Computer Vision IST Seminar

Final Project Report

Luu Huu Phuc
Kyoto University

Exercise 1: estimate the LABEL of image x

- New dataloader:
  - +) adjust the dataloader → output batches of (x, label)
- New networks:
  - +) Hiragana model: output size = 956 (total classes of labels)
  - +) a model based on TinyYOLO model and Hiragana model

```
self.ConvBlock = nn.Sequential(                           # in : 64 x 64 x 1
    nn.Conv2d(1, 32, 3, 1, 1), nn.BatchNorm2d(32), self.act,  # 64 x 64 x 32
    nn.Conv2d(32, 32, 3, 1, 1), nn.BatchNorm2d(32), self.act,  # 64 x 64 x 32
    self.pool,                                            #  32 x 32 x 32
    nn.Conv2d(32, 64, 3, 1, 1), nn.BatchNorm2d(64), self.act,  # 32 x 32 x 64
    nn.Conv2d(64, 64, 3, 1, 1), nn.BatchNorm2d(64), self.act,  # 32 x 32 x 64
    self.pool,                                            #  16 x 16 x 64
    nn.Conv2d(64, 128, 3, 1, 1), nn.BatchNorm2d(128), self.act,  # 16 x 16 x 128
    nn.Conv2d(128, 128, 3, 1, 1), nn.BatchNorm2d(128), self.act, # 16 x 16 x 128
    self.pool,                                            # 8 x 8 x 128
    nn.Conv2d(128, 256, 3, 1, 1), nn.BatchNorm2d(256), self.act,  # 8 x 8 x 256
    nn.Conv2d(256, 256, 3, 1, 1), nn.BatchNorm2d(256), self.act,  # 8 x 8 x 256
    self.pool,                                            # 4 x 4 x 256
    nn.Conv2d(256, 512, 3, 1, 1), nn.BatchNorm2d(512), self.act,  # 4 x 4 x 512
    nn.Conv2d(512, 512, 3, 1, 1), nn.BatchNorm2d(512), self.act,  # 4 x 4 x 512
    self.pool,                                            # 2 x 2 x 512
    nn.Conv2d(512, 1024, 3, 1, 1), nn.BatchNorm2d(1024), self.act,  # 2 x 2 x 1024
    nn.Conv2d(1024, 1024, 3, 1, 1), nn.BatchNorm2d(1024), self.act,  # 2 x 2 x 1024
    self.pool,                                            # 1 x 1 x 1024
)

self.FC_as_Conv = nn.Sequential(
    nn.Conv2d(1024, 4096, 1, 1, 0), self.act, self.drop, # 1 x 1 x 4096
    nn.Conv2d(4096, 4096, 1, 1, 0), self.act, self.drop, # 1 x 1 x 4096
    nn.Conv2d(4096, 956, 1, 1, 0),        # 1 x 1 x 956
)
```

Model architecture:

ConvBlock + FC_as_Conv block

+) ConvBlock = 6 small blocks

Each small block:

Reduce spatial size by half
Doule the depth

+) In: 64 x 64 x 1
+) Out: 1 x 1 x 956

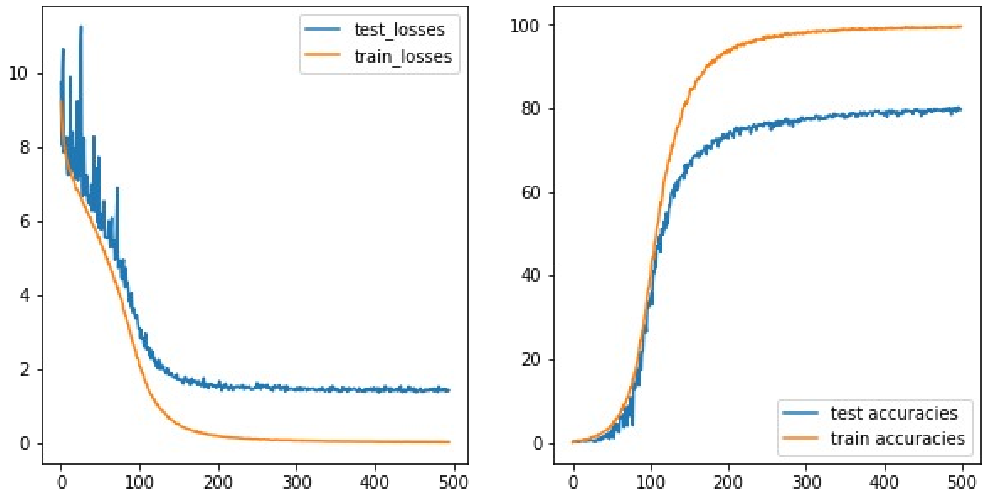
Exercise 1: estimate the LABEL of image x

- Classification results

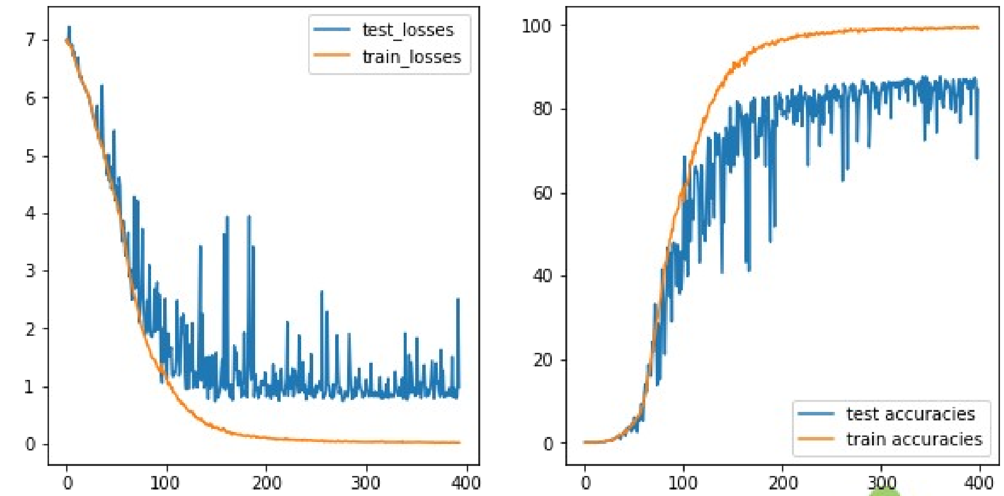
  +) Hiragana model: Train 99% , Test 80%, stable

  +) TinyYOLO model: Train 99%, Test 87%, not so stable

- Train accuracy ~ 100% => The two models can learn the distribution of train data
- To avoid overfit: apply random transform (crop, translate, zoom out, flip, …) when training



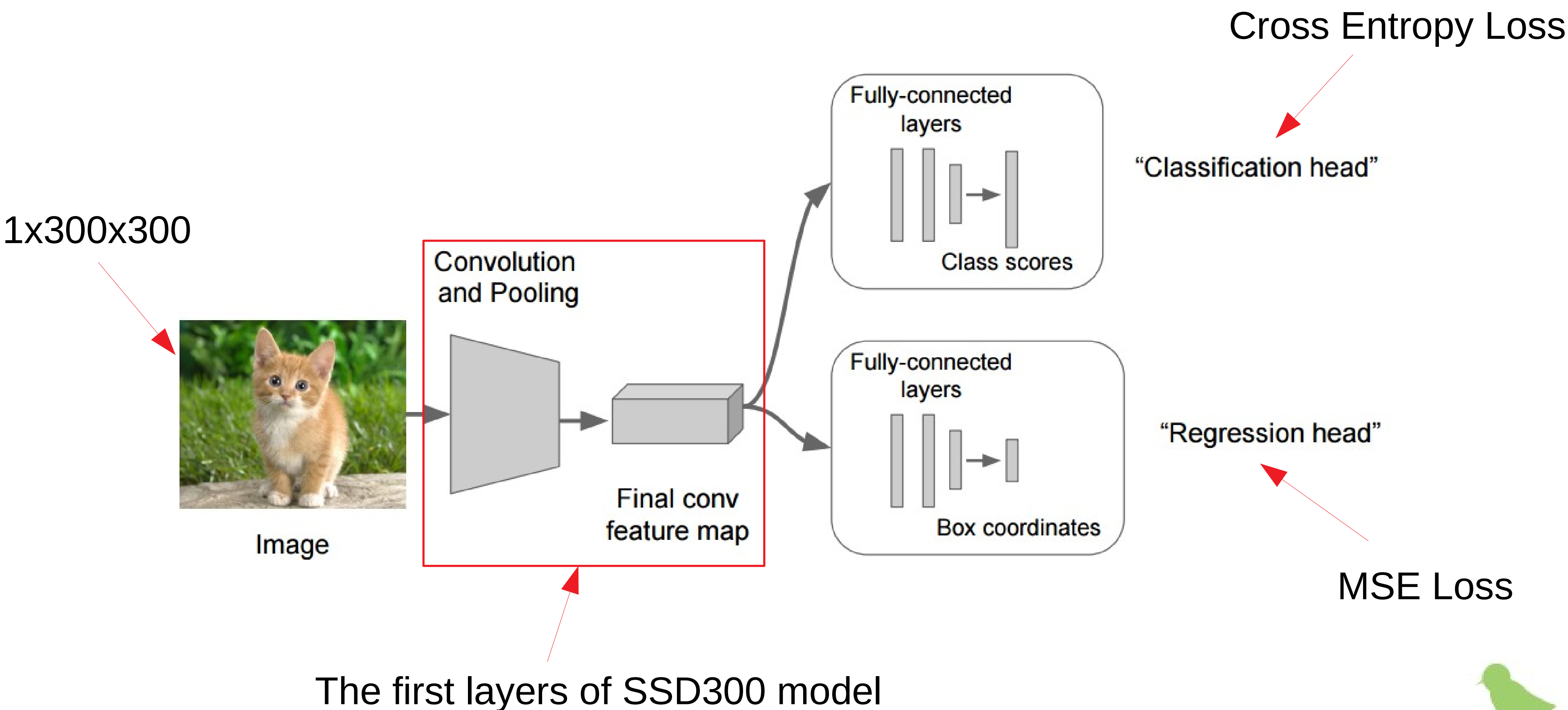
Hiragana model



TinyYOLO model

Exercise 2: Hiragana moji detection

- First try:
  - +) Use a conv model to extract feature from the images
  - +) Then split into 2 branches: label branch + bbox branch



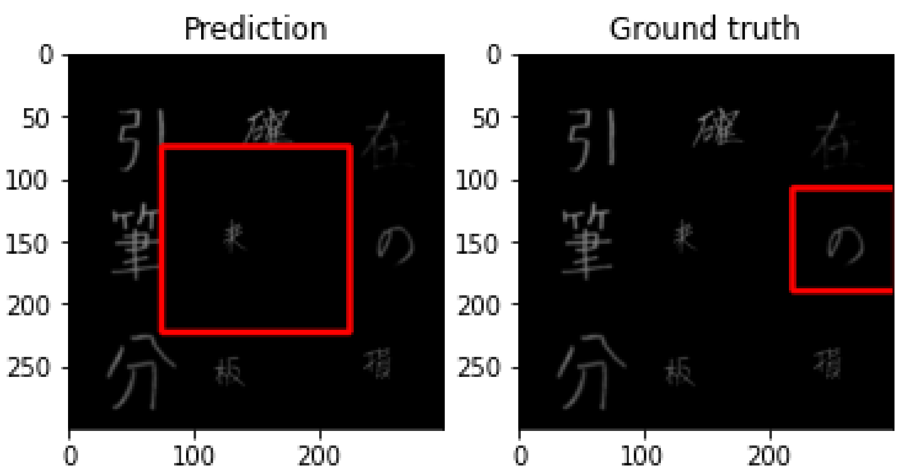
Exercise 2: Hiragana moji detection

- First try: **Fails**

  => But the Loss did not reduces

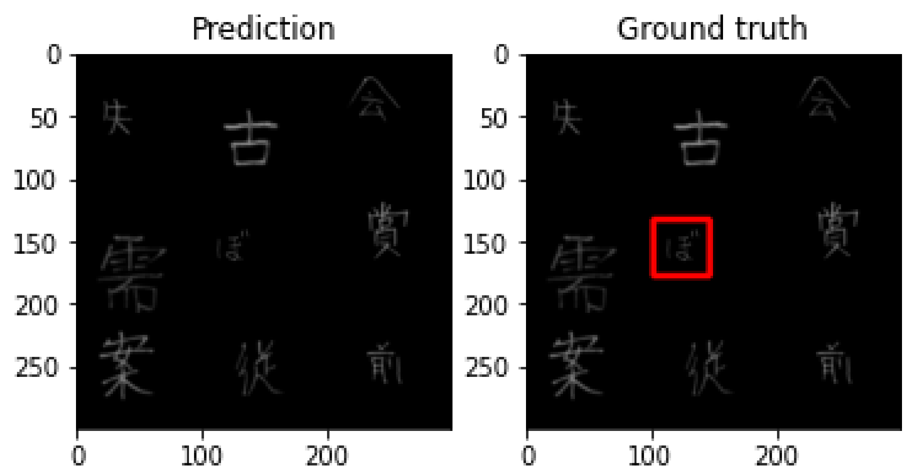
  The model outputs are all 0s

Before training




```
[ 74  75 150 148]
[218 108  82  82]
```

After training




```
Prediction:   [0 0 0 0]
Ground truth:  [101 133  45  45]
```

Exercise 2: Hiragana moji detection with YOLO algorithm

- I tried again with YOLO algorithm

- Model:

  +) TinyYOLO model: the same as in Ex1, except the output channel is 80

  +) 80 = (1+4)x1 + 75: 1 confidence score, 4 bbox location, 75 class probabilities

- Resize images:

  +) Pad around each image with 10 pixels

  +) Size: 300x300 → 320x320

  => Each image can be divided into 5x5 grids of size 64x64

```
self.ConvBlock = nn.Sequential(
    nn.Conv2d(1, 32, 3, 1, 1), nn.BatchNorm2d(32), self.act,
    nn.Conv2d(32, 32, 3, 1, 1), nn.BatchNorm2d(32), self.act,
    self.pool,
    nn.Conv2d(32, 64, 3, 1, 1), nn.BatchNorm2d(64), self.act,
    nn.Conv2d(64, 64, 3, 1, 1), nn.BatchNorm2d(64), self.act,
    self.pool,
    nn.Conv2d(64, 128, 3, 1, 1), nn.BatchNorm2d(128), self.act,
    nn.Conv2d(128, 128, 3, 1, 1), nn.BatchNorm2d(128), self.act,
    self.pool,
    nn.Conv2d(128, 256, 3, 1, 1), nn.BatchNorm2d(256), self.act,
    nn.Conv2d(256, 256, 3, 1, 1), nn.BatchNorm2d(256), self.act,
    self.pool,
    nn.Conv2d(256, 512, 3, 1, 1), nn.BatchNorm2d(512), self.act,
    nn.Conv2d(512, 512, 3, 1, 1), nn.BatchNorm2d(512), self.act,
    self.pool,
    nn.Conv2d(512, 1024, 3, 1, 1), nn.BatchNorm2d(1024), self.act,
    nn.Conv2d(1024, 1024, 3, 1, 1), nn.BatchNorm2d(1024), self.act,
    self.pool,
)

self.FC_as_Conv = nn.Sequential(
    nn.Conv2d(1024, 4096, 1, 1, 0), self.act, self.drop, # it is okie to
    nn.Conv2d(4096, 4096, 1, 1, 0), self.act, self.drop,
    nn.Conv2d(4096, 80, 1, 1, 0),  # 80 = 5 + 75
)
```

Exercise 2: Hiragana moji detection with YOLO algorithm

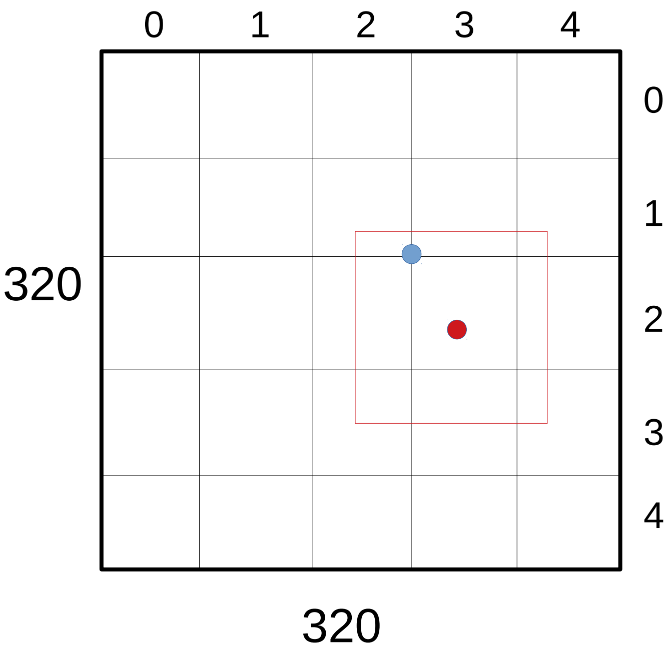
- Model:

  +) Input: 320x320x1

  +) Output: 5x5x80

```
input shape:  torch.Size([8, 1, 320, 320])
output shape:  torch.Size([8, 5, 5, 80])
```

```
input shape:  torch.Size([8, 1, 64, 64])
output shape:  torch.Size([8, 1, 1, 80])
```



- Prepare training data:

  +) convert label, bbox → 5x5x80 tensor

  +) Each tensor 1x1x80 contain:

  [conf, x, y, w, h, p1, …, p75]

  +) conf: value 1 if the moji is in the grid, 0 otherwise

  +) x, y: coord of the bbox’s center, values in (0,1)

  +) w, h: width, height of the bbox, values in (0,5)

  +) p_i: value 1 if the moji’s label is i, 0 otherwise

=> The model looks at each grid and detects if the moji is in that grid or not

Exercise 2: Hiragana moji detection with YOLO algorithm

- Loss function:

$$
\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
$$

Original YOLO loss function

$$
\lambda_{coord} \sum_{i=o}^{25} 1_i^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] + \sum_i^{25} 1_i^{obj} (c_i - \hat{c}_i)^2 + \lambda_{noobj} \sum_i^{25} 1_i^{noobj} (c_i - \hat{c}_i)^2 + \sum_i^{25} 1_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2
$$

- Lambda_coord = 5.0
- Lambda_noobj = 0.5

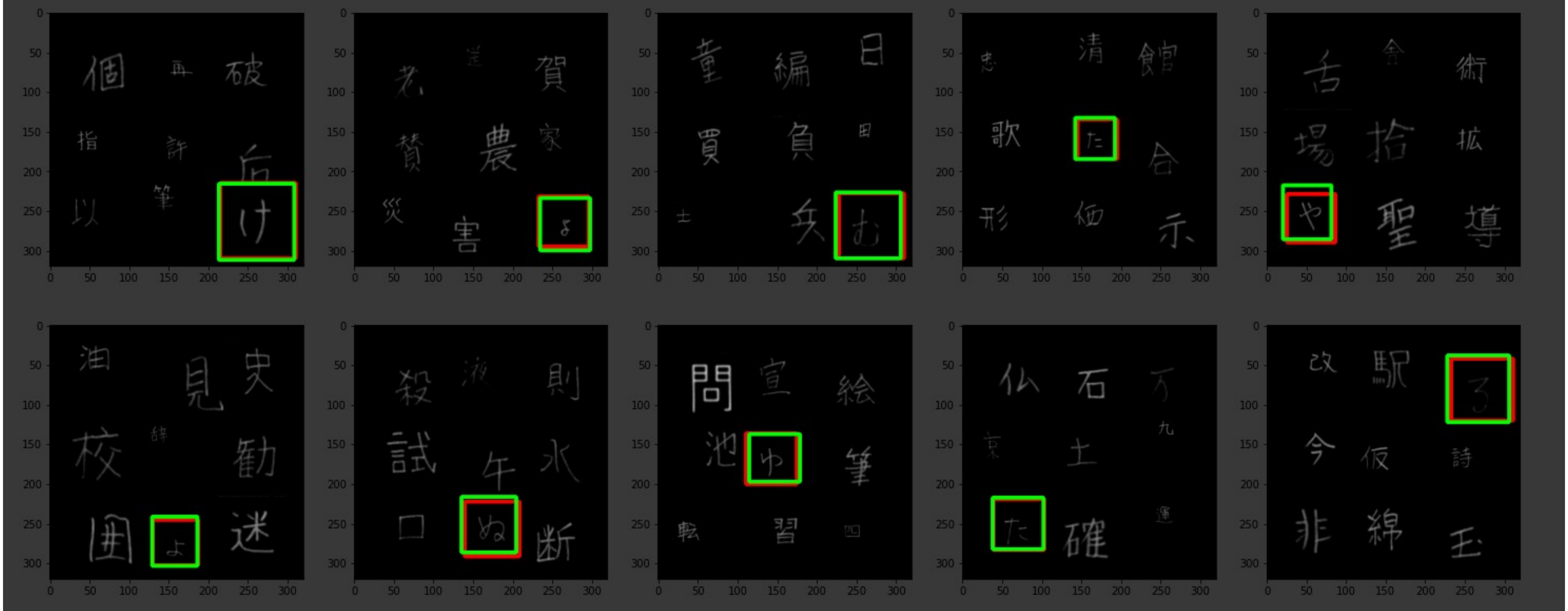
Exercise 2: Hiragana moji detection with YOLO algorithm

- Results:
  +) Correctly detect the moji's location: bbox
  +) Fails to classify moji's label

```
Mean IoU: 0.7959
Box accuracy: 0.9367
Grid accuracy: 0.9390
Label accuracy: 0.0140
```

1/75 ~ 0.01333

```
Pred label:
 ['を' 'を' 'こ' 'ほ' 'ぞ' 'を' 'ぞ' 'こ' 'み' 'を']
True label:
 ['け' 'よ' 'む' 'た' 'や' 'よ' 'ぬ' 'ゆ' 'た' 'る']
```



Exercise 2: Hiragana moji detection with YOLO algorithm

- Loss function:

$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$
$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$
$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

Original YOLO loss function

$$\lambda_{coord} \sum_{i=o}^{25} 1_i^{obj}[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] + \sum_i^{25} 1_i^{obj}(c_i - \hat{c}_i)^2 + \lambda_{noobj} \sum_i^{25} 1_i^{noobj}(c_i - \hat{c}_i)^2 + \sum_i^{25} \boxed{1_i^{obj} CrossEntropyLoss(p_i, \hat{p}_i)}$$

- Lambda_coord = 5.0
- Lambda_noobj = 0.5

Use CrossEntropyLoss instead of MSELoss

Exercise 2: Hiragana moji detection with YOLO algorithm

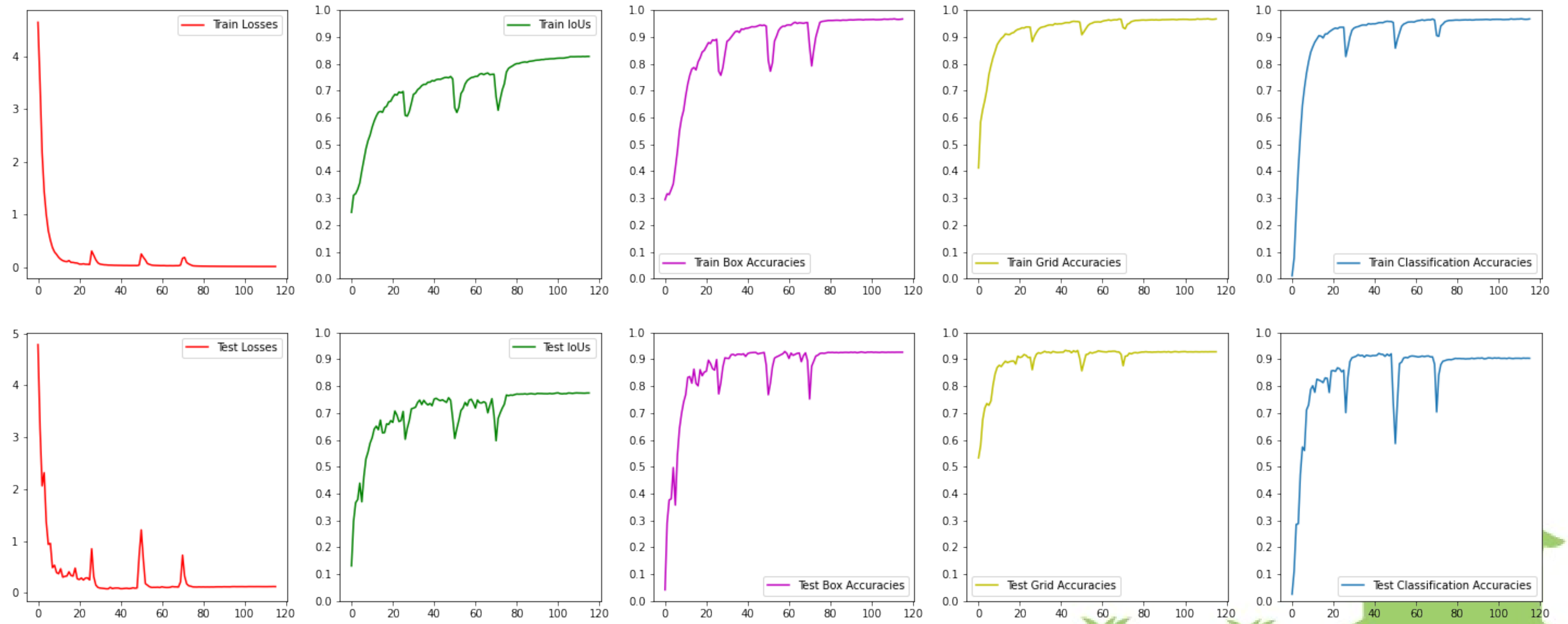
- Results:

  +) Correctly detect the moji’s location: bbox

  +) Correctly classify the moji’s label

  +) Box prediction is seen as Correct if IoU (pred_box, true_box) > 0.5

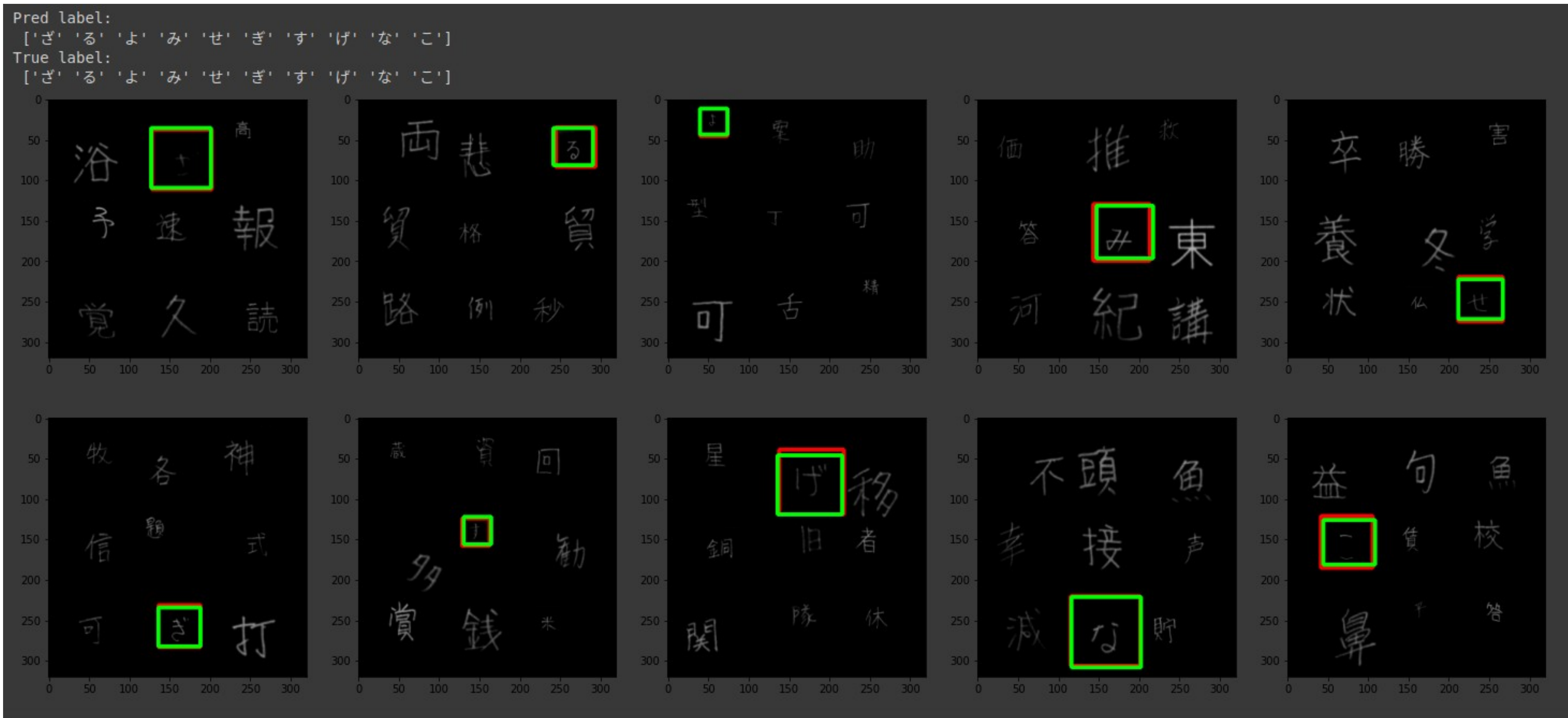
  +) Grid accuracy: correctly predict the grid that contains the moji



Exercise 2: Hiragana moji detection with YOLO algorithm

- Results:
  +) Correctly detect the moji's location: bbox
  +) Correctly classify the moji's label

```
Mean IoU: 0.7988
Box accuracy: 0.9577
Grid accuracy: 0.9583
Label accuracy: 0.9533
```



Conclusion

- Implemented the YOLO algorithm in Ex2
- To improve model's performance:

  +) randomly apply augmentation in train data (crop, translate, zoom in, flip, …)

  +) pretrain the TinyYOLO model on the 64x64x1 images of hiragana moji in Ex1