DEFECT INSPECTION FROM SCRATCH TO PRODUCTION

Andrew Liu, Ryan Shen

Deep Learning Solution Architect

NVIDIA

AGENDA

Defect Inspection and its challenges

NGC Docker images

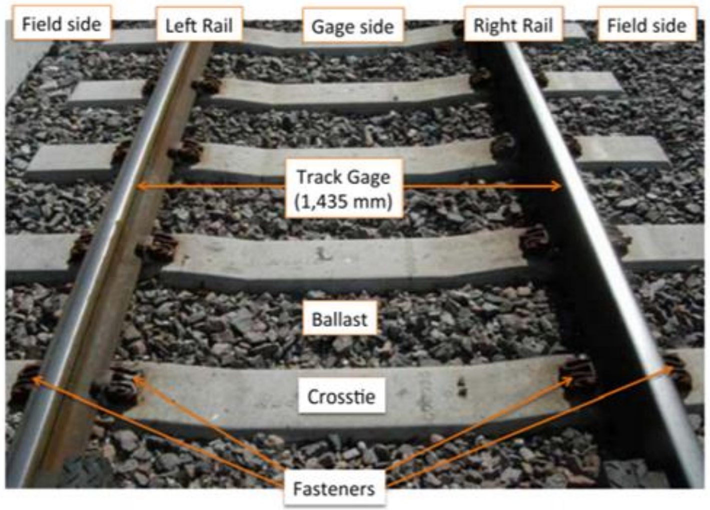
Model set up - Unet

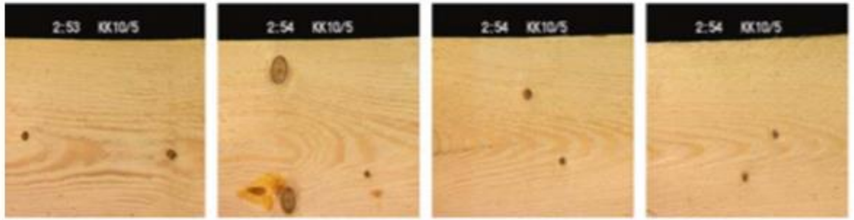
Data preparation - DAGM

Production - Speed up with TensorRT

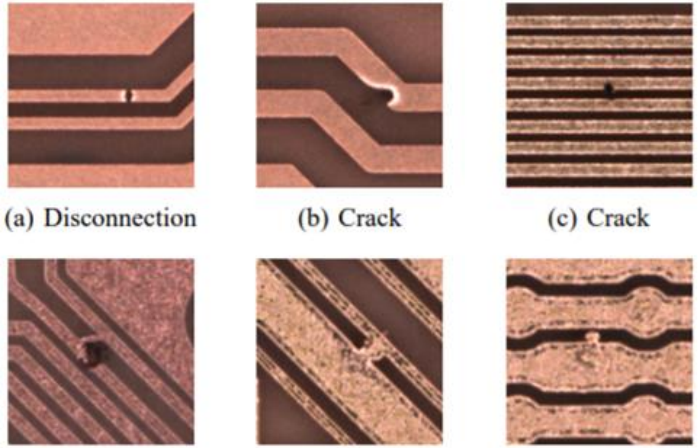
INDUSTRIAL
DEFECT INSPECTION

INDUSTRIAL DEFECT INSPECTION





(a) (b) (c) (d)



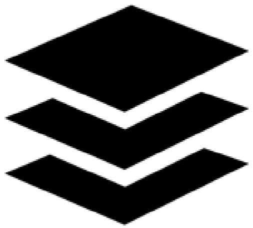
(a) Disconnection (b) Crack (c) Crack

(d) Connection (e) Connection (f) Projection

NGC DOCKER IMAGES

Benefits for Deep Learning Workflow

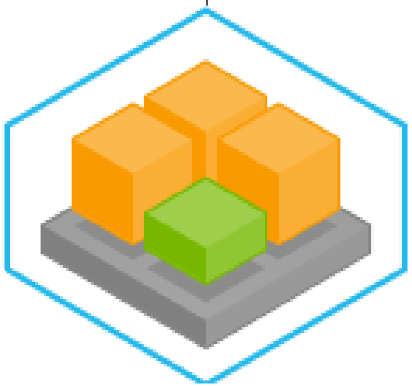
High Level Benefits and Feature Set



Single software stack

Develop once, deploy anywhere

Scale across teams of practitioners
Developer, DevOp, QC

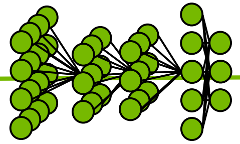


Defect inspection Workflow

from scratch to production within container

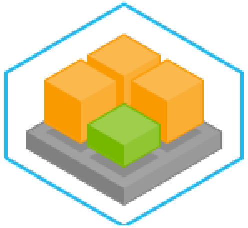
Training

Inference

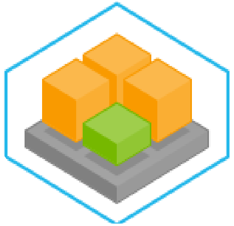


NGC optimized docker image

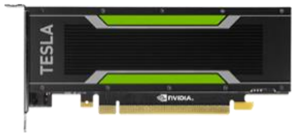
TENSORRT • GRE



docker pull
nvcr.io/nvidia/**tensorflow**
:18.02-py3



docker pull
nvcr.io/nvidia/**tensorrt**:
18.02-py2

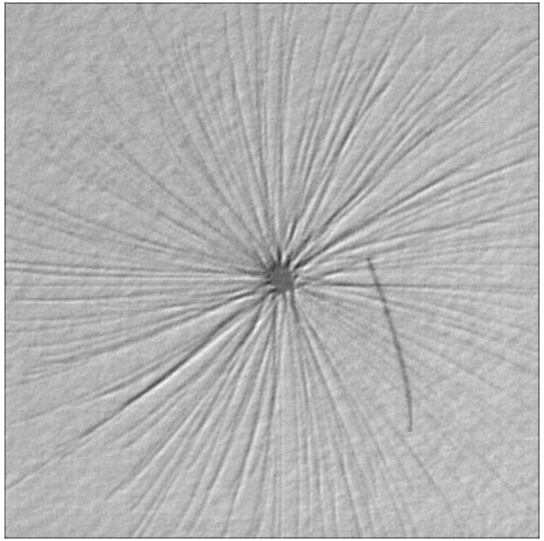


MODEL SET UP

WAYS FOR DEFECT INSPECTION

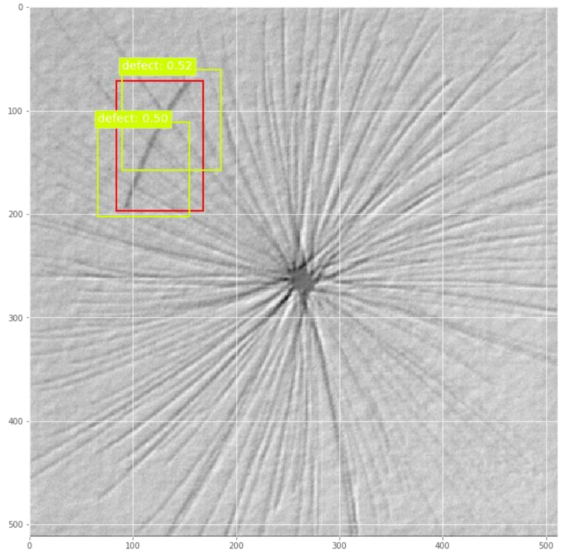
Depends on domain adaptation

Image Classification



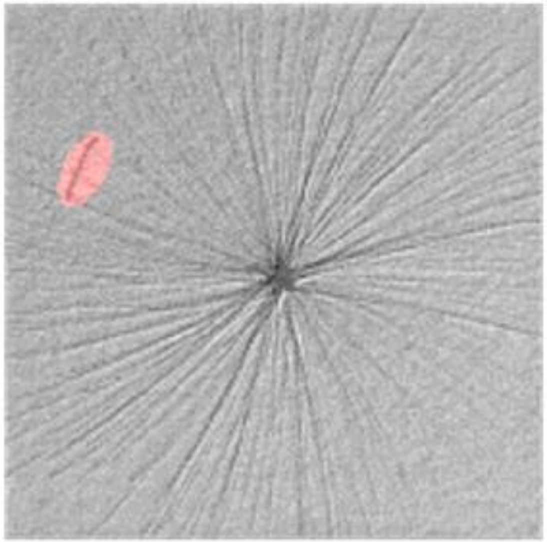
Label: 0 / 1
(Defect / Non Defect)

Object Detection



Label: Bounding-Box

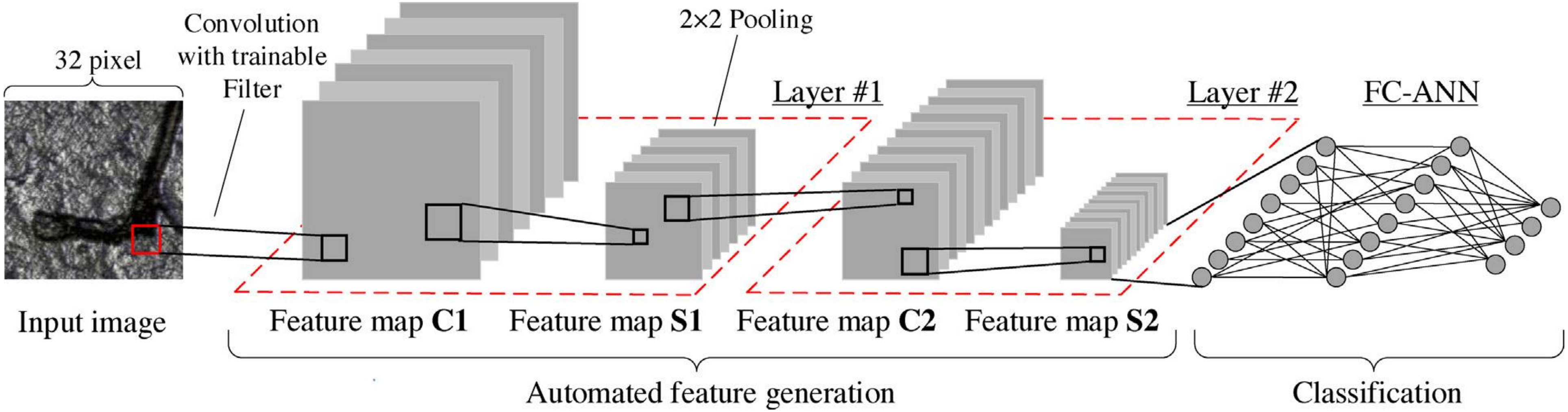
Image Segmentation



Label: Polygons Mask

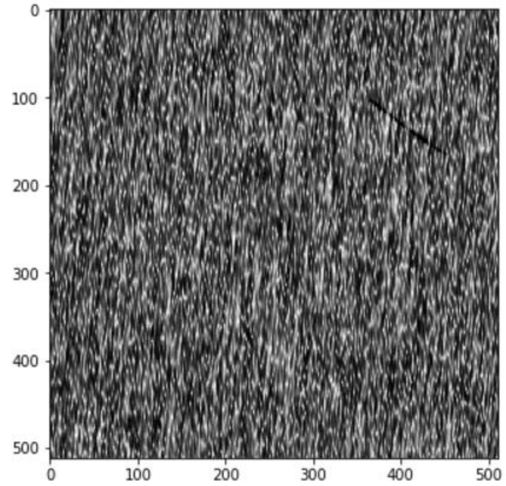
CNN STRUCTURE

LeNet

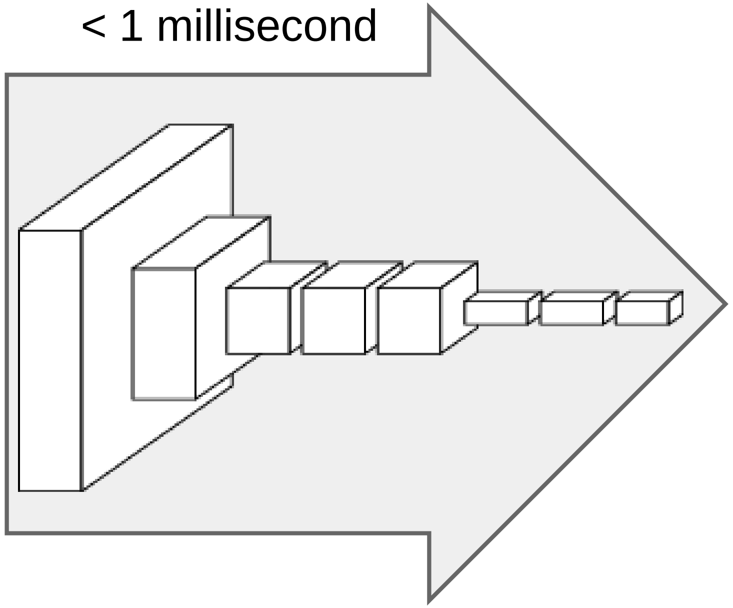


Source: Design of Deep Convolutional Neural Network Architectures for Automated Feature Extraction in Industrial Inspection, D. Weimer et al, 2016

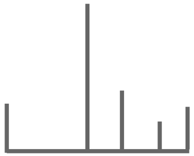
convnets perform classification



< 1 millisecond



end-to-end learning

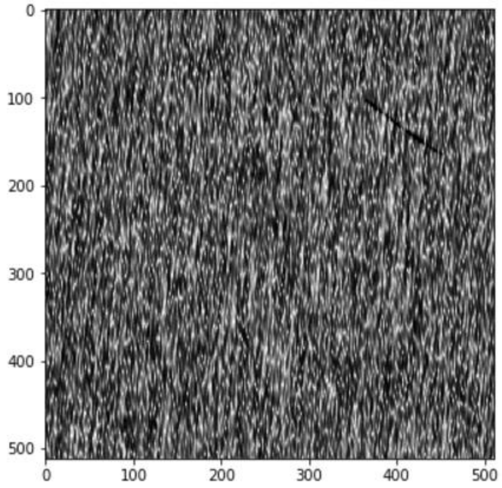


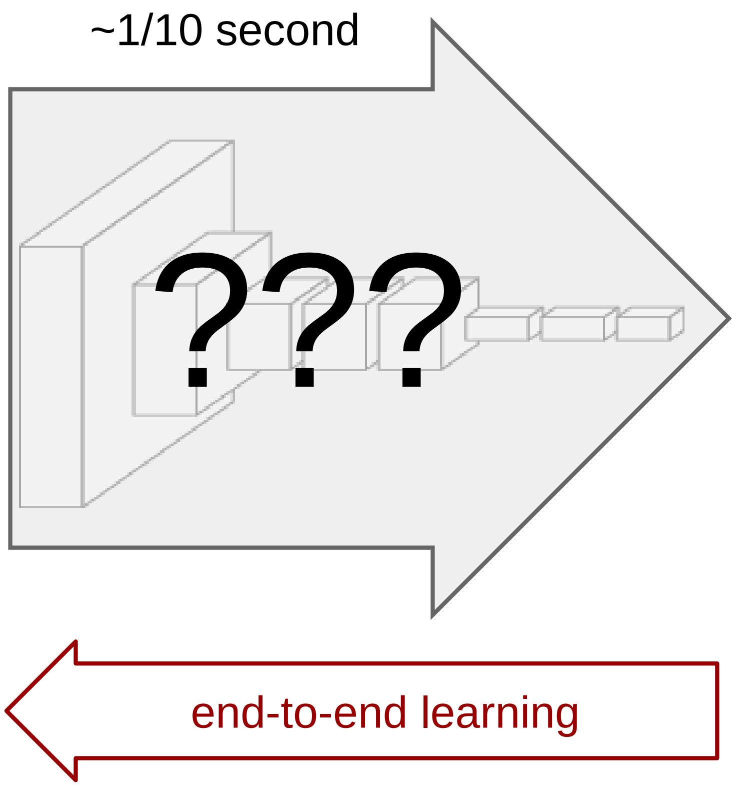
1000-dim vector

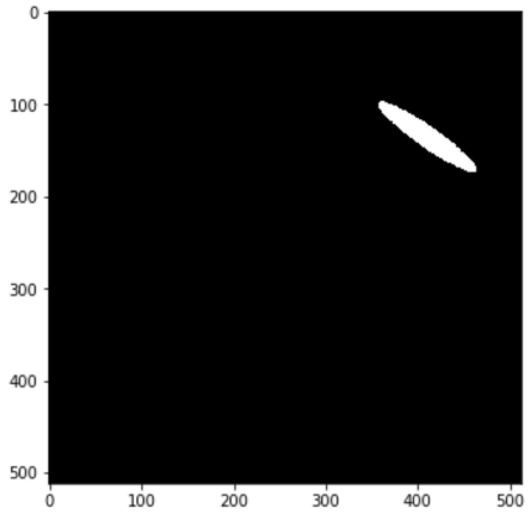
“defect class1”

https://arxiv.org/abs/1605.06211

lots of pixels, little time?

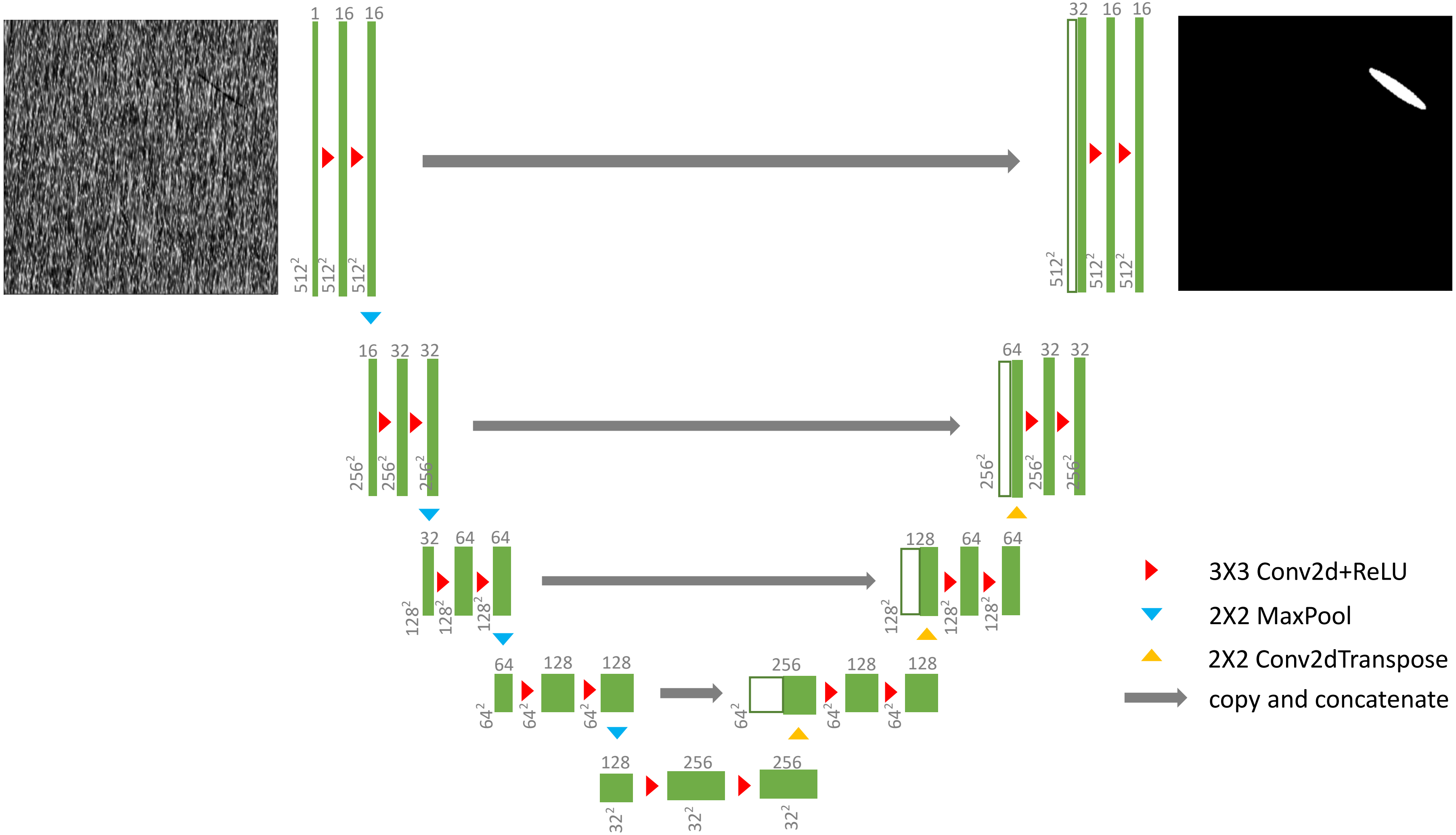






https://arxiv.org/abs/1605.06211

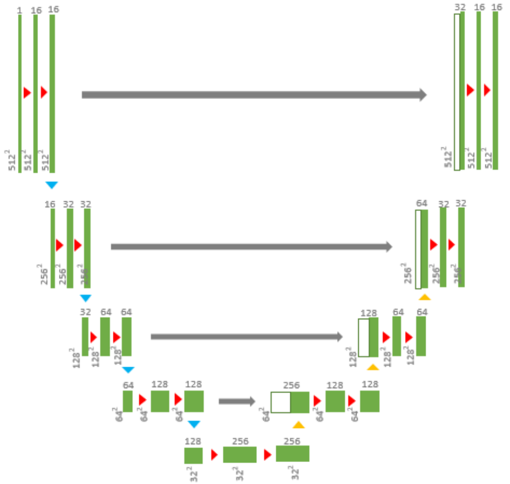
U-Net structure



KERAS IMPLEMENTATION

Convolution

```
inputs = Input((IMAGE_HEIGHT, IMAGE_WIDTH, IMAGE_CHANNELS))
inputs_norm = Lambda(lambda x: x/127.5 - 1.)
conv1 = Conv2D(8, (3, 3), activation='relu', padding='same')(inputs)
conv1 = Conv2D(8, (3, 3), activation='relu', padding='same')(conv1)
pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)

conv2 = Conv2D(16, (3, 3), activation='relu', padding='same')(pool1)
conv2 = Conv2D(16, (3, 3), activation='relu', padding='same')(conv2)
pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)

conv3 = Conv2D(32, (3, 3), activation='relu', padding='same')(pool2)
conv3 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv3)
pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)

conv4 = Conv2D(64, (3, 3), activation='relu', padding='same')(pool3)
conv4 = Conv2D(64, (3, 3), activation='relu', padding='same')(conv4)
pool4 = MaxPooling2D(pool_size=(2, 2))(conv4)

conv5 = Conv2D(128, (3, 3), activation='relu', padding='same')(pool4)
conv5 = Conv2D(128, (3, 3), activation='relu', padding='same')(conv5)
```



DECODING

deconvolution

```
up6 = merge([UpSampling2D(size=(2, 2))(conv5), conv4], mode='concat',
conv6 = Conv2D(64, (3, 3), activation='relu', padding='same')(up6)
conv6 = Conv2D(64, (3, 3), activation='relu', padding='same')(conv6)

up7 = merge([UpSampling2D(size=(2, 2))(conv6), conv3], mode='concat',
conv7 = Conv2D(32, (3, 3), activation='relu', padding='same')(up7)
conv7 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv7)

up8 = merge([UpSampling2D(size=(2, 2))(conv7), conv2], mode='concat',
conv8 = Conv2D(16, (3, 3), activation='relu', padding='same')(up8)
conv8 = Conv2D(16, (3, 3), activation='relu', padding='same')(conv8)

up9 = merge([UpSampling2D(size=(2, 2))(conv8), conv1], mode='concat', concat_axis=3)
conv9 = Conv2D(8, (3, 3), activation='relu', padding='same')(up9)
conv9 = Conv2D(8, (3, 3), activation='relu', padding='same')(conv9)

conv10 = Conv2D(1, (1, 1), activation='sigmoid')(conv9)

model = Model(inputs=inputs, outputs=conv10)
```

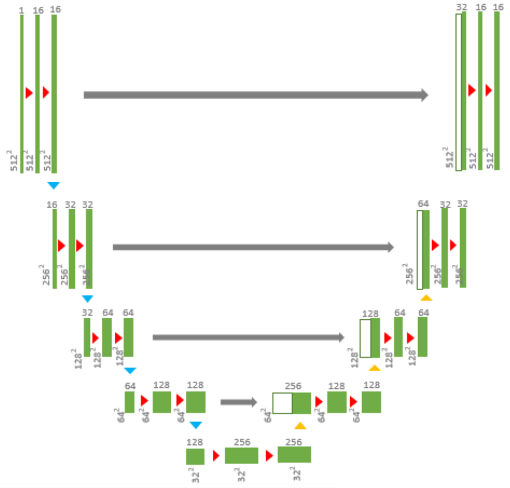
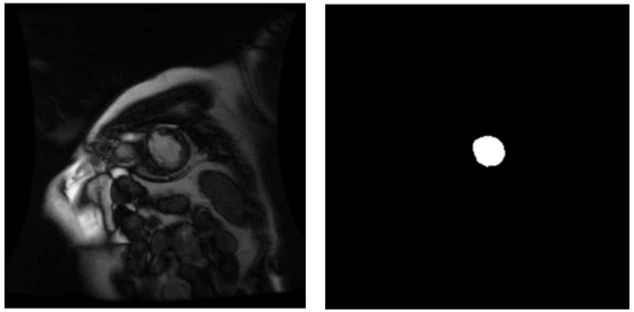


Image segmentation on medical images

Same process among various use cases

Data Science BOWL
2016

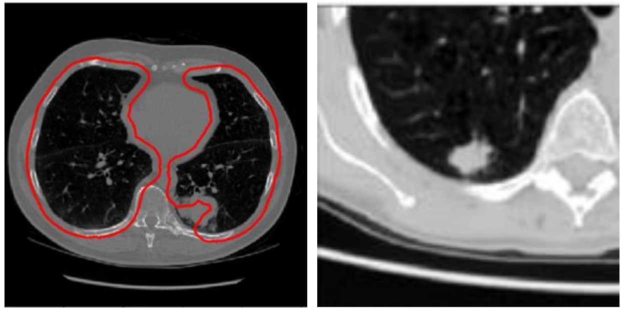


MRI image

Left ventricle

heart disease

Data Science BOWL
2017

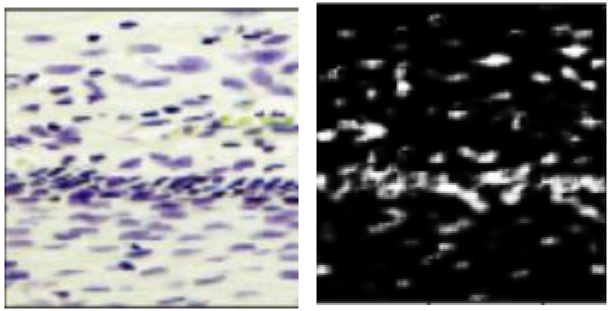


CT image

Nodule

Lung cancer

Data Science BOWL
2018



Image

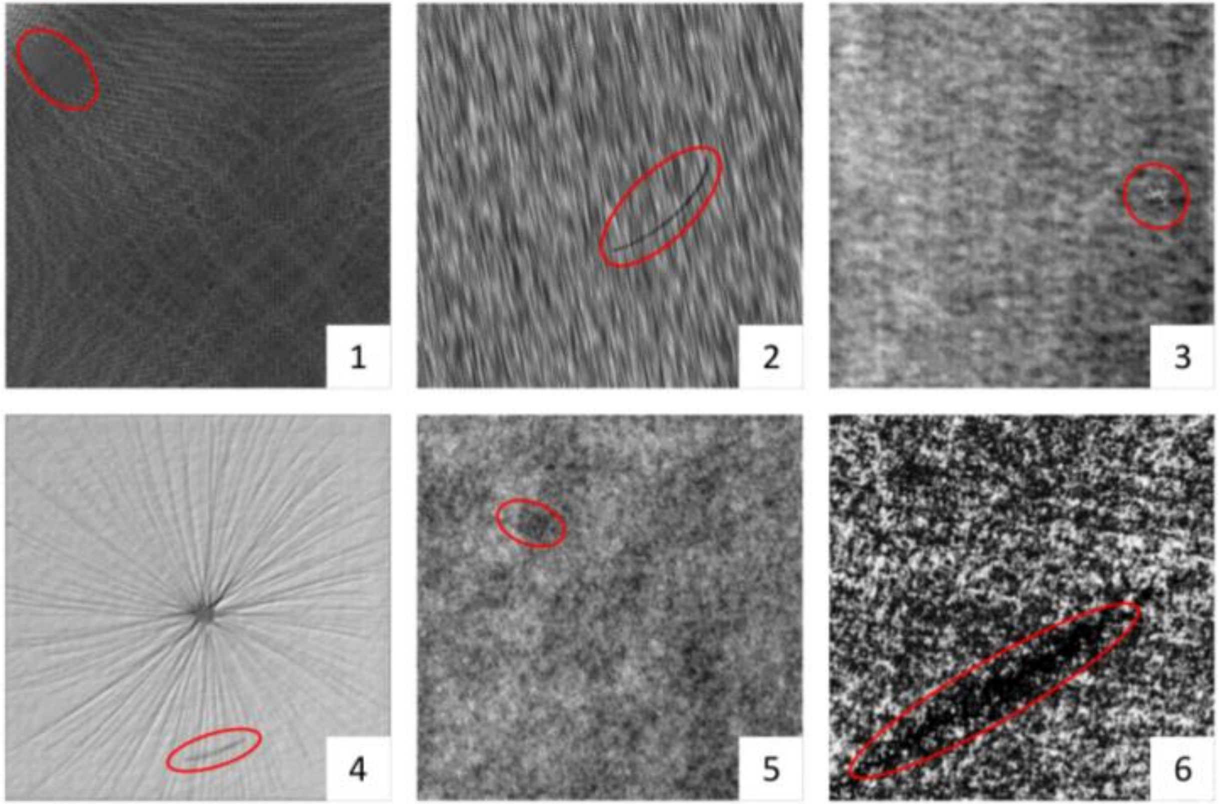
Nuclei

Drug discovery

DATA PREPARATION

INDUSTRIAL OPTICAL INSPECTION

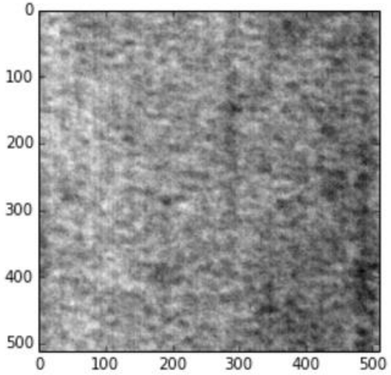
German Association for Pattern Recognition



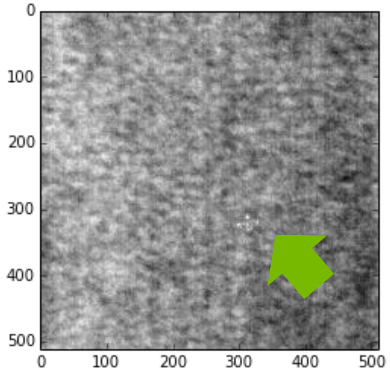
- http://resources.mpi-inf.mpg.de/conferences/dagm/2007/prizes.html

INDUSTRIAL OPTICAL INSPECTION

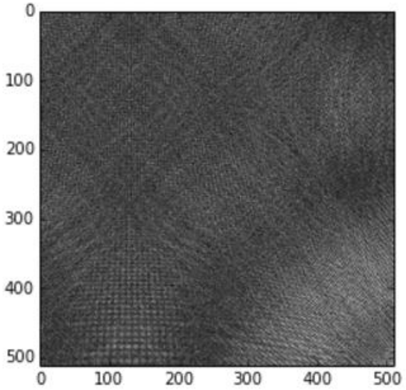
German Association for Pattern Recognition



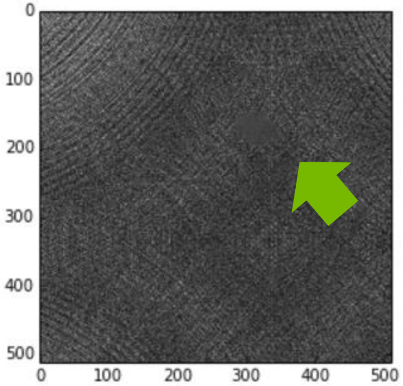
Pass



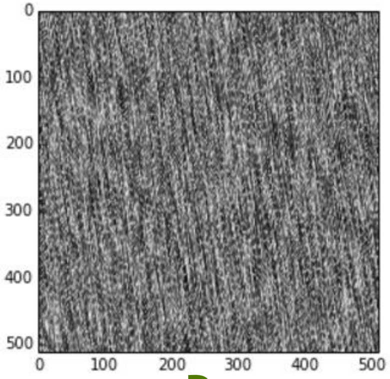
NG



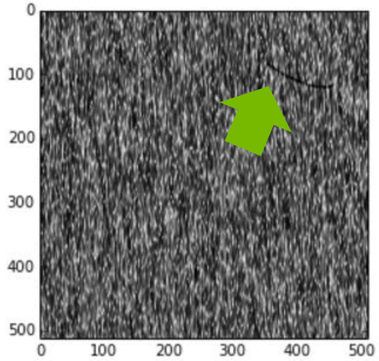
Pass



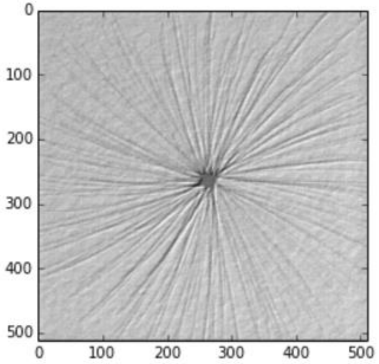
NG



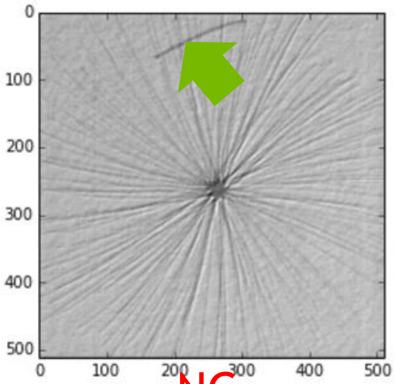
Pass



NG



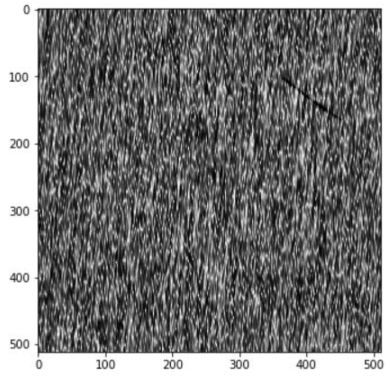
Pass

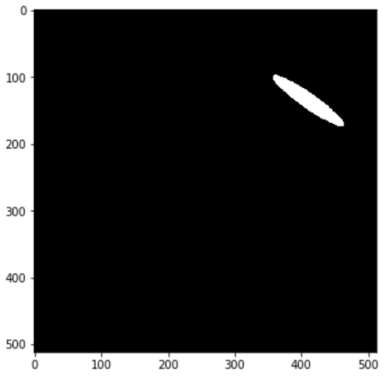


NG

DATA DETAILS

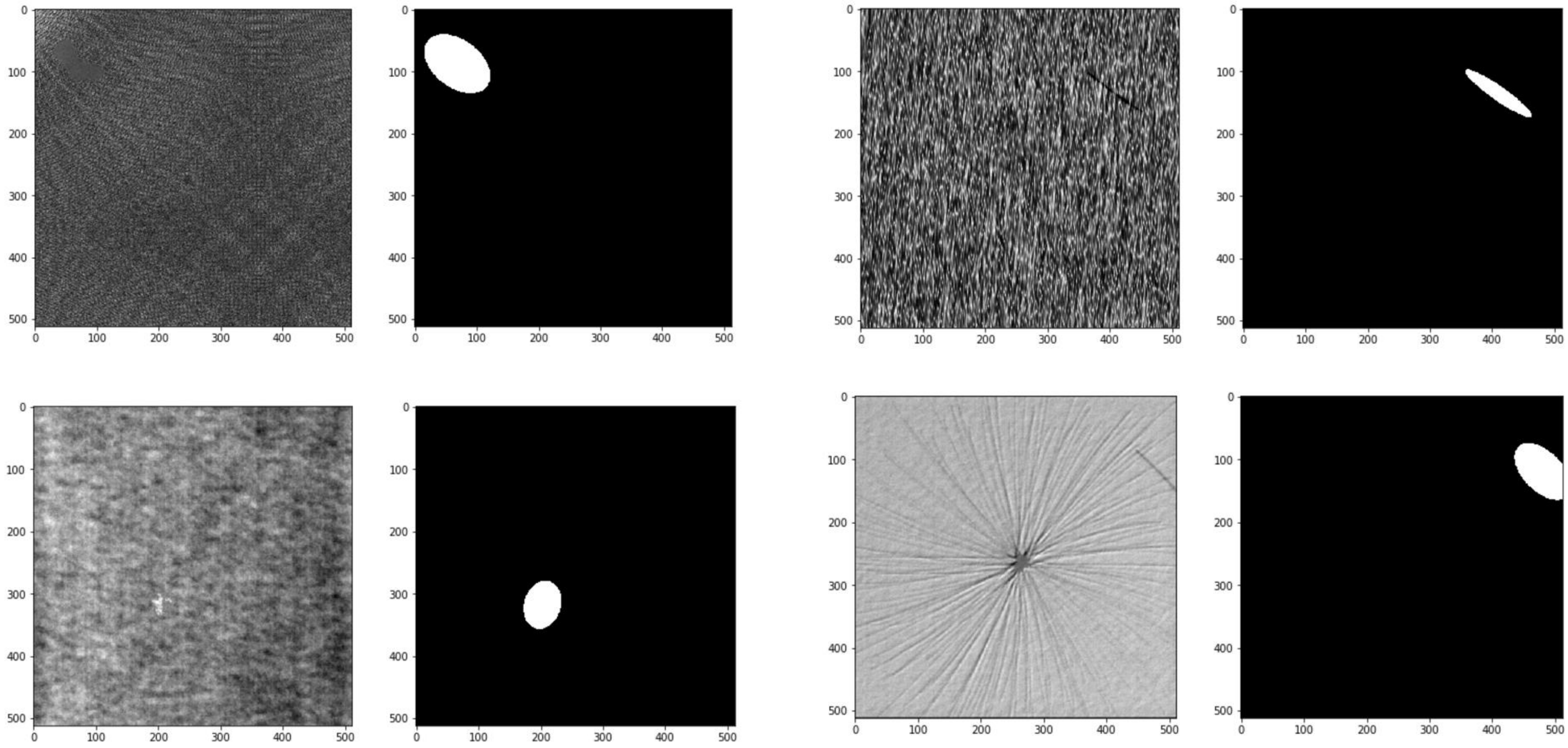
- Original images are 512 x 512 grayscale format
- Output is a tensor of size 512 x 512 x 1
  - Each pixel belongs to one of two classes





- Training set consist of 100 images
- Validation set consist of 50 images

MORE EXAMPLES

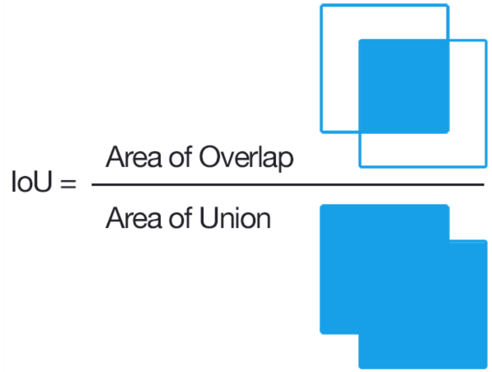


IMBALANCE DATA

Dice Metric (IOU)

- Metric to compare the similarity of two samples:

$$\frac{2A_{nl}}{A_n + A_l}$$



  - Where:
    - A_n is the area of the contour predicted by the network
    - A_l is the area of the contour from the label
    - A_{nl} is the intersection of the two
      - The area of the contour that is predicted correctly by the network
      - 1.0 means perfect score.

- More accurately compute how well we're predicting the contour against the label

- We can just count pixels to give us the respective areas

LOSS FUNCTION WITH KERAS

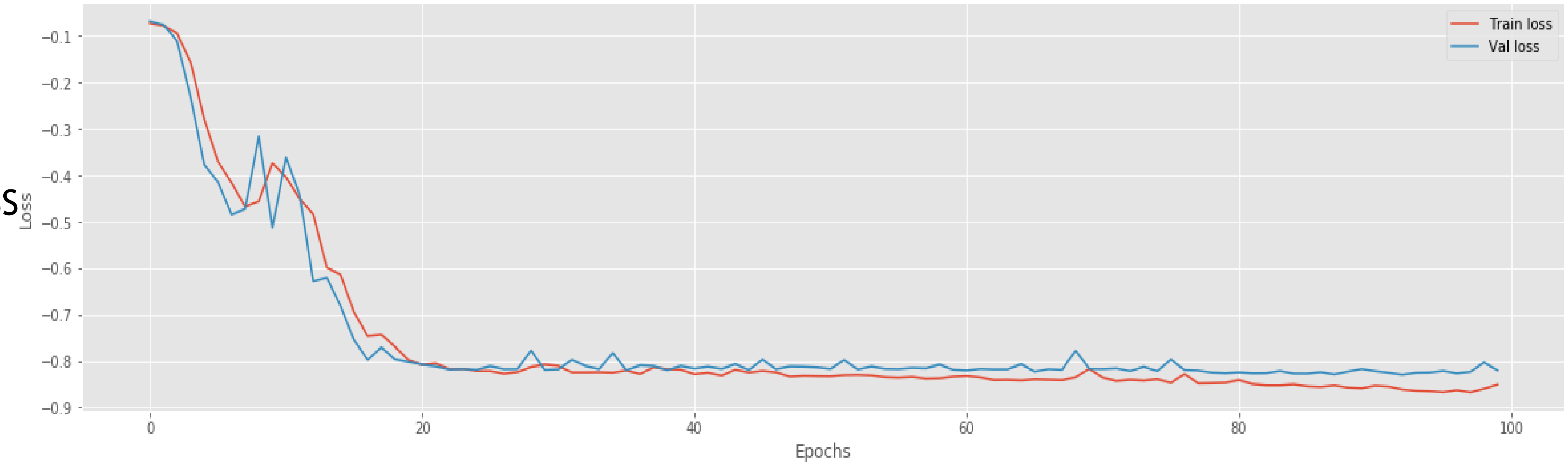
```
def IOU_calc(y_true, y_pred):
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(y_true_f * y_pred_f)

    return 2*(intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) + smooth)
```

```
def IOU_calc_loss(y_true, y_pred):
    return -IOU_calc(y_true, y_pred)
```

```
model.compile(optimizer=Adam(lr=1e-4), loss=IOU_calc_loss, metrics=[IOU_calc])
```

LEARNING CURVES

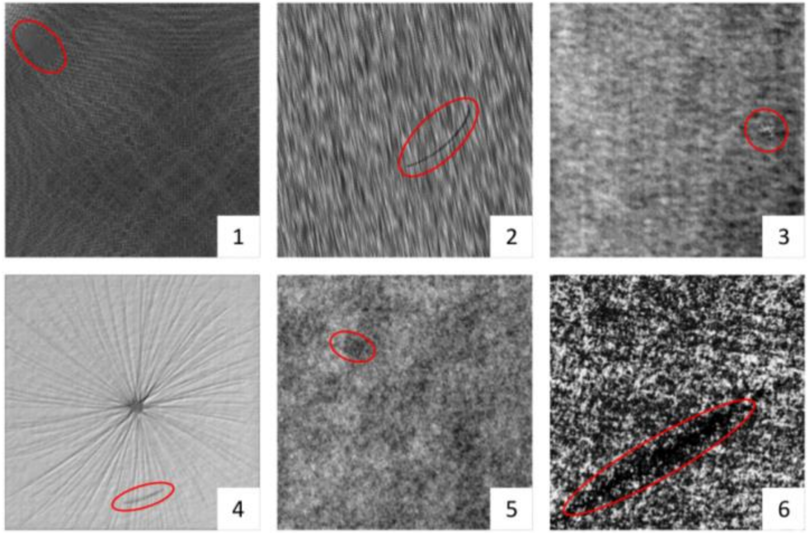
LOSS

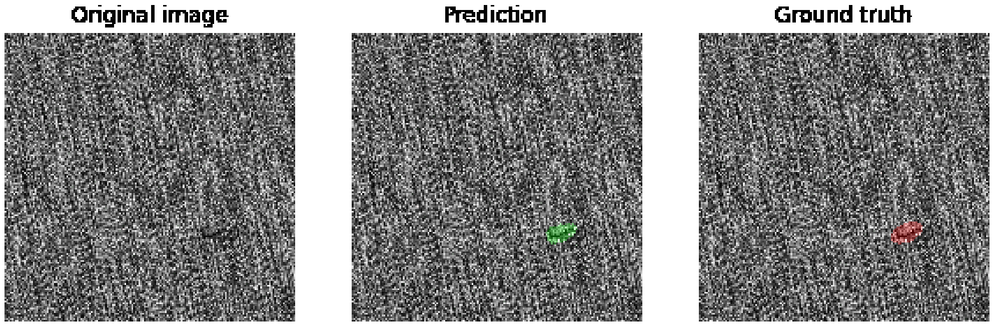


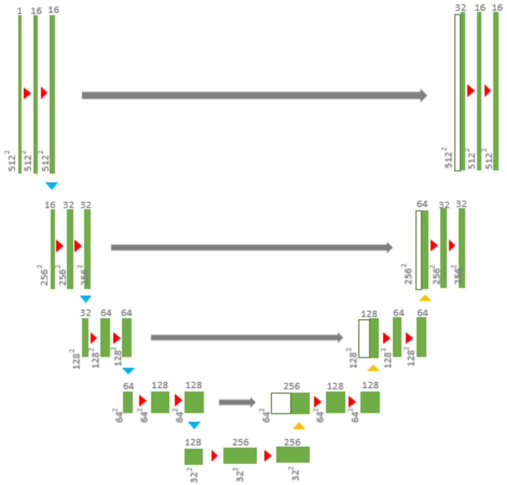
APPLICATION: INDUSTRIAL INSPECTION

NVIDIA

- **1000** defect-free, **150** defect images
- **Challenges**: Not all deviations from the texture are necessarily defects.



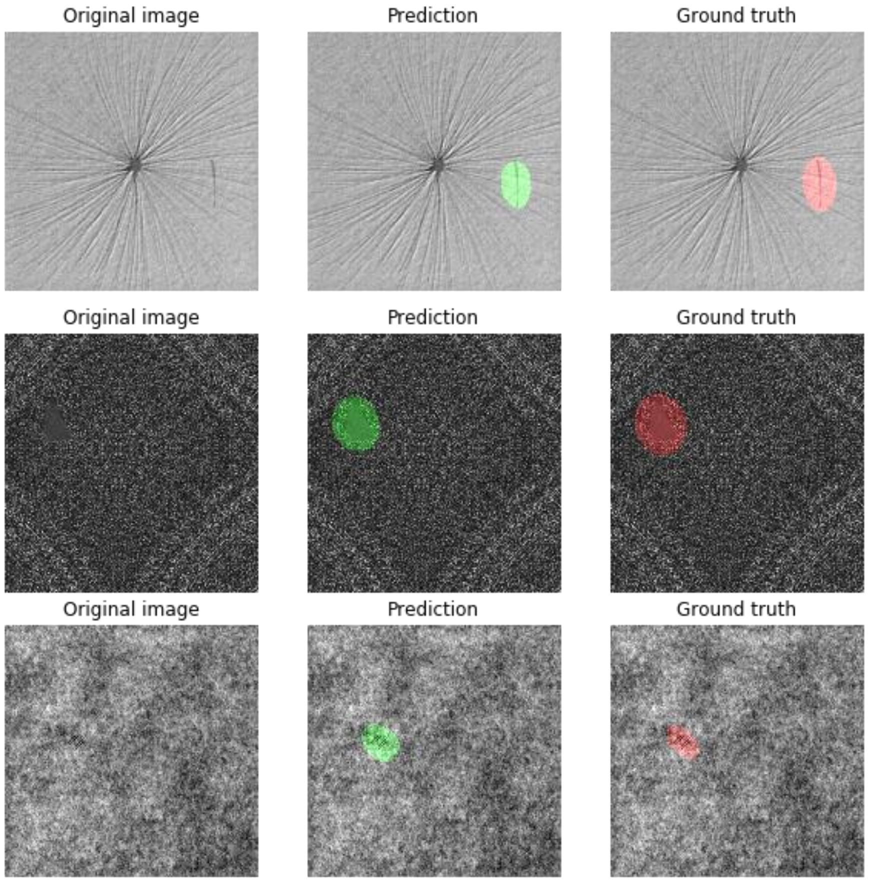




PRODUCTION

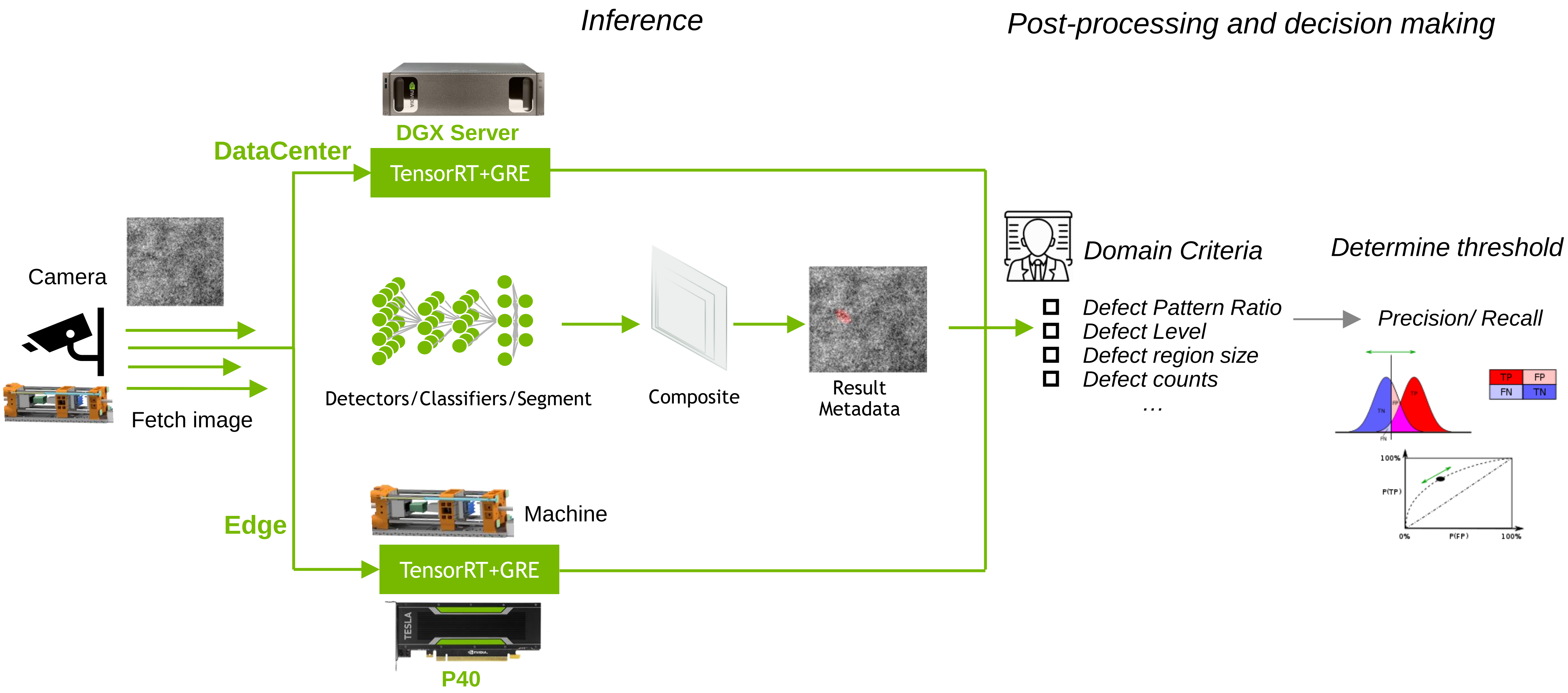
FINAL DECISION

Plus Human Logic

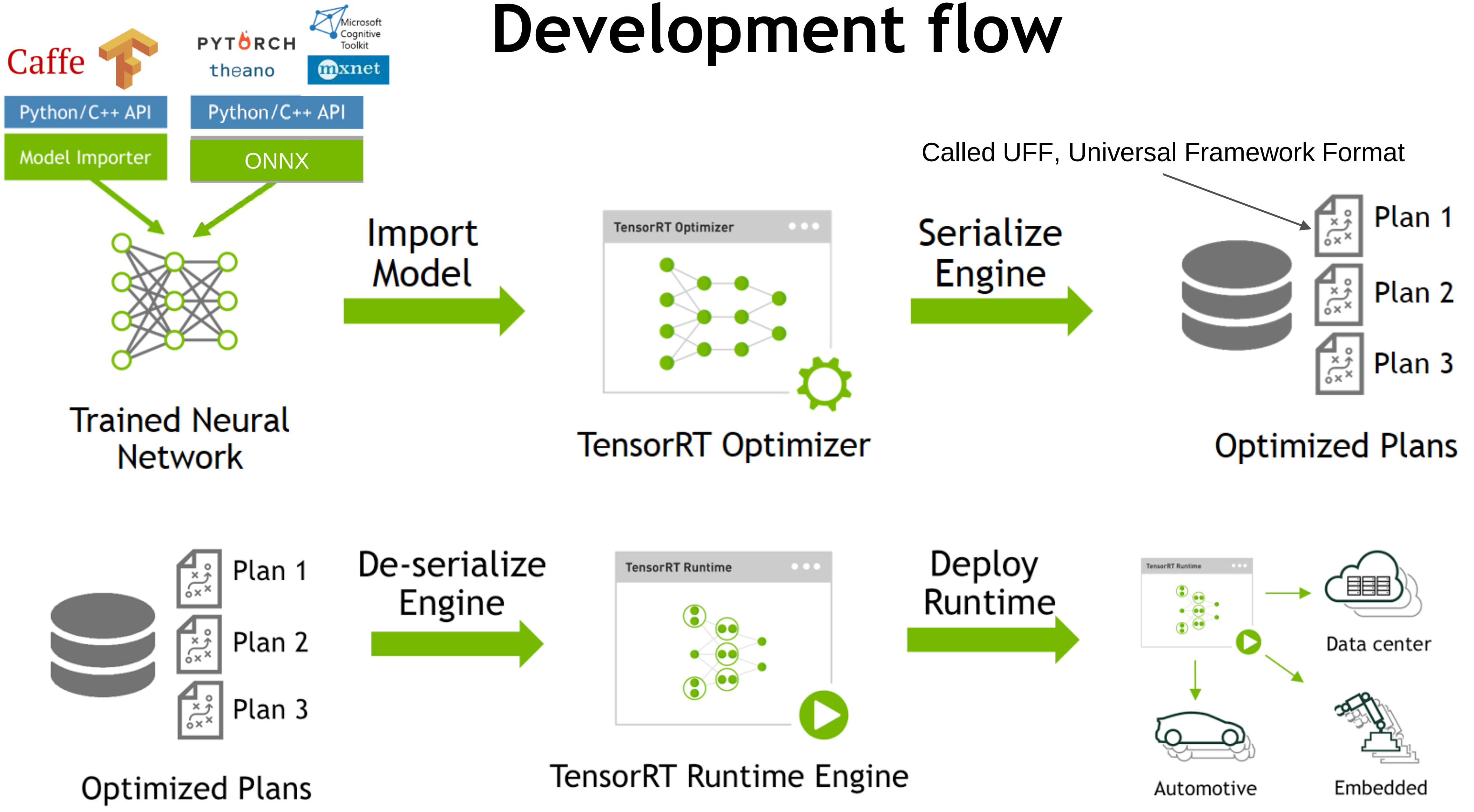


Size, Position, … etc

INFERENCE PIPELINE



Development flow

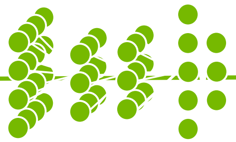


Defect inspection Workflow

from scratch to production within container

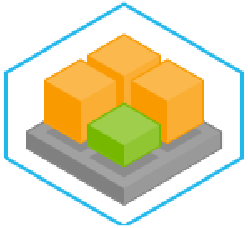
Training

Inference

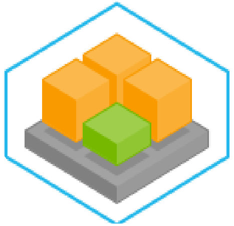


NGC optimized docker image

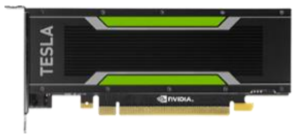
TENSORRT • GRE



docker pull
nvcr.io/nvidia/**tensorflow**
:18.02-py3



docker pull
nvcr.io/nvidia/**tensorrt**:
18.02-py3



SUMMARY

Challenges	Delivers
Training , inference environment is hard to build, maintain, share.	Using NGC Docker images.
Model optimizations and speed up throughput.	TensorRT + GRE SDK
So many deep learning model out there, how to choose the right model?	If your dataset, demand requirement fit the scenario like we do. U-Net model is great choice for segmentation task.

Thank You