roboflow

Axis-Aligned Bounding Box (AABB - txt file

class_id coordinates (x_center, y_center) (width) (height)

```
0 0.51328125 0.67265625 0.20859375 0.20546875
```

Oriented Bounding Box (OBB) - txt file

class_id coordinates (x_center, y_center) (width) (height) (Rotation angle (θ))

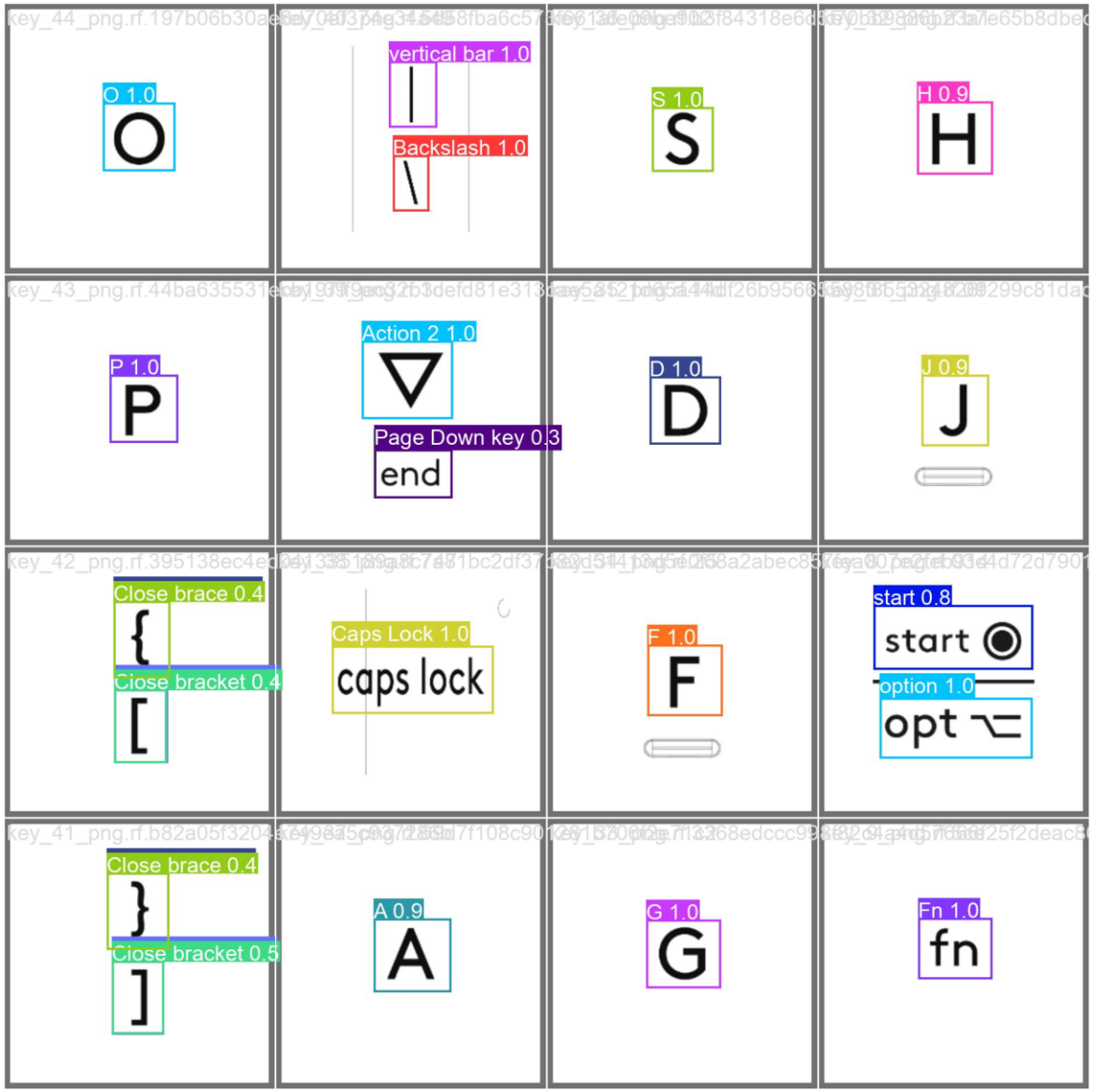

Pothole
test_096_jpg.rf.4

```python
from ultralytics import YOLO

def main():
    # Load a model
    model = YOLO("yolov8n.yaml")  # build a new model from scratch
    model = YOLO("yolov8n.pt")  # load a pretrained model (recommended for training)

    # Use the model
    model.train(data="momo640.yaml", epochs=100)  # train the model
    metrics = model.val()  # evaluate model performance on the validation set
    results = model("https://attach.setn.com/newsimages/2019/07/09/2010347-XXL.jpg")
    path = model.export(format="onnx")  # export the model to ONNX format

if __name__ == '__main__':
    main()
```

```yaml
train: ./momo640/train # training dataset
val: ./momo640/valid  # validation dataset
nc: 1
names: ['Pothole']
```


Pothole 0.6
Pothole 0.4
Pothole 0.8
Pothole 0.3.5
Pothole 0.8
Pothole 0.3
Pothole 0.4
Pothole 0.7
Pothole 0.7
Pothole 0.3
Pothole 0.7
Pothole 0.9
Pothole 0.5
Pothole 0.4
Pothole 0.4
Pothole 0.3
Pothole 0.6
Pothole 0.6
Pothole 0.4
Pothole 0.8

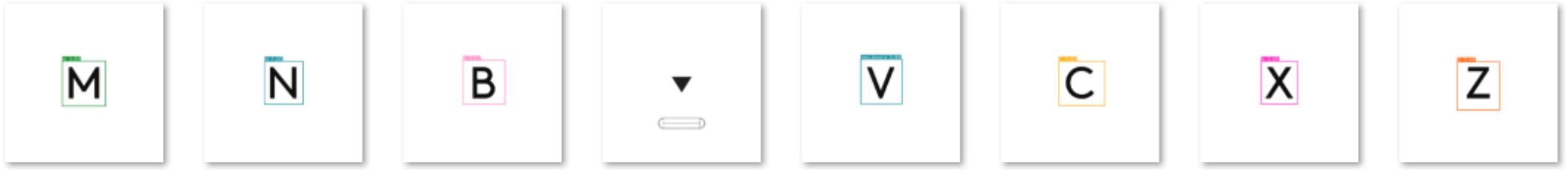
- Multi Class
- Inhouse training
- Customized test program
- YOLO v9 VS v8
- Downlaod Ultrlytics, my code
- Use more threshold to analyze efficientnet backbone




```
train: ./momo640/train # training dataset
val: ./momo640/valid  # validation dataset
nc: 118
names: ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'Action 2',
'Action 3', 'Action 4', 'Alt', 'Ampersand', 'Asterisk', 'At sign', 'B',
'BackSpace', 'Backslash', 'Brightness Down', 'Brightness Up', 'C', 'Caps
Lock', 'Caret', 'Close brace', 'Close bracket', 'Close parenthesis', 'Colon',
'Command', 'Ctrl', 'D', 'Dictation', 'Dollar sign', 'Double dash', 'E', 'Easy
Switch channel 1', 'Easy Switch channel 2', 'Easy Switch channel 3', 'Emoji',
'Equals', 'F', 'F1', 'F10', 'F11', 'F12', 'F2', 'F3', 'F4', 'F5', 'F6', 'F7',
'F8', 'F9', 'Fn', 'Fn Lock', 'G', 'Greater than', 'H', 'Home Key', 'Hyphen',
'I', 'Insert Key', 'J', 'K', 'L', 'Less than', 'M', 'Mute Sound', 'N', 'O',
'Open brace', 'Open bracket', 'Open parenthesis', 'P', 'Page Down key', 'Page
Up key', 'Percent', 'Play Pause', 'Plus', 'Pound sign', 'Q', 'Question mark',
'Quote', 'R', 'S', 'Screen Capture', 'Screen Lock', 'Search Spotlight',
'Semicolon', 'Show Desktop', 'Single quote', 'T', 'Tilde', 'U', 'Volume Down',
'Volume Up', 'W', 'X', 'Y', 'backtick', 'comma', 'delete', 'dot', 'end',
'enter', 'esc', 'exclamation', 'forwar_slash', 'key_board', 'option', 'right',
'shift', 'start', 'tab', 'up', 'vertical bar']
```

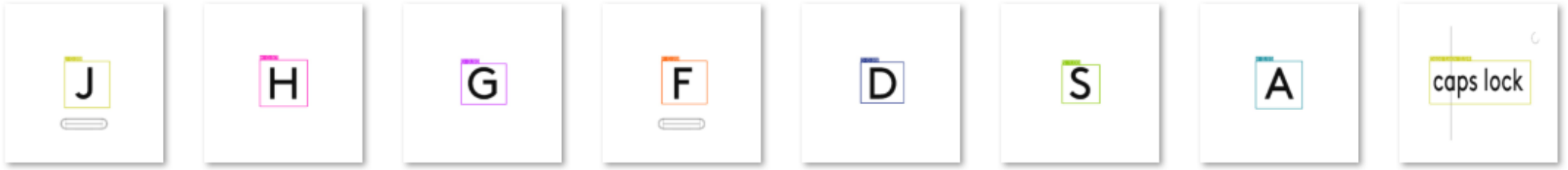
```python
model = YOLO('best.pt')


image_dir = 'word_key_yolo_used'
image_paths = [os.path.join(image_dir, file) for file in os.listdir(image_dir) if file.endswith(('.png', '.jpg', '.jpeg'))]


results = model(image_paths)


save_dir = 'detected_images'
os.makedirs(save_dir, exist_ok=True)


for i, (result, img_path) in enumerate(zip(results, image_paths)):
    image_name = Path(img_path).name
    detections = []


    if result.boxes:
        boxes = result.boxes.data
        for box in boxes:
            class_id = int(box[5])
            confidence = box[4]
            class_name = model.names[class_id]
            detections.append(f"{class_name} ({confidence:.2f})")


    print(f"Image: {image_name}, Detections: {', '.join(detections)}")


    save_path = os.path.join(save_dir, f'key_{i}.jpg')
    result.save(save_path)
```

key_0.jpg key_1.jpg key_2.jpg key_3.jpg key_4.jpg key_5.jpg key_6.jpg key_7.jpg



key_8.jpg key_9.jpg key_10.jpg key_11.jpg key_12.jpg key_13.jpg key_14.jpg key_15.jpg

key_16.jpg key_17.jpg key_18.jpg key_19.jpg key_20.jpg key_21.jpg key_22.jpg key_23.jpg



key_24.jpg key_25.jpg key_26.jpg key_27.jpg key_28.jpg key_29.jpg key_30.jpg key_31.jpg


Volume Up 0.93
F12 0.97
F12
Alt 0.98
alt
Command 0.81
cmd ⌘
Screen Capture 0.98
F8 0.93
F8
Action 2 0.79
end 0.96
end

YOLO V8n

```python
from ultralytics import YOLO

def main():
    # Load a model
    model = YOLO("yolov8n.yaml")  # build a new model from scratch
    model = YOLO("yolov8n.pt")  # load a pretrained model (recommended for training)

    # Use the model
    model.train(data="momo640.yaml", epochs=100)  # train the model
    metrics = model.val()  # evaluate model performance on the validation set
    results = model("https://attach.setn.com/newsimages/2019/07/09/2010347-XXL.jpg")
    path = model.export(format="onnx")  # export the model to ONNX format

if __name__ == '__main__':
    main()
```

```yaml
# YOLOv8.0n backbone
backbone:
  # [from, repeats, module, args]
  - [-1, 1, Conv, [64, 3, 2]] # 0-P1/2
  - [-1, 1, Conv, [128, 3, 2]] # 1-P2/4
  - [-1, 3, C2f, [128, True]]
  - [-1, 1, Conv, [256, 3, 2]] # 3-P3/8
  - [-1, 6, C2f, [256, True]]
  - [-1, 1, Conv, [512, 3, 2]] # 5-P4/16
  - [-1, 6, C2f, [512, True]]
  - [-1, 1, Conv, [1024, 3, 2]] # 7-P5/32
  - [-1, 3, C2f, [1024, True]]
  - [-1, 1, SPPF, [1024, 5]] # 9

# YOLOv8.0n head
head:
  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 6], 1, Concat, [1]] # cat backbone P4
  - [-1, 3, C2f, [512]] # 12

  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 4], 1, Concat, [1]] # cat backbone P3
  - [-1, 3, C2f, [256]] # 15 (P3/8-small)

  - [-1, 1, Conv, [256, 3, 2]]
  - [[-1, 12], 1, Concat, [1]] # cat head P4
  - [-1, 3, C2f, [512]] # 18 (P4/16-medium)

  - [-1, 1, Conv, [512, 3, 2]]
  - [[-1, 9], 1, Concat, [1]] # cat head P5
  - [-1, 3, C2f, [1024]] # 21 (P5/32-large)

  - [[15, 18, 21], 1, Detect, [nc]] # Detect(P3, P4, P5)
```

YOLO V9c

```python
from ultralytics import YOLO

def main():
    # Load a model
    model = YOLO("yolov9c.yaml")  # build a new model from scratch
    model = YOLO("yolov9c.pt")  # load a pretrained model (recommended for training)

    # Use the model
    model.train(data="momo640.yaml", epochs=100)  # train the model
    metrics = model.val()  # evaluate model performance on the validation set
    results = model("https://attach.setn.com/newsimages/2019/07/09/2010347-XXL.jpg")
    path = model.export(format="onnx")  # export the model to ONNX format

if __name__ == '__main__':
    main()
```

```yaml
# gelan backbone
backbone:
  - [-1, 1, Conv, [64, 3, 2]]  # 0-P1/2
  - [-1, 1, Conv, [128, 3, 2]]  # 1-P2/4
  - [-1, 1, RepNCSPELAN4, [256, 128, 64, 1]]  # 2
  - [-1, 1, ADown, [256]]  # 3-P3/8
  - [-1, 1, RepNCSPELAN4, [512, 256, 128, 1]]  # 4
  - [-1, 1, ADown, [512]]  # 5-P4/16
  - [-1, 1, RepNCSPELAN4, [512, 512, 256, 1]]  # 6
  - [-1, 1, ADown, [512]]  # 7-P5/32
  - [-1, 1, RepNCSPELAN4, [512, 512, 256, 1]]  # 8
  - [-1, 1, SPPELAN, [512, 256]]  # 9

head:
  - [-1, 1, nn.Upsample, [None, 2, 'nearest']]
  - [[-1, 6], 1, Concat, [1]]  # cat backbone P4
  - [-1, 1, RepNCSPELAN4, [512, 512, 256, 1]]  # 12

  - [-1, 1, nn.Upsample, [None, 2, 'nearest']]
  - [[-1, 4], 1, Concat, [1]]  # cat backbone P3
  - [-1, 1, RepNCSPELAN4, [256, 256, 128, 1]]  # 15 (P3/8-small)

  - [-1, 1, ADown, [256]]
  - [[-1, 12], 1, Concat, [1]]  # cat head P4
  - [-1, 1, RepNCSPELAN4, [512, 512, 256, 1]]  # 18 (P4/16-medium)

  - [-1, 1, ADown, [512]]
  - [[-1, 9], 1, Concat, [1]]  # cat head P5
  - [-1, 1, RepNCSPELAN4, [512, 512, 256, 1]]  # 21 (P5/32-large)

  - [[15, 18, 21], 1, Detect, [nc]]  # Detect(P3, P4, P5)
```

YOLO V8n

YOLO V9c