Final Graduation Project and Master of Research

Implementation and Development of a Multi-Sensor Mobile Vision System (2D Color, 3D, IR)

Host organization: Vilmorin Mikado

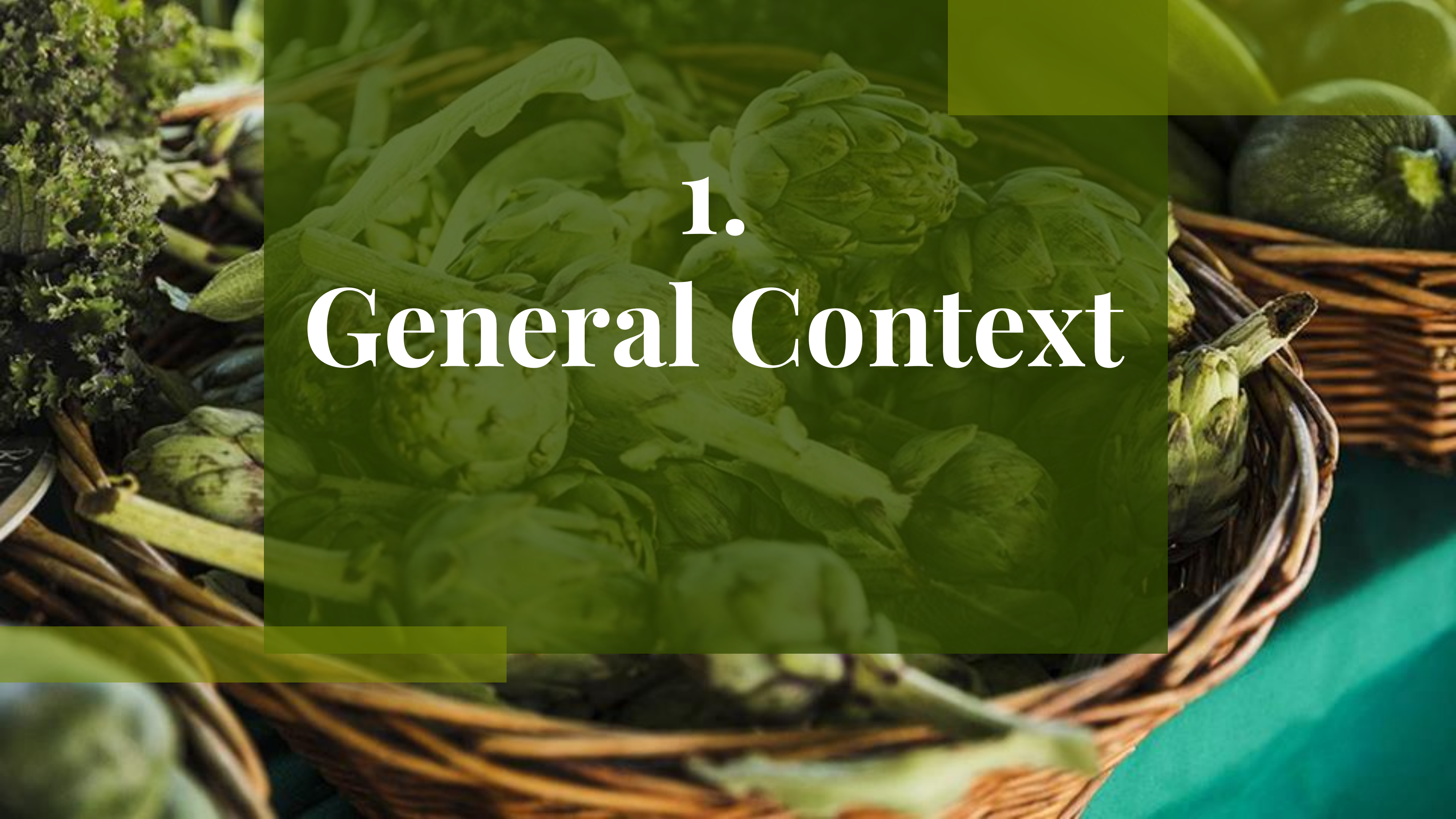
Author: Chayma MOUSSA

Advisor: Dr. Feryel BEJI

Co-adviser: Dr. Ali BOUDJEDRA

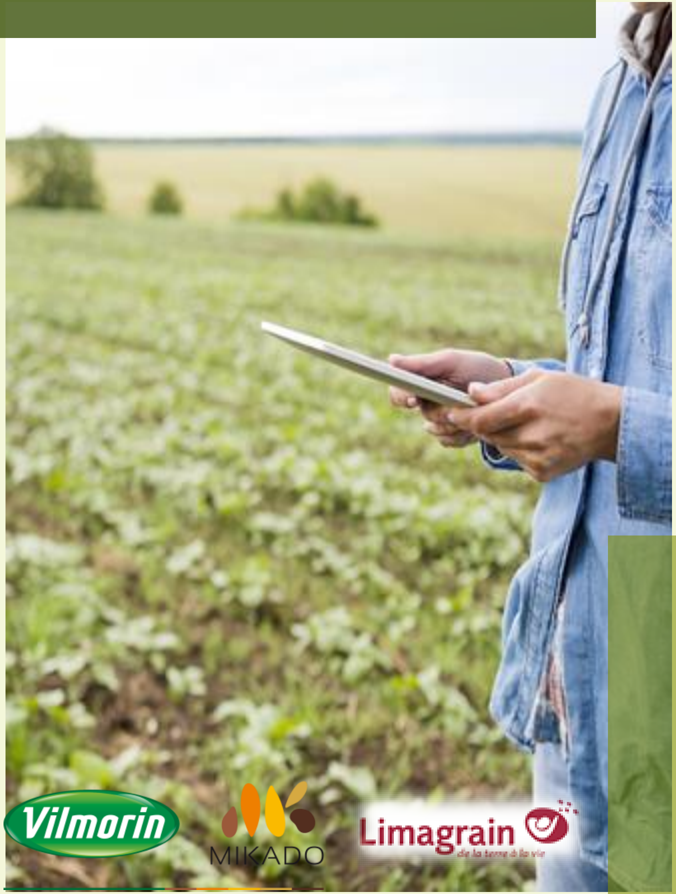
Academic year: 2020/2021

Content



1. General Context

Host Organization



- This project was created by the Limagrain Group and was hosted within the Artificial Vision and Automation R&D lab

- Research is an important aspect for Vilmorin Mikado

- The Artificial Vison and Automation R&D lab includes artificial vision and automation activities and image processing activities

4

Problem Statement (1/3)

Lettuce is one of the Limagrain product

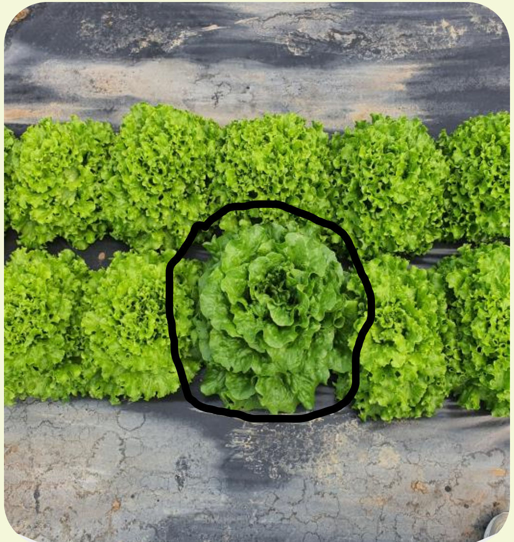
Problem Statement (2/3)

Among the planted lettuces, some are **off-types**

Smaller volume and
lighter color

Different color

Completely different



Problem Statement (3/3)

- Limagrain experts cannot visit frequently lettuce fields which are located in **several countries**

- An off-type is detected according to its neighbors

- Producers do not remove these off-types as they will decrease their production

Objectives

Goal 1

Produce good image quality

Goal 2

Classify lettuces to detect off-types

- **Develop a model to generate a good quality image and a model to be used to detect off-types**

Project cycle

1

State of The Art

2

Data Acquisition

3

Data Preprocessing and preparation

4

Merging the three image sources (2D color, 3D, IR)

5

Classification model to detect off-types

2.
Literature Review

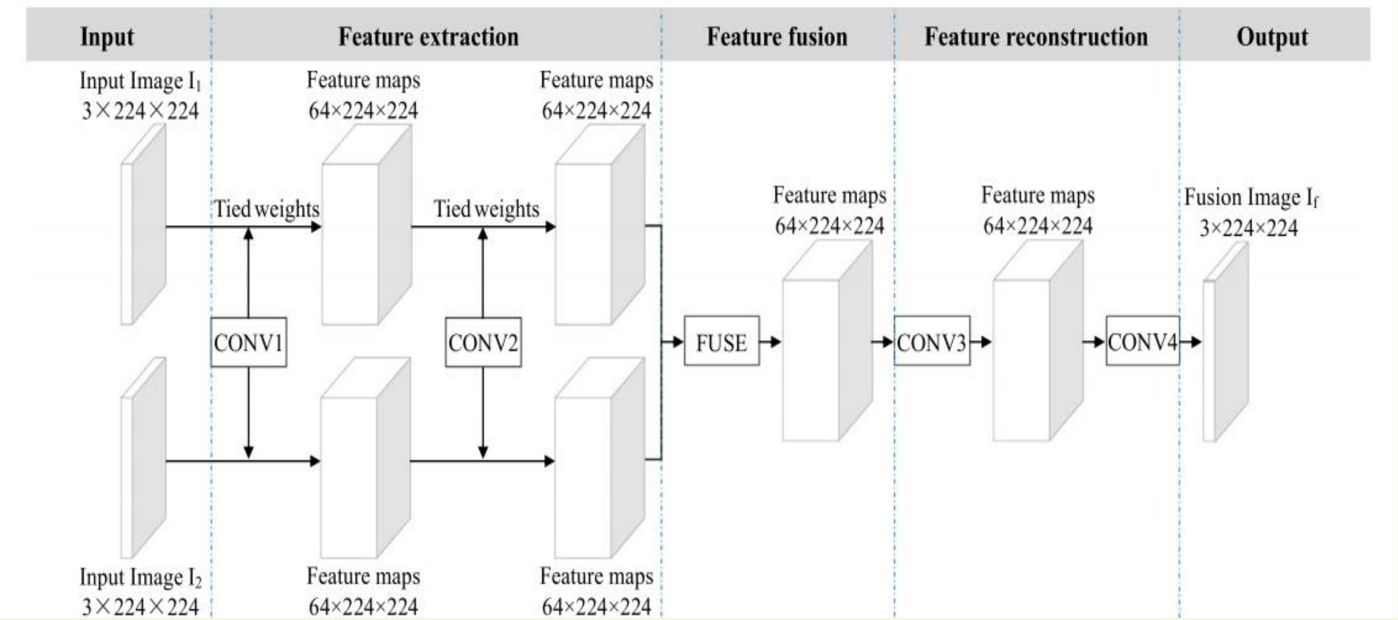
Fusing Infrared and Color Images (1/3)

- Color camera performs well in **good illumination conditions** and provide rich colors and detail information
- Poor light conditions comprise the effectiveness of color camera
- Thermal camera performs well in **limited visibility conditions**
- Thermal camera is ineffective under direct bright sunlight
- Thermal camera is still more **expensive** than their color countpart having the same resolution

➢ Integrating captured information from different camera such as color and thermal offers **rich information** to improve the quality of the image taken in varying lighting conditions

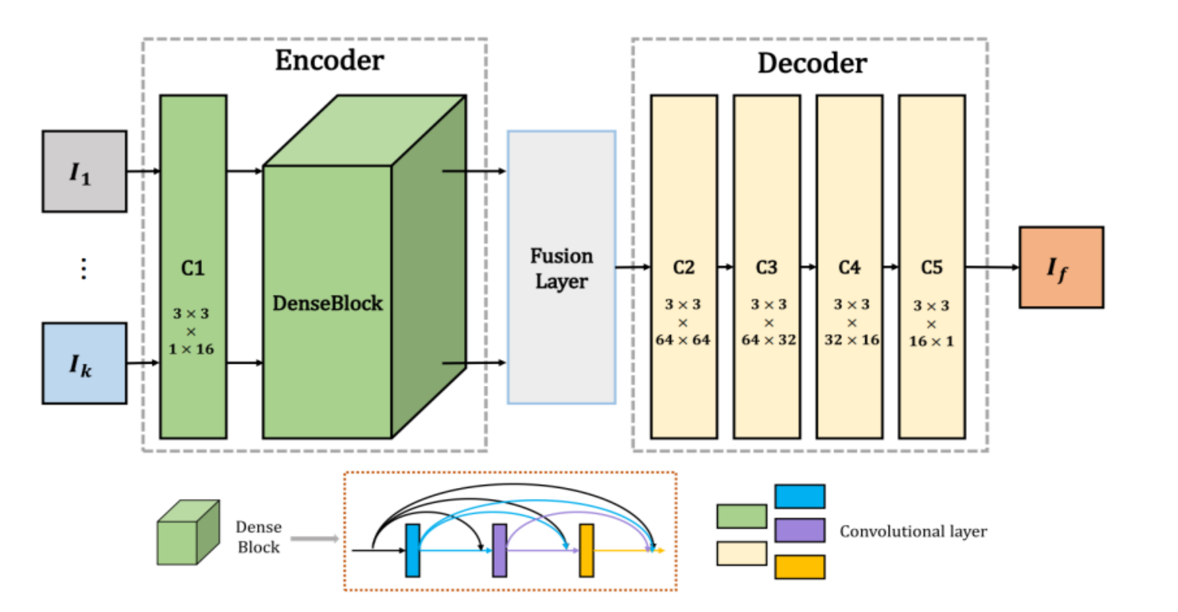
Fusing Infrared and Color Images (2/3)

Yu Zhanga and all 2020. **IFCNN: A general image fusion framework based on convolutional neural network**



Fusing Infrared and Color Images (3/3)

Hui Li & Xiao-Jun Wu 2019. **DenseFuse: A Fusion Approach to Infrared and Visible Images**



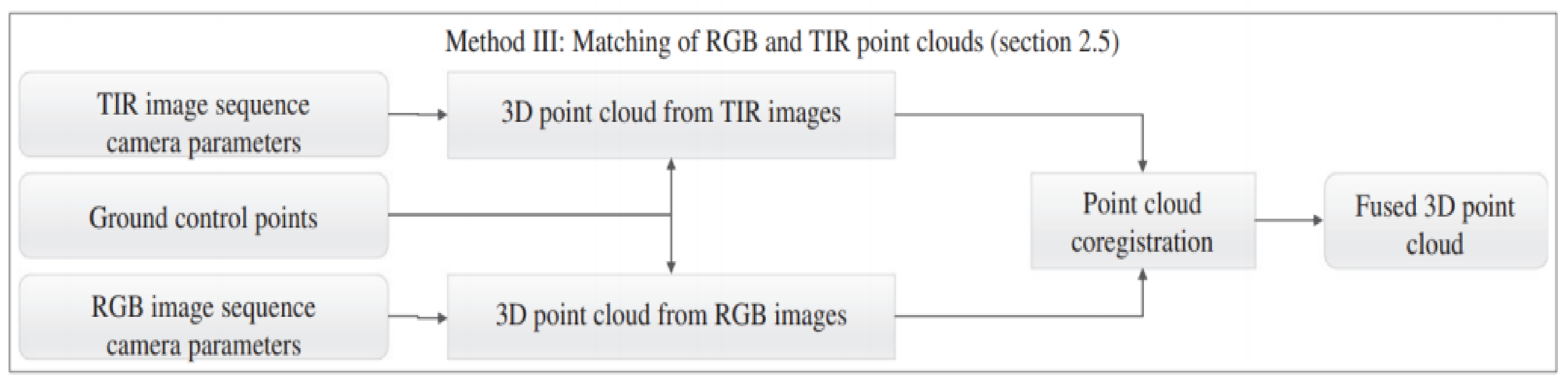
Fusing 2D Color and 3D Images (1/2)

- 2D images have progressed to be reasonably accurate under controlled conditions
- 2D have proved that their performance decreases significantly when pose or brightening variations are present in the pictures
- The characteristics of 3D point clouds and 2D digital images are thought to be complementary

➢ To improve the image quality under these conditions, merging 2D image and 3D image should be studied.

Fusing 2D Color and 3D Images

L. Hoegner & and al 2018. **Mobile thermal mapping for matching of infrared images with 3D building models and 3D point clouds**

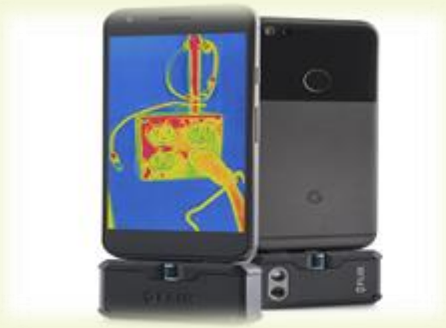


3. Project Life Cycle

Data Acquisition (1/3)

Galaxy Note 10+
Camera resolution: 4032x3024 pixels
pixel size: 14um
3 rear cameras:
- 12 Mpx
- 12 Mpx,telephoto
- 16 Mpx (wide-angle)

+ TOF(Time Of Flight) sensor



FLIR ONE PRO ANDROID
1440*1080 pixel

Data Acquisition (2/3)

Acquired **Lettuce typologies :**

Multifeuille

Batavia

Chene

Iceberg

Romaine

Beurre

Data Acquisition (3/3)

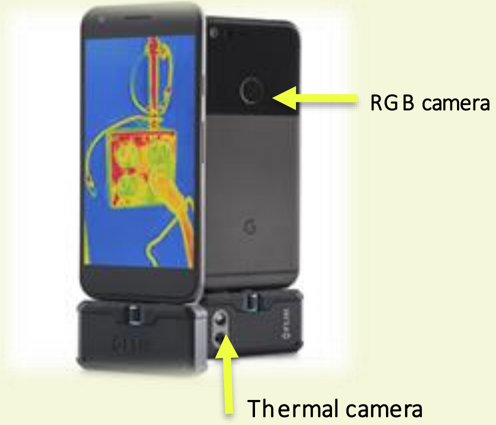
- We used **Scanner 3D** in order to take the 3D images but due to the overlapping lettuces planted in greenhouses could not be detected

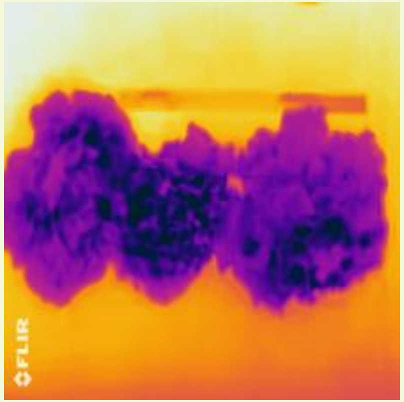
Typology	Images with RGB images taken by Galaxy Note 10+		Images with RGB images taken by Flir One Pro		Other RGB images
	RGB	IR	RGB	IR	
Batavia	59	59	57	57	32
Multifeuille	83	83	10	10	4
Beurre	0	0	63	63	0
Chene	0	0	62	62	0
Romain	12	12	5	5	1
Iceberg	8	8	11	11	8

Global dataset

Data Preprocessing (1/2)

- The **offset** between the two cameras induces a **shift** between the images

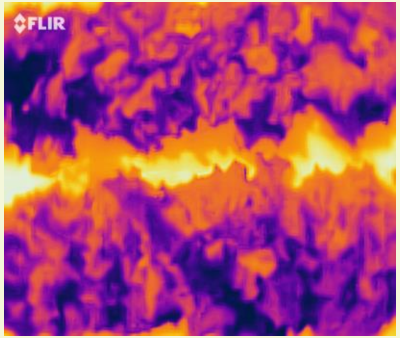




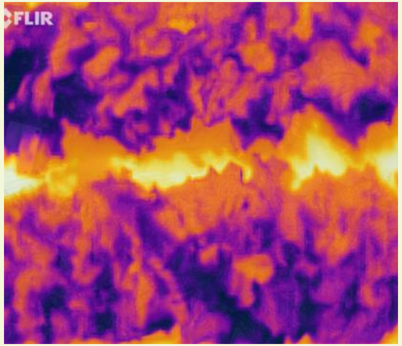
Data Preprocessing (2/2)

- Correcting the shift between images using online tool named **overlay.imageonline**

Before

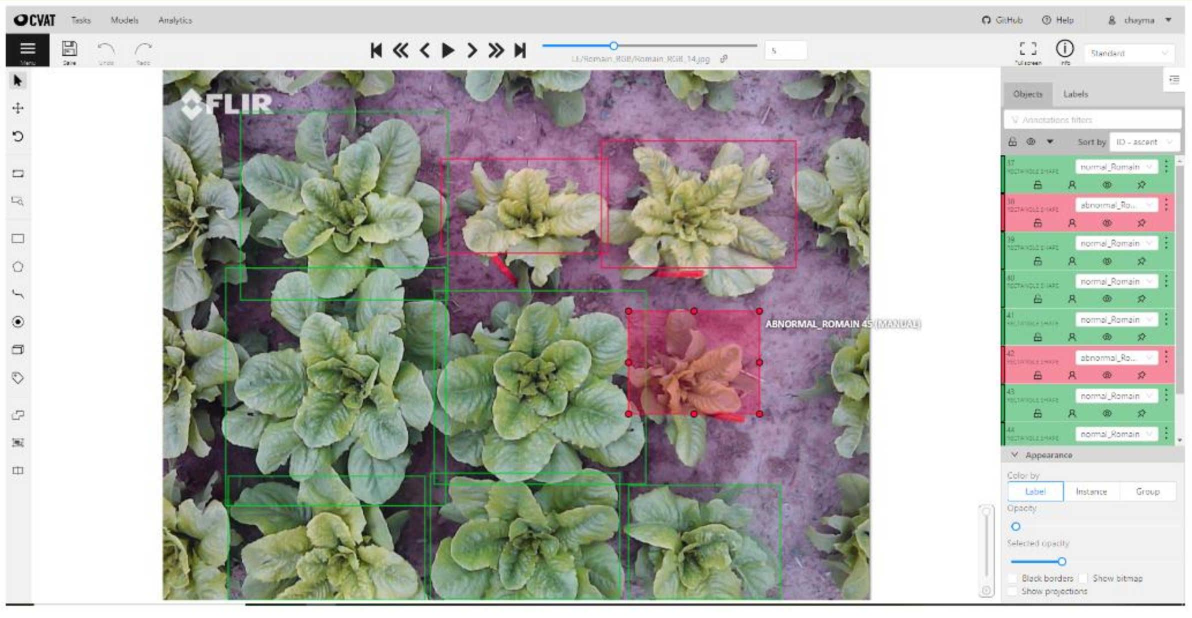


After



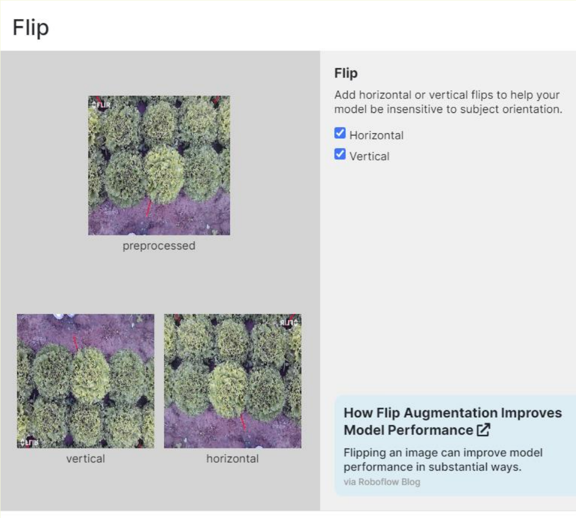
Data Preparation (1/2)

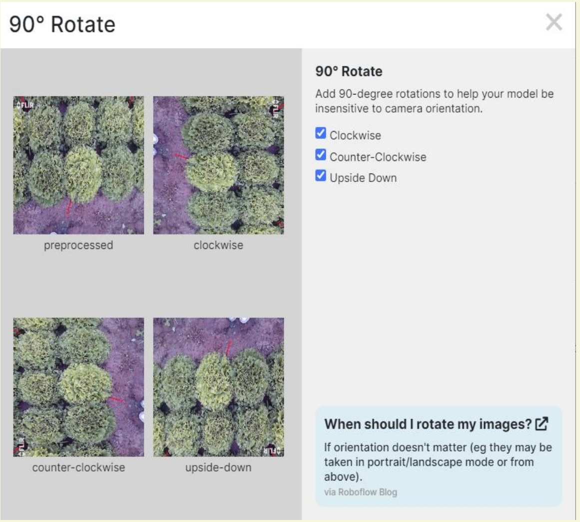
Data Annotation using CVAT

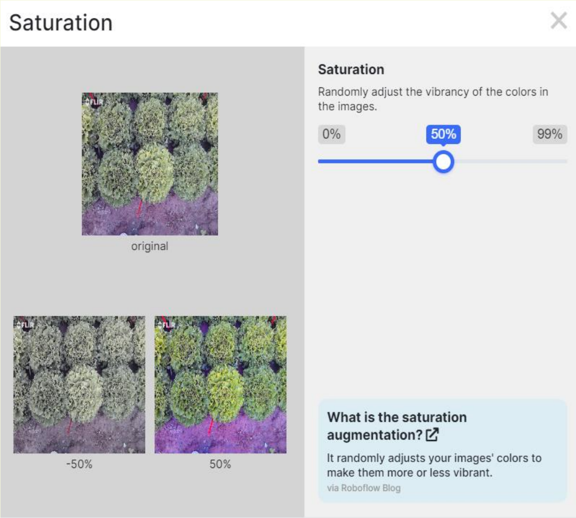


Data Preparation (2/2)

Data Augmentation







Data Description (1/2)

Description of Batavia data set

	Without data augmentation	With data augmentation: Rotation & Flip
Total images	74	165
Images annotated	72	163
Objects	275	780
Normals	233	660
Abnormals	42	120

Data Description (2/2)

Description of RGB Batavia data set

	Without data augmentation	With data augmentation	
		Rotation & Flip	Rotation & Flip & Saturation
Total images	148	257	303
Images annotated	108	257	303
Objects	623	1726	2594
Normals	510	1488	2219
Abnormals	113	238	375

Work Environment (1/2)

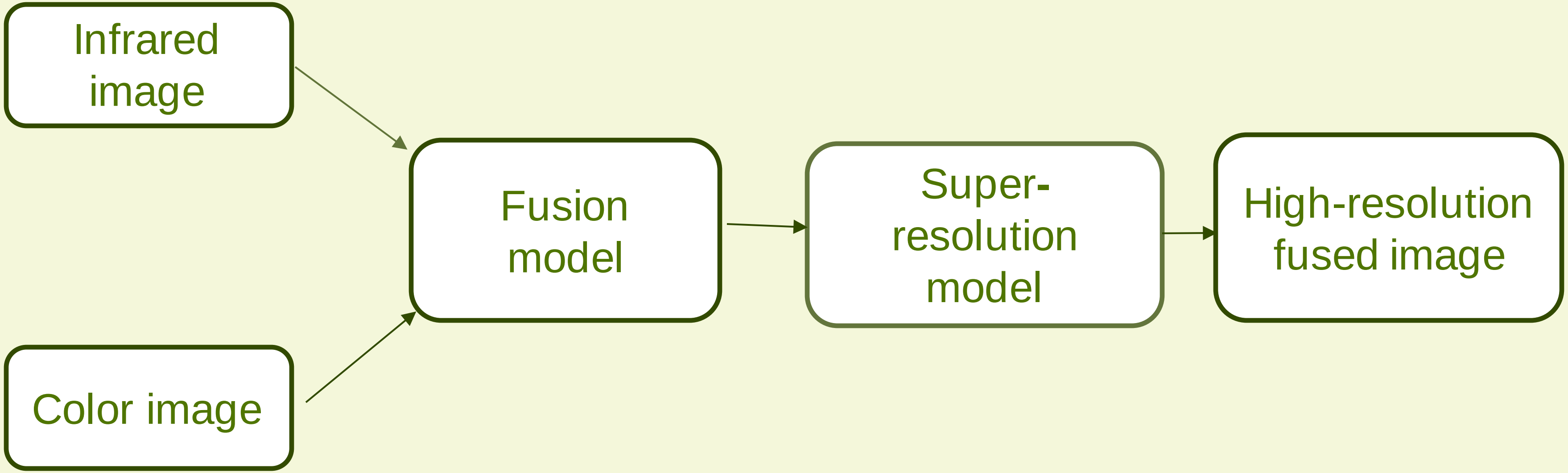
Used software

Work Environment (2/2)

Used technologies: libraries

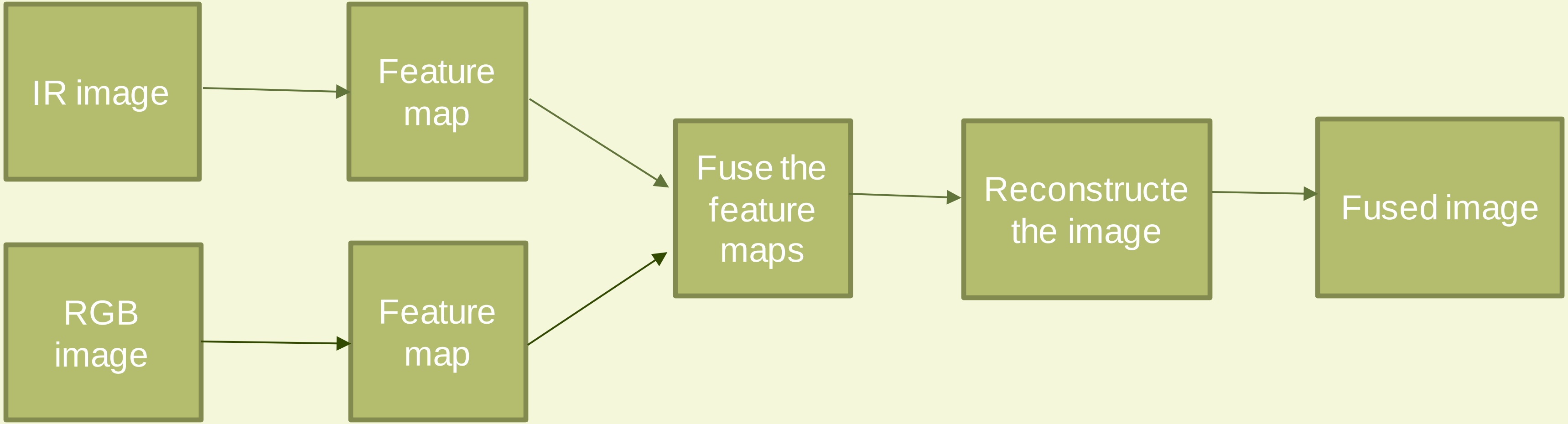
Modeling (1/11)

Image fusion methodology



Modeling (2/11)

Fusion model



Modeling (3/11)

Fusion model

IFCNN1.ipynb

File Edit View Insert Runtime Tools Help Last edited on July 23

Comment Share

+ Code + Text Connect Editing

```
[ ] def myIFCNN(fuse_scheme=0):
        # pretrained resnet101
        resnet = models.resnet101(pretrained=True)
        # our model
        model = IFCNN(resnet, fuse_scheme=fuse_scheme)
        return model
```

```
[ ] model = myIFCNN(0)
    model_name = "IFCNN-MAX"
    model.load_state_dict(torch.load("C:/Users\Chayma.MOUSSA\IFCNN\Code\snapshots\IFCNN-MAX.pth", map_location=torch.device('c
    model.eval()
```

```
IFCNN(
  (conv2): ConvBlock(
    (conv): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
  )
  (conv3): ConvBlock(
    (conv): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
  )
  (conv4): Conv2d(64, 3, kernel_size=(1, 1), stride=(1, 1))
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=1, bias=False)
)
```

Modeling (4/11)

Fusion model

IFCNN1.ipynb

File Edit View Insert Runtime Tools Help Last edited on July 23

```python
def load_pair_images(i):
    try:
        RGB_name_img="MF_r_RGB_"+ str(i)+".jpg"
        IR_name_img="MF_r_IR_"+ str(i)+".jpg"
        IR = Image.open(os.path.join("P:/Chayma MOUSSA/Dataset/Dataset_lettuce/Multifeuille/MF_rouge/No_shift/IR/", IR_nam
        RGB = Image.open(os.path.join("P:/Chayma MOUSSA/Dataset/Dataset_lettuce/Multifeuille/MF_rouge/No_shift/RGB/", RGB_
        img1 = np.asarray(IR)
        img1 = torchvision.transforms.ToTensor()(img1)
        img1 = torchvision.transforms.Normalize(mean=mean, std=std)(img1)
        img1 = torch.unsqueeze(img1, 0)
        img2 = np.asarray(RGB)
        img2 = torchvision.transforms.ToTensor()(img2)
        img2 = torchvision.transforms.Normalize(mean=mean, std=std)(img2)
        img2 = torch.unsqueeze(img2, 0)
        return img1 ,img2
    except:
        return "no other images"


i=0
while(1):
    i+=1
    mean=[0.485, 0.456, 0.406] # normalization parameters
    std=[0.229, 0.224, 0.225]
    # load source images
```

Modeling (5/11)

Fusion model

IFCNN1.ipynb

File Edit View Insert Runtime Tools Help Last edited on July 23

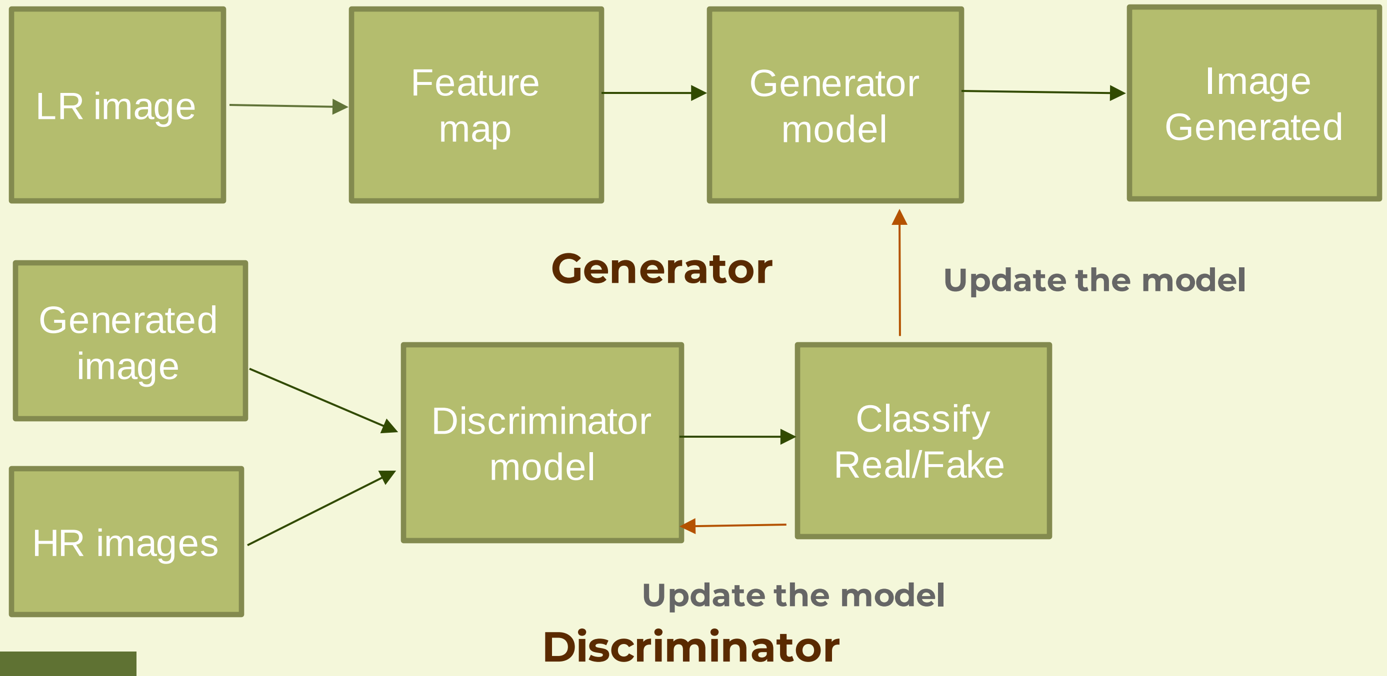
Comment Share

+ Code + Text Connect Editing

```python
            return img1 ,img2
        except:
            return "no other images"


i=0
while(1):
    i+=1
    mean=[0.485, 0.456, 0.406] # normalization parameters
    std=[0.229, 0.224, 0.225]
    # load source images
    img1,img2 = load_pair_images(i)


        # perform image fusion
    with torch.no_grad():
        res = model(img1, img2)
        res = denorm(mean, std, res[0]).clamp(0, 1) * 255
        res_img = res.cpu().data.numpy().astype('uint8')
        img = res_img.transpose([1,2,0])
        img = Image.fromarray(img)
        img.show()
        filename = "MF_r_"+str(i)
        img.save(r"P:\Chayma MOUSSA\Dataset\resultats fusion RGB IR\IFCNN fusion sheme=MAX\MF rouge\{:>12s}.jpg".format(fi
```

Modeling (6/11)

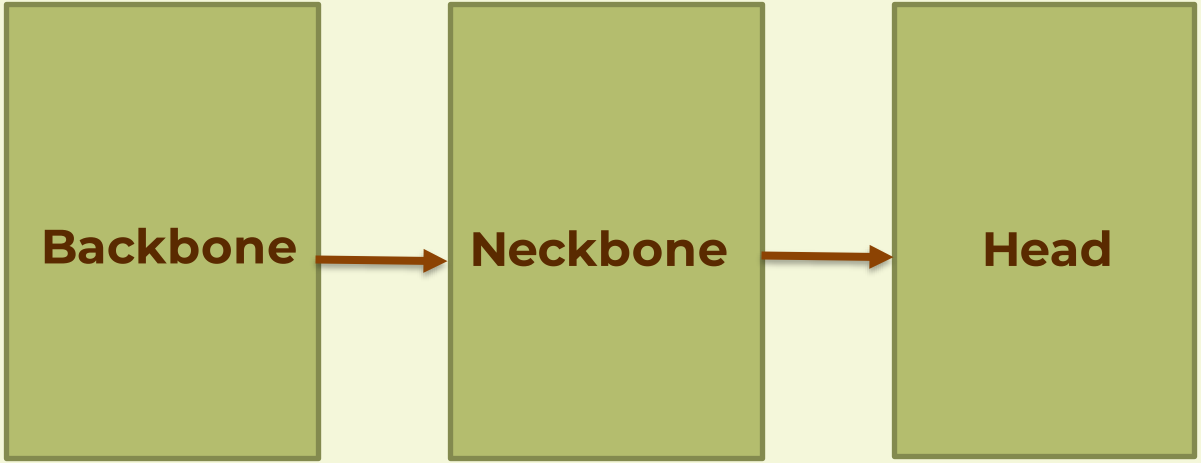
Super-Resolution Generative Adversarial Network(SR-GAN)



Modeling (7/11)

YOLOv5

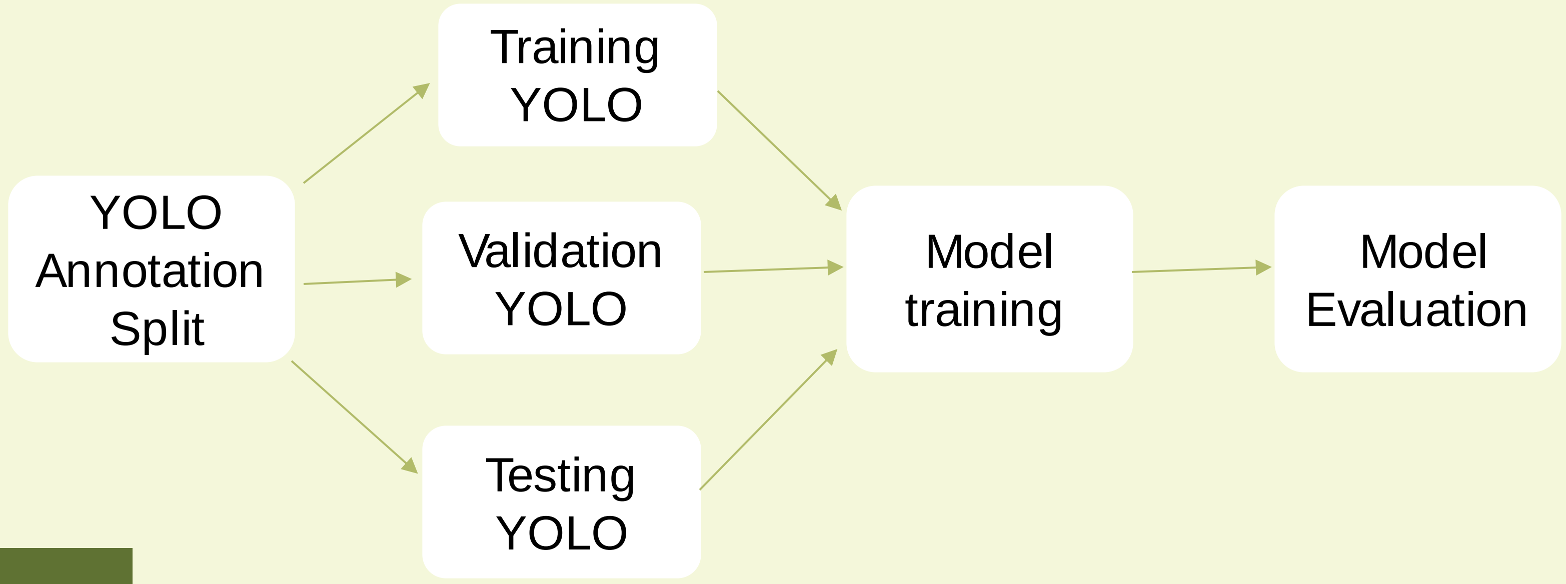
The network architecture of Yolov5 consists of three parts:



Modeling (8/11)

YOLOv5

YOLOv5 is a very suitable model for object detection and classification



Modeling (9/11)

YOLOv5

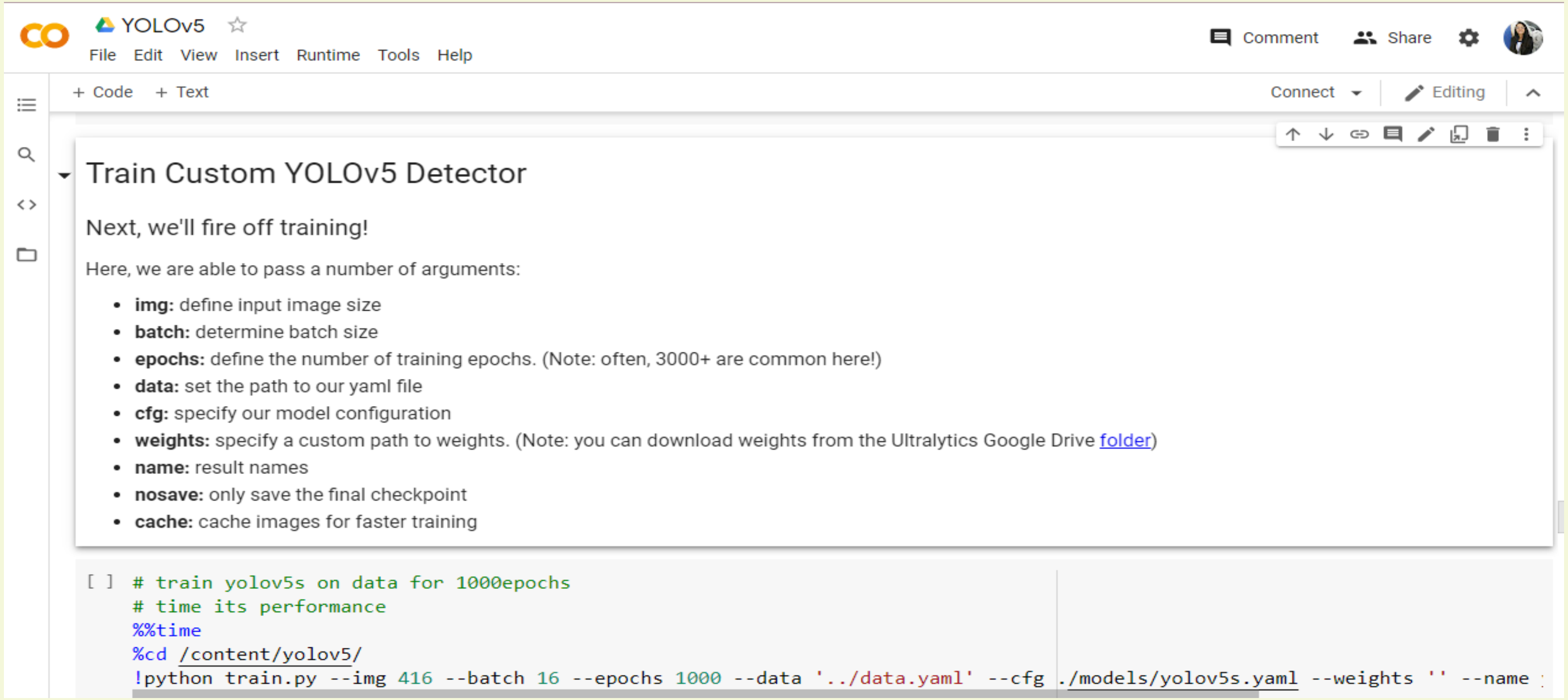
YOLOv5
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Connect Editing

```
#this is the model configuration we will use for our tutorial
%cat /content/yolov5/models/yolov5s.yaml
```

```
# parameters
nc: 2  # number of classes
depth_multiple: 0.33  # model depth multiple
width_multiple: 0.50  # layer channel multiple

# anchors
anchors:
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32

# YOLOv5 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Focus, [64, 3]],  # 0-P1/2
   [-1, 1, Conv, [128, 3, 2]],  # 1-P2/4
   [-1, 3, BottleneckCSP, [128]],
   [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
   [-1, 9, BottleneckCSP, [256]],
   [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16
   [-1, 9, BottleneckCSP, [512]],
   [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
   [-1, 1, SPP, [1024, [5, 9, 13]]],
   [-1, 3, BottleneckCSP, [1024, False]],  # 9
  ]

# YOLOv5 head
```

Modeling (10/11)

YOLOv5



Modeling (11/11)

YOLOv5

YOLOv5
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Connect Editing

Double-click (or enter) to edit

```
# when we ran this, we saw .007 second inference time. That is 140 FPS on a TESLA P100!
# use the best weights!
%cd /content/yolov5/
!python detect.py --weights ./runs/train/yolov5s_results6/weights/best.pt --img 416 --conf 0.4 --source ../test/images
```

```
/content/yolov5
Namespace(agnostic_nms=False, augment=False, classes=None, conf_thres=0.4, device='', exist_ok=False, img_size=416, iou_thres=0.45, name='exp', proje
YOLOv5 v4.0-126-g886f1c0 torch 1.9.0+cu102 CUDA:0 (Tesla K80, 11441.1875MB)

Fusing layers...
/usr/local/lib/python3.7/dist-packages/torch/nn/functional.py:718: UserWarning: Named tensors and all their associated APIs are an experimental featu
  return torch.max_pool2d(input, kernel_size, stride, padding, dilation, ceil_mode)
Model Summary: 232 layers, 7249215 parameters, 0 gradients, 16.8 GFLOPS
image 1/5 /content/yolov5/../test/images/BBL_RGB_10_jpg.rf.410f4d9881cd6e45efe6d777fc59f89c.jpg: 416x416 5 NORs, Done. (0.032s)
image 2/5 /content/yolov5/../test/images/BBL_RGB_12_jpg.rf.67dc3277e0e79b8804c5bf751911458e.jpg: 416x416 8 NORs, Done. (0.025s)
image 3/5 /content/yolov5/../test/images/BBL_RGB_4_jpg.rf.1ad4f5c3271ff2eede768978717c10a0.jpg: 416x416 5 NORs, Done. (0.025s)
image 4/5 /content/yolov5/../test/images/BBL_RGB_7_jpg.rf.bd7031a12a8d3956267274126a7528e8.jpg: 416x416 6 NORs, Done. (0.026s)
image 5/5 /content/yolov5/../test/images/Bv6_RGB_4_jpg.rf.39e90b3dd01665012dd3f29387d8d814.jpg: 416x416 8 NORs, Done. (0.025s)
Results saved to runs/detect/exp3
Done. (0.243s)
```

```
#display inference on ALL test images
#this looks much better with longer training above

import glob
```

Evaluation Metrics (1/2)

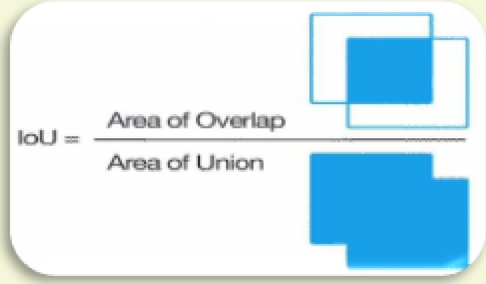
Classification

- **Accuracy:** $$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$
- **Precision:** $$Precision = \frac{TP}{TP + FP} = \frac{TP}{Alldetections}$$
- **Recall:** $$Recall = \frac{TP}{TP + FN} = \frac{TP}{Allpositiveinstances}$$
- **F1:** $$F_1 = \frac{2 \times Precision + Recall}{Precision + Recall}$$

Evaluation Metrics (2/2)

Object detection

- **Intersection over Union:** To decide whether a prediction is correct with respect to an object, **IoU**



- **Mean Average Precision (mAP)** is the mean of all classes' Average precision
- it is commonly designed as mAP@[.5:.95] signifies average mAP over various IoU limits

Results (1/7)

RGB dataset D2

Typology	Data augmentation		Recall	Accuracy	Precision	F_1	mAP
	Without	Flip & Rotation					
Batavia	x		0.618	0.38	0.486	0.544	0.329
Batavia		x	0.5538	0.44	0.6240	0.5868	0.3873
Multifeuille	x		0.720	0.46	0.367	0.486	0.4189
Multifeuille		x	0.672	0.765	0.723	0.697	0.544
Beurre	x		0.8564	0.77	0.8057	0.8328	0.6758
Beurre		x	0.8774	0.83	0.8997	0.8884	0.6945
Chene	x		0.7261	0.555	0.7928	0.759	0.5797
Chene		x	0.7464	0.69	0.8246	0.8126	0.5937
All typologies	x		0.790	0.835	0.801	0.795	0.560
All typologies		x	0.78	0.835	0.81	0.795	0.59

Results(2/7)

Fused dataset with the method SUM

Typology	Data augmentation		Recall	Accuracy	Precision	F_1	mAP
	without	Flip & Rotation					
Batavia	x		0.751	0.385	0.457	0.568	0.382
Batavia		x	0.6060	0.495	0.5935	0.5997	0.3490
Multifeuille	x		0.6726	0.42	0.4022	0.5034	0.4326
Multifeuille		x	0.733	0.562	0.824	0.776	0.5715
Beurre	x		0.810	0.795	0.798	0.804	0.570
Beurre		x	0.9189	0.735	0.7401	0.8199	0.6324
Chene	x		0.860	0.54	0.589	0.589	0.3745
Chene		x	0.782	0.59	0.492	0.604	0.428
All typologies	x		0.649	0.61	0.657	0.653	0.472
All typologies		x	0.710	0.705	0.708	0.709	0.508

Results (3/7)

Fused dataset with the method MAX

Typology	Data augmentation		Recall	Accuracy	Precision	F_1	mAP
	without	Flip & Rotation					
Batavia	x		0.751	0.385	0.457	0.568	0.382
Batavia		x	0.6060	0.495	0.5935	0.5997	0.3490
Multifeuille	x		0.6726	0.42	0.4022	0.5034	0.4326
Multifeuille		x	0.733	0.562	0.824	0.776	0.5715
Beurre	x		0.810	0.795	0.798	0.804	0.570
Beurre		x	0.9189	0.735	0.7401	0.8199	0.6324
Chene	x		0.860	0.54	0.589	0.589	0.3745
Chene		x	0.782	0.59	0.492	0.604	0.428
All typologies	x		0.649	0.61	0.657	0.653	0.472
All typologies		x	0.710	0.705	0.708	0.709	0.508

Results (4/7)

Fused Dataset with the method Mean

Typology	Data augmentation		Recall	Accuracy	Precision	F_1	mAP
	without	Flip & Rotation					
Batavia	x		0.672	0.4	0.5325	0.5942	0.3927
Batavia		x	0.779	0.625	0.553	0.647	0.484
Multifeuille	x		0.753	0.43	0.408	0.529	0.4398
Multifeuille		x	0.6897	0.74	0.7287	0.7087	0.578
Beurre	x		0.7966	0.79	0.8613	0.8277	0.6044
Beurre		x	0.855	0.815	0.872	0.863	0.644
Chene	x		0.6432	0.555	0.6369	0.6400	0.4215
Chene		x	0.7270	0.725	0.6859	0.7059	0.4369
All typologies	x		0.6503	0.665	0.6217	0.6357	0.4431
All typologies		x	0.6704	0.72	0.6681	0.6693	0.4591

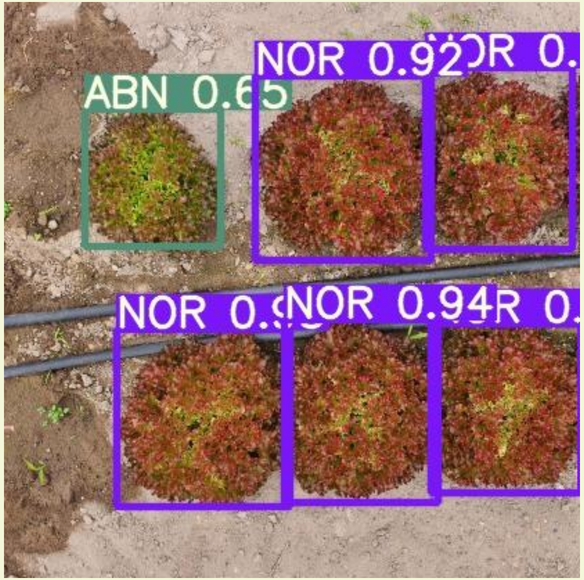
Results (5/7)

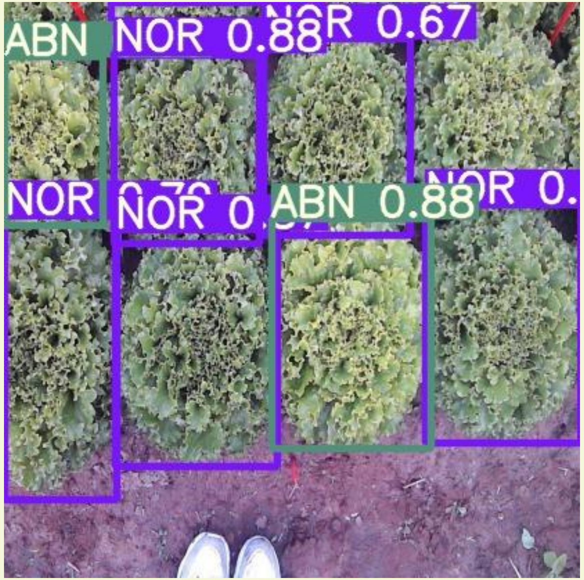
RGB dataset D1

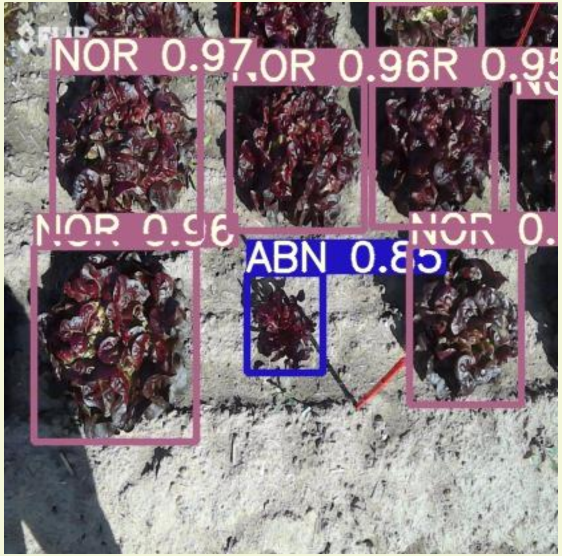
Typology	Data augmentation		Recall	Accuracy	Precision	F_1	mAP
	Flip&Rotation	Flip&Rotation&Saturation					
Batavia	NO	NO	0.7231	0.69	0.8334	0.7743	0.6210
Batavia	YES	NO	0.6965	0.7	0.8119	0.7498	0.6297
Batavia	NO	YES	0.7106	0.685	0.8201	0.7614	0.6321
Multifeu.	NO	NO	0.7164	0.445	0.5937	0.6493	0.4556
Multifeu.	YES	NO	0.7735	0.745	0.7909	0.7821	0.5604
Multifeu.	NO	YES	0.7584	0.71	0.8493	0.8013	0.5896
Beurre	NO	NO	0.8564	0.77	0.8057	0.8328	0.6758
Beurre	YES	NO	0.8774	0.83	0.8997	0.8884	0.6945
Beurre	NO	YES	0.9215	0.88	0.9037	0.9125	0.7482
Chene	NO	NO	0.7261	0.555	0.7928	0.7590	0.5797
Chene	YES	NO	0.7464	0.69	0.8246	0.8126	0.5937
Chene	NO	YES	0.7604	0.86	0.9067	0.8271	0.6264
All	NO	NO	0.8231	0.8	0.8462	0.8345	0.7324
All	YES	NO	0.7986	0.785	0.8645	0.8302	0.7452
All	NO	YES	0.8551	0.86	0.8864	0.8705	0.7695

Results (6/7)

Tested images on RGB dataset

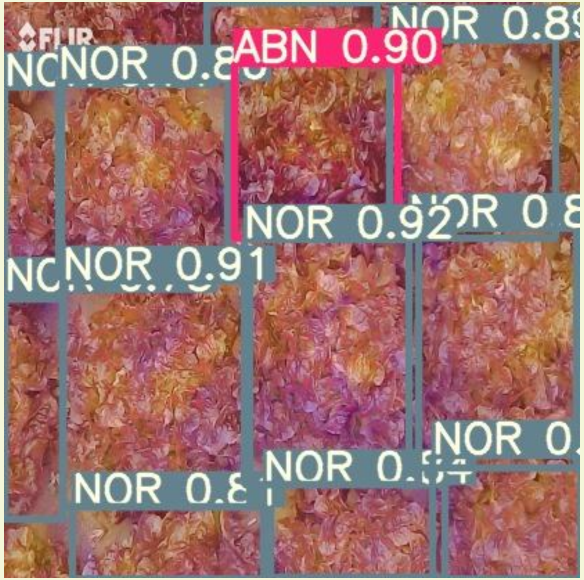


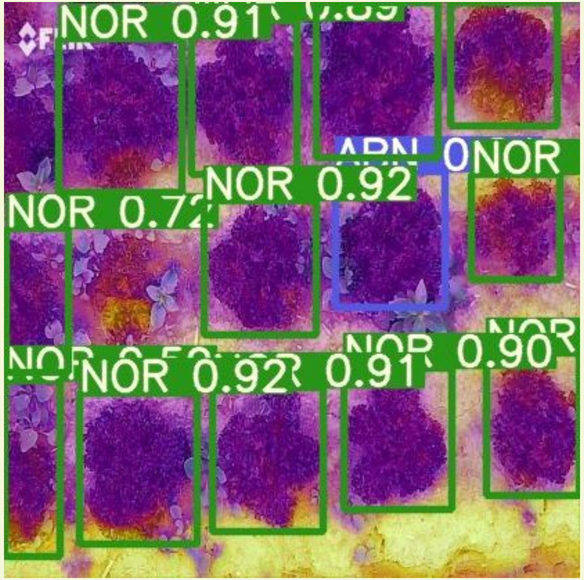


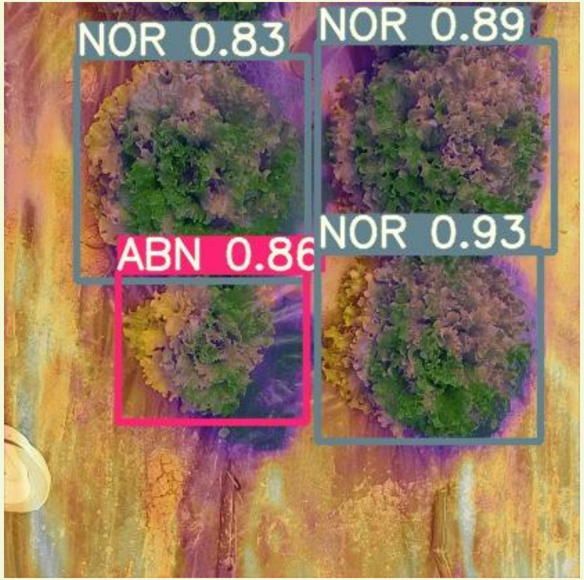


Results (7/7)

Tested images on fused dataset







Conclusion And Future Work

Conclusion and Future Work (1/2)

- We achieved a mAP of 0.7695% with all typology datasets
- We realised a precision of 0.9037% and an F1 score of 0.9125% with Beurre typology
- We found that RGB datasets have generally better results than fused datasets in terms of object detection and classification
- The results on the data augmentation data sets showed significant improvement with most data sets

Conclusion and Future Work (2/2)

- The model object detection performance can be improved by studying how to segment the overlapping lettuces
- Performing more experiments with different model backbones, training on additional data and optimizing the hyperparameters can increase the performance of the model to meet better the industry requirements
- We can deploy the model and develop an application that can be ran through microservices architecture

Thank you for your attention

Code

IFCNN1.ipynb

File Edit View Insert Runtime Tools Help Last edited on July 23

Comment Share

+ Code + Text Connect Editing

```
# My Convolution Block
class ConvBlock(nn.Module):
    def __init__(self, inplane, outplane):
        super(ConvBlock, self).__init__()
        self.padding = (1, 1, 1, 1)
        self.conv = nn.Conv2d(inplane, outplane, kernel_size=3, padding=0, stride=1, bias=False)
        self.bn = nn.BatchNorm2d(outplane)
        self.relu = nn.ReLU(inplace=True)

    def forward(self, x):
        out = F.pad(x, self.padding)
        out = self.conv(out)
        out = self.bn(out)#batch norm
        out = self.relu(out)
        return out
```

```
class IFCNN(nn.Module):
    def __init__(self, resnet, fuse_scheme=0):
        super(IFCNN, self).__init__()
        self.fuse_scheme = fuse_scheme # MAX, MEAN, SUM
        self.conv2 = ConvBlock(64, 64)
        self.conv3 = ConvBlock(64, 64)
        self.conv4 = nn.Conv2d(64, 3, kernel_size=1, padding=0, stride=1, bias=True)

        # Initialize parameters for other parameters
```

Code

IFCNN1.ipynb

File Edit View Insert Runtime Tools Help Last edited on July 23

Comment Share

+ Code + Text

Connect Editing

[ ]

```
        # Initialize parameters for other parameters
        for m in self.modules():
            if isinstance(m, nn.Conv2d):
                n = m.kernel_size[0] * m.kernel_size[1] * m.out_channels
                m.weight.data.normal_(0, math.sqrt(2. / n))

        # Initialize conv1 with the pretrained resnet101 and freeze its parameters
        for p in resnet.parameters():
            p.requires_grad = False
        self.conv1 = resnet.conv1
        self.conv1.stride = 1
        self.conv1.padding = (0, 0)

    def tensor_max(self, tensors):
        max_tensor = None
        for i, tensor in enumerate(tensors):
            if i == 0:
                max_tensor = tensor
            else:
                max_tensor = torch.max(max_tensor, tensor)
        return max_tensor

    def tensor_sum(self, tensors):
        sum_tensor = None
        for i, tensor in enumerate(tensors):
```

Code

IFCNN1.ipynb

File Edit View Insert Runtime Tools Help Last edited on July 23

Comment Share

+ Code + Text Connect Editing

```
        # Initialize parameters for other parameters
        for m in self.modules():
            if isinstance(m, nn.Conv2d):
                n = m.kernel_size[0] * m.kernel_size[1] * m.out_channels
                m.weight.data.normal_(0, math.sqrt(2. / n))

        # Initialize conv1 with the pretrained resnet101 and freeze its parameters
        for p in resnet.parameters():
            p.requires_grad = False
        self.conv1 = resnet.conv1
        self.conv1.stride = 1
        self.conv1.padding = (0, 0)

    def tensor_max(self, tensors):
        max_tensor = None
        for i, tensor in enumerate(tensors):
            if i == 0:
                max_tensor = tensor
            else:
                max_tensor = torch.max(max_tensor, tensor)
        return max_tensor

    def tensor_sum(self, tensors):
        sum_tensor = None
        for i, tensor in enumerate(tensors):
```

Code

IFCNN1.ipynb

File Edit View Insert Runtime Tools Help Last edited on July 23

Comment Share

+ Code + Text Connect Editing

```python
        return out_tensors

    def forward(self, *tensors):
        # Feature extraction
        outs = self.tensor_padding(tensors=tensors, padding=(3, 3, 3, 3), mode='replicate')
        outs = self.operate(self.conv1, outs)
        outs = self.operate(self.conv2, outs)

        # Feature fusion
        if self.fuse_scheme == 0: # MAX
            out = self.tensor_max(outs)
        elif self.fuse_scheme == 1: # SUM
            out = self.tensor_sum(outs)
        elif self.fuse_scheme == 2: # MEAN
            out = self.tensor_mean(outs)
        else: # Default: MAX
            out = self.tensor_max(outs)

        # Feature reconstruction
        #print(out.shape)
        out = self.conv3(out)
        #print(out.shape)
        out = self.conv4(out)
        #print(out.shape)
        return out
```

Code

IFCNN1.ipynb

File Edit View Insert Runtime Tools Help Last edited on July 23

Comment Share

+ Code + Text

Connect Editing

```
[ ] def myIFCNN(fuse_scheme=0):
        # pretrained resnet101
        resnet = models.resnet101(pretrained=True)
        # our model
        model = IFCNN(resnet, fuse_scheme=fuse_scheme)
        return model
```

```
[ ] model = myIFCNN(0)
    model_name = "IFCNN-MAX"
    model.load_state_dict(torch.load("C:/Users\Chayma.MOUSSA\IFCNN\Code\snapshots\IFCNN-MAX.pth", map_location=torch.device('c
    model.eval()
```

```
IFCNN(
  (conv2): ConvBlock(
    (conv): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
  )
  (conv3): ConvBlock(
    (conv): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
  )
  (conv4): Conv2d(64, 3, kernel_size=(1, 1), stride=(1, 1))
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=1, bias=False)
)
```

Code

```python
# My Convolution Block
class ConvBlock(nn.Module):
    def __init__(self, inplane, outplane):
        super(ConvBlock, self).__init__()
        self.padding = (1, 1, 1, 1)
        self.conv = nn.Conv2d(inplane, outplane, kernel_size=3, padding=0, stride=1, bias=False)
        self.bn = nn.BatchNorm2d(outplane)
        self.relu = nn.ReLU(inplace=True)

    def forward(self, x):
        out = F.pad(x, self.padding)
        out = self.conv(out)
        out = self.bn(out)#batch norm
        out = self.relu(out)
        return out
```

```python
class IFCNN(nn.Module):
    def __init__(self, resnet, fuse_scheme=0):
        super(IFCNN, self).__init__()
        self.fuse_scheme = fuse_scheme # MAX, MEAN, SUM
        self.conv2 = ConvBlock(64, 64)
        self.conv3 = ConvBlock(64, 64)
        self.conv4 = nn.Conv2d(64, 3, kernel_size=1, padding=0, stride=1, bias=True)

        # Initialize parameters for other parameters
```

Calculate the shift between the IR and the RGB images

```
RGB = cv2.imread("C:/Users/Chayma.MOUSSA/Desktop/IR_1.jpg", 1)
RGB_resized = cv2.resize(RGB, (1080,1440 ))
plt.imshow(RGB_resized)

IR = cv2.imread("C:/Users/Chayma.MOUSSA/Desktop/IR_3.jpg", 1)
IR_gray = cv2.cvtColor(IR, cv2.COLOR_BGR2GRAY)

#register the translation between the IR image and the RGB image
shift, error, diffphase = register_translation(IR_gray, RGB_resized)

print(shift)
print(error)
print(diffphase)
#apply the offset to the RGB image
offset_image = fourier_shift(np.fft.fftn(RGB_resized), shift)
offset_image = np.fft.ifftn(offset_image)
plt.imshow(offset_image.real)
```