Assignment 3A: Instance Segmentation with MaskRCNN

Venkata Sai Nikhil, Thodupunuri

62716136

1 Backbone architecture

The entire network is implemented using 4 network classes. Here are the name conventions used.

- **Base Network** : This class implements the backbone architecture.
- **Intermediate Network**: This class implements the intermediate layer used by both the regressor and classifier component. This class depends on the base network to generate the output. Base class is passed as a constructor parameter.
- **Regression Network**: This class implements the regression specific layers. this class depends on intermediate object (passed as a constructor parameter).
- **Classification Network**: This class implements the classification specific layers. this class depends on intermediate object (passed as a constructor parameter).

Code:

```
1
2  import torch
3  import torch.nn as nn
4
5  class BaseNetwork(torch.nn.Module):
6    def __init__(self):
7      super().__init__()
8      self.conv1 = nn.Conv2d(3, 16, 5, padding=2)
9
10     self.conv1_seq = nn.Sequential(self.conv1, nn.BatchNorm2d(num_features = 16),
11                                    nn.ReLU())
12
13     self.pool1 = nn.MaxPool2d(2, stride=2, padding = 0)
14
15
16     self.conv2 = nn.Conv2d(16, 32, 5, padding=2)
17
18     self.conv2_seq = nn.Sequential(self.conv2, nn.BatchNorm2d(num_features = 32),
19                                    nn.ReLU())
20
21     self.pool2 = nn.MaxPool2d(2, stride=2, padding = 0)
22
23
24     self.conv3 = nn.Conv2d(32, 64, 5, padding=2)
25
26     self.conv3_seq = nn.Sequential(self.conv3, nn.BatchNorm2d(num_features = 64),
27                                    nn.ReLU())
28
29     self.pool3 = nn.MaxPool2d(2, stride=2, padding = 0)
30
31
32     self.conv4 = nn.Conv2d(64, 128, 5, padding=2)
33
```

```
    self.conv4_seq = nn.Sequential(self.conv4, nn.BatchNorm2d(num_features = 128),
                                   nn.ReLU())

    self.pool4 = nn.MaxPool2d(2, stride=2, padding = 0)

    self.conv5 = nn.Conv2d(128, 256, 5, padding=2)

    self.conv5_seq = nn.Sequential(self.conv5, nn.BatchNorm2d(num_features = 256),
                                   nn.ReLU())


    self.base_network = nn.Sequential(self.conv1_seq, self.pool1, self.conv2_seq,
                                      self.pool2, self.conv3_seq, self.pool3,
                                      self.conv4_seq, self.pool4, self.conv5_seq)

  def forward(self, X):
    return self.base_network.forward(X)


class Intermediate_Network(torch.nn.Module):
  def __init__(self, base_model):
    super().__init__()
    self.base_model = base_model
    self.layer1 = nn.Conv2d(256, 256, 3, padding=1)
    self.layer1_seq = nn.Sequential(self.layer1, nn.BatchNorm2d(num_features = 256),
                                   nn.ReLU())

  @staticmethod
  def from_static_init(base_model, layer1_seq):
    model = Intermediate_Network(base_model)
    model.layer1_seq = layer1_seq
    return model

  def forward(self, X):
    base_output = self.base_model.forward(X)
    return self.layer1_seq.forward(base_output)


class ClassificationNetwork(torch.nn.Module):
  def __init__(self, intermediate_model):
    super().__init__()
    self.intermediate_model = intermediate_model
    self.layer2 = nn.Conv2d(256, 1, 1)
    self.sigmoid = nn.Sigmoid()
    self.classification_network = nn.Sequential(self.layer2, self.sigmoid)

  @staticmethod
  def from_static_init(intermediate_model, classification_network):
    model = ClassificationNetwork(intermediate_model)
    model.classification_network = classification_network
    return model

  def forward(self, X):
    intermediate_output = self.intermediate_model.forward(X)
    return self.classification_network.forward(intermediate_output)


class RegressionNetwork(torch.nn.Module):
  def __init__(self, intermediate_model):
    super().__init__()
    self.intermediate_model = intermediate_model
    self.layer2 = nn.Conv2d(256, 4, 1)
    self.regression_network = nn.Sequential(self.layer2)

  @staticmethod
  def from_static_init(intermediate_model, regression_network):
    model = RegressionNetwork(intermediate_model)
    model.regression_network = regression_network
```

```
    return model

  def forward(self, X):
    intermediate_output = self.intermediate_model.forward(X)
    return self.regression_network.forward(intermediate_output)



## Sample Initialization.
base_model = BaseNetwork()
intermediate_model = Intermediate_Network(base_model)

c_model = ClassificationNetwork(intermediate_model)
r_model = RegressionNetwork(intermediate_model)
```

2 Histogram of the area and aspect ratios

Scale: Scale is termed as area.

Mode changes if the number of bins are changed. Mean and variance of bins are analysed to get the perspective divergence of bin.

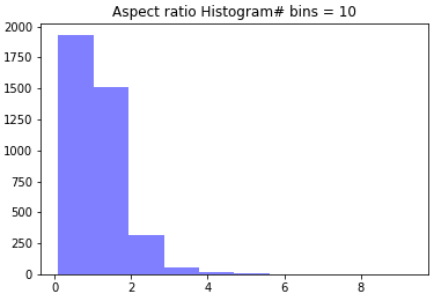
Aspect ratio

Experimented with 10 different bin sizes.

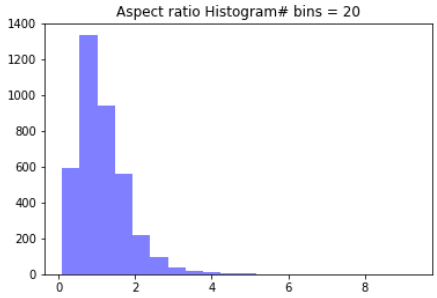
Mean and Variance of Mode bins:

Bin Size	Bin Number	Bin Count	Mean	Variance	Percentage
10	1	1931	0.6686	0.0397	50.2%
20	2	1339	0.7753	0.016	34.84%
30	3	858	0.855	0.0075	22.32%
40	3	715	0.670	0.0044	18.6%
50	4	555	0.739	0.0027	14.44%
60	4	481	0.6329	0.0019	12.51%
70	5	411	0.6856	0.0014	10.69%
80	6	359	0.7276	0.0011	9.34%
90	6	321	0.658	0.0008	8.35%
100	6	293	0.602	0.0007	7.62%

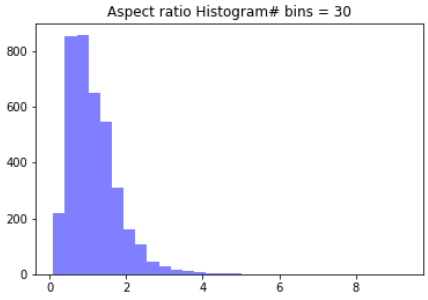
Plots:



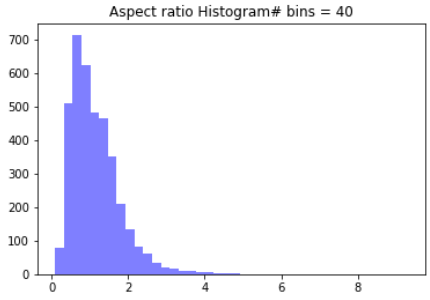
10 bins



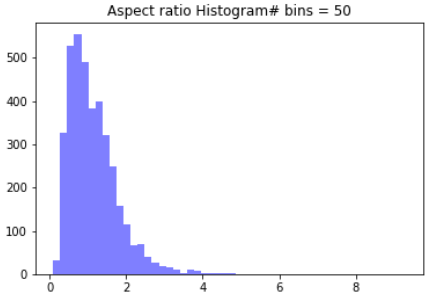
20 bins



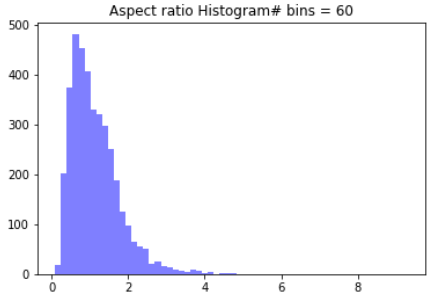
30 bins



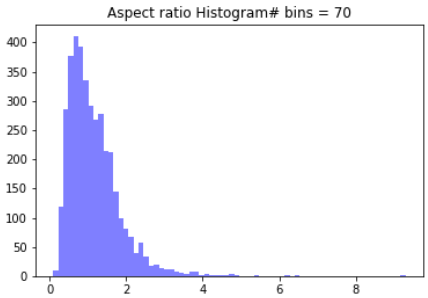
40 bins



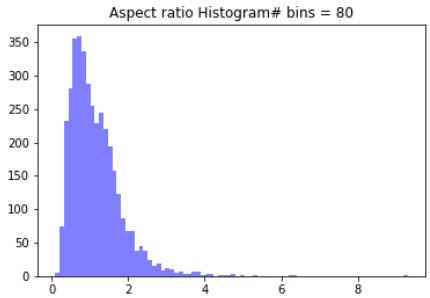
50 bins



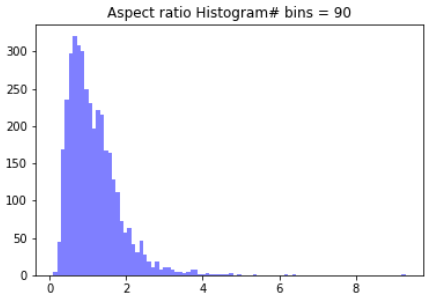
60 bins



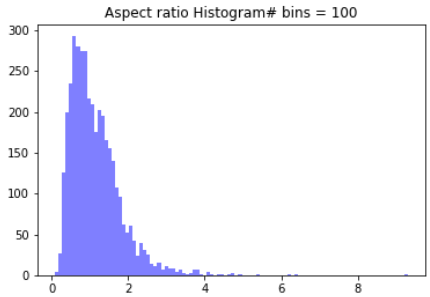
70 bins



80 bins



90 bins



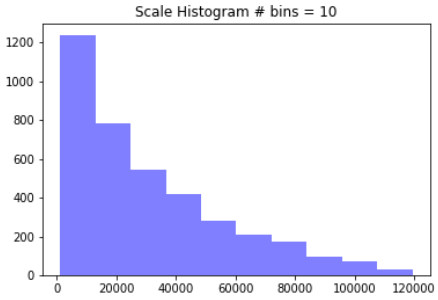
100 bins

Area bins

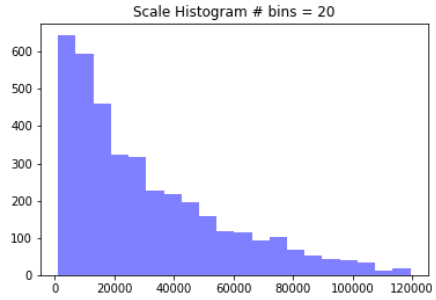
Mean and Variance of Mode bins:

Bin Size	Bin Number	Bin Count	Mean	Variance	Percentage
10	1	1238	6847	10768083	32.21%
20	1	644	4107	2476964	16.75%
30	1	440	3221	1003043	11.44%
40	1	325	2781	594366	8.45%
50	2	276	4541	464472	7.18%
60	2	249	3970	284233	6.47%
70	2	227	3622	231787	5.9%
80	2	193	3323	178471	5.02%
90	2	168	3064	142375	4.37%
100	3	163	4049	114948	4.24%

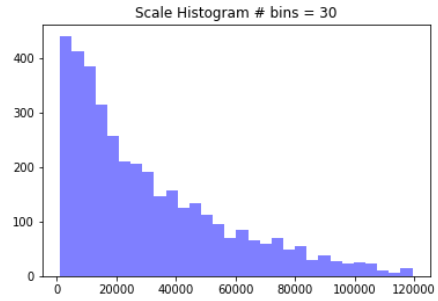
Plots:



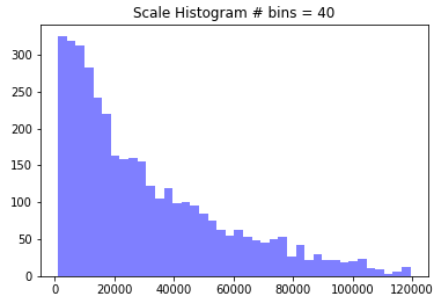
10 bins



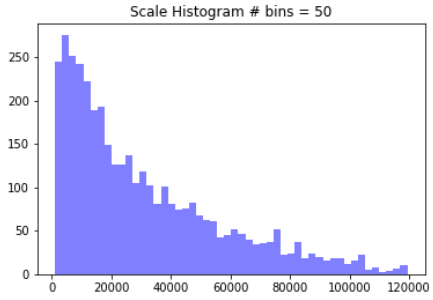
20 bins



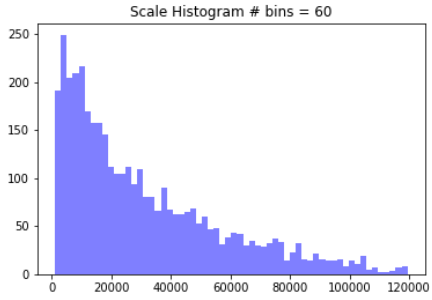
30 bins



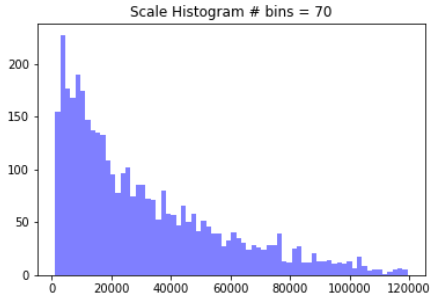
40 bins



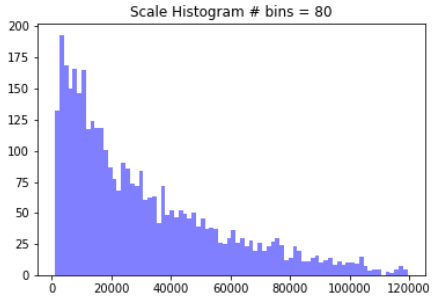
50 bins



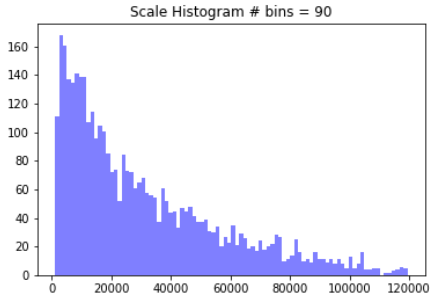
60 bins



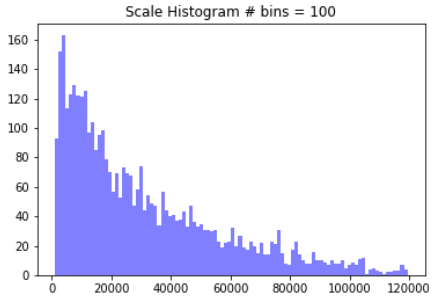
70 bins



80 bins



90 bins



100 bins

Mode Values

Chosen mode values for

Area	$4107 \sim 64^2$
Aspect Ratio	0.66

Effects

Pros of choosing mode

- If only one anchor is used, Single mode helps us to generalize max number of samples. Mode helps us to repair some of the effects of single anchor.
- Even with multiple anchors. we can choose the multiple max bins and use those scale values.

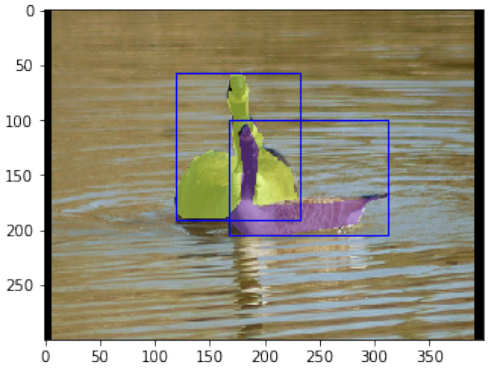
Cons:

- If we look at the percentage from the above tables, Mode only covers up-to 16.75% for area and 18.6% for aspect ratio. Rest of the ranges will be ignored.
- Also the variance in mode bin is to be considered. If the bin has high variance, then mode doesn't help. (Inspect histograms with different bin sizes, to choose the mode wisely).
- If the test data is not known prior, then these generalizations will not hold and will result in poor accuracies.
- Prefer using multiple anchors over mode if possible.

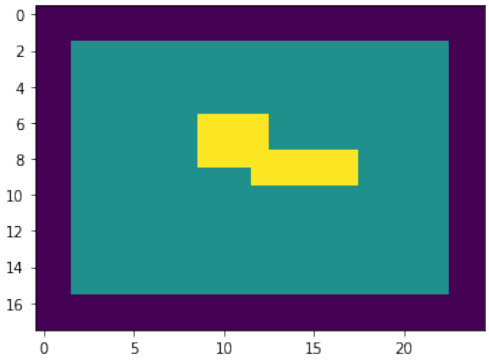
3 Ground truth for RPN training

Feature Vector Color Code : Yellow - positive boxes, Green - Negative boxes, Purple - Ignore boxes.

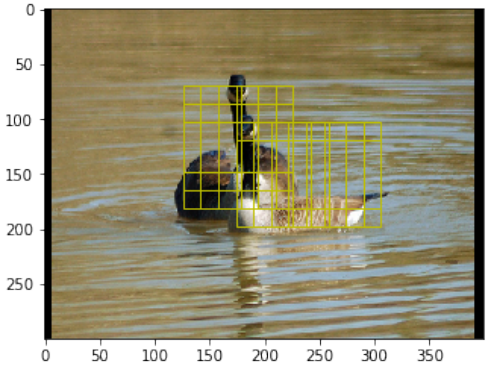
Sample 1



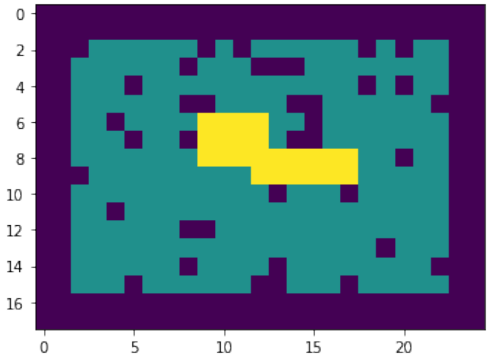
Ground Truth



Feature vector

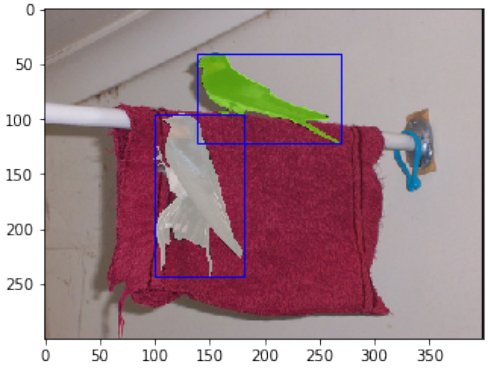


Allocated Anchor boxes

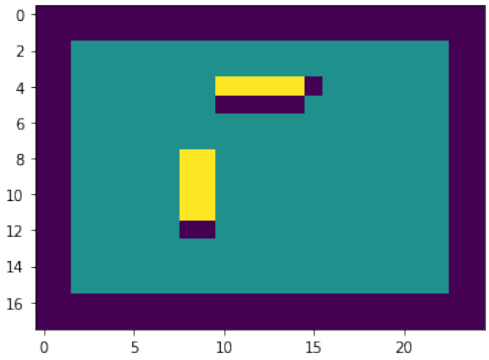


Sampling 256 anchors

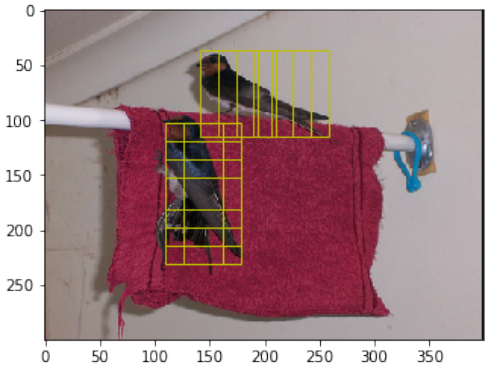
Sample 2



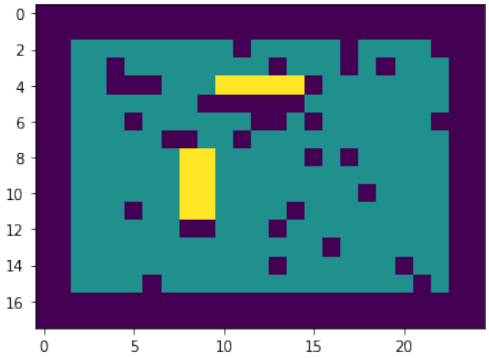
Ground Truth



Feature vector



Allocated Anchor boxes



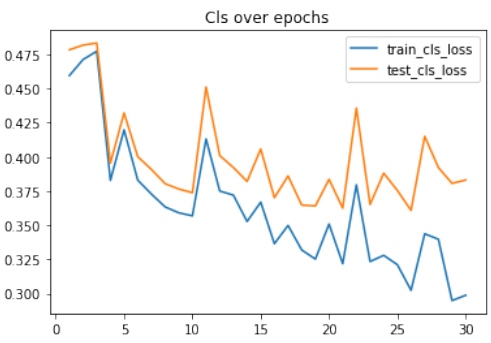
Sampling 256 anchors

4 RPN classification branch

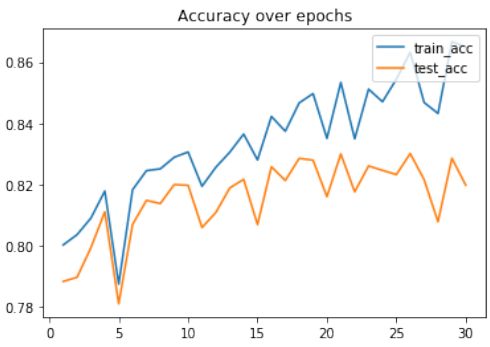
Sampling strategy: Sampled 256 anchors for each image.

Plots

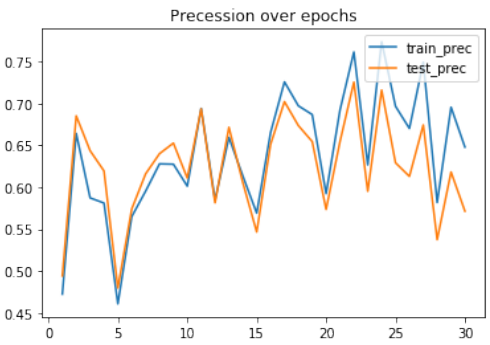
Batch size: 30, **LR:** 0.001. Each image in the batch has equal number of positive and negative anchors. From each image around 256 anchors are sampled.



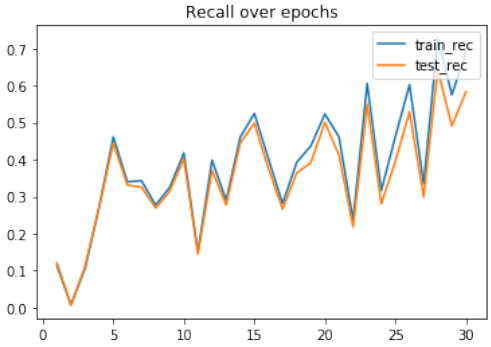
Classifier loss function



Accuracy over epochs

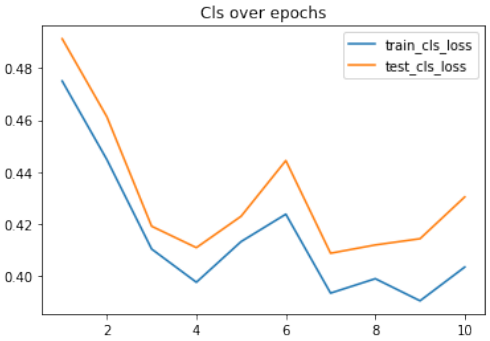


Precession over epochs

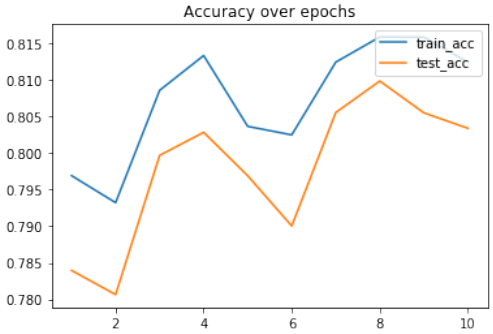


Recall over epochs

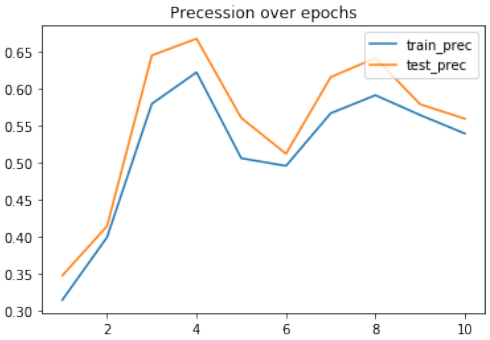
Batch size: 1, **LR:** 0.00001. Each image has equal number of positive and negative anchors. From each image around 100 anchors are sampled.



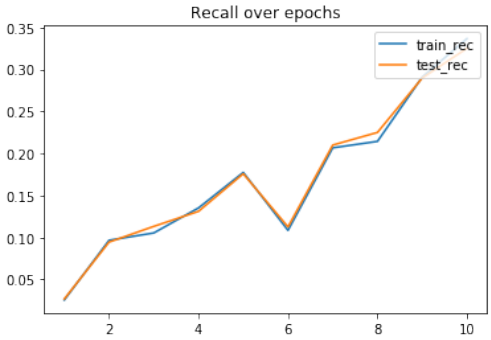
Classifier loss function



Accuracy over epochs



Precession over epochs



Recall over epochs

5 Receptive field

Logic for calculating receptive field.

- The logic assumes that the filter is symmetrical.
- Each layer (CONV, POOL) can be represented using 2 values (K_t, S_t). where K_t is the kernel size, S_t is the stride.
- Let ES_t be the effective stride after t layers. R_t be the receptive field after t layers.

- Relation:
  - $ES_t = S_t * ES_{t-1}$
  - $R_t = (K_t - 1) * ES_{t-1} + R_{t-1}$,
- Base case:
  - $R_0 = 1$, $ES_0 = 1$

Base Network Receptive Calculation

Layer Number	Layer Type	Kernel Size	Stride	Effective Stride	Receptive field
1	CONV	5x5	1	1	5x5
2	POOL	2x2	2	2	6x6
3	CONV	5x5	1	2	14x14
4	POOL	2x2	2	4	16x16
5	CONV	5x5	1	4	32x32
6	POOL	2x2	2	8	36x36
7	CONV	5x5	1	8	68x68
8	POOL	2x2	2	16	76x76
9	CONV	5x5	1	16	140x140

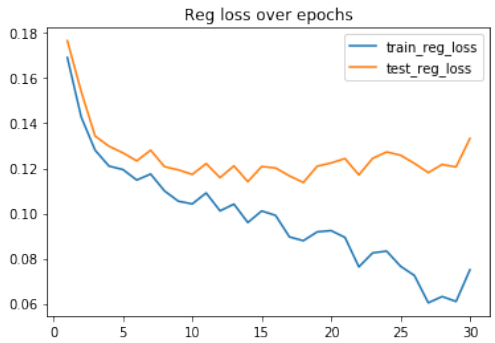
Intermediate Network Receptive Calculation

Layer Number	Layer Type	Kernel Size	Stride	Effective Stride	Receptive field
1	CONV	3x3	1	16	172x172

The anchor box should be inside of the receptive field (Size of anchor box should be less than receptive filed both in height and width). If the receptive field is smaller, then the network will not have enough information(responses is limited to only receptive field) to predict boxes (which are larger than the receptive field).

6 RPN regression branch

Batch size: 30, **LR:** 0.001.



Regression loss over epochs

Batch size: 1, **LR:** 0.00001.

Regression loss over epochs

Note: For only regression training, all anchors are used (sampling is not done).

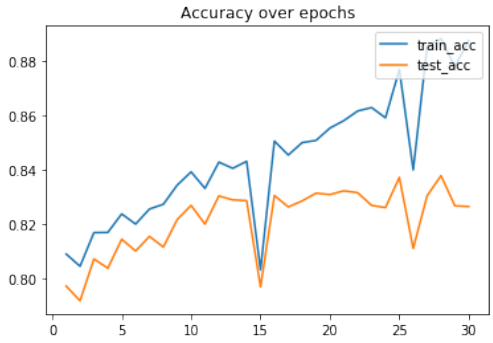
7 Combined Training

Sampling strategy: Sampled 256 anchors for each image.

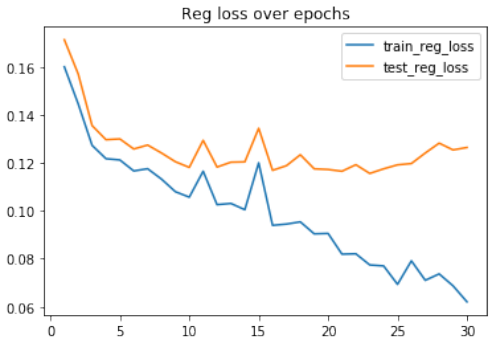
Plots

Batch size: 30, **LR:** 0.001. Each image in the batch has equal number of positive and negative anchors. From each image around 256 anchors are sampled.

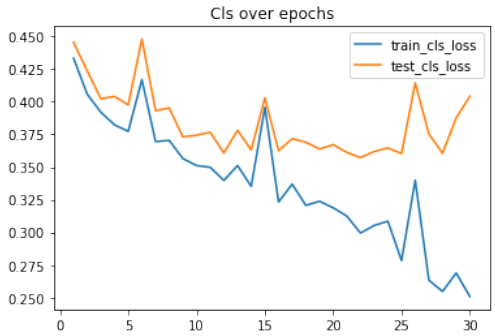
Total loss function



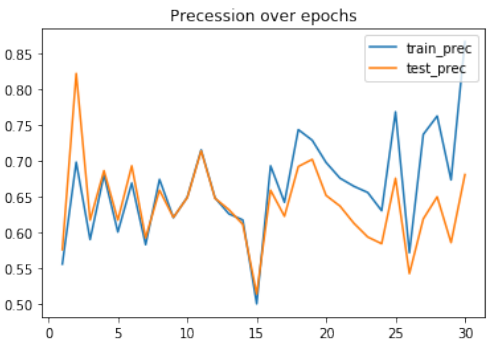
Point-wise Accuracy over epochs



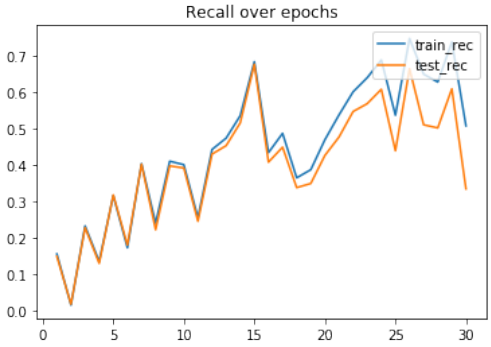
Regression loss over epochs



Classifier loss over epochs

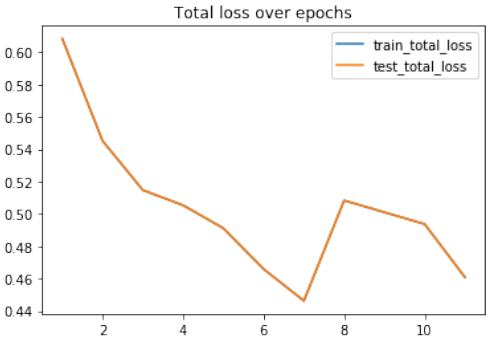


Precession over epochs

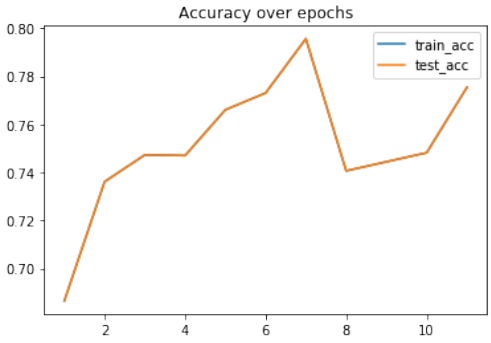


Recall loss over epochs

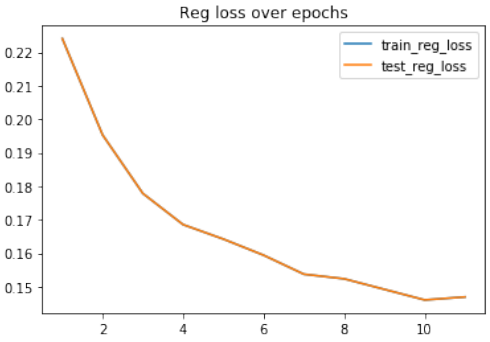
Batch size: 1, **LR:** 0.000001. Each image has equal number of positive and negative anchors. From each image around 100 anchors are sampled.



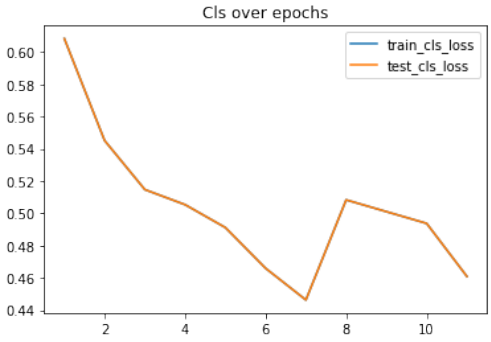
Total loss function



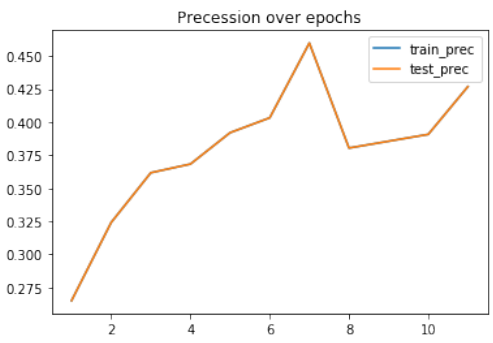
Point-wise Accuracy over epochs



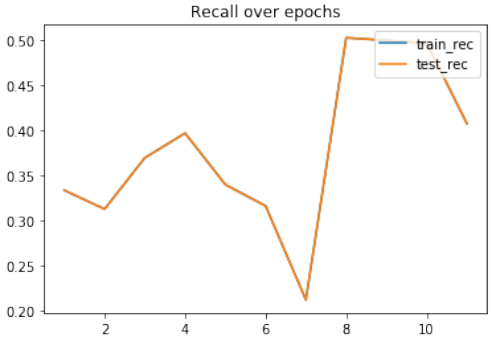
Regression loss over epochs



Classifier loss over epochs



Precession over epochs

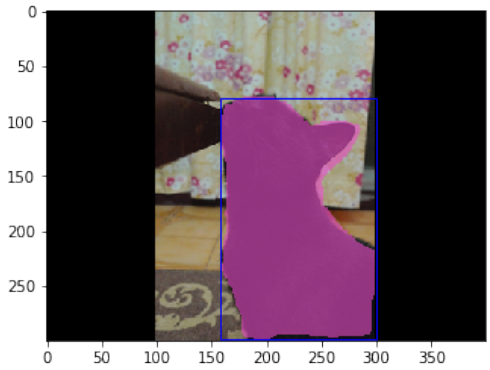


Recall loss over epochs

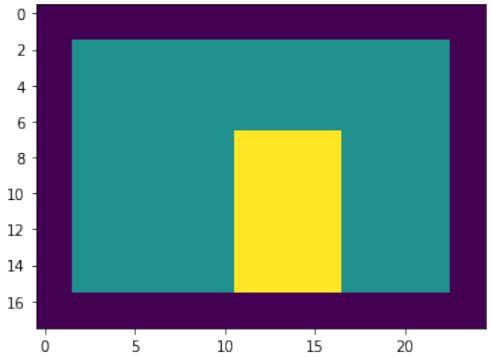
Visualizing test images

Color scheme: For prediction, Yellow boxes are predicted positive boxes. Red are predicted negative boxes.

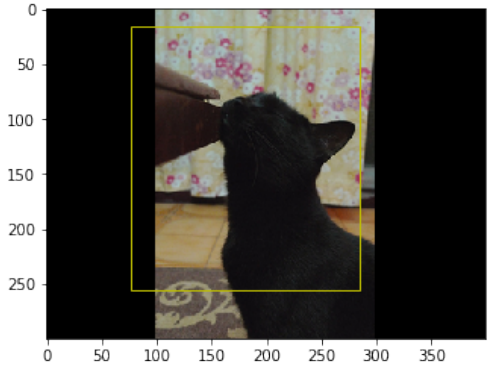
Sample 1



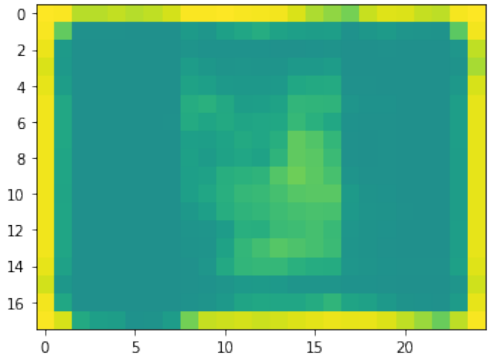
Ground truth boxes



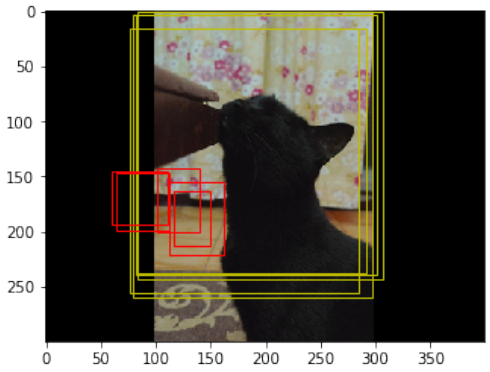
Ground truth feature vector(objectness)



Prediction with NMS

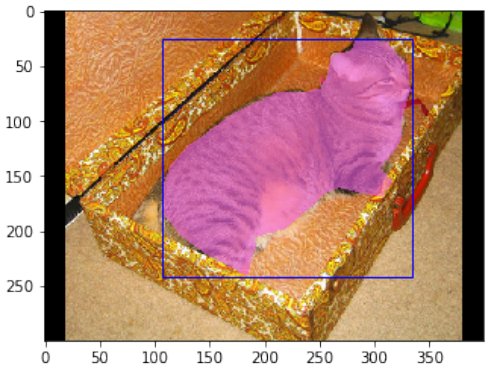


Prediction feature vector(objectness)

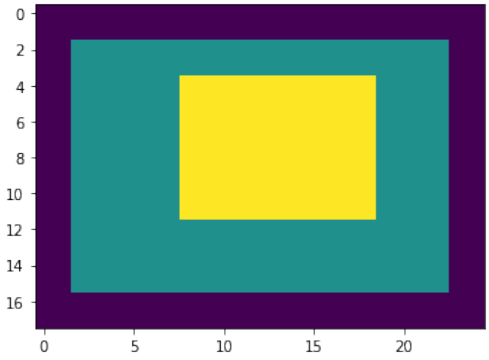


Prediction with negative boxes

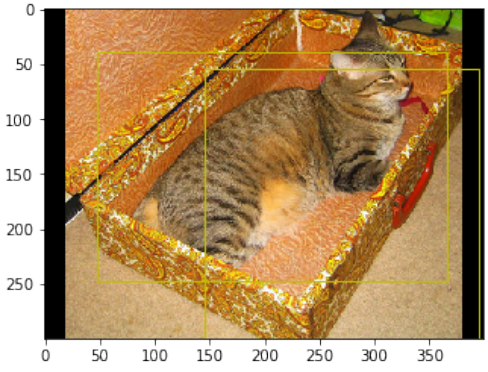
Sample 2



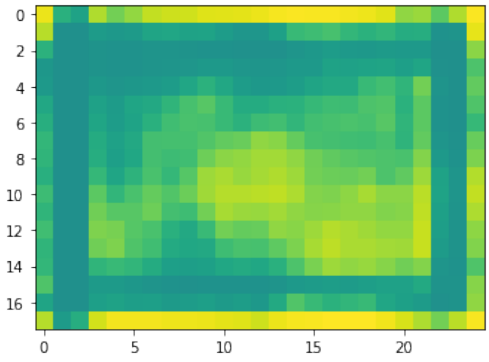
Ground truth boxes



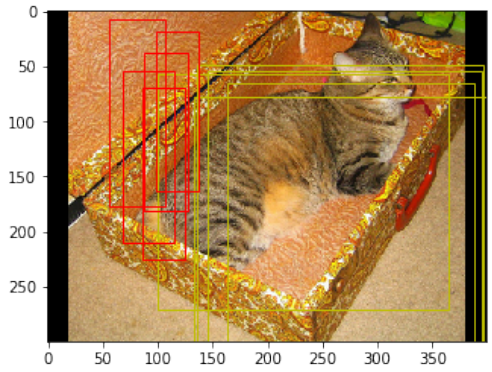
Ground truth feature vector(objectness)



Prediction with NMS

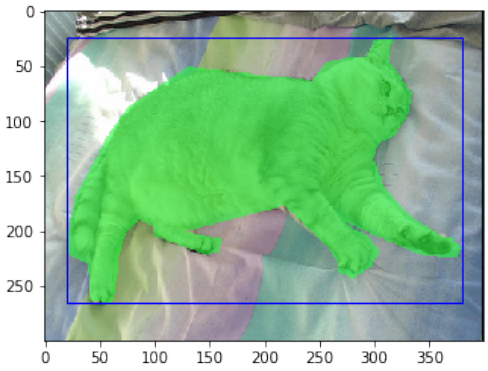


Prediction feature vector(objectness)

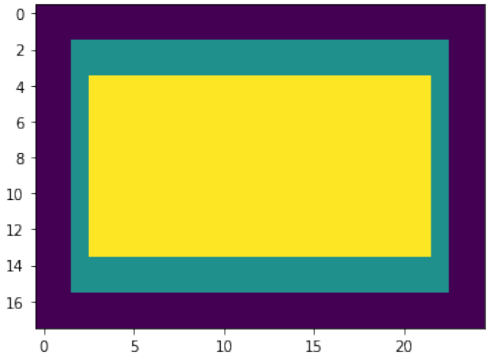


Prediction with negative boxes

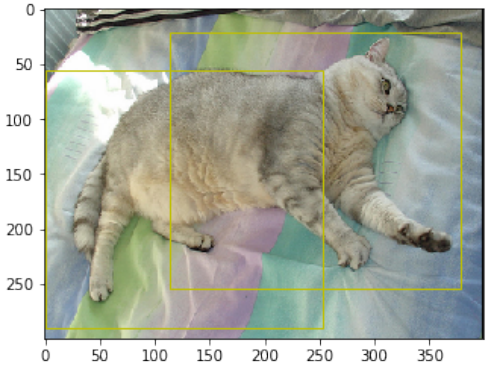
Sample 3



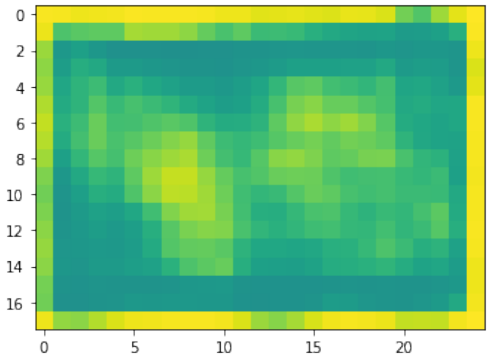
Ground truth boxes



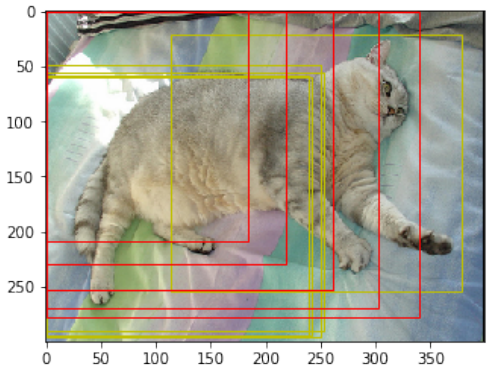
Ground truth feature vector(objectness)



Prediction with NMS

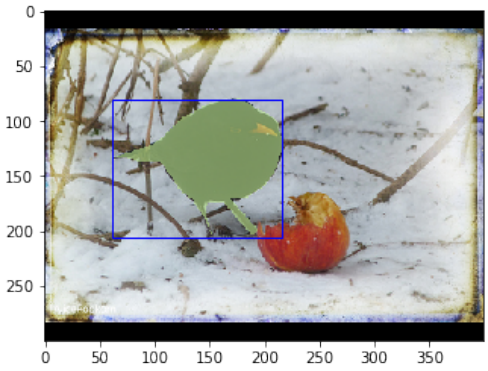


Prediction feature vector(objectness)

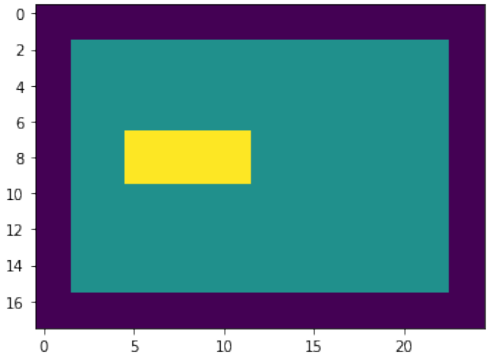


Prediction with negative boxes

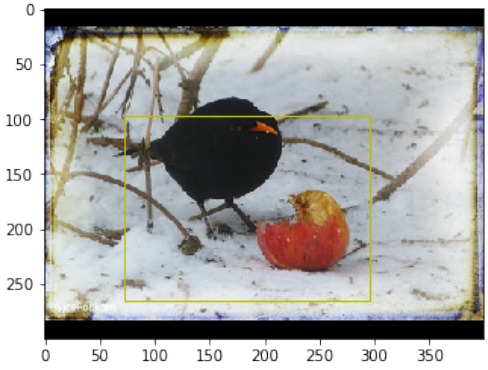
Sample 4



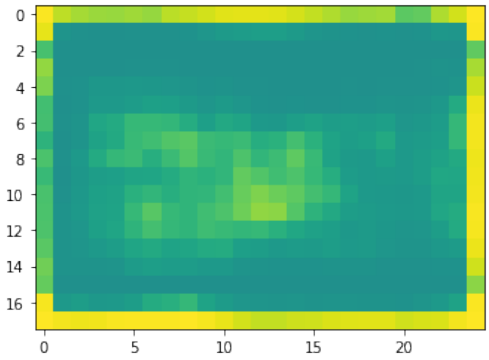
Ground truth boxes



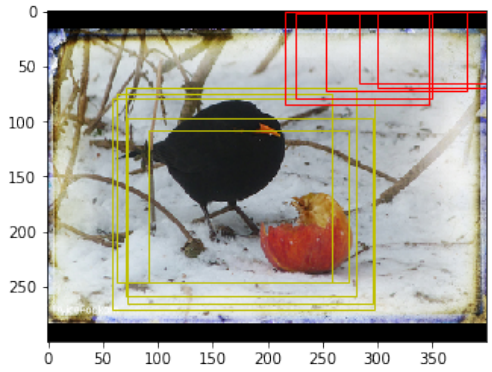
Ground truth feature vector(objectness)



Prediction with NMS



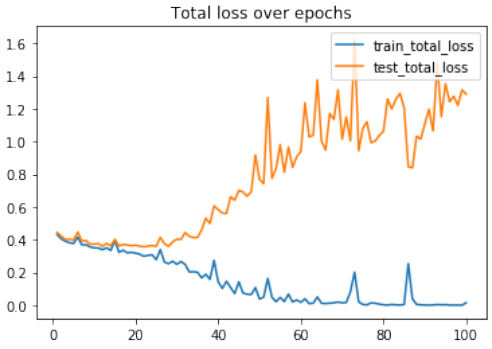
Prediction feature vector(objectness)



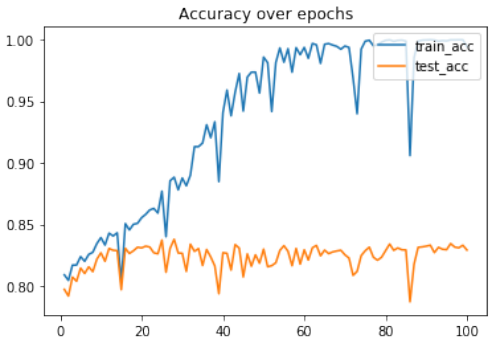
Prediction with negative boxes

Over-fitting example

Training for long epochs, resulted in over-fitting. After 35 epochs, the test loss stopped decreasing and started increasing.

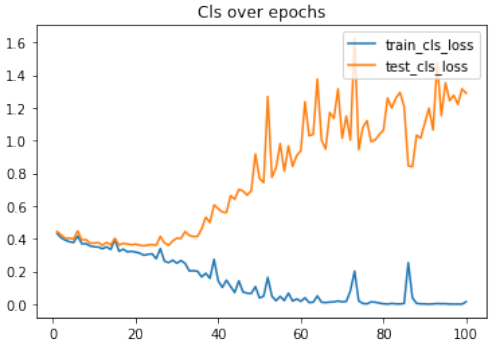


Total loss function

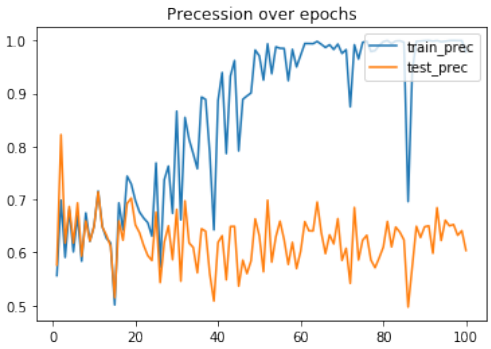


Point-wise Accuracy over epochs

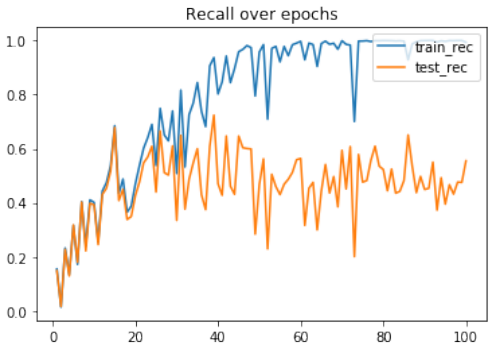
Regression loss over epochs



Classifier loss over epochs



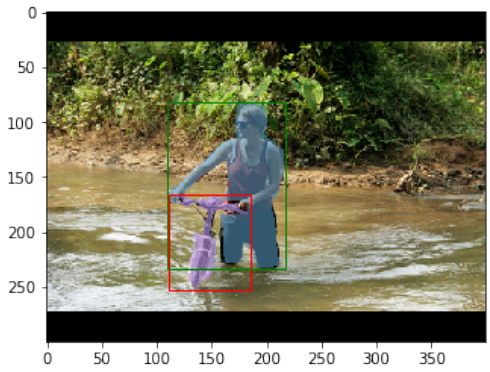
Precession over epochs



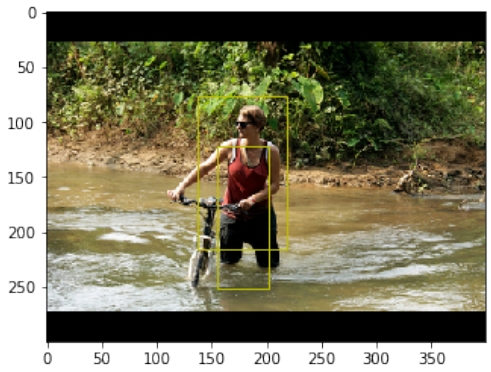
Recall loss over epochs

8 Decoding

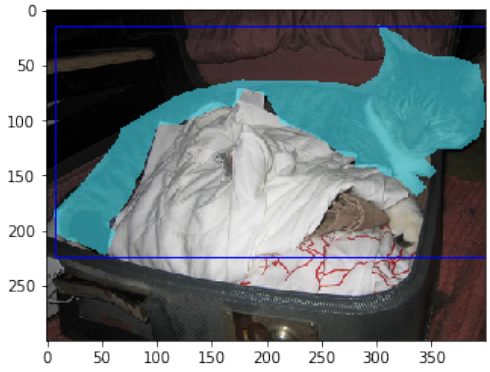
Hyper parameter = 2 for the following decoding.



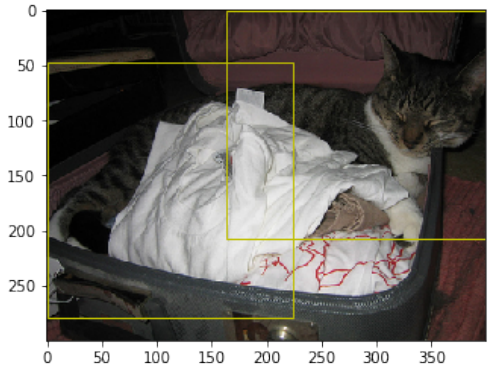
Ground truth



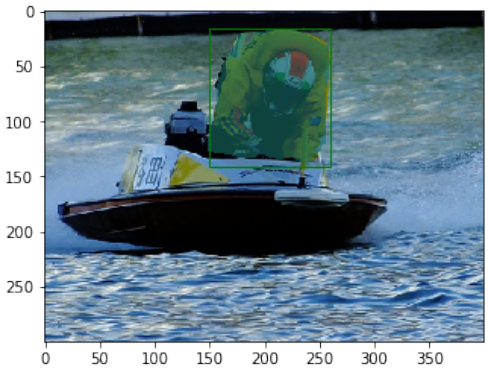
Prediction



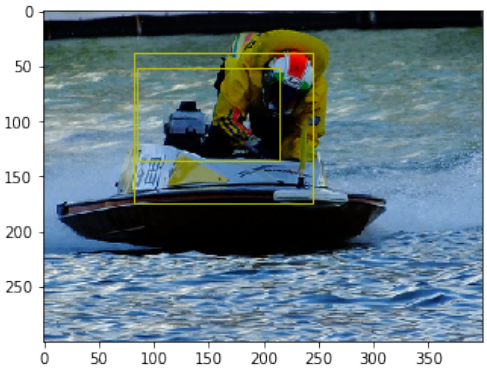
Ground truth



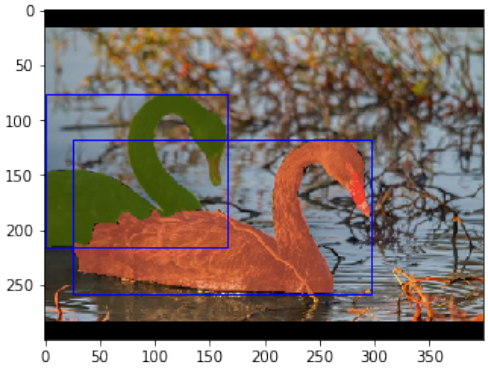
Prediction



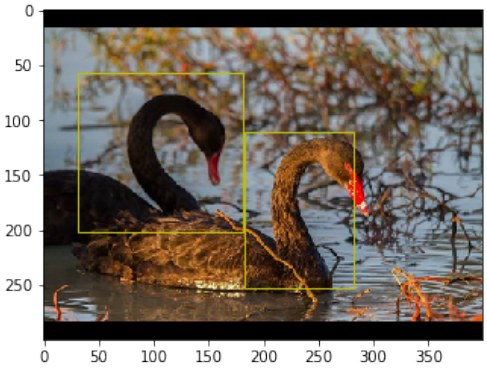
Ground truth



Prediction



Ground truth



Prediction