validate_model

April 16, 2025

[18]:
```
print("hi")
```

```
hi
```

[19]:
```
from ultralytics import YOLO
import os
%matplotlib inline
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import numpy as np
```

[20]:
```
# Load the YOLO model
base_dir = '/home/ian/intellicook/ingredient-recognition/model'
weights_path = os.path.join(base_dir, 'train/runs/detect/train/weights/best.pt')
model = YOLO(weights_path)
```

[21]:
```
# Validate the model
data_path = os.path.join(base_dir, 'data/Food Ingredient Recognition.v4i.
↪yolov11/data.yaml')
results = model.val(data=data_path, split='val')
```

```
Ultralytics 8.3.55  Python-3.12.2 torch-2.6.0+cu124 CUDA:0 (NVIDIA GeForce RTX
4090, 24195MiB)
YOLO11m summary (fused): 303 layers, 20,077,063 parameters, 0 gradients, 67.9
GFLOPs

val: Scanning /home/ian/intellicook/ingredient-
recognition/model/data/Food Ingredient
Recognition.v4i.yolov11/valid/labels.cache… 780 images, 67 backgrounds, 0
corrupt: 100%|          | 780/780 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|          | 49/49 [00:04<00:00, 11.94it/s]

                   all        780       2517        0.8      0.815      0.858
0.594
                 apple         10         20      0.947        0.9      0.919
0.81
             asparagus         12         47      0.532      0.298      0.371
```

0.117						
	avocado	13	49	0.934	0.871	0.963
0.709						
	banana	15	25	0.768	0.92	0.878
0.633						
	bell pepper	13	59	0.851	0.972	0.966
0.544						
	bitter gourd	10	33	0.581	0.606	0.625
0.258						
	bok choy	17	49	0.752	0.682	0.796
0.337						
	broccoli	7	27	0.833	0.889	0.886
0.498						
	cabbage	10	19	0.864	0.947	0.969
0.674						
	carrot	14	39	0.705	0.538	0.672
0.267						
	cashew	14	94	0.836	0.921	0.92
0.434						
	cauliflower	14	17	0.932	0.882	0.983
0.576						
	chayote	10	51	0.87	0.961	0.953
0.545						
	chicken breast	20	38	0.909	0.895	0.97
0.633						
	chicken thigh	18	31	0.979	0.935	0.978
0.741						
	chicken wing	13	34	0.859	0.735	0.866
0.531						
	chilli	13	55	0.53	0.709	0.63
0.41						
	coconut	19	49	0.709	0.918	0.895
0.742						
	coconuts	3	10	0.93	0.9	0.978
0.642						
	corn	18	79	0.736	0.835	0.822
0.501						
	crab	11	18	0.511	0.444	0.515
0.316						
	cucumber	15	71	0.815	0.808	0.899
0.562						
	egg_	13	61	0.977	0.934	0.97
0.861						
	eggplant	16	60	0.785	0.75	0.777
0.573						
	garlic	13	53	0.947	0.906	0.962
0.731						
	ginger	6	14	0.675	0.5	0.568

0.421	grapes	11	20	0.764	0.75	0.84
0.584	lemon	26	65	0.815	0.678	0.848
0.657	lettuce	10	13	0.723	0.604	0.628
0.424	lobster tails	11	28	0.832	0.786	0.887
0.539	mango	14	47	0.856	0.809	0.937
0.818	melon	20	41	0.864	0.854	0.888
0.665	onion	8	53	0.596	1	0.933
0.726	orange	10	36	0.871	0.917	0.98
0.768	oysters	12	87	0.891	0.862	0.878
0.658	pawpaw	11	33	0.691	0.909	0.841
0.631	peanuts	20	222	0.726	0.824	0.863
0.584	peas	7	12	0.704	0.667	0.73
0.348	pineapple	17	27	0.924	0.815	0.893
0.657	pork belly	13	29	0.692	0.759	0.841
0.62	potato	16	71	0.922	0.986	0.988
0.868	pumpkin	11	19	0.849	0.947	0.965
0.87	radishes	13	19	0.713	0.785	0.793
0.436	red rice	14	23	0.79	0.826	0.94
0.741	salmon	17	27	0.884	0.848	0.954
0.695	sea scallops	6	43	0.761	0.907	0.86
0.651	shrimp	10	47	0.717	0.66	0.674
0.409	spinach	20	156	0.71	0.763	0.806
0.504	strawberry	12	59	0.79	0.814	0.904
0.591	sweet potato	19	77	0.865	0.922	0.957

```
0.754
                tempeh        11         34      0.788      0.877      0.854
0.503
                  tofu        15         58      0.863      0.862      0.942
0.704
                tomato        15         43       0.75       0.93      0.877
0.773
                  tuna        10         16      0.936      0.812      0.953
0.702
            white rice         8         10      0.948          1      0.995
0.695
Speed: 0.3ms preprocess, 3.5ms inference, 0.0ms loss, 0.3ms postprocess per
image
Results saved to runs/detect/val2
```

[22]:

```
# Extract metrics
metrics = results.results_dict
print(metrics.keys())

for key in metrics:
    print(f"{key}: {metrics[key]}")
```

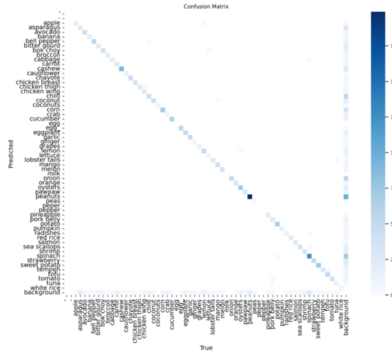
```
dict_keys(['metrics/precision(B)', 'metrics/recall(B)', 'metrics/mAP50(B)',
'metrics/mAP50-95(B)', 'fitness'])
metrics/precision(B): 0.8000694712113787
metrics/recall(B): 0.8150705497445575
metrics/mAP50(B): 0.8577925524501078
metrics/mAP50-95(B): 0.5935191361677661
fitness: 0.6199464777960002
```

[23]:

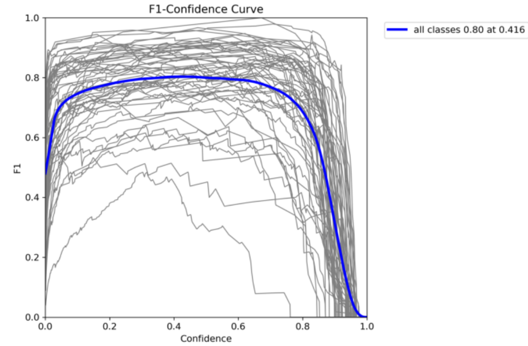
```
image_paths = [
  os.path.join(base_dir, 'train/runs/detect/val/confusion_matrix.png'),
  os.path.join(base_dir, 'train/runs/detect/val/F1_curve.png'),
  os.path.join(base_dir, 'train/runs/detect/val/PR_curve.png'),
  os.path.join(base_dir, 'train/runs/detect/val/P_curve.png'),
  os.path.join(base_dir, 'train/runs/detect/val/R_curve.png'),
  os.path.join(base_dir, 'train/runs/detect/val/confusion_matrix_normalized.
↪png'),
  os.path.join(base_dir, 'train/runs/detect/val/val_batch0_labels.jpg'),
  os.path.join(base_dir, 'train/runs/detect/val/val_batch0_pred.jpg')
]

# Display all images
fig, axes = plt.subplots(nrows=4, ncols=2, figsize=(15, 20))
axes = axes.flatten()

for ax, img_path in zip(axes, image_paths):
```

```
    img = plt.imread(img_path)
    ax.imshow(img)
    ax.axis('off')  # Hide the axis
    ax.set_title(os.path.basename(img_path))

plt.tight_layout()
plt.show()
```

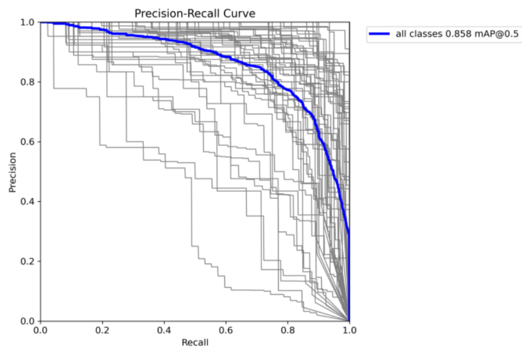
confusion_matrix.png



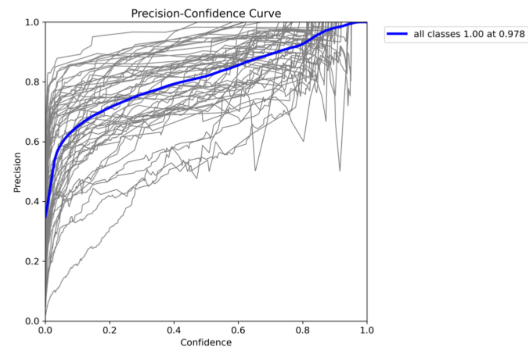
F1_curve.png



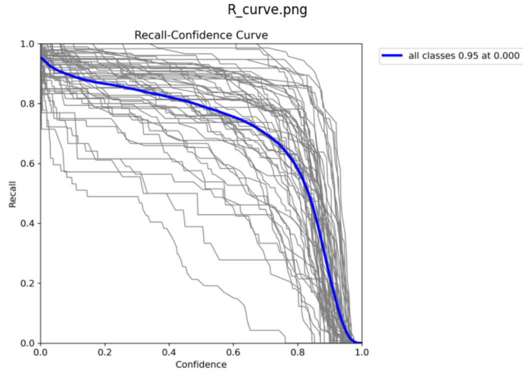
PR_curve.png



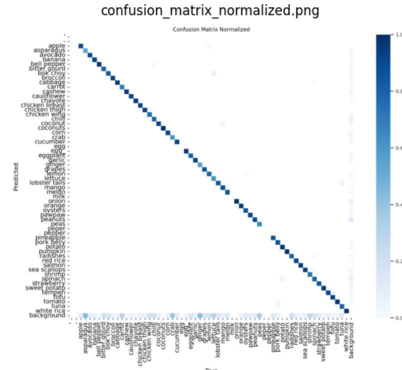
P_curve.png



R_curve.png



confusion_matrix_normalized.png



val_batch0_labels.jpg

val_batch0_pred.jpg