Octane'20
Review Presentation

Project : To find low cost vision system for paint finish inspection.

Department : Engineering

Mentor : Sandeep Magdum Sir

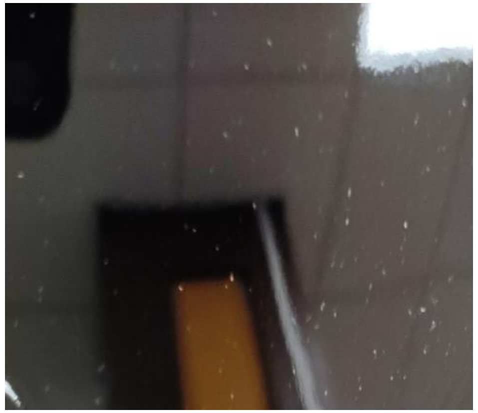
BU-HR : Vijay Sonar Sir

Paint defects and challenges

- Identifying paint defects on petrol tank.

1. Runs
2. Dust
3. Craters

Runs



Dust

Crater

OK

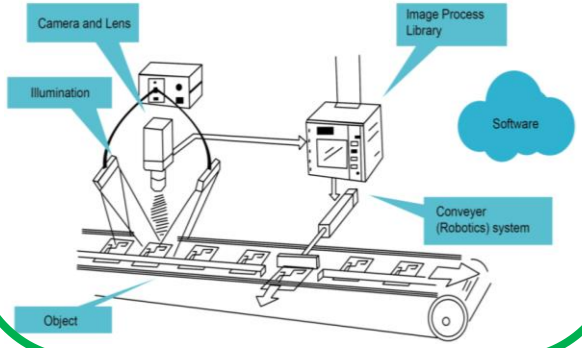
Challenges

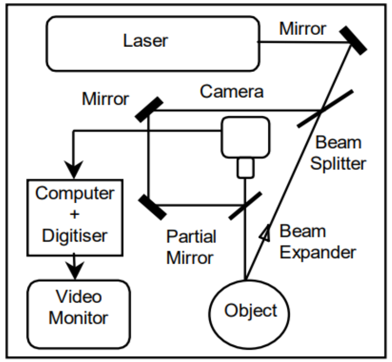
- Glare on specular curved surface hides paint defects.
- Reflections on specular tank surface surfaces change pixel intensities of image.

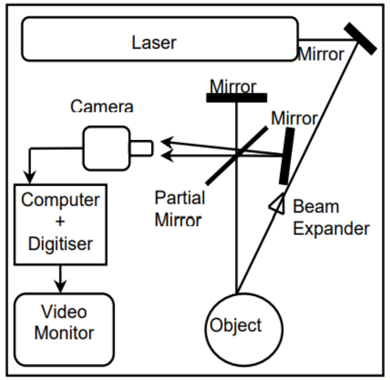
Defect inspection method

Inspection methods

- Machine vision
  - Cost < 2 lakhs
    Less components
    Simple data analysis
- Interferometry
  - Cost>7 lakhs
    More components
    Complex data analysis
- Shearograpgy
  - Cost>7 lakhs
    More components
    Complex data analysis







Vision System for Paint finish inspection

Features of Vision System

- High Accuracy
- Low inference time
- Cost effective
- Simple Layout
- Reliable

Inspection system

- Visual representation of paint defect
- Data collection by camera
- Data transfer through a channel
- Data processing unit
- Defect classification

Paint defect → Camera and lighting → Interface → Edge computing → Image preprocessing → Image classification

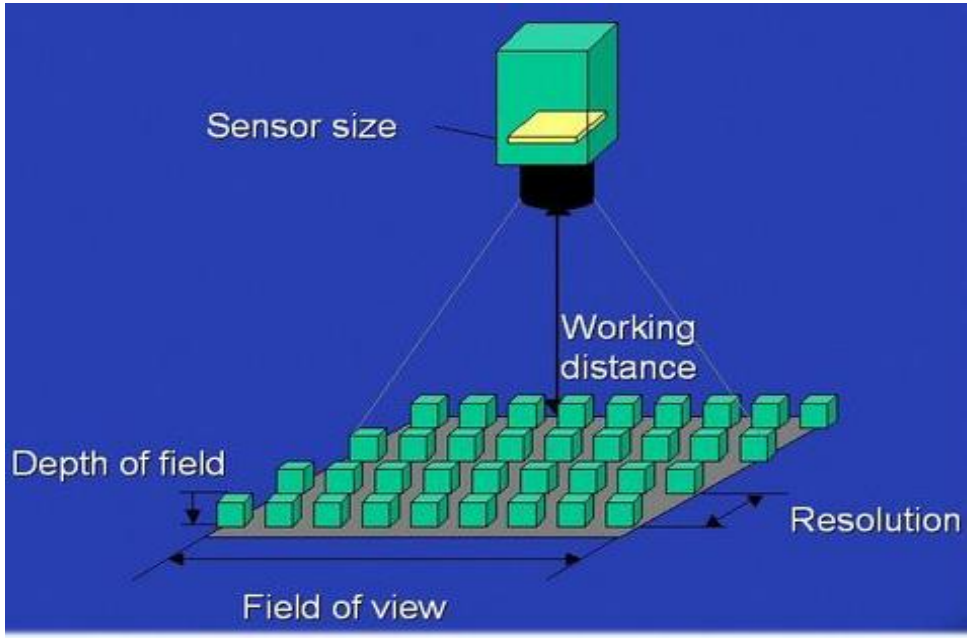
Camera

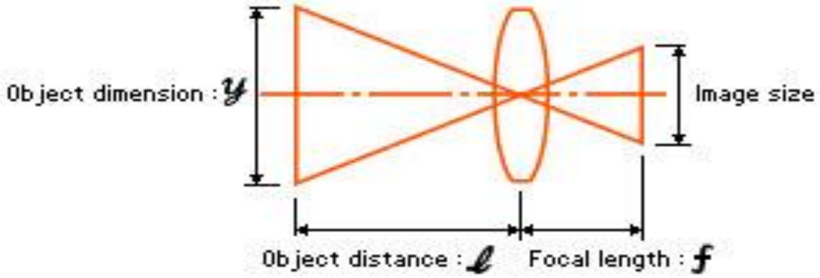
Pixel size & camera resolution

"At least 2 pixels must fit in an defect"

- Work depth(WD)=500mm.
- Field of view(FOV)=300mm
- Focal length(FL)=16mm
- FOV* FL=Sensor* WD
- Pixel size=5 microns
- Pixel size= Sensor/Resolution
- **Resolution>=6MP**

Defect	Dimension (micrometer)
Dust	20
Pinhole	50
Crater	100
Solvent popping	10-50



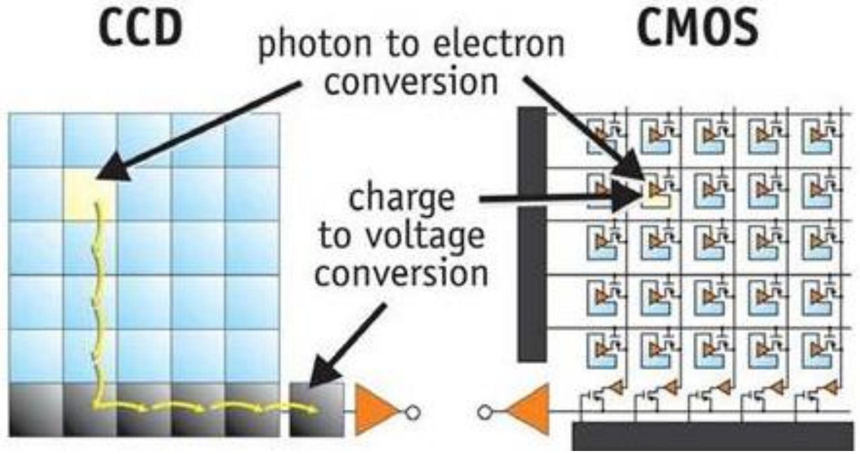


- **Selected Resolution can easily find minimum size defect.**
- **Defects like runs can be detected by 5MP resolution as well.**

Camera

1.CMOS Sensor

- Compact size.
- High processing speed.
- Consumes less power.
- Glare resistant.
- Better image resolution and field of view.
- **Global Shutter**.



2. Interface

	USB 3.0	GigE
Bandwidth (Mbps)	400	125
Camera Cost	Low	Low
Cable Cost	Low	Low
CPU Usage	Low	Medium
Multicamera Setup	Excellent	Good
Cable length	4.6m	100m

- **USB 3.0** is selected for low cost.

Low cost

Sensor

- CMOS sensor has less cost than CCD sensor.
- CMOS sensor is widely manufactured and offers more varieties.

Resolution

- Right sizing of camera prevents over sizing of camera specifications.
- Detailed study was done in camera resolution instead of going for higher resolution cameras like 21MP.
- 6MP resolution has been chosen with exactly matching our requirements.

Interface

- USB 3.0 is least cost interface for higher bandwidth.

Camera and Lighting

- Resolution: 6MP
- Focal length: 16mm
- Sensor size: 2/3”
- Sensor: CMOS
- USB 3.0

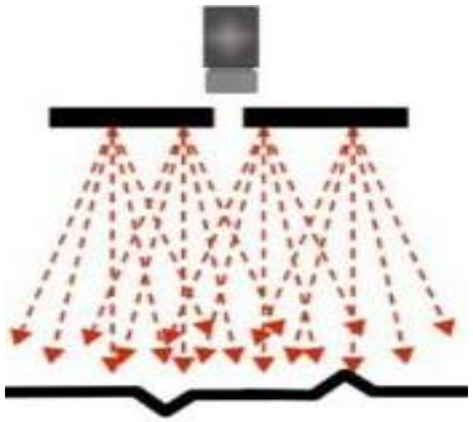
acA3088-57uc - Basler ace

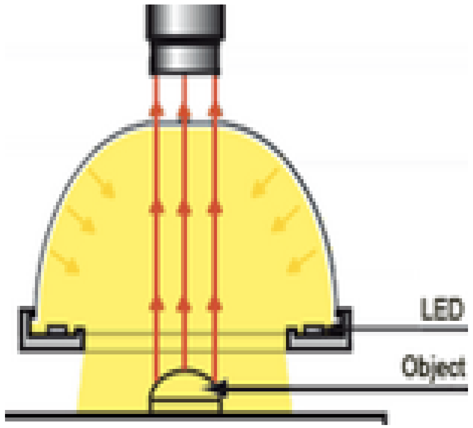
1.LED light is chosen as light source.

- High life span
- High efficiency
- High CRI

2. Bright field illumination is the illumination technique

3. Dome light is chosen as light source geometry.

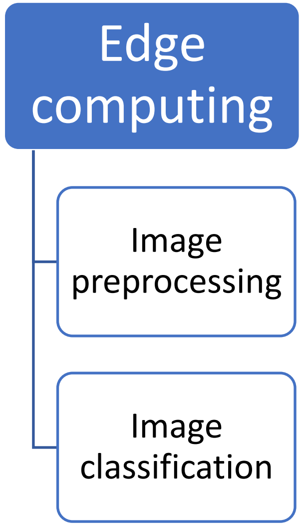




Edge computing

Edge Computing Solution

- Placing computing device near to camera.
- Decreases inference time and interface cost.
- Enables real time service.
- Device will perform Image preprocessing and classification.
- Python code will be uploaded to the device.
- Edge computing devices: Raspberry pi and Nvidia Jetson nano.



Key features

- Latency
- Scalability
- Cost effective

Low cost

- Right sized processing device has been chosen for image processing.
- Standard machine vision processing units come along with PC costing more.
- Right sizing allows to reduce cost significantly.
- Provided solution allows scalability at lower cost.
- Low cost solution for multi camera setup.

Defect Inspection

TEMPLATE MATCHING

- Comparing all frames captured with image dataset.
- Passing selected frames to classification algorithms.

Library: open-cv and matplotlib, Statistics

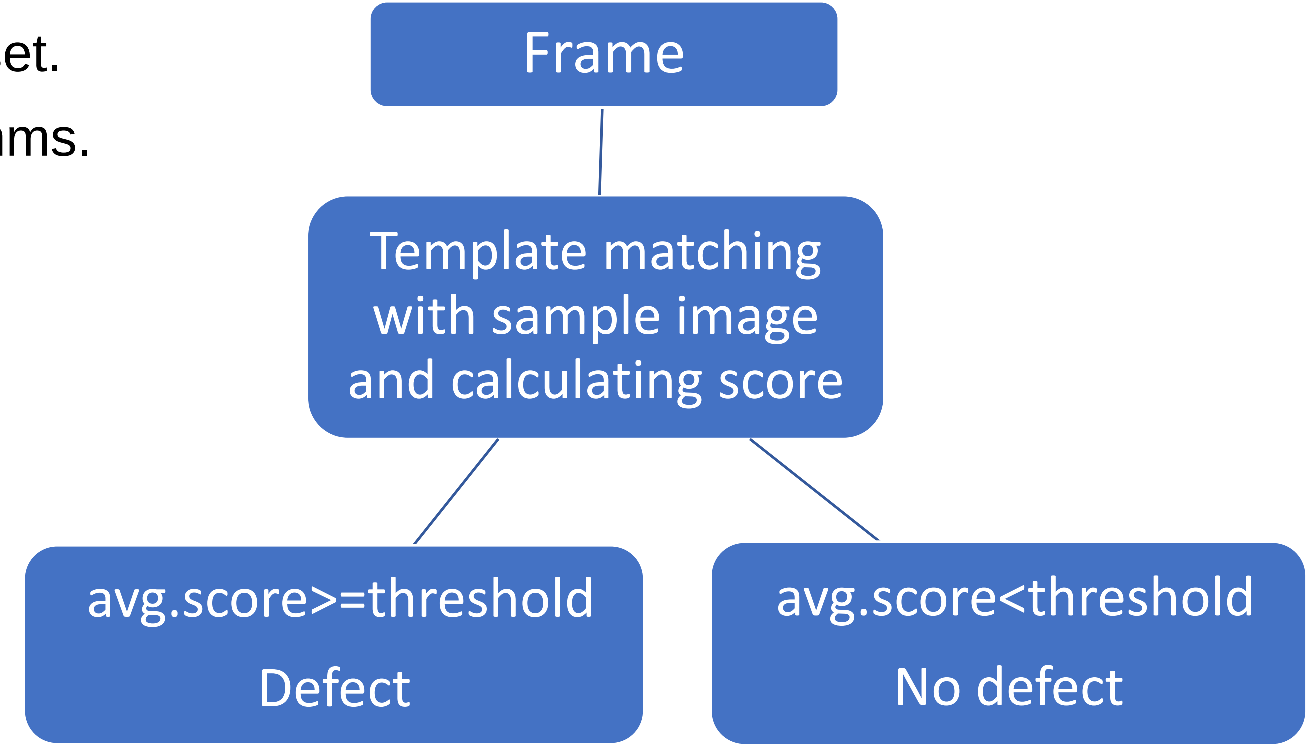
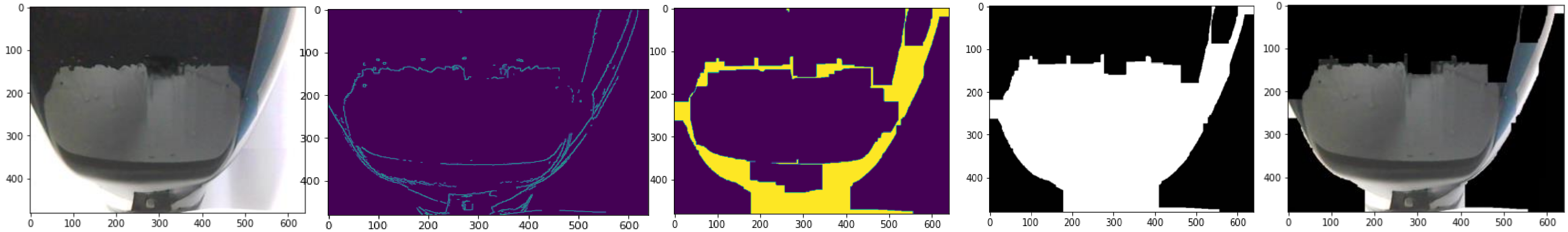


Image preprocessing

Defect image → Edge detection → Image morphing → Contour detection → Contour extraction → Masked image → Bitwise operation



Python Libraries

- Open-cv
- Skimage
- Matplolib
- Numpy

Reason

- Preparing image for classification

Cost

- All methods and algorithms are self developed.
- Open source software.

Deep learning models for classification

Criteria for Selection

1. High accuracy
2. Low inference time
3. Low space requirement

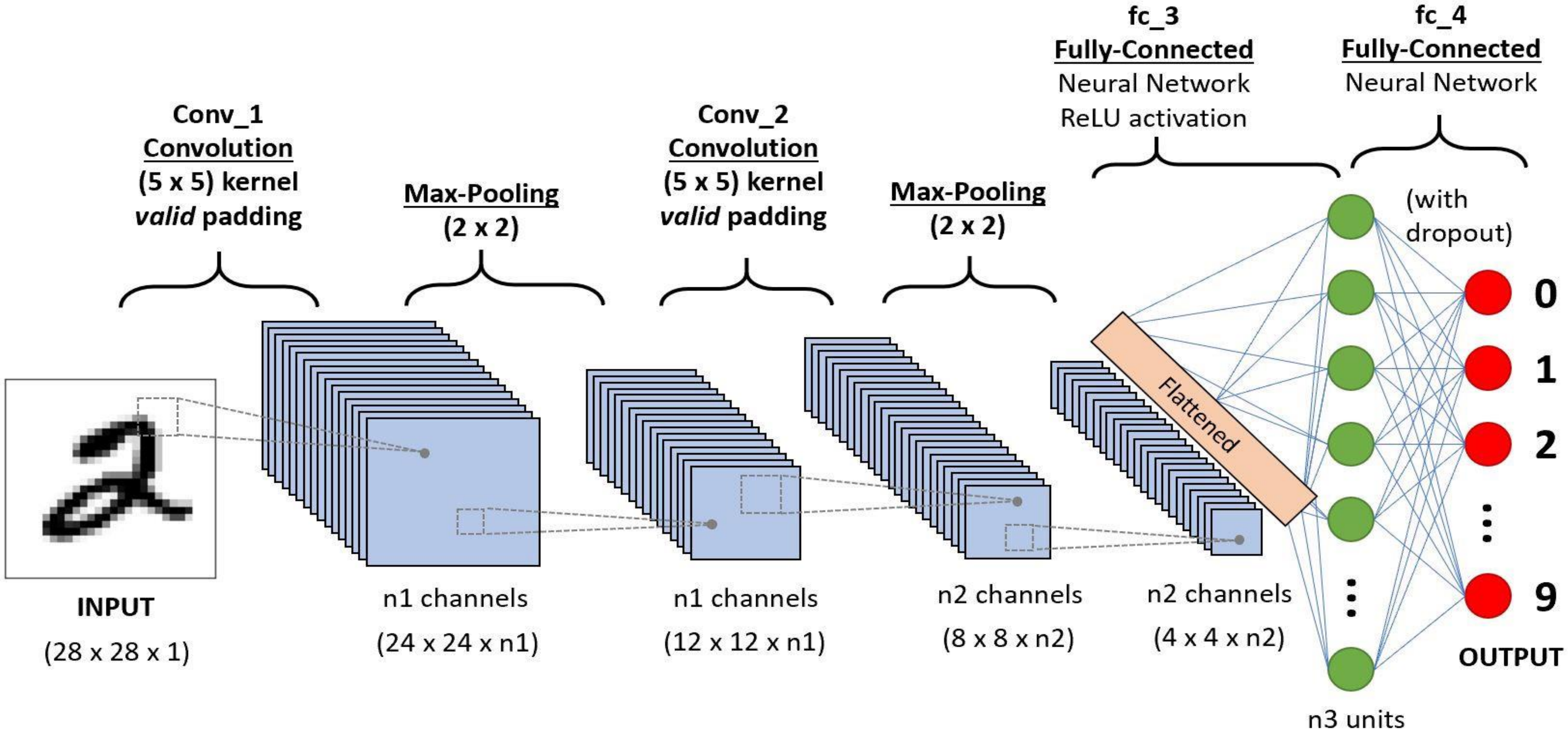
Model Selection

- Model with lowest inference time, space requirement with enough accuracy to classify our image.

Deep learning models for image classification

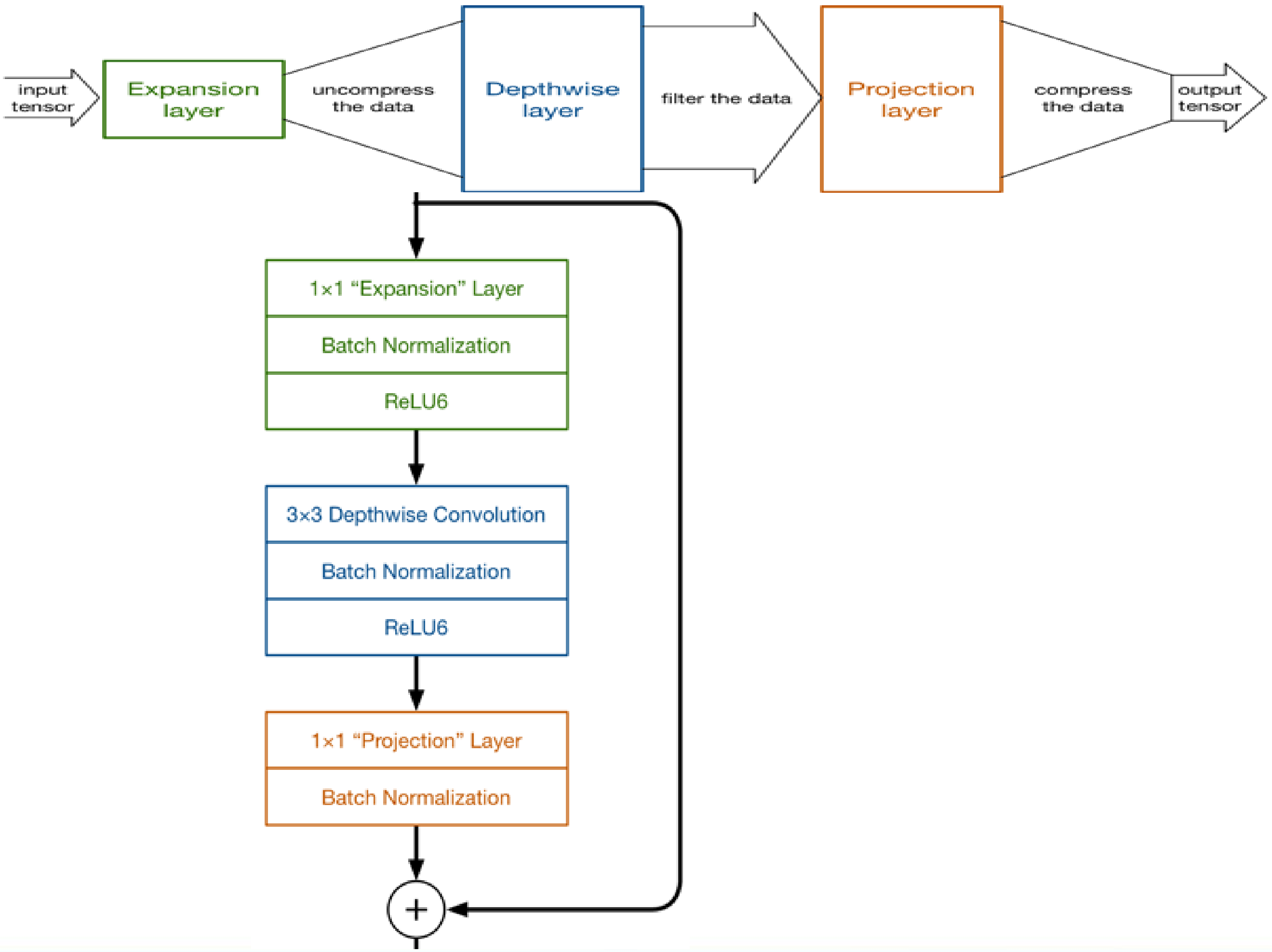
- Squeeze Net, Resnet-18, MobileNetv2 and Shuffle Net have low inference time and low space.
- Resnet-18 and MobileNetv2 have good accuracy for our application.
- Both models can be easily deployed on raspberry pi and Nvidia Jetson Nano.

Convolution neural network(CNN)

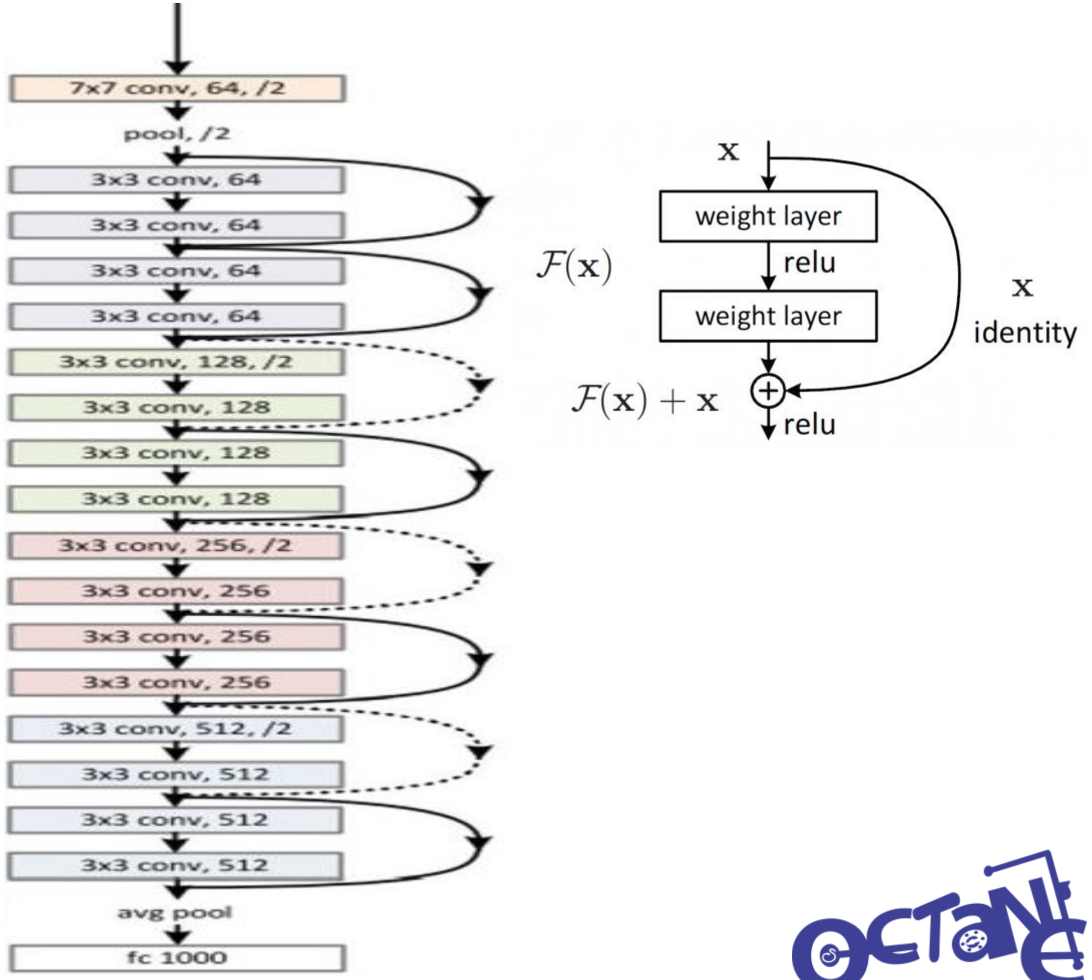


Deep learning models

MobileNetv2



Resnet-18



Classification

- Model training and validation accuracy

	Train accuracy	Validation accuracy	Inference time(sec)
Resnet-18	0.8648	0.9412	3.4
Quantized Resnet-18	0.9082	0.9504	2.5
MobileNetV2	0.9846	0.9917	1.3

MobileNetv2 has better accuracy than resnet-18 and requires less space than resnet-18

Cost

- MobileNetV2 allows to reduce RAM cost for CPU and provides good latency reducing production cost.

Image classification

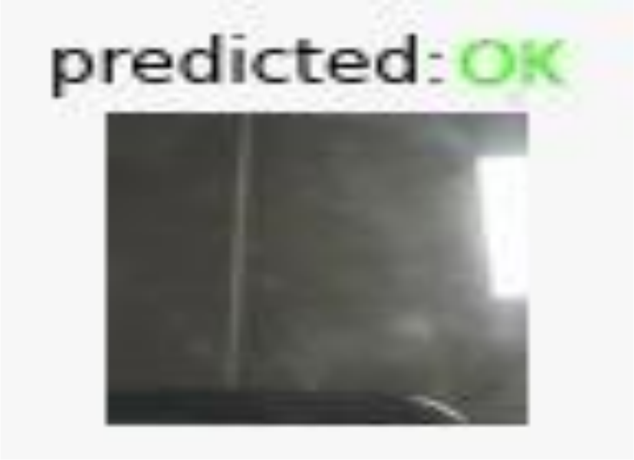
- Image dataset(Train and validation)
- Image transformation to tensors.
- Importing pre trained model in pytorch.
- Modifying model for our application.
- Result: Type of defect or OK image

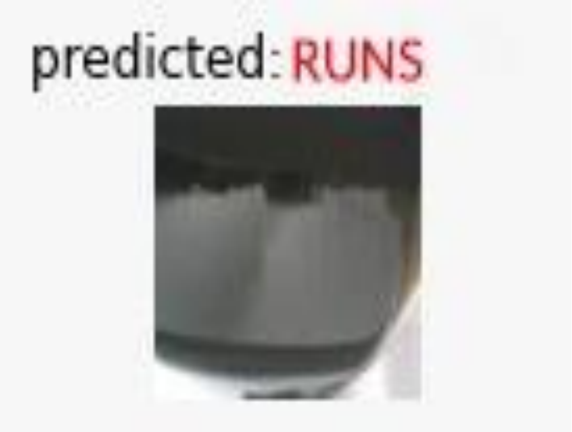
Video frames → Template matching → Defect image → Edge detection → Image morphing → Contour detection → Contour extraction → Masked image → Bitwise operation

Image dataset → Image transformation → Pre-trained model → Fine tuning → Test image transformation → Classification

Results

- Predictions obtained have 100% accuracy.





FRAME

TEMPLATE MATCHING

Defect detection

```
import os
import cv2
import numpy as np
from matplotlib import pyplot as plt
DATADIR="C:/Users/User pc/Desktop/Data science/Tank Images/Train/NOK_RUN"
path=os.path.join(DATADIR)
score=[]
for img in os.listdir(path):
    img1=cv2.imread(os.path.join(path,img))
    img2=cv2.imread("opencv_frame_1.png")
    h1,w1,c1=img1.shape[::1]
    c2,w2,h2=img2.shape[::-1]
    w=int((w1+w2)/2)
    h=int((h1+h2)/2)
    img1 = cv2.resize(img1, (w, h))
    img2 = cv2.resize(img2, (w, h))
    #gray_image=cv2.cvtColor(img1,cv2.COLOR_BGR2GRAY)
    #template=cv2.cvtColor(img2,cv2.COLOR_BGR2GRAY)
    method=cv2.TM_CCORR_NORMED
    res=cv2.matchTemplate(img1,img2,method)
    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
    score.append(max_val)
import statistics as s
average=s.mean(score)
if (average>=0.90):
    print("NOT OK")
else:
    print("OK")
```

```
NOT OK
```

Image Pre processing

```
img2=cv2.imread("opencv_frame_1.png")
gray_scale=cv2.cvtColor(img2,cv2.COLOR_BGR2GRAY)
edge=cv2.Canny(gray_scale,80,100)
plt.imshow(edge)
from skimage import morphology
kernel=np.ones((42,42),np.uint8)
opening=cv2.morphologyEx(edge,cv2.MORPH_CLOSE,kernel)
cleaned = morphology.remove_small_objects(opening, min_size=25000, connectivity=10000)
plt.imshow(cleaned)
from PIL import Image
im1,contours, hierarchy = cv2.findContours(cleaned,cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
c = max(contours, key=cv2.contourArea)
height,width=img2.shape[:2]
mask = np.ones([height,width,3], dtype="uint8")
mask=cv2.drawContours(mask,[c],-1,(255,255,255),-1)
print(height,width)
plt.imshow(mask)
image=cv2.bitwise_and(img2,mask)
plt.imshow(image)
```

Machine learning model

- MobileNetV2

```
model_ft = models.mobilenet_v2(pretrained=True)
model_ft.classifier=nn.Sequential(nn.Dropout(p=0.2,inplace=False),
                                  nn.Linear(in_features=1280,out_features=3,bias=True))

model_ft = model_ft.to(device)

criterion = nn.CrossEntropyLoss()

# Observe that all parameters are being optimized
optimizer_ft = optim.SGD(model_ft.parameters(), lr=0.001, momentum=0.9)
# Decay LR by a factor of 0.1 every 7 epochs
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_ft, step_size=7, gamma=0.1)
model_ft = train_model(model_ft, criterion, optimizer_ft, exp_lr_scheduler,
                       num_epochs=25)
```

```
Epoch 22/24
----------
Train Loss: 0.0508 Acc: 0.9903
val Loss: 0.0046 Acc: 1.0000

Epoch 23/24
----------
Train Loss: 0.0141 Acc: 0.9981
val Loss: 0.0085 Acc: 1.0000

Epoch 24/24
----------
Train Loss: 0.0150 Acc: 1.0000
val Loss: 0.0091 Acc: 1.0000

Training complete in 134m 57s
Best val Acc: 1.000000
```

Output

```python
def visualize_model(model, num_images=6):
    was_training = model.training
    model.eval()
    images_so_far = 0
    fig = plt.figure()

    with torch.no_grad():
        for i, (inputs, labels) in enumerate(dataloaders['val']):
            inputs = inputs.to(device)
            labels = labels.to(device)

            outputs = model(inputs)
            _, preds = torch.max(outputs, 1)

            for j in range(inputs.size()[0]):
                images_so_far += 1
                ax = plt.subplot(num_images//2, 2, images_so_far)
                ax.axis('off')
                ax.set_title('predicted: {}'.format(class_names[preds[j]]))
                imshow(inputs.cpu().data[j])

                if images_so_far == num_images:
                    model.train(mode=was_training)
                    return
        model.train(mode=was_training)
```

Cost reduction

- Solution functions effectively and accurately on defect images and provides low cost real time detection of defects.

Component	Component
Camera	CMOS sensor is chosen against CCD sensor for its low cost. Least possible camera resolution is chosen for application.
Interface	USB 3.0 is the least cost interface for higher bandwidth against camera link, CoaXPress.
Computing	Raspberry pi and Nvidia Jetson Nano have less cost than standard PC systems.
Artificial Intelligence & Machine learning	All codes and algorithms have been self developed. ML model of low size is chosen to reduce RAM size.
Software	Open source software is used for model development.

Thank You!