Object detection and localization

Tahereh Hematian pour fard

Data 606: Capstone

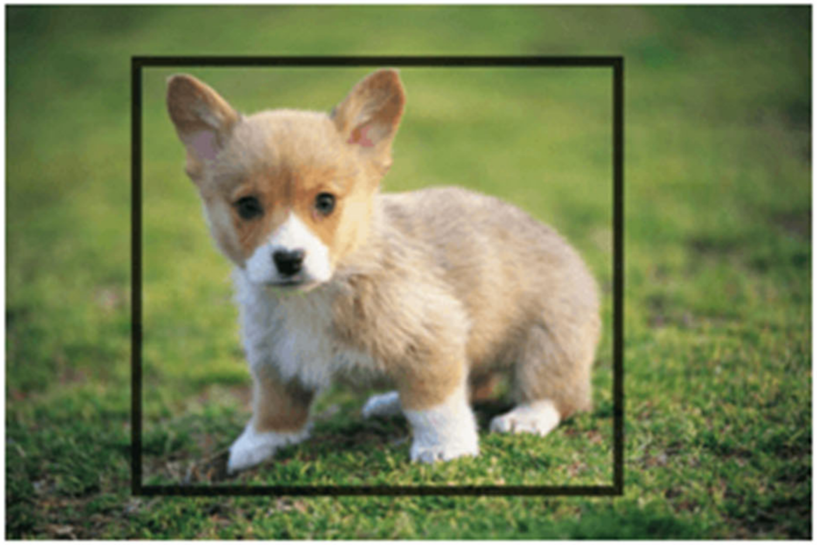
Dr. Chaojie Wang, Nov 22nd, 2022

Outlines

- Data generator
- Machine Learning Dataset preparation
- Neural Network architecture
- Hyper-parameters for training the model
- Predictions
- Challenging objects
- References

Object Detection and Localization

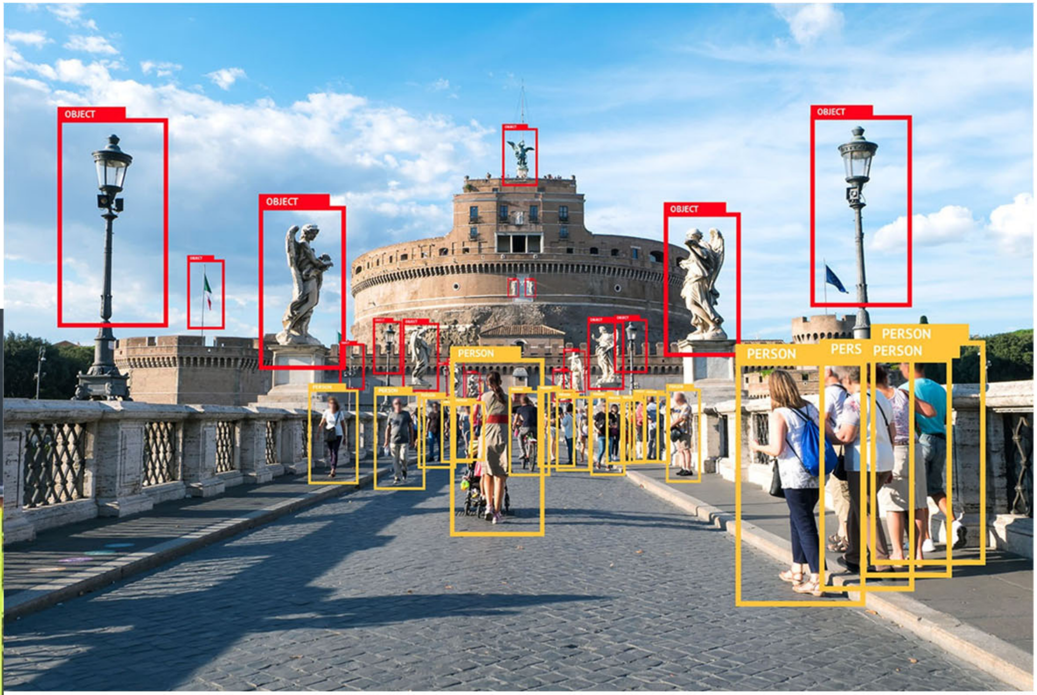
Object Classification is the task of identifying that picture is a dog

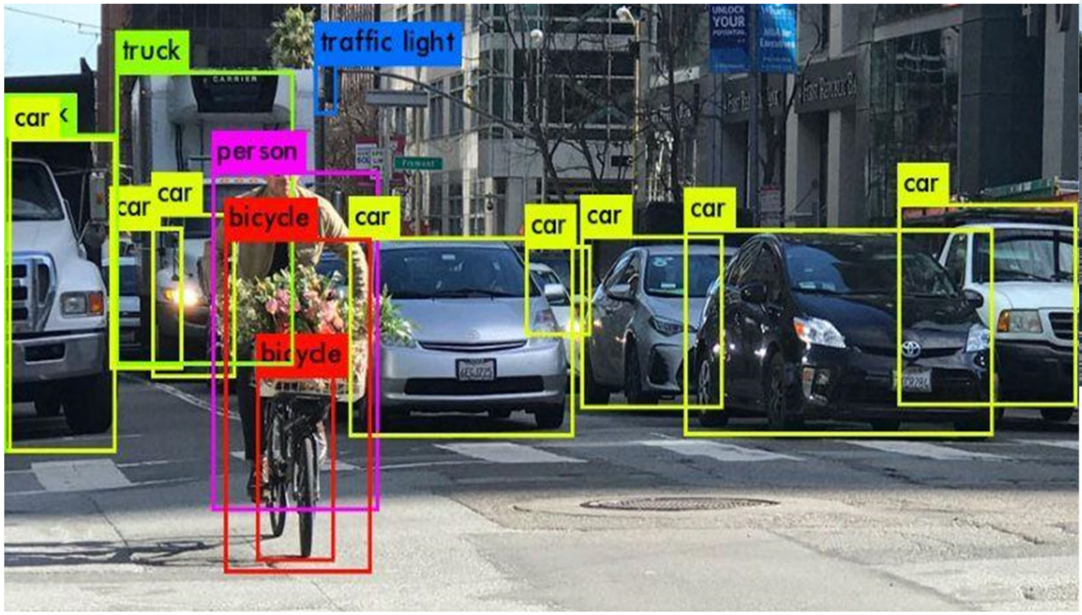


Object Localization involves the class label as well as a bounding box to show where the object is located.

Application

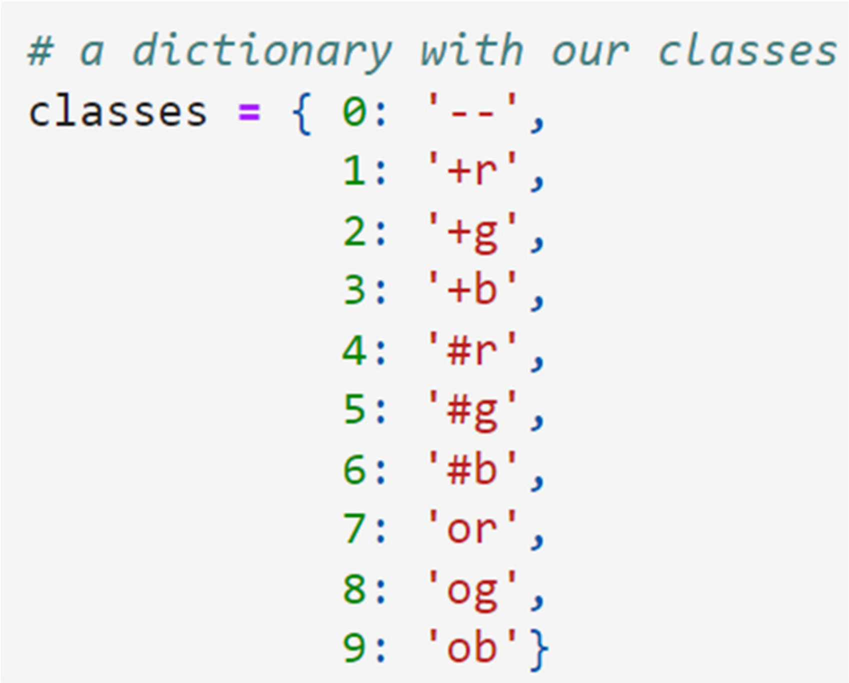
- Self-driving cars
- Video surveillance
- Climate change
- And …



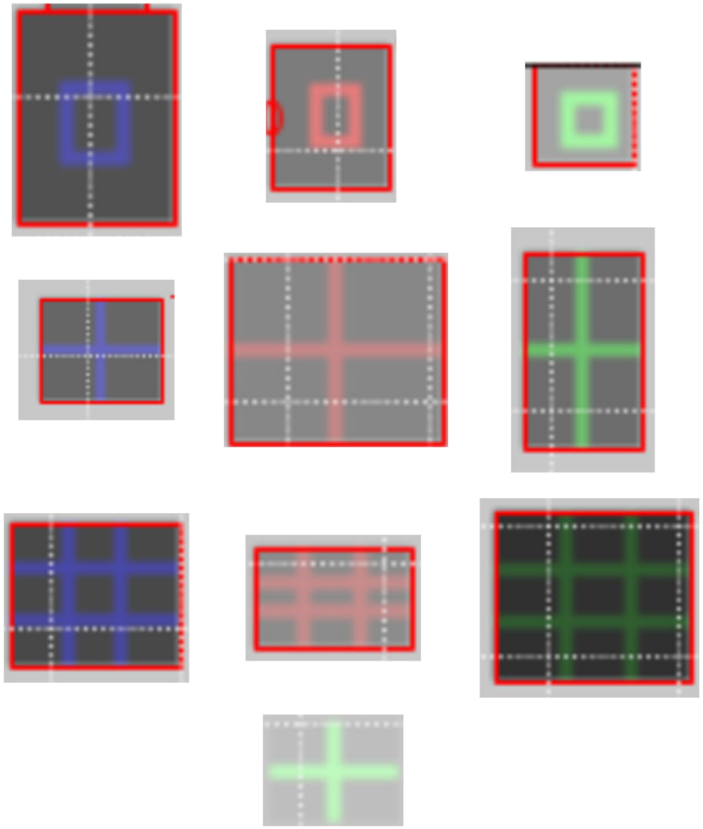


Data Generator

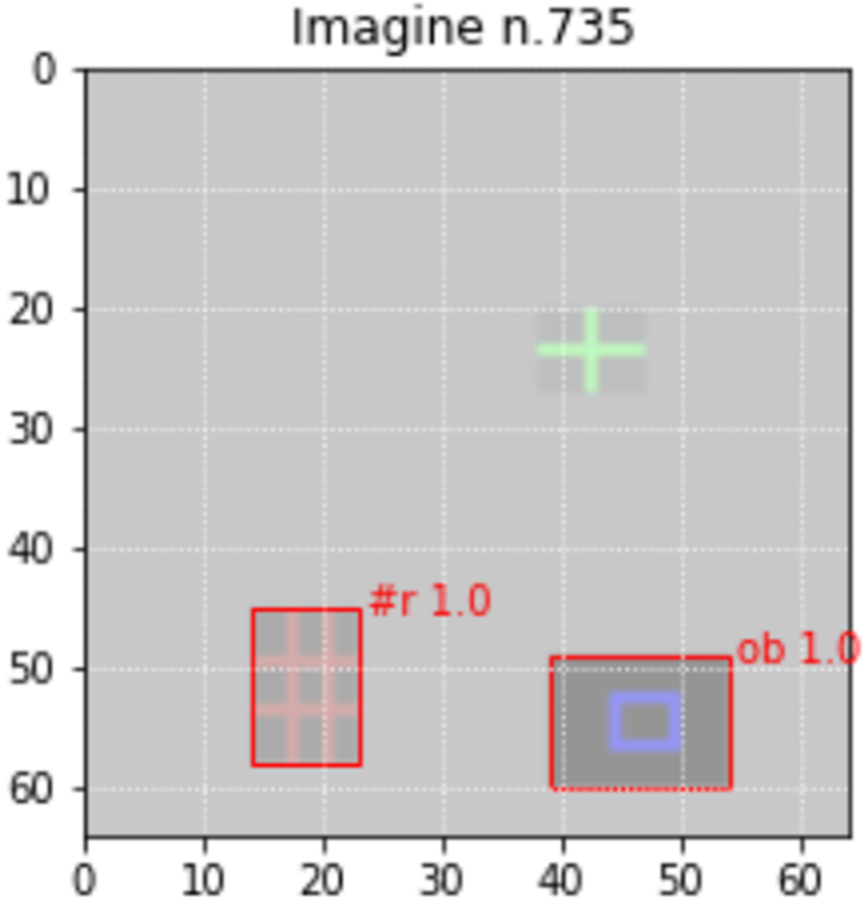
- ❑ **Synthetic data**
- ❑ **10000 images**
- ❑ **Size of each image is 64*64**
- ❑ **At most 3 objects**



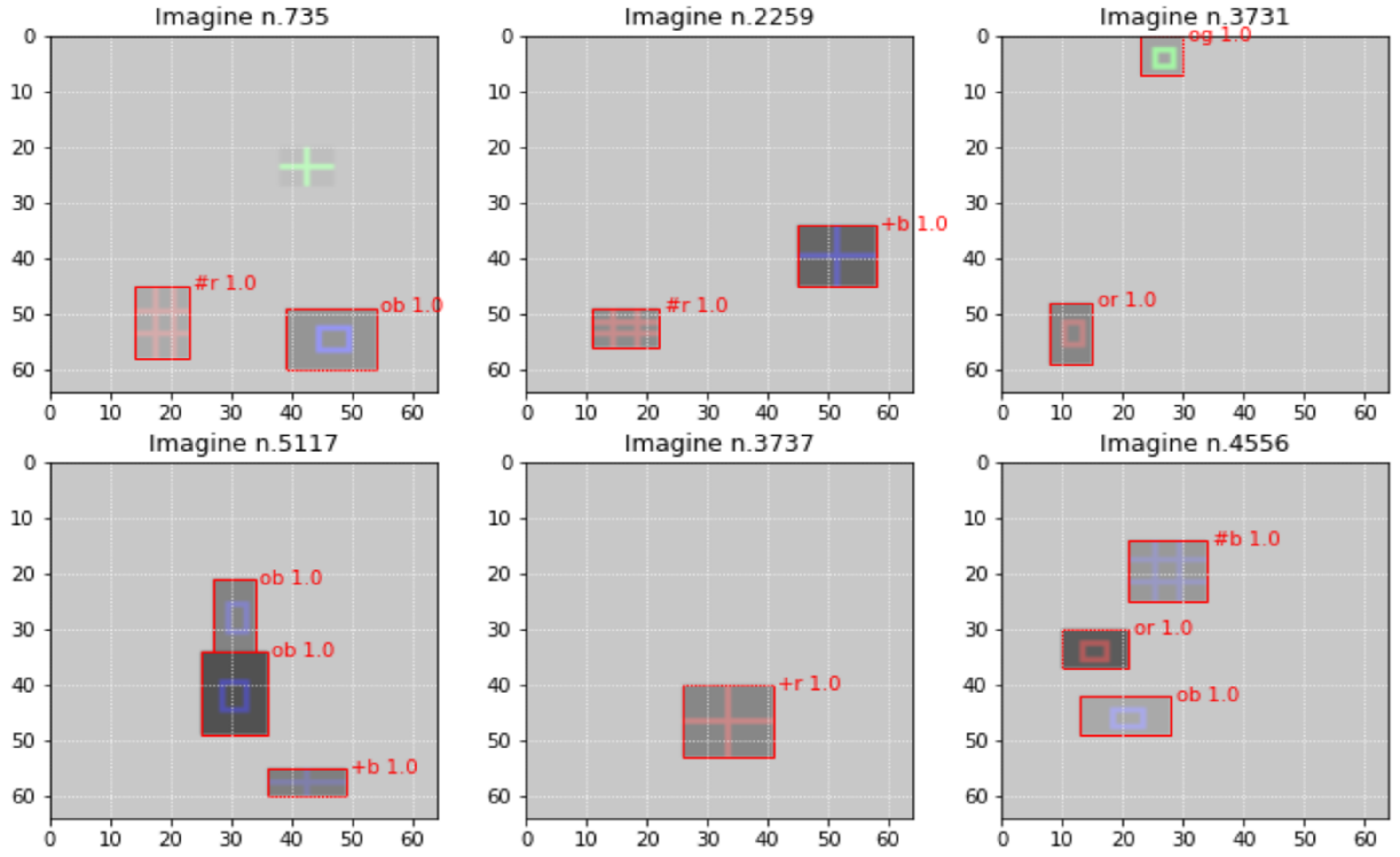
```
# a dictionary with our classes
classes = { 0: '--',
            1: '+r',
            2: '+g',
            3: '+b',
            4: '#r',
            5: '#g',
            6: '#b',
            7: 'or',
            8: 'og',
            9: 'ob'}
```



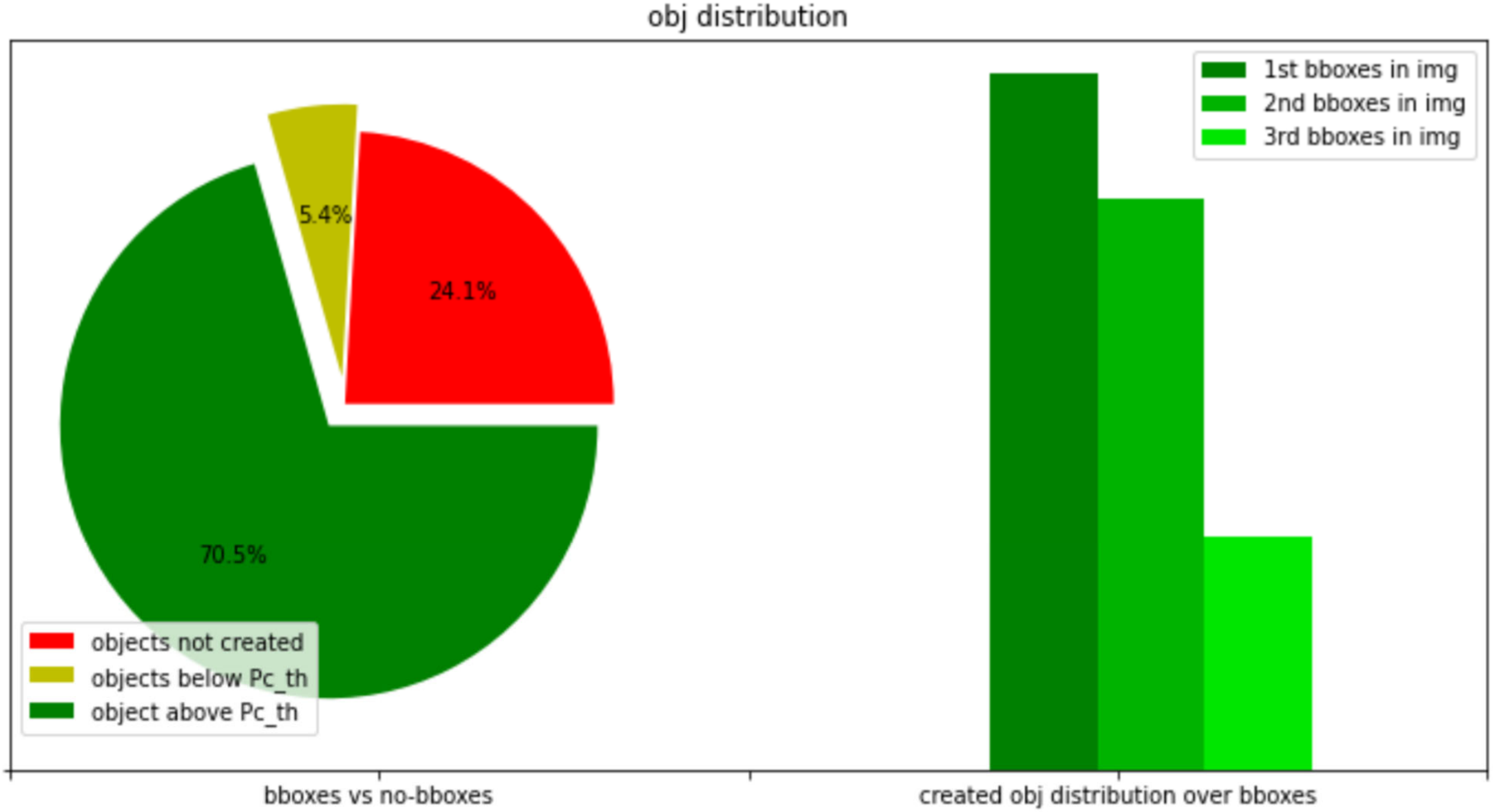
Data Generator



Data Generator



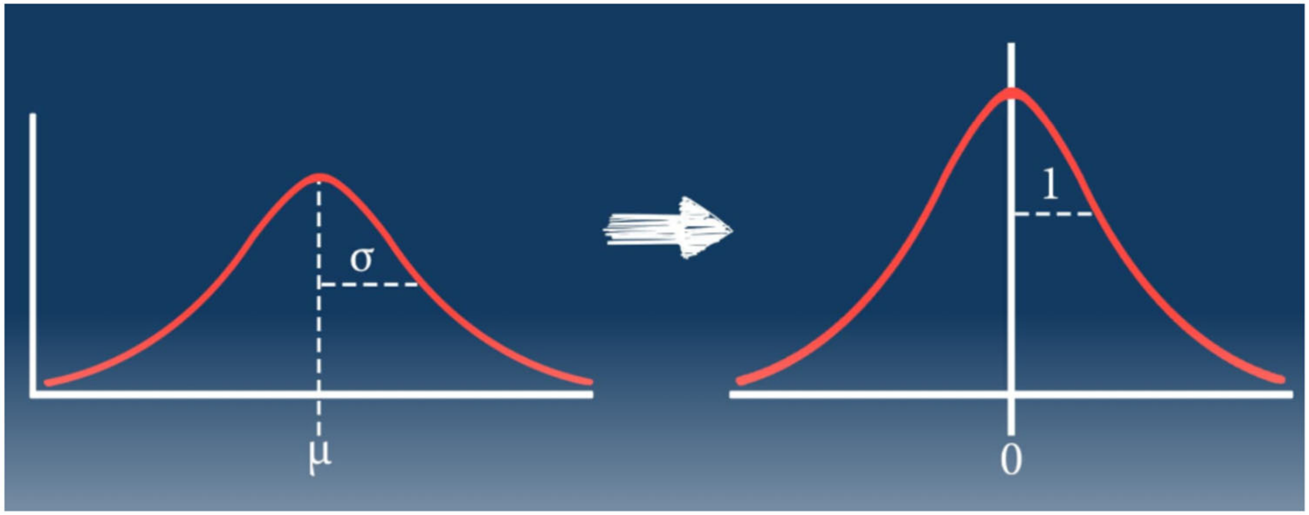
Data Generator



Machine Learning Dataset preparation

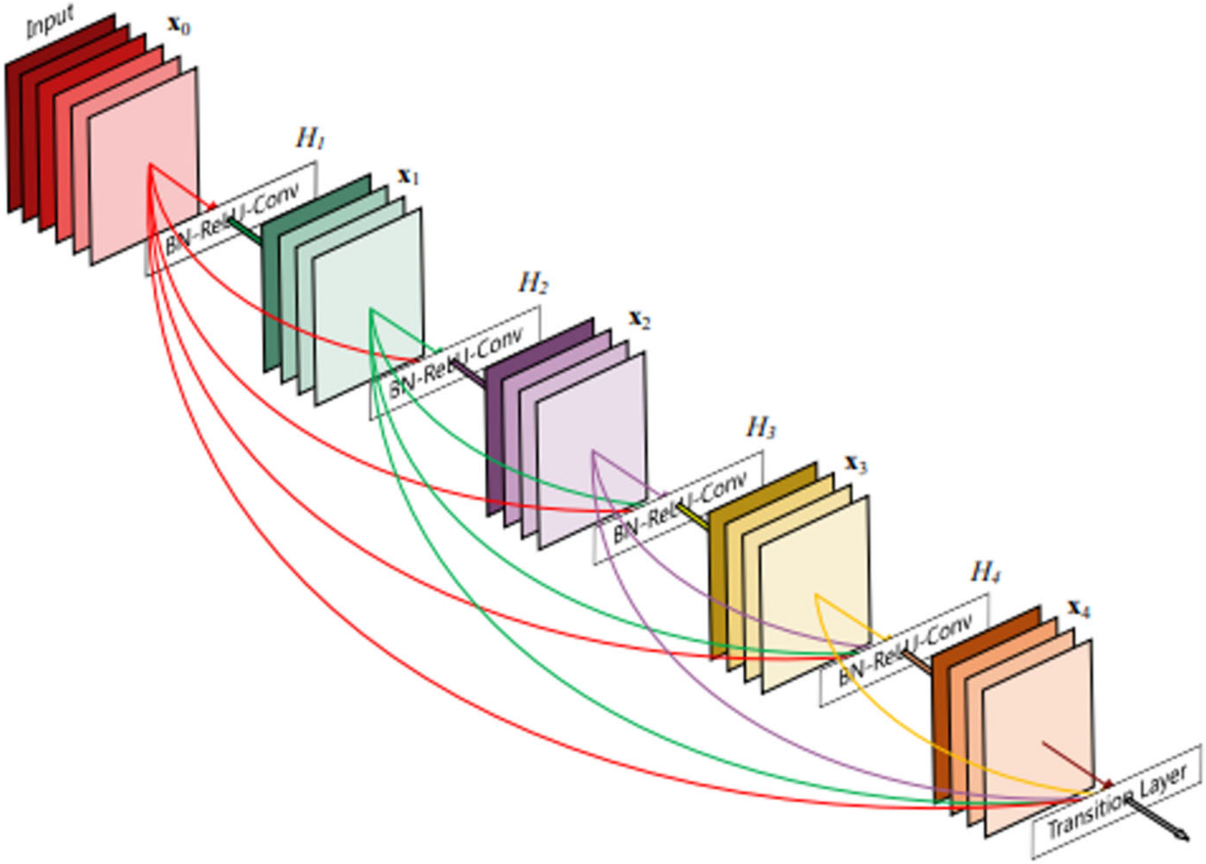
- ❑ **Average value: 196.25**
- ❑ **Standard deviation: 18.41**

$$x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{standard deviation }(x)}$$



Neural Network Architecture

- ❑ **DenseNet 200**
- ✓ **Cornell University**
- ✓ **Tsinghua University**
- ✓ **Facebook AI Research (FAIR)**



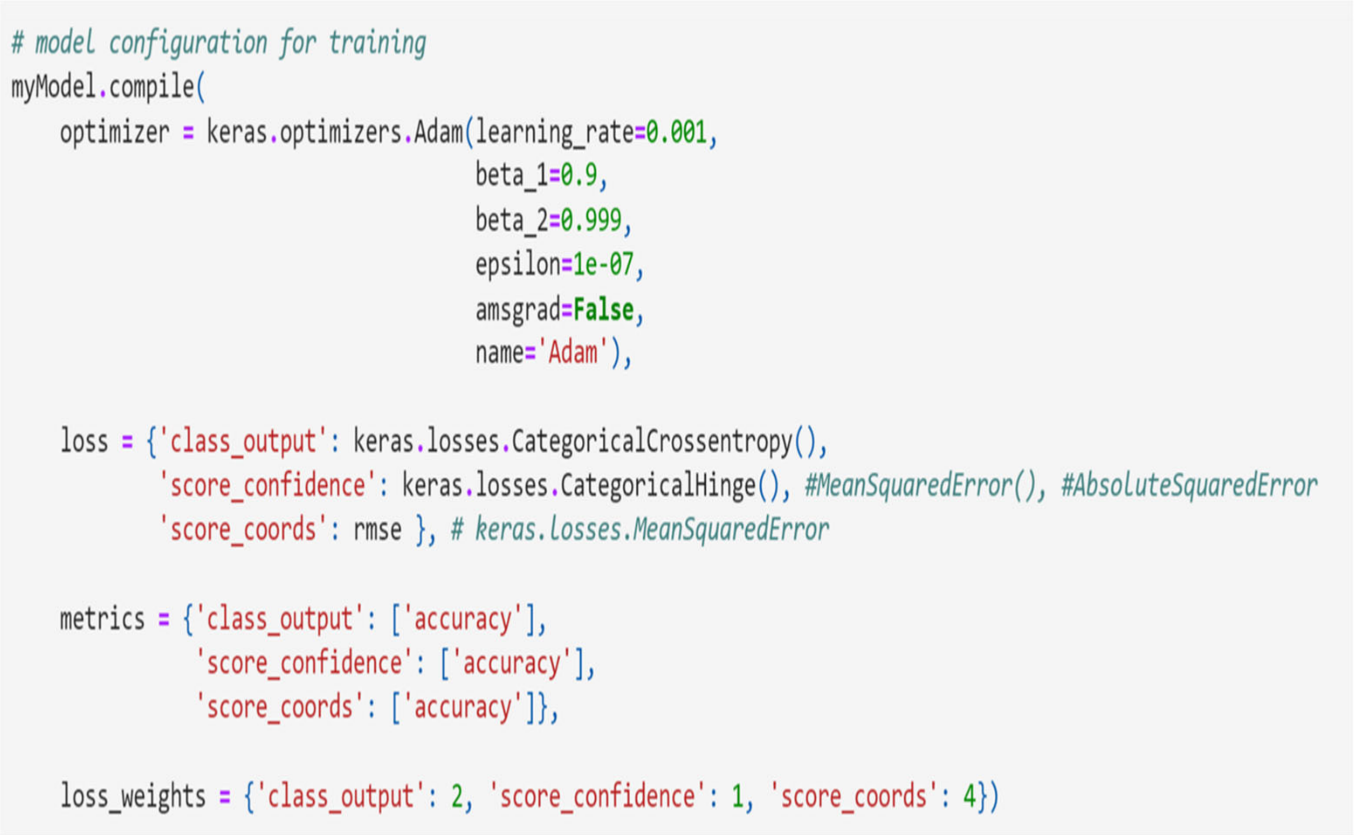
Hyper-parameters for training the model

❑ N-ep = 100

❑ Batches = 64

❑ Learning rate = 0.001

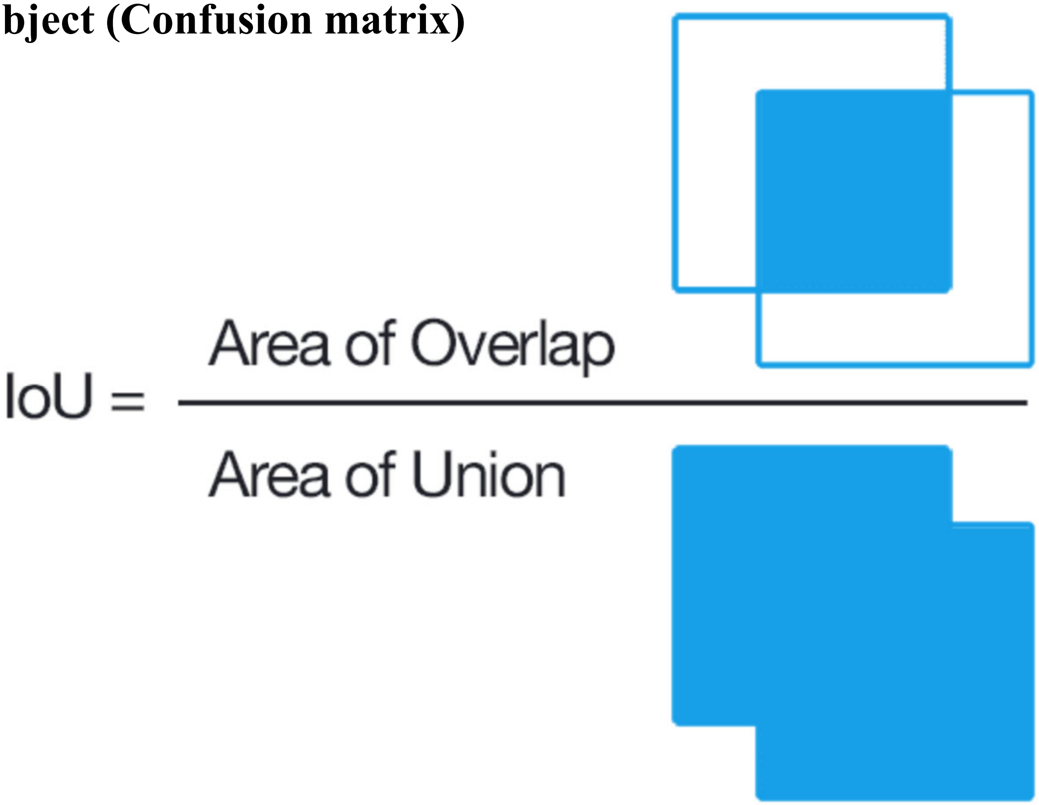
❑ Optimizer = Adam



```
# model configuration for training
myModel.compile(
    optimizer = keras.optimizers.Adam(learning_rate=0.001,
                                      beta_1=0.9,
                                      beta_2=0.999,
                                      epsilon=1e-07,
                                      amsgrad=False,
                                      name='Adam'),

    loss = {'class_output': keras.losses.CategoricalCrossentropy(),
            'score_confidence': keras.losses.CategoricalHinge(), #MeanSquaredError(), #AbsoluteSquaredError
            'score_coords': rmse }, # keras.losses.MeanSquaredError

    metrics = {'class_output': ['accuracy'],
               'score_confidence': ['accuracy'],
               'score_coords': ['accuracy']},

    loss_weights = {'class_output': 2, 'score_confidence': 1, 'score_coords': 4})
```

Predictions

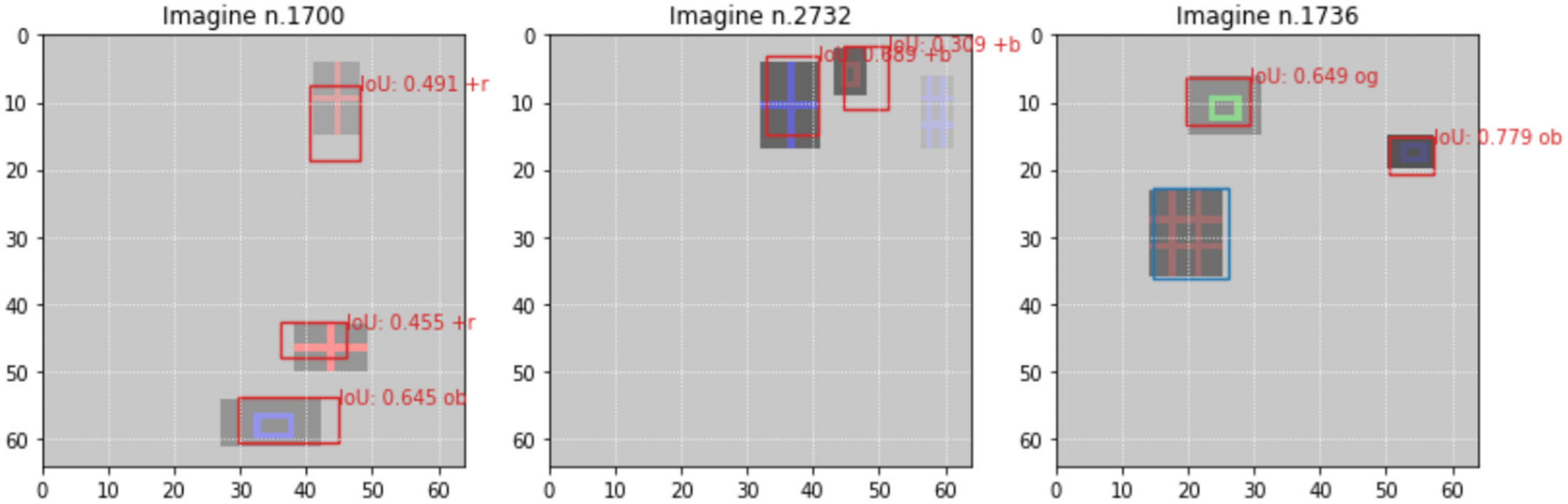
- ❑ **Identification of bounding boxes, intersection over union (IoU)**
- ❑ **Classification of the identified object (Confusion matrix)**



$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

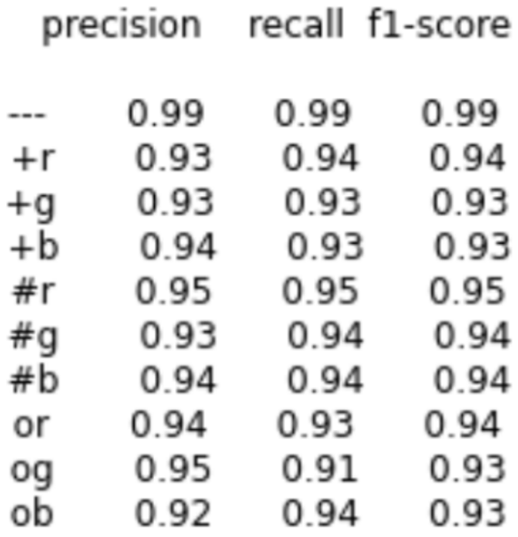
Bounding box predictions

- ❑ **Intersection over Union (IoU)**

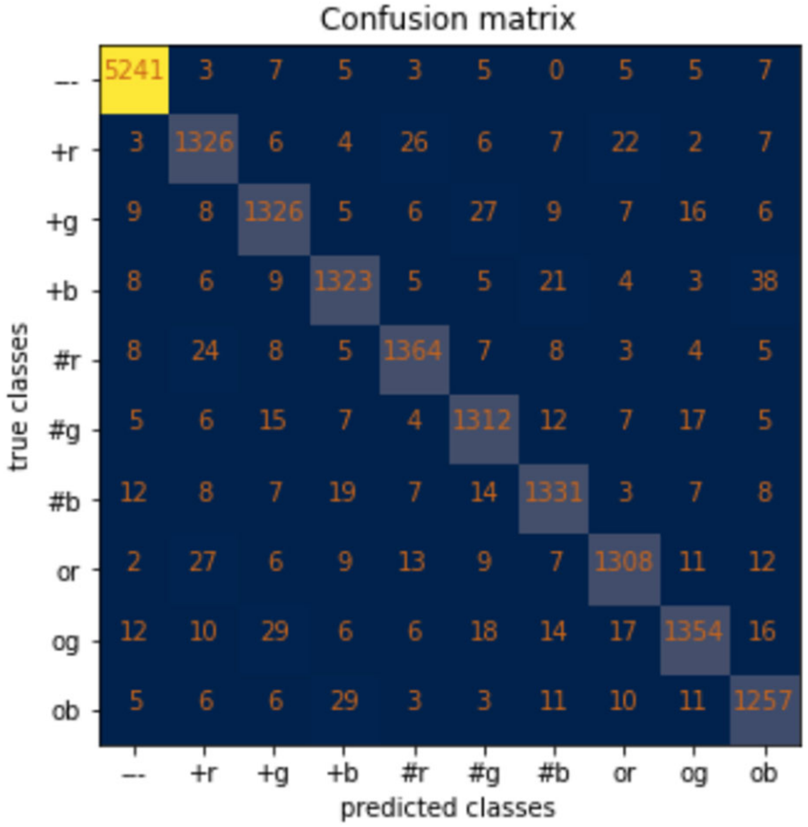


Classification Metric

- **Confusion matrix**



	precision	recall	f1-score
---	0.99	0.99	0.99
+r	0.93	0.94	0.94
+g	0.93	0.93	0.93
+b	0.94	0.93	0.93
#r	0.95	0.95	0.95
#g	0.93	0.94	0.94
#b	0.94	0.94	0.94
or	0.94	0.93	0.94
og	0.95	0.91	0.93
ob	0.92	0.94	0.93

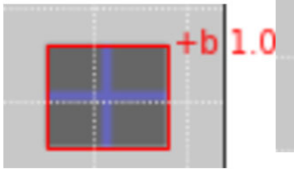


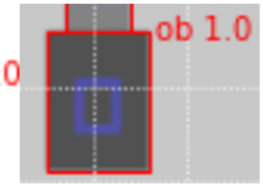
	--	+r	+g	+b	#r	#g	#b	or	og	ob
--	5241	3	7	5	3	5	0	5	5	7
+r	3	1326	6	4	26	6	7	22	2	7
+g	9	8	1326	5	6	27	9	7	16	6
+b	8	6	9	1323	5	5	21	4	3	38
#r	8	24	8	5	1364	7	8	3	4	5
#g	5	6	15	7	4	1312	12	7	17	5
#b	12	8	7	19	7	14	1331	3	7	8
or	2	27	6	9	13	9	7	1308	11	12
og	12	10	29	6	6	18	14	17	1354	16
ob	5	6	6	29	3	3	11	10	11	1257

Challenging objects

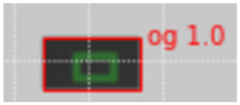
❑ **Recurrent Neural Network**

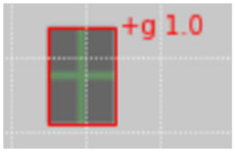
38 errors



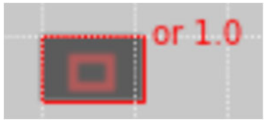


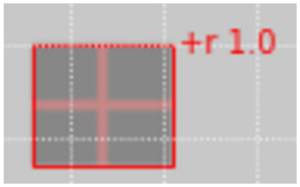
29 errors

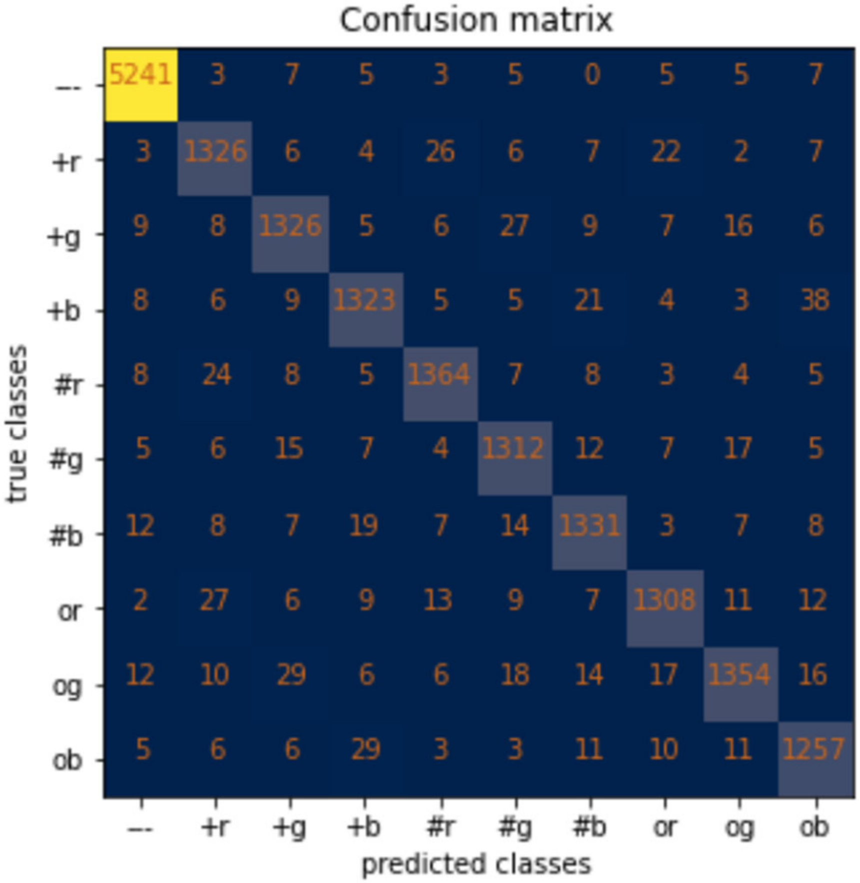




27 errors







	--	+r	+g	+b	#r	#g	#b	or	og	ob
--	5241	3	7	5	3	5	0	5	5	7
+r	3	1326	6	4	26	6	7	22	2	7
+g	9	8	1326	5	6	27	9	7	16	6
+b	8	6	9	1323	5	5	21	4	3	38
#r	8	24	8	5	1364	7	8	3	4	5
#g	5	6	15	7	4	1312	12	7	17	5
#b	12	8	7	19	7	14	1331	3	7	8
or	2	27	6	9	13	9	7	1308	11	12
og	12	10	29	6	6	18	14	17	1354	16
ob	5	6	6	29	3	3	11	10	11	1257

Conclusion

- ❑ **Synthetic data with 10000 images and at most 3 objects**
- ❑ **DensNet200: deep neural network with 200 layers**
- ❑ **IoU metric : 70%**
- ❑ **Confusion matrix: 92%**

References

- https://arxiv.org/abs/1608.06993
- https://github.com/MarcoFosci/Computer_Vision/blob/master/Shape_Detection/Different_CNN_architectures/test_Part_II_.ipynb

Thank you