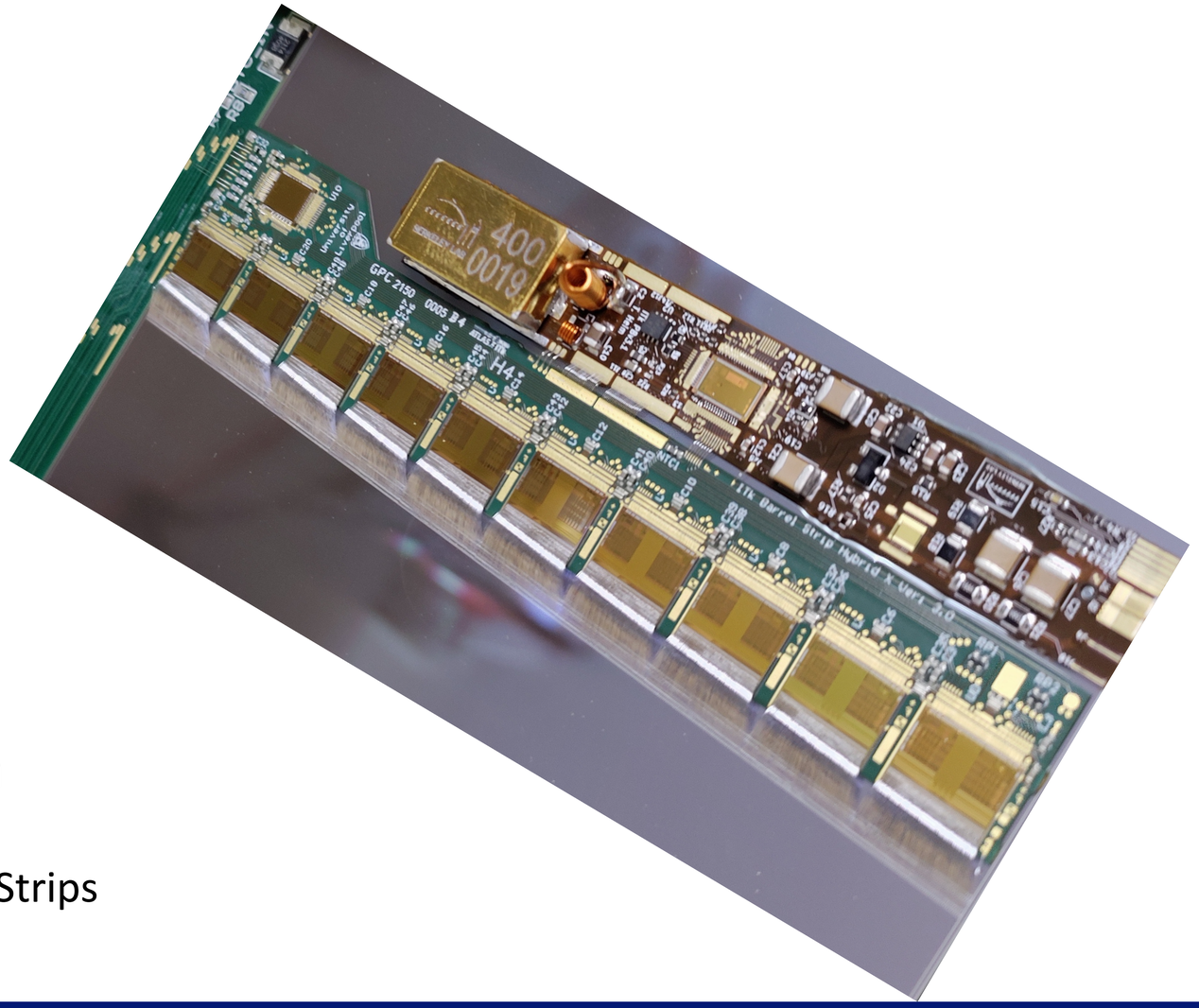
UNIVERSITY OF LIVERPOOL

Automated visual inspection system for ATLAS ITk strip hybrids



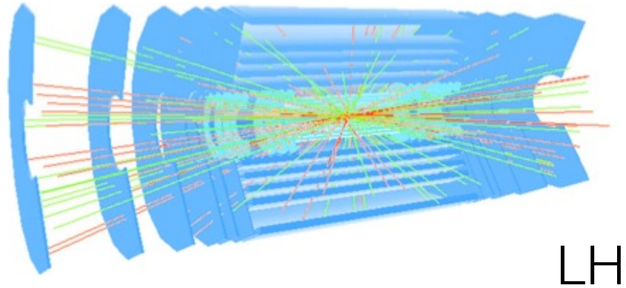
Paul J Dervan

On behalf of ATLAS ITk Strips

Outline

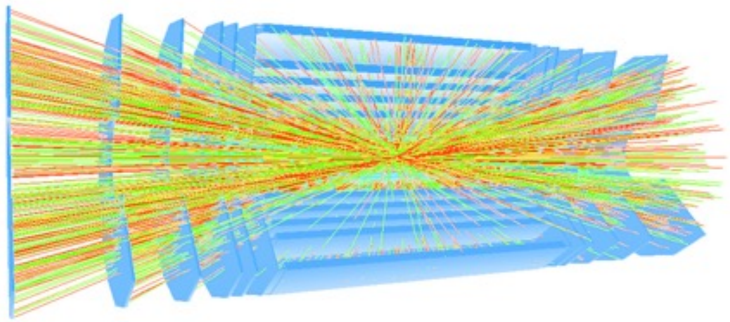
- The ATLAS Upgrade
- The problem
- Computer Vision
- Classification
- Object Detection

ATLAS Inner Tracker (ITk)

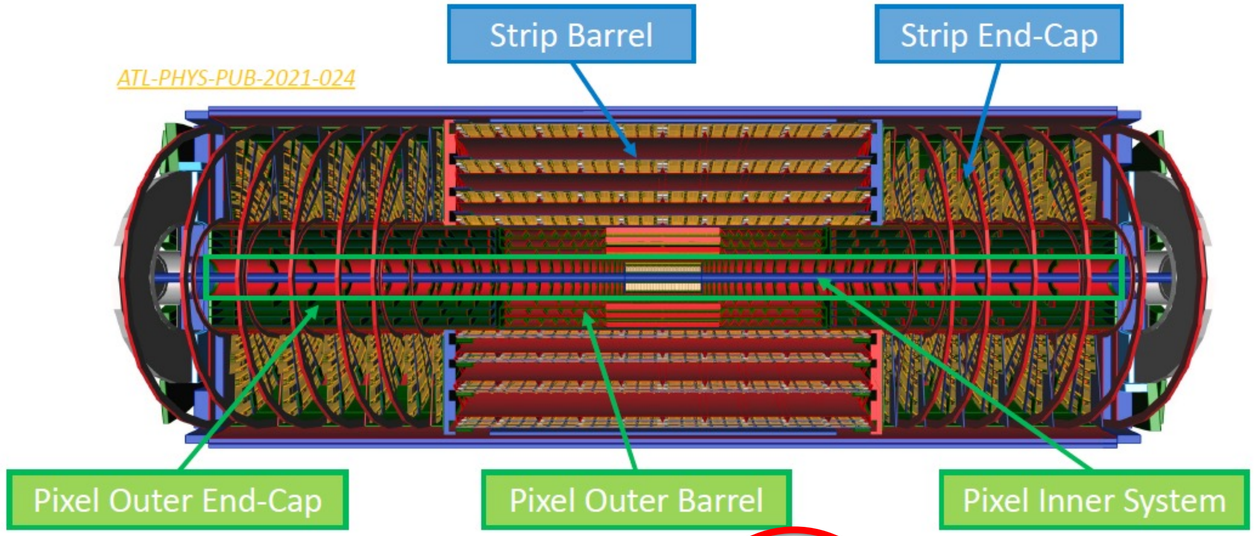


LHC

- All new silicon tracker for HL-LHC
- Strip detector has of O(20k) modules
- Pixel detector has of O(10k) modules



HL-LHC

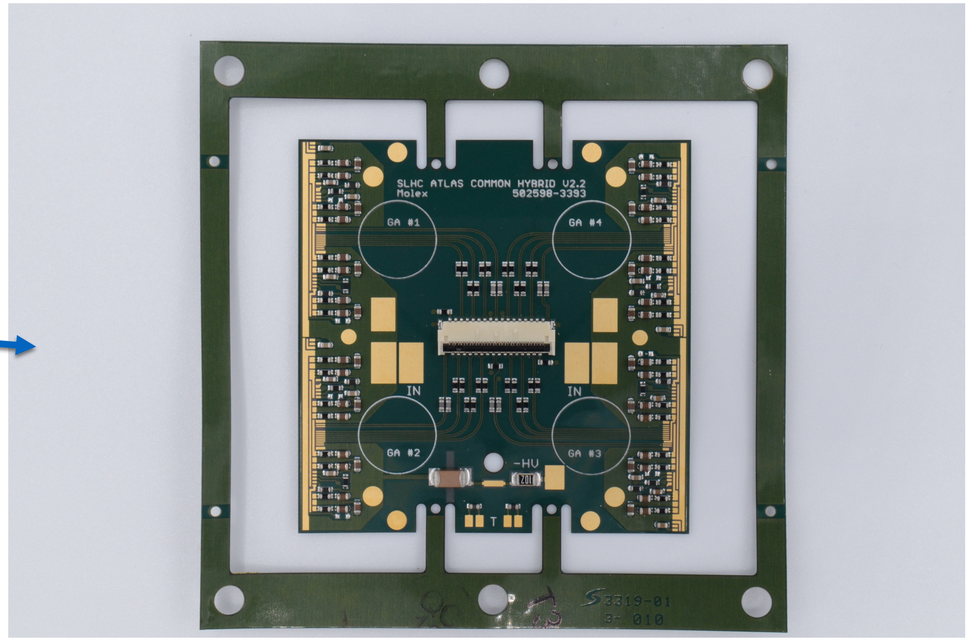


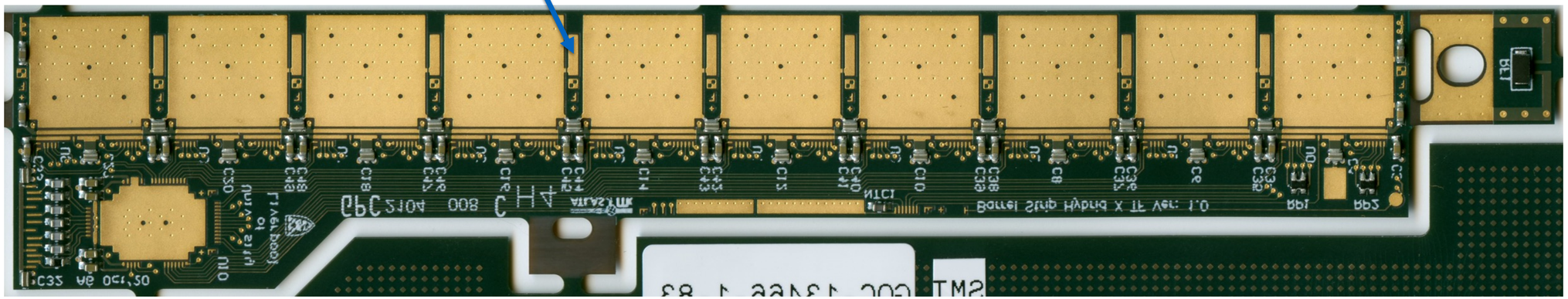
ITk (ID)	Area (m^2)	# Modules	# Channels (M)
Pixels	13 (1.6)	9164 (2000)	5100 (92)
Strips	165 (61)	17888 (4088)	60 (6.3)

The Hybrids

Strips
(focus of today)

Pixels





Outlining the problem

- The problem:
  - Lots of hybrids that need visual inspection (O(20k) strip hybrids)
  - Usually done by a human inspecting the object using a microscope
  - Time consuming, boring and tiring
  - Mistakes happen when people are tried/bored

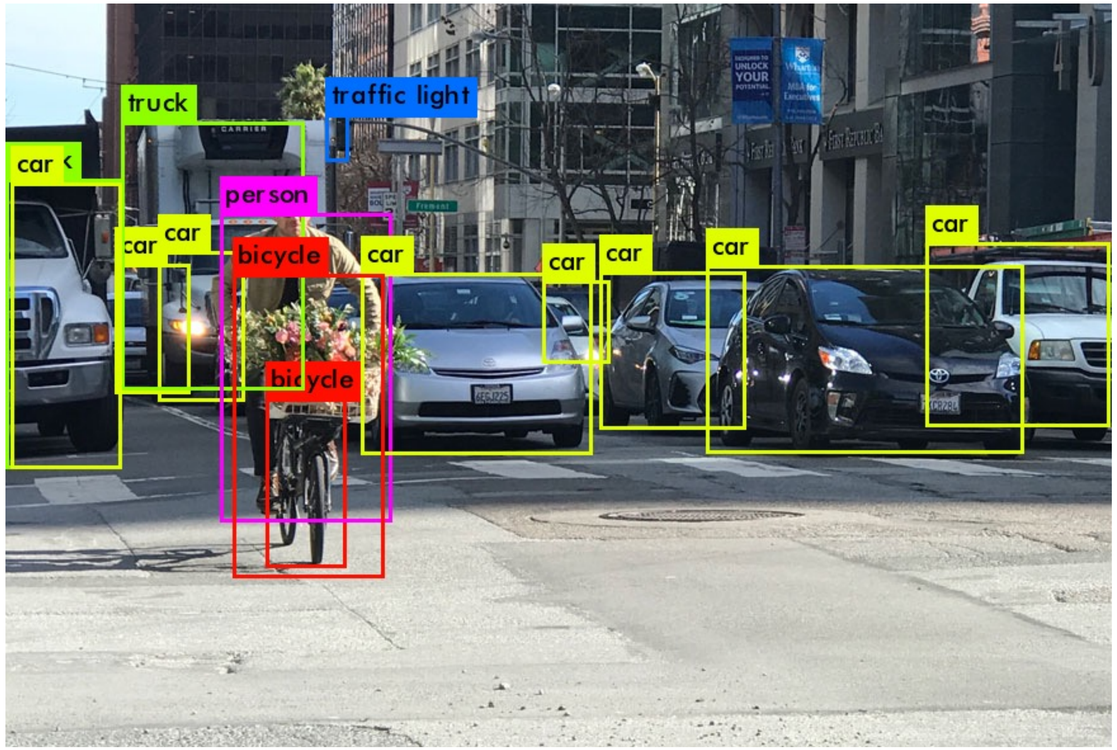
- Could computers help?

- I started to look at Machine Learning and Deep Neural Networks to see if they could help
- This work is for both strips and pixels (just strips shown today)
  - Early work was on the Classification of objects
  - Extended this work to Object Detection

Goal

Ultimate GOAL : Fully automated visual inspection of ATLAS hybrids

- Everyone has heard of machine learning and AI especially in computer vision
- It is used everywhere. Even on your mobile phone (have a look for the camera icon in amazon.co.uk)
- The state of the art is the self driving car
- For fun, could we do something similar for our visual inspection of hybrids?
- Ultimate goal would be to inspect a hybrid and find all good features and bad features

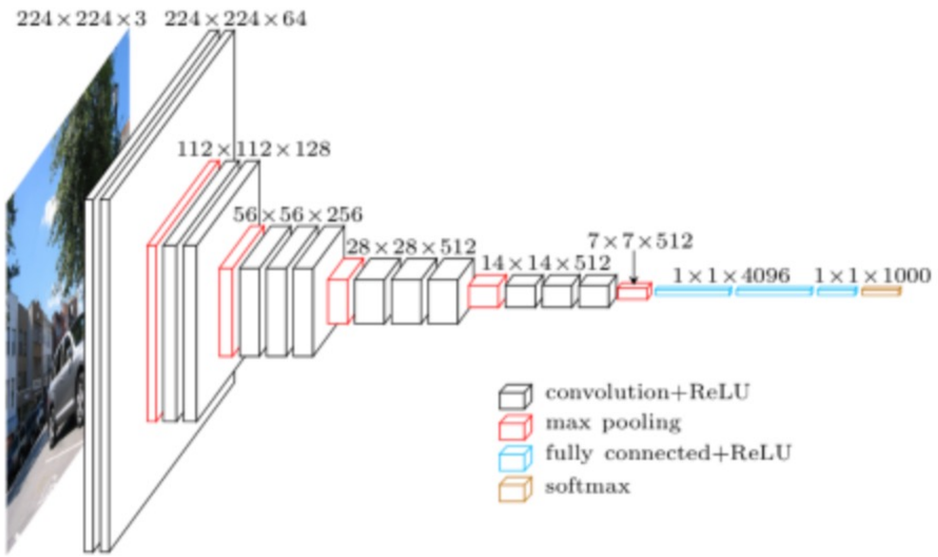


Convolution Neural network (CNN) - Classification

A convolutional neural network is a class of deep neural networks commonly applied to analyzing visual imagery.
They consist of convolutional, pooling and fully connected layers.

The most common are:

- VGG (image to right)
- LeNet
- AlexNet
- GoogleNet
- ResNet
- Inception

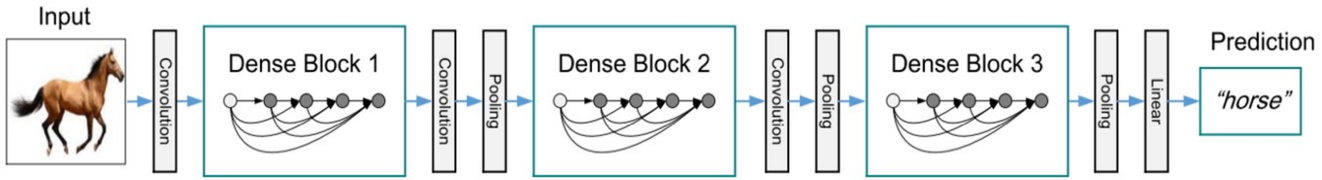


Architecture of VGG16

These are trained on vast amounts of labelled images and are used for detecting (“CLASSIFYING”) objects e.g. dogs, cats, people, apples, laptops, cars, trains, planes etc …..

They take weeks to train using the state of the art CPU’s and GPU’s. (They are fed data for example from “I’m not a robot” images.)

Input a picture of a horse → output is it’s a “Horse” with a percentage probability

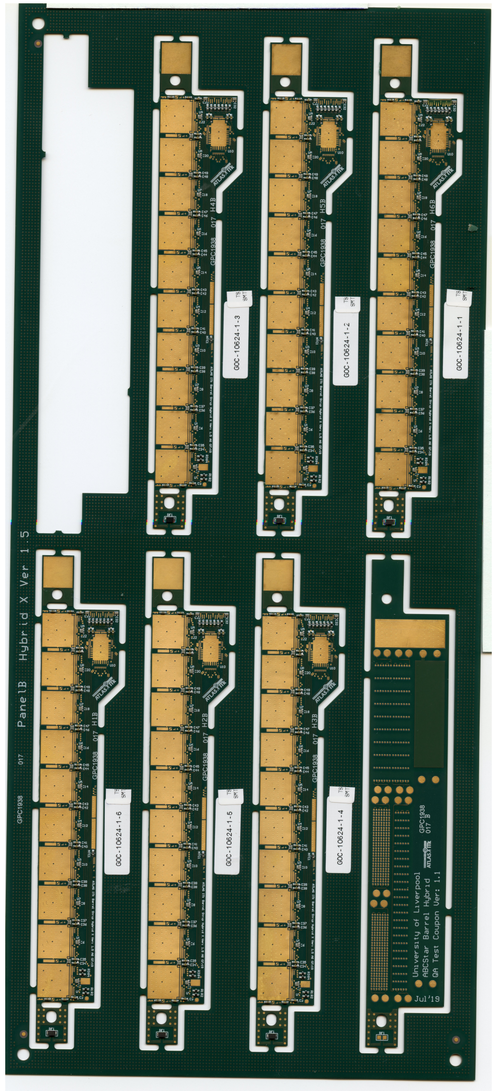


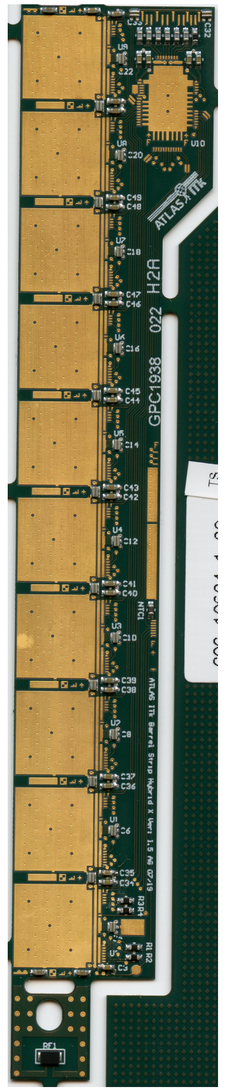
Luckily you can train smaller ones on our computers.

Image taking/processing

- Hybrids come in panels of six/seven hybrids
- These panels are scanned (below)
- So single hybrids need to be removed for inspection (this is done semi automatically)
- The colour can be corrected (using histogram matching). This corrects for any light changes or setting for the scanner set incorrectly

Epson Perfection 850 scanner
Frame Size = ~850MB
Hybrid = ~35MB





Find components (Template Match)

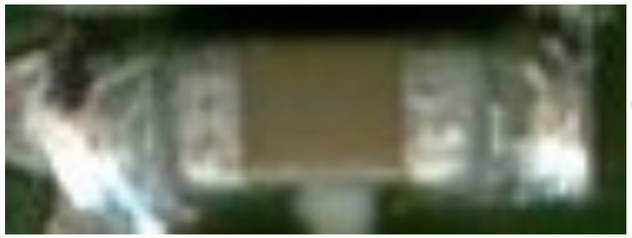
We need images of the components to train the CNN

Using an image of an e.g. horizontal capacitor, run OpenCV “cv2.matchTemplate”

This finds objects that are similar to the input image
(its not always correct, if it was my work would be done!).

Using this we can then produce a set of image libraries

e.g: capacitor



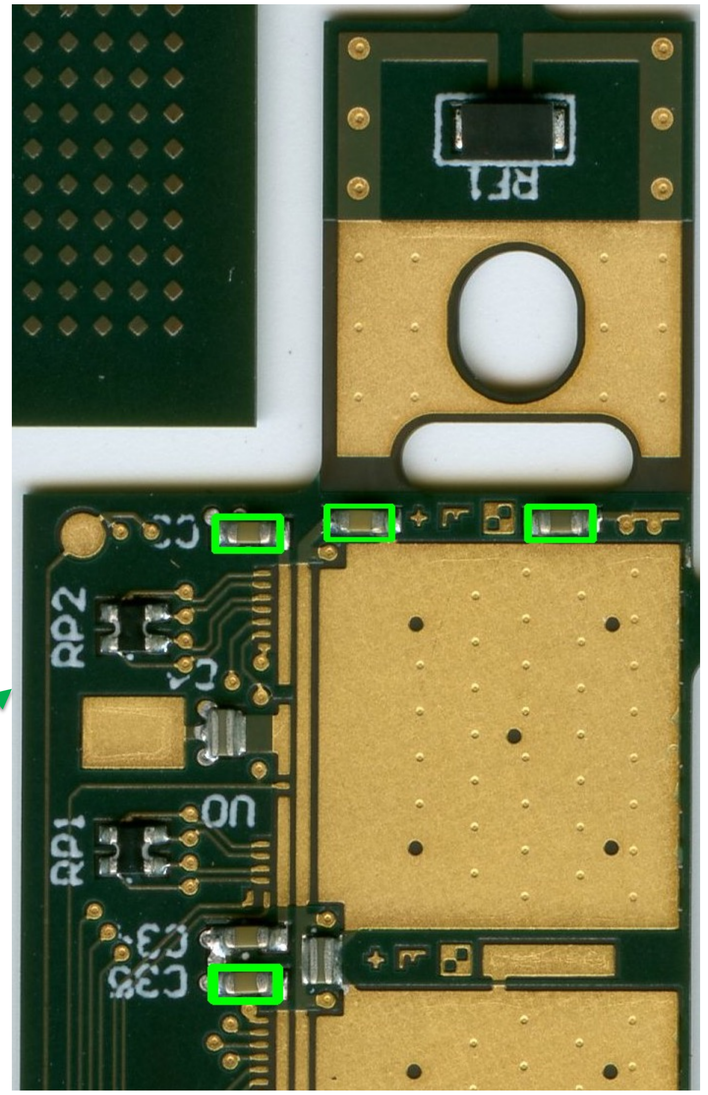
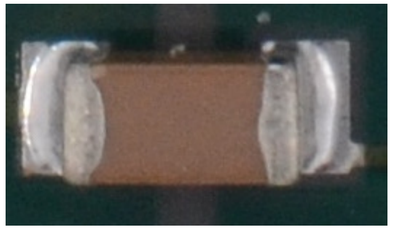
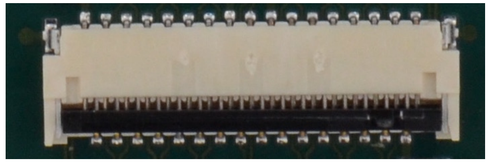
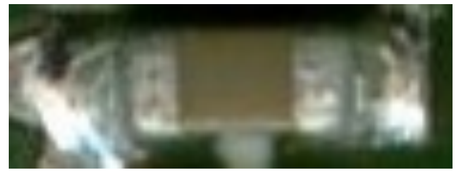


Image library

- Capacitor HV
- Capacitor
- Capacitor Vertical
- Double 1
- Double 2
- Opening
- Resistor Pack
- Capacitor Mur
- Resistor Vertical
- Ziff
- Other
- …..

Split into test and train




My CNN

Build a CNN using python and Keras/tensorflow

- Image pre-processing (shape, color etc …)
- Image augmentation

- 3 convolutional layers
- Pooling layers
- Fully connected layer
- Dropout layer
- Soft max activation layer

Lots of weights (parameters) to fit

Layer (type)	Output Shape	Param #
conv2d_13 (Conv2D)	(None, 148, 248, 32)	896
max_pooling2d_13 (MaxPooling	(None, 74, 124, 32)	0
conv2d_14 (Conv2D)	(None, 72, 122, 64)	18496
max_pooling2d_14 (MaxPooling	(None, 36, 61, 64)	0
conv2d_15 (Conv2D)	(None, 34, 59, 64)	36928
max_pooling2d_15 (MaxPooling	(None, 17, 29, 64)	0
flatten_5 (Flatten)	(None, 31552)	0
dense_9 (Dense)	(None, 128)	4038784
activation_9 (Activation)	(None, 128)	0
dropout_5 (Dropout)	(None, 128)	0
dense_10 (Dense)	(None, 10)	1290
activation_10 (Activation)	(None, 10)	0

Total params: 4,096,394
Trainable params: 4,096,394
Non-trainable params: 0

Training of the CNN

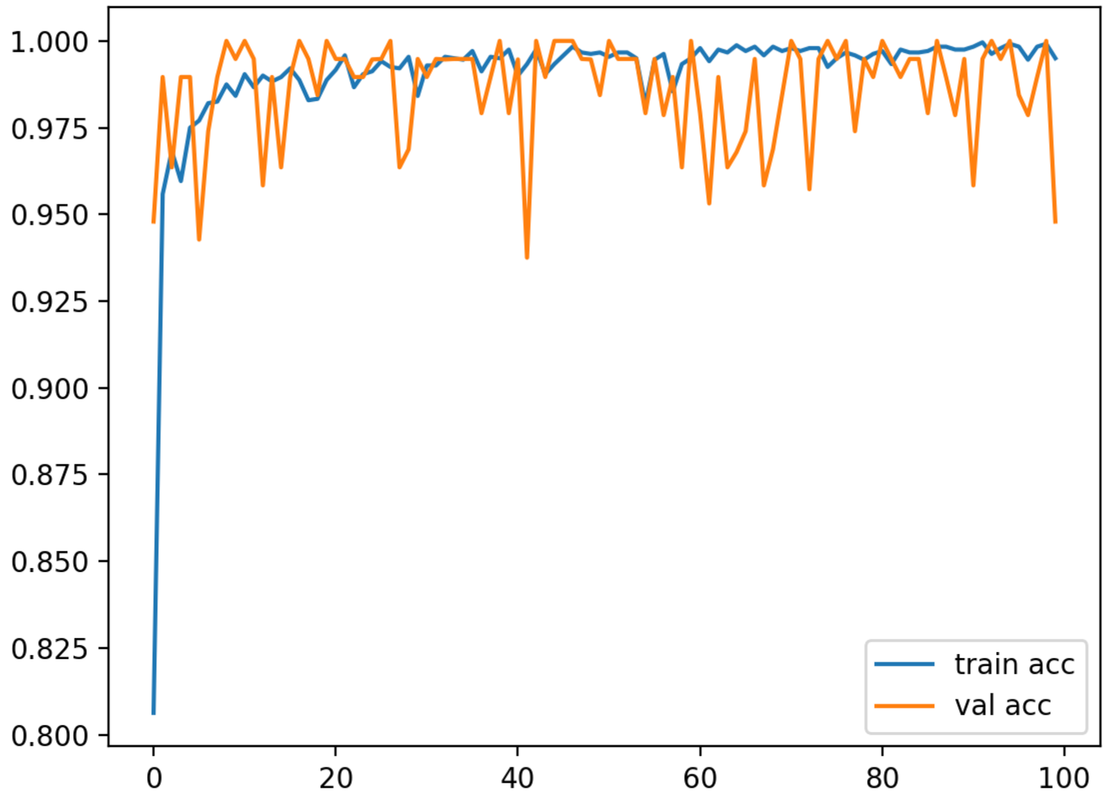
A few hours later …
(100 epochs)

Result of training of the CNN

The training can be run over various epochs e.g. 100. This can be varied.

Basically, you want the training and the test accuracy to increase together. Sometimes with over fitting the training accuracy increases but the test accuracy doesn't → Not enough images!

This plot shows how accurate the CNN is functioning on the training and test data



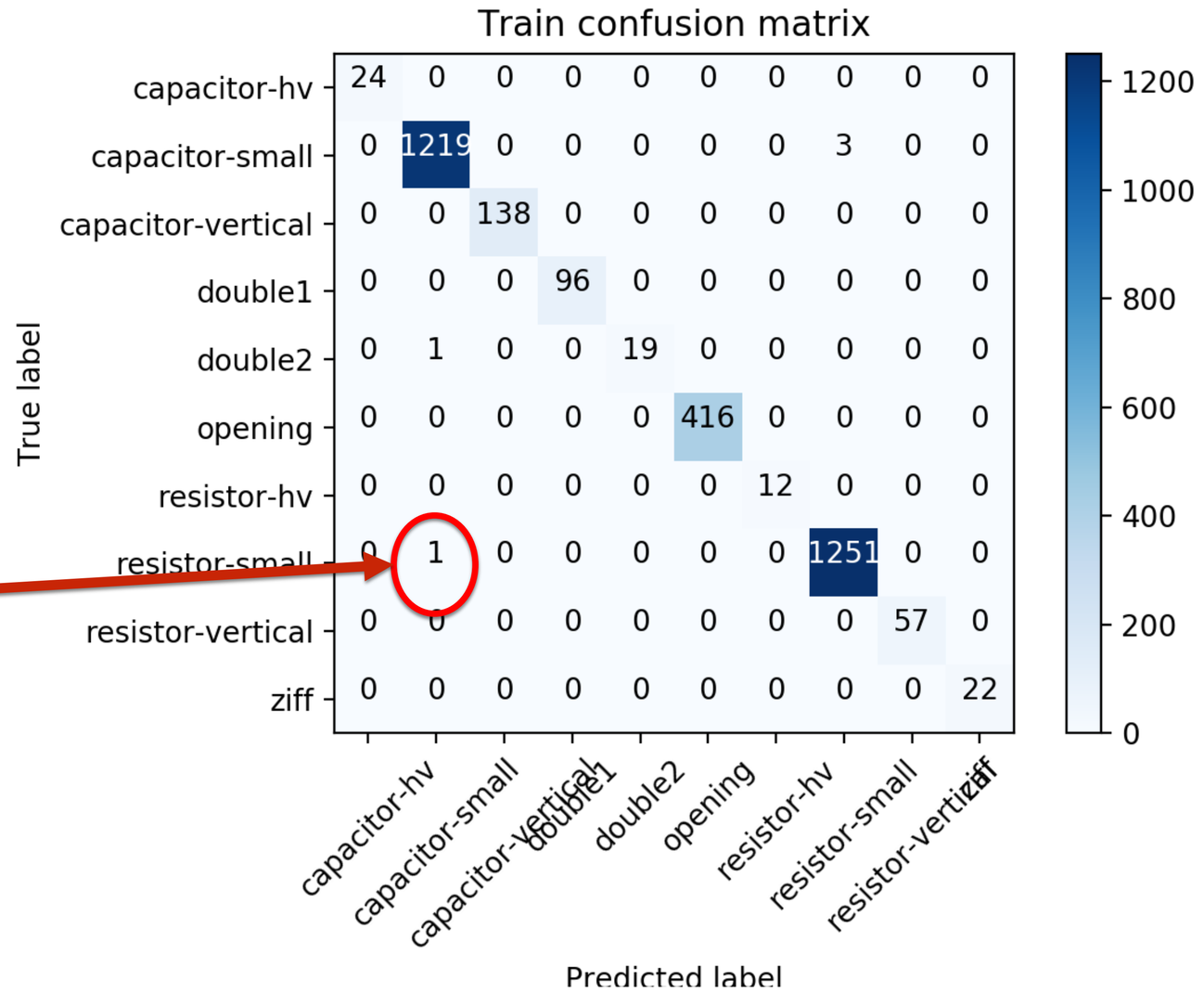
Result of training of the CNN (confusion matrix)

The confusion Matrix:

Tests how predictive the CNN is. You feed in a known image and compare it to what it says it is.

e.g a resistor is predicted to be a capacitor, 1 out of 1252 images was wrong.

Let’s try it on a completely unseen image!

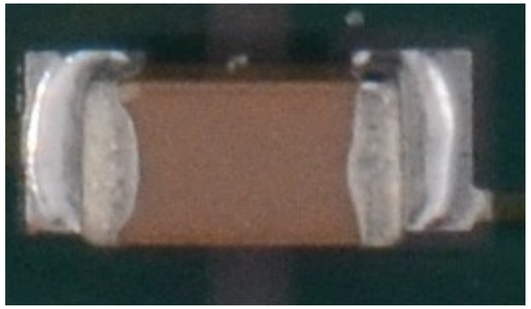


Trying the CNN

First test is to pass in a test image (one it wasn't trained on). e.g. a capacitor hv

The CNN Predicts:

'capacitor-hv': 9.9999440e-01
'capacitor-small': 5.6219887e-06
'capacitor-vertical': 1.1677365e-12
'double1': 2.8908326e-15
'double2': 1.3033322e-13
'opening': 2.3757464e-17
'resistor-hv': 2.3757464e-17
'resistor-small': 7.6223750e-15
'resistor-vertical': 7.5907114e-13
'ziff': 3.5378621e-20

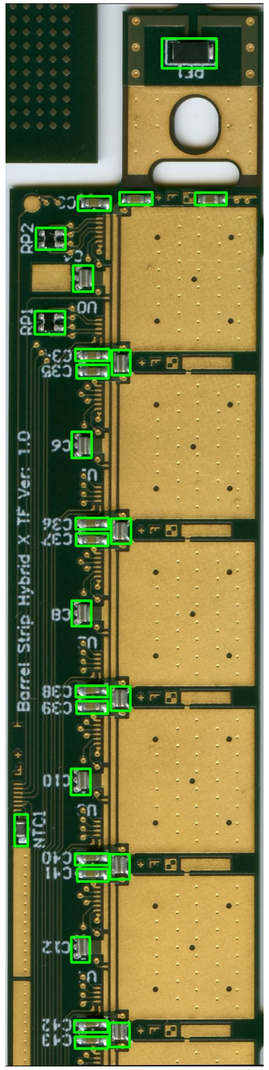


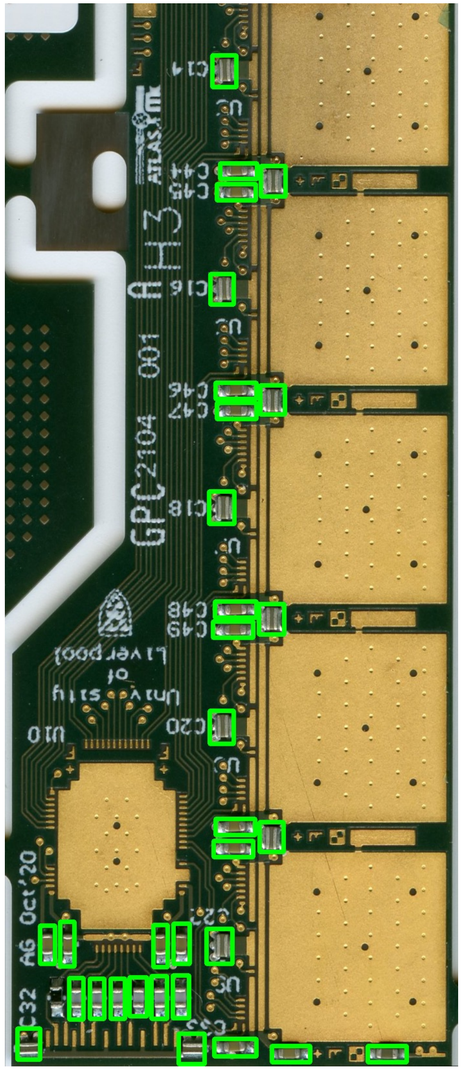
Similar results for the other components

Using the CNN

- Using a Region of Interest (ROI) map, based on the bill of materials, we can pick out the images from the new hybrid
- Use the CNN to predict what it is
- Compare to the known/expected answer
- Green means a good agreement
- Good agreement is seen on most hybrids (not all)

After Alignment



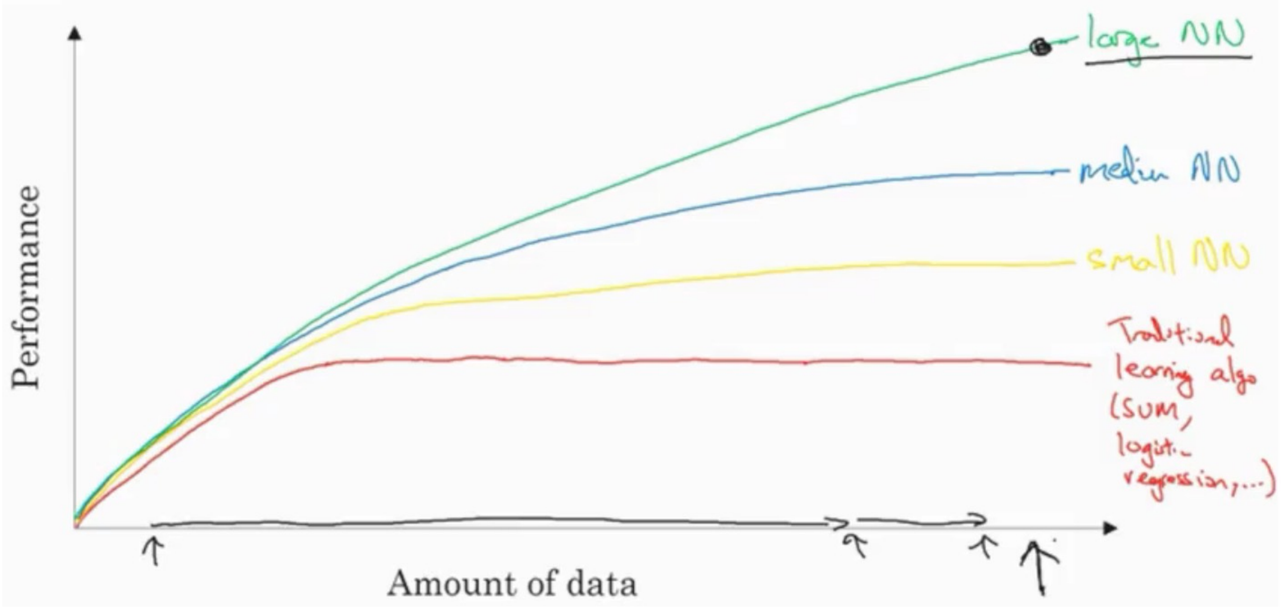


Summary for object classification

- Trained a CNN to classify SMD components
- First tests were encouraging

- Need more and more images → ”Big Data”

- Further improvements:
  - -- more images will improve things
  - -- tweak to the CNN design
  - -- number of epochs



- Need more images. And especially ones with defects.
  - -- More real images are not likely – it’s the same in industry
  - -- Use GANs to generate real looking fake images

- Alignment of the images was a problem

- “Object detection” may make this better…

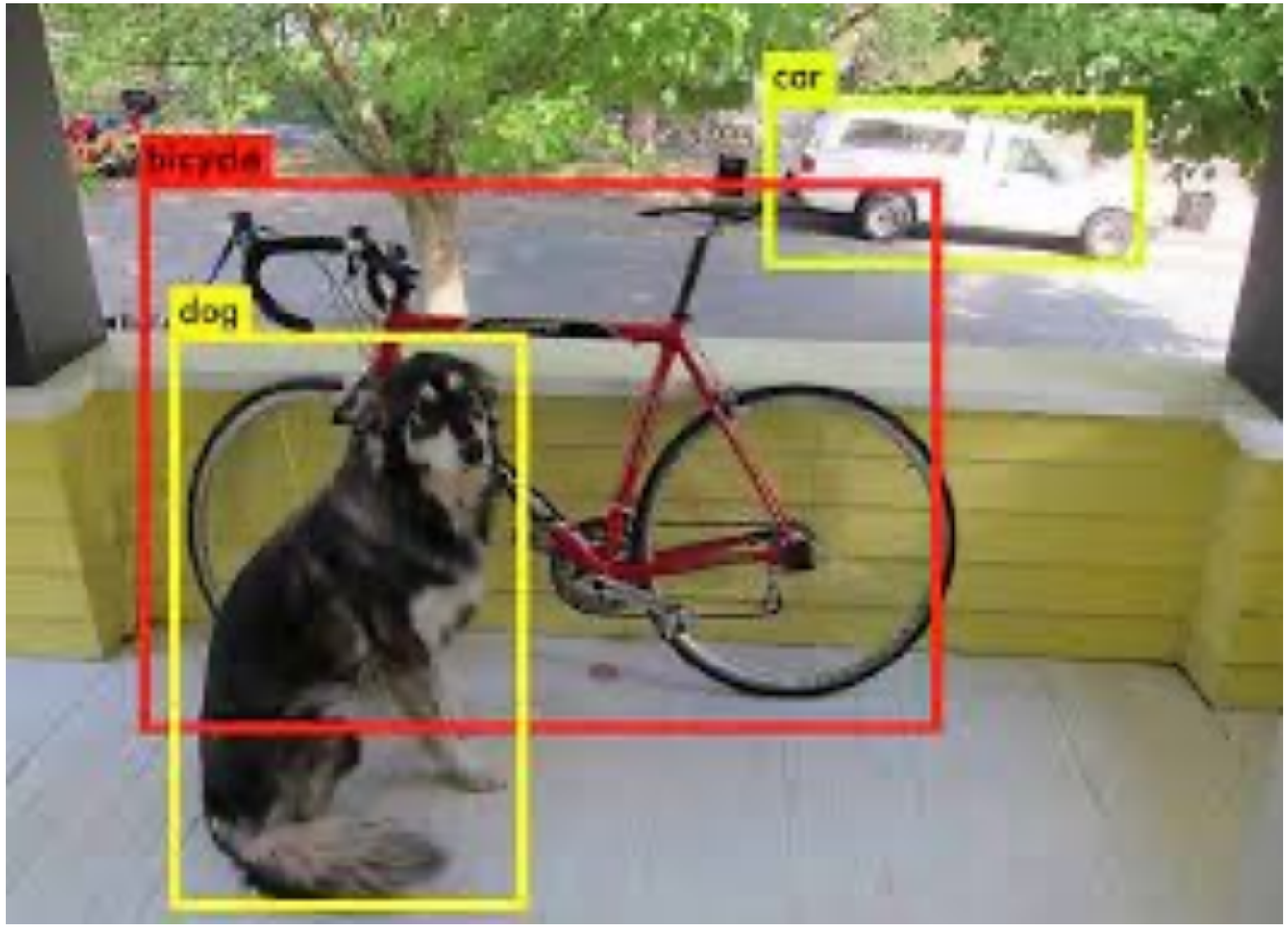
Object Detection

Object Detection:

This is finding the objects in an image and classifying them.

So far, we have classified an object (HV capacitor, Ziff etc ...). But we had to know where it was using ROI.

What if we could find them in the image as well!
(getting closer to self driving inspection)



YOLO v5 (Object Detection)

YOLO (You Only Look Once) Object Detection:

This is one of the “in” methods for object detection. But there are others. RCNN, SSD to still try.

We need to train these from scratch

These use clever processing of the convolutional layers so that you only have to process an image once (after training).

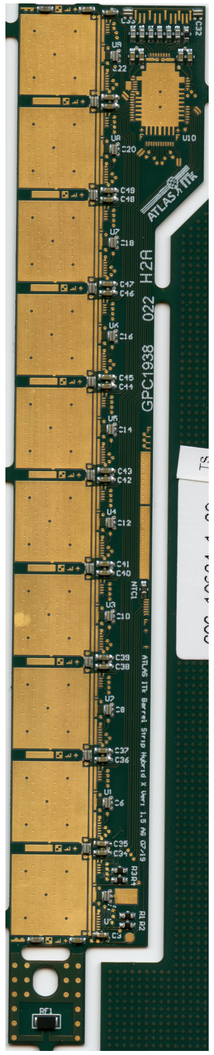
There are a few versions of YOLOv5: nano, small, medium, large (this is the number of layers in the NN):

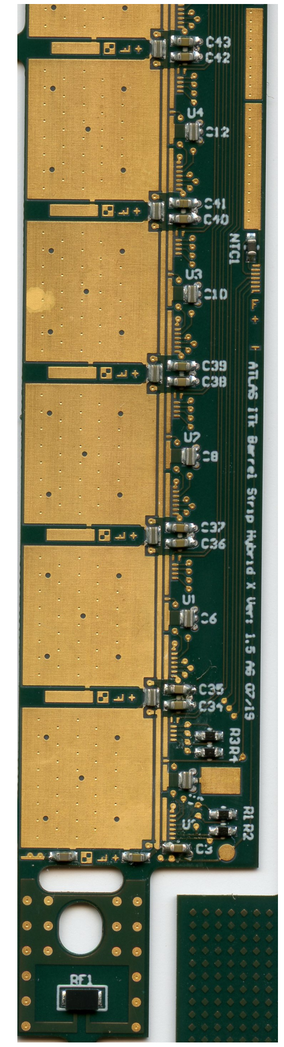
- medium YOLO has 290 layers. → Remember my CNN (3 convolutional layers)

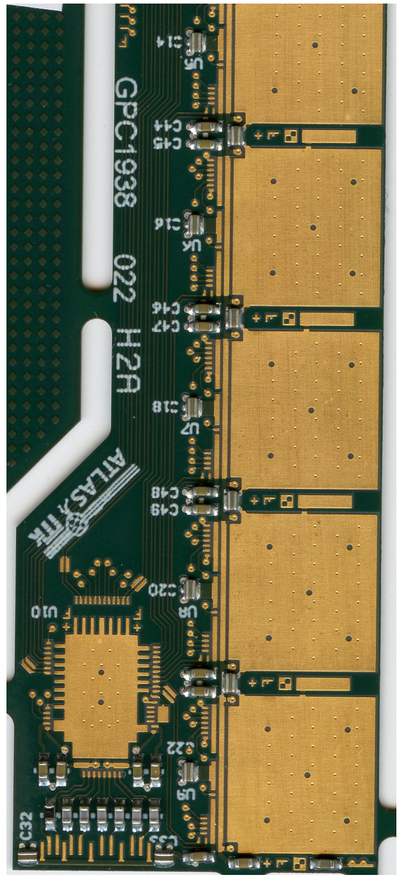
Strip hybrids in half

- The YOLO object detector doesn't work very well with a whole hybrid. The SMDs are too small.

- So, we need to split the hybrid.

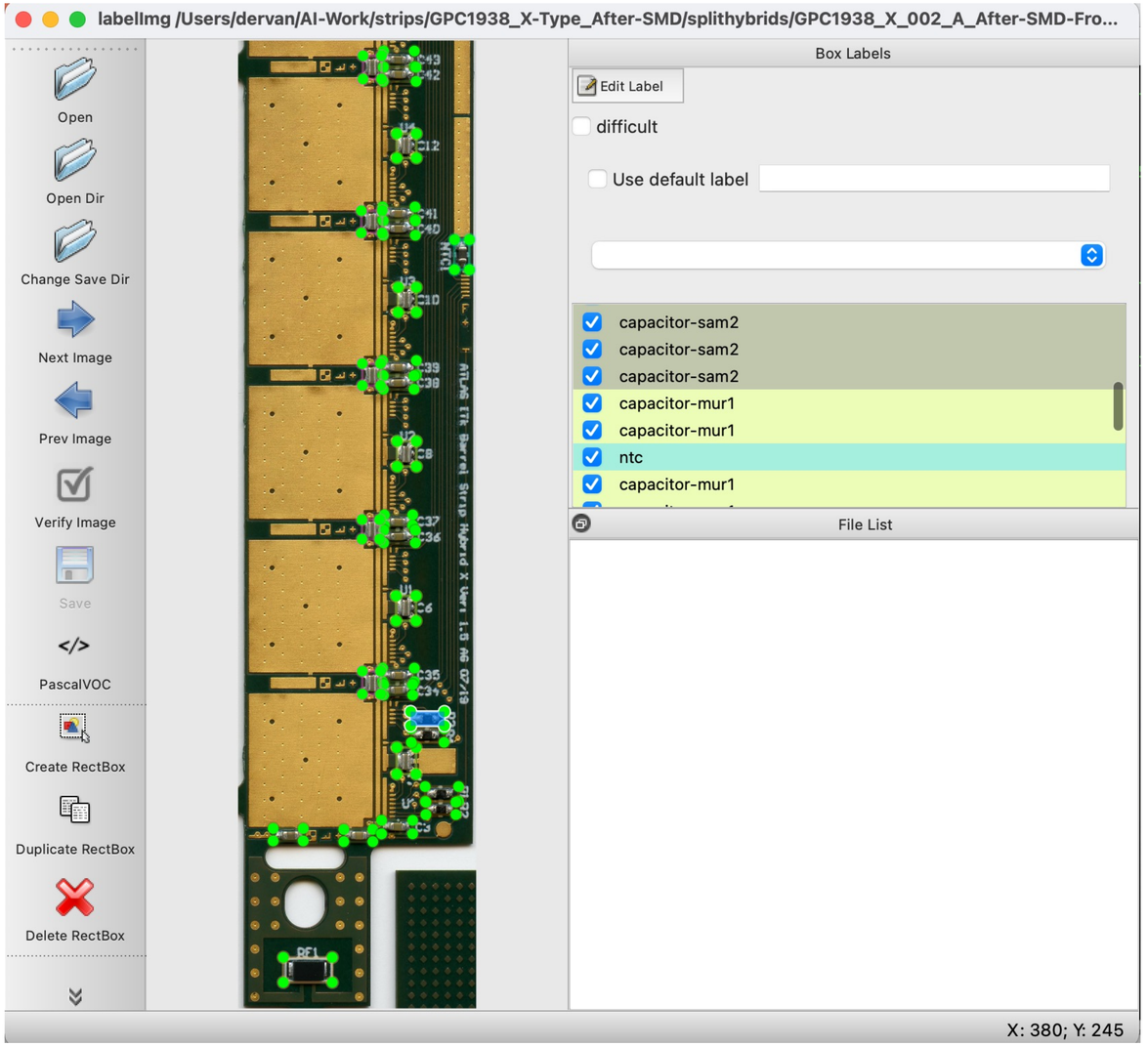






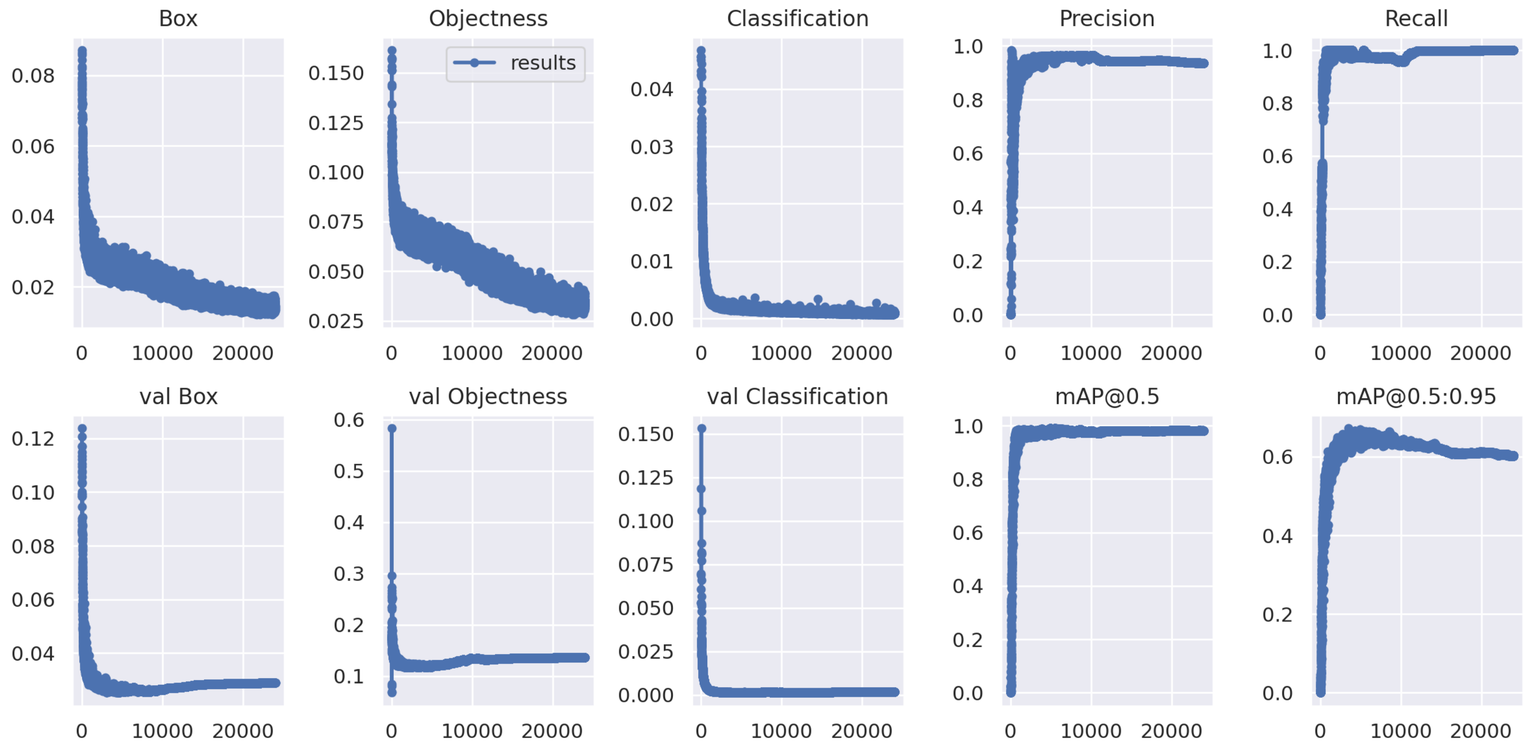
Annotate the images

- To train an object detection network we need annotated data.
- This is images with the interesting parts (SMDs) identified with a bounding box and a label.
- This is also done by hand using annotation software (labelImg in this case)



Train the YOLO network

- Split the annotated images into test and train data sets (usually 30/70 split).
- There are 2 sets of hybrids:
  GPC2104 and GPC2105
- Train on the YOLO network over 27000 epochs
- Each epoch does a full pass through all the images that are annotated
- Expect the loss to decrease.
- And precision to increase.
- This takes 2 days on a GPU



Apply the weights to make predictions

Once trained lets:

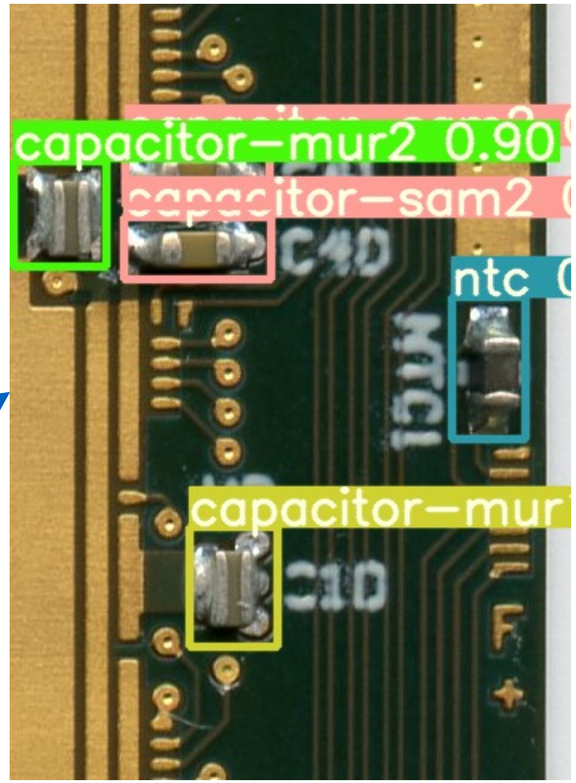
Try on a hybrid it wasn't trained with

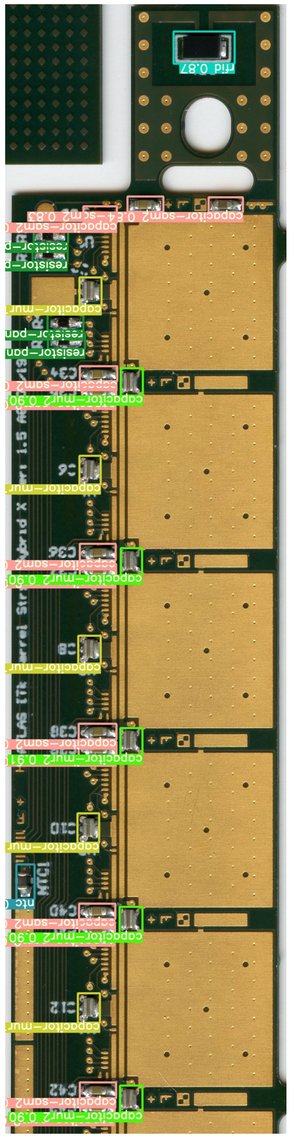
This is what it looks like after the yolo algorithm has run on a hybrid. All the components have been found.

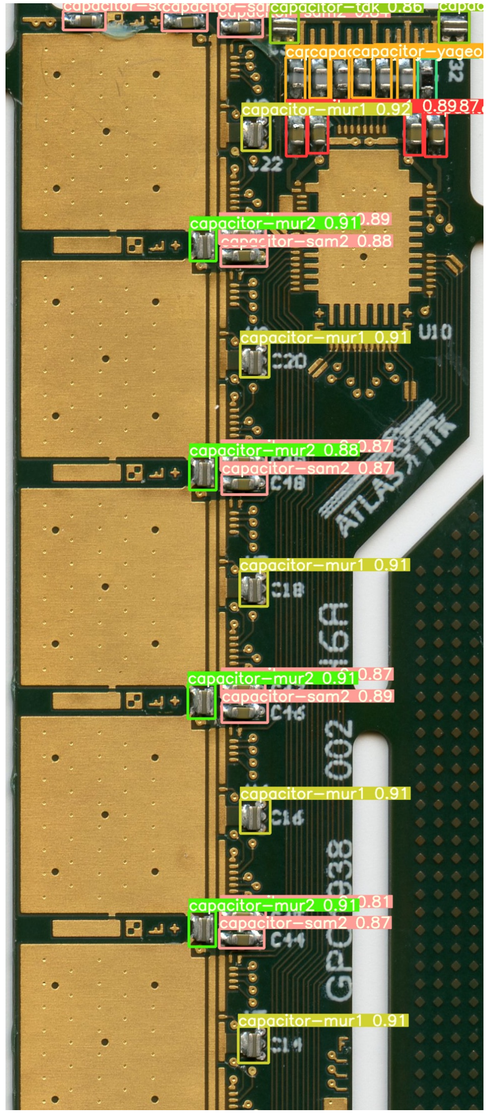
Time taken per hybrid is about 0.5 seconds

Do a simple component count to find any differences

Zoomed in



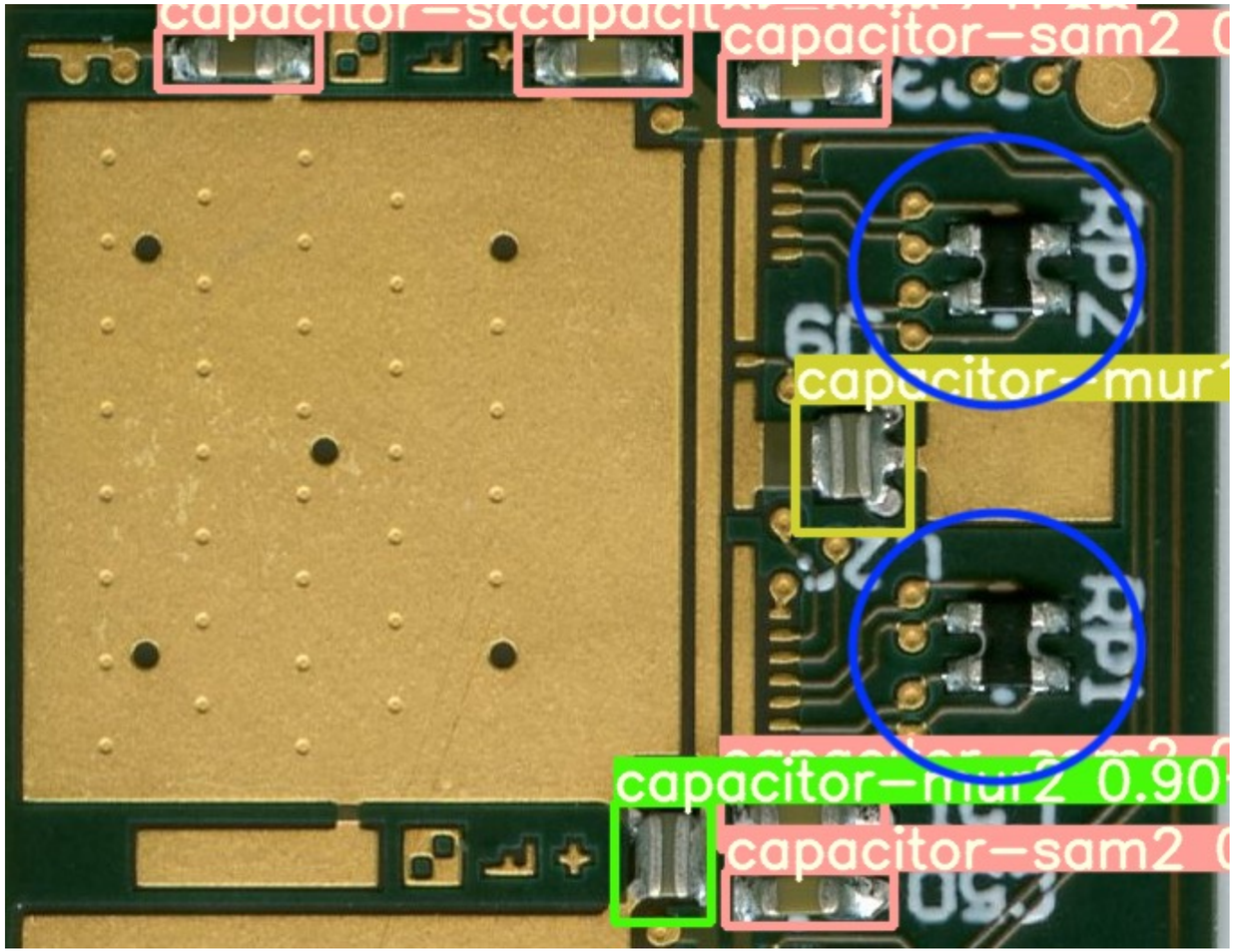




Now use it on two batches

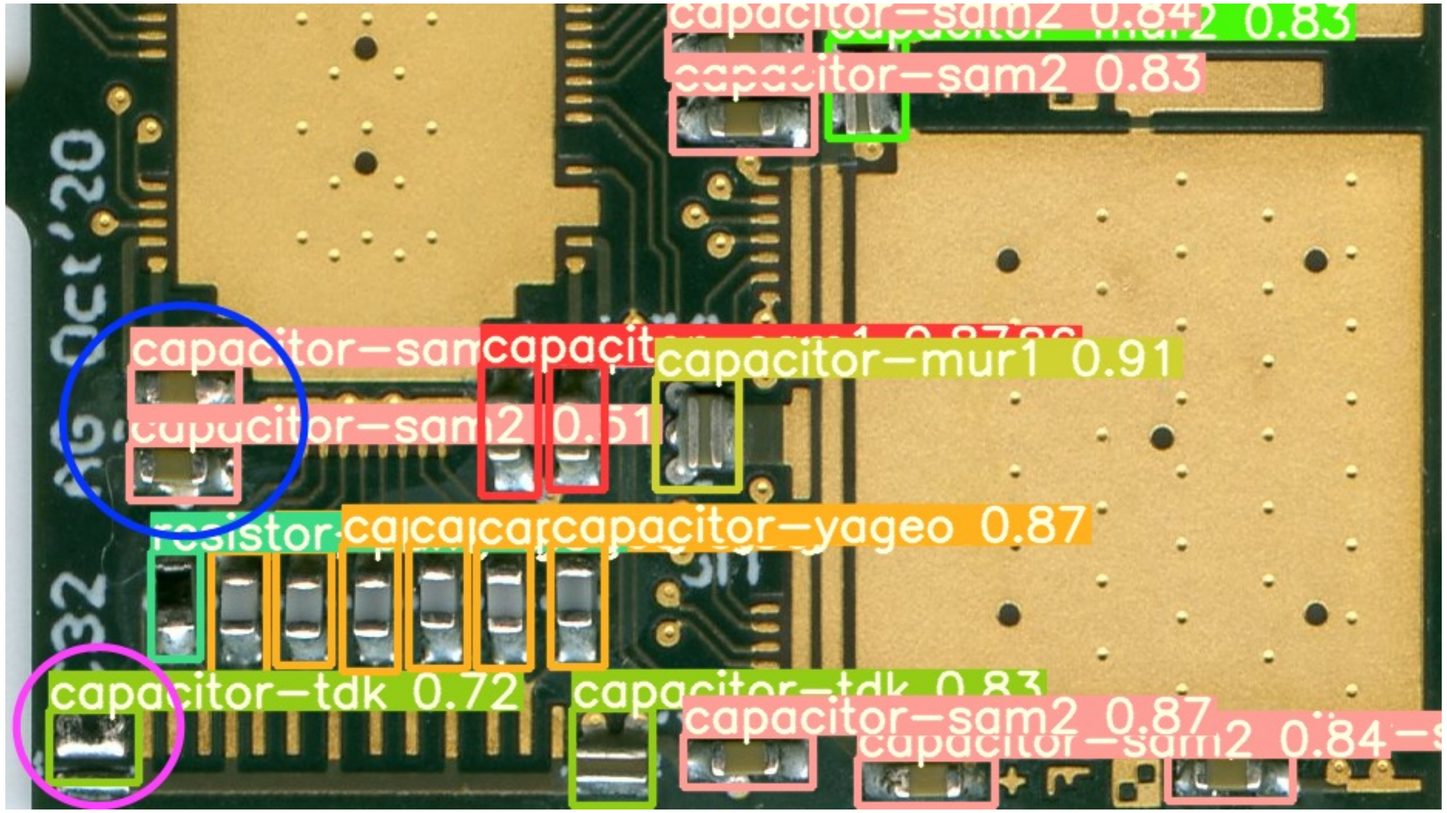
	GPC2104 (83)		GPC2105 (84)	
	PASS	FAIL	PASS	FAIL
P1	83	0	83	1
P2	81	1	79	5
TOTAL		1		6
TOTAL (%)		1%		7%

Some Examples of what it flagged up



- The resistor packs were missed by the neural network
- This could be improved by more images of the components during the training

Some Examples of what it flagged up



- Two capacitors are rotated WRONGLY during assembly
- The neural network system picked this up
- The company's AOI system didn't
- So, we can already see that this work is worthwhile

Summary

- Classification:
  - worked well with aligned hybrids
  - can be used when you know where things are (ROI, Alignment, BOM)
  - has been trained including defective/missing components

- Object Detection:
  - shown to work well (with yolov5)
  - this gives you the position as well as the classification
  - but you no longer know where things **should** be

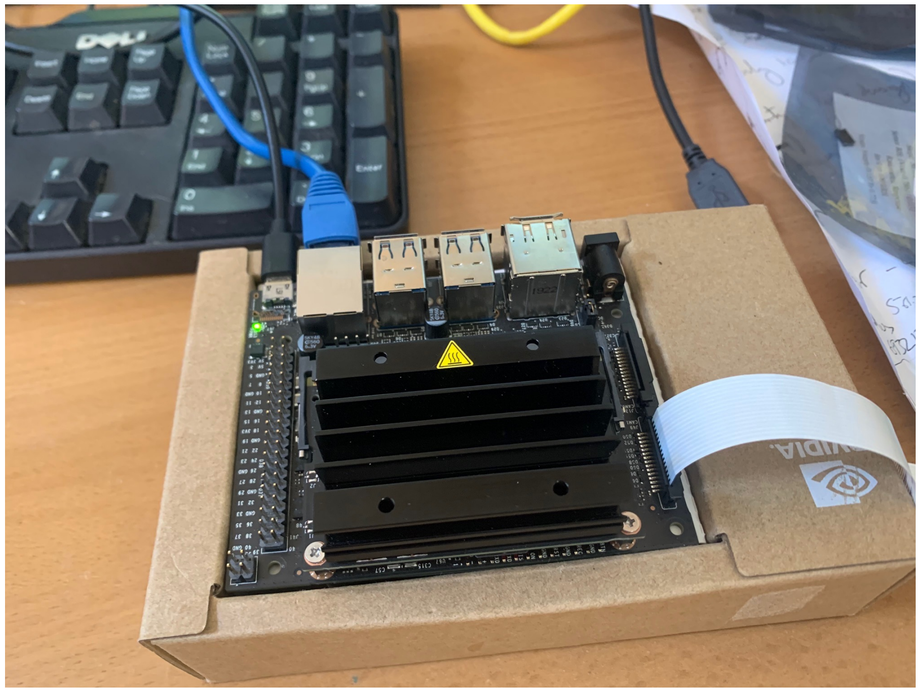
Both methods already show promising results – **We only need to check less than 10% of the modules by eye**

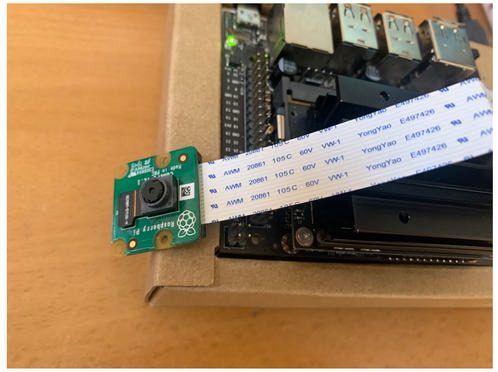
Future/Continuing work:

Some combination of the two is probably needed
Identifying Solder splash is ongoing (various methods)
There are not enough pictures of defects (investigating GANs to produce real fake images)

A final thought/wish

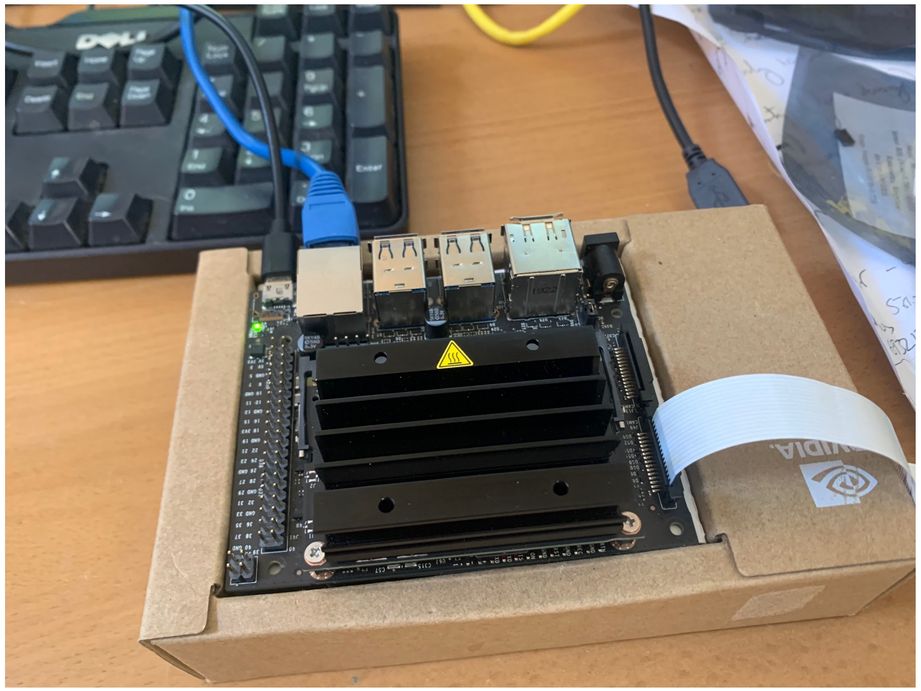
- Once you train a YOLO network they can run on video or real time cameras.
- YOLO can also work on “edge” devices
- Now playing with this on NVIDIA Jetson GPU and a Raspberry pi

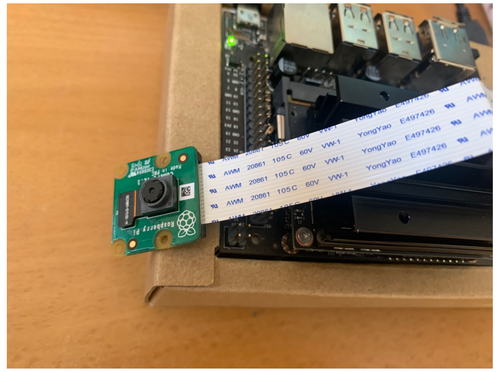




A final thought/wish

- Once you train a YOLO network they can run on video or real time cameras.
- YOLO can also work on “edge” devices
- Now playing with this on NVIDIA Jetson GPU and a Raspberry pi





→ Self driving hybrid Inspection is getting closer