Image Manipulation Detection with R-CNN

Hongyi Zheng, Paul Zhu

Executive Summary

Problem:

- Detect the objects in an image, and determine whether this object is manipulated or not

Solution approach:

- R-CNN Network

Benefits:

- One model, multiple objectives:
  - Locate the object
  - Determine whether the object is manipulated
- Help people better distinguish modified photos from original ones

Problem Motivation

- Increasing usage of photoshop tools and image manipulation techniques
- Manipulated images can lead to misinformation and adversarial attacks
- A very useful application of convolutional neural network

Background Work

Zhou, Peng, et al. “Learning Rich Features for Image Manipulation Detection.”

- Offer the approach of implementing Faster R-CNN for Manipulation Detection

https://github.com/pengzhou1108/RGB-N

- Creating the COCO synthetic dataset (based on COCO 2014) for training, validation, and test

Writing Custom Datasets, Dataloaders and Transformers

- Instructions for customizing Dataloaders for the COCO synthetic dataset

J;, Ren S;He K;Girshick R;Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.”

- Detailed explanation of RPN and R-CNN

Object_detection_hw.ipynb from NYU CSCI-GA.2271

- Basic framework. We gave it a complete overhaul to fit our objective

Technical Challenges & Innovations

- Datasets consists of images from two separate sources (COCO 2014 & COCO Synthetic), with standalone annotation files
  - Define custom dataset
- The bounding boxes needs to be rescaled along with the image, which means both targets and labels have to be transformed.
  - Define custom transformation function
- Only a small proportion of the images in COCO 2014 dataset is used, but the COCO 2014 dataset itself is very large
  - Write image clean-up scripts to reduce the dataset size to ensure they could be uploaded to HPC server
- DataLoader is very slow
  - Apply transformations during dataset initialization: 5x faster loading

Technical Challenges & Innovations

- Imbalanced anchor classes (most anchors does not contain object)
  - Implementation of Online Hard Example Mining
- Different from traditional R-CNN: besides of predicting whether there is an object inside the anchor, this network also needs to classify whether the contained object is tampered.
  - Use multi-class classification instead of binary classification
    - 0 - No object in the anchor
    - 1- Tampered object in the anchor
    - 2 - Authentic object in the anchor
- High model complexity
  - Use powerful GPUs on NYU HPC
  - Use parallel training and speed-up techniques

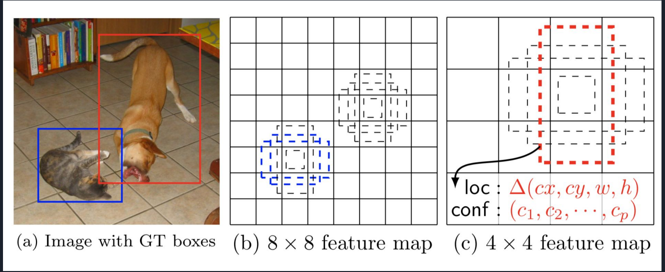
Approach

- Rescale image, convert image to tensor and put it into the ResNet18 encoder

- Implementing the Regional Proposal Network: Extract the 3rd output feature map of size 8 x 8 from the ResNet18 encoder and apply a 3x3 convolutional layer on top of it. Then, apply two sibling 1x1 convolutional layers for classification and regression respectively.

Approach - Regression Head Target

For each anchor, determine the closest true bounding box by selecting the max IoU score between this anchor and each ground-truth bounding box. If the max IoU score for an anchor is greater than a certain threshold (in our case, 0.7), then we say this anchor actually contains an object. We then calculate the transformation parameters between this anchor and the true bounding box.


loc : $\Delta(cx, cy, w, h)$
conf : $(c_1, c_2, \cdots, c_p)$
(a) Image with GT boxes
(b) 8×8 feature map
(c) 4×4 feature map

The four parameters are:

- Difference in x and y-coordinate between the center of the anchor and the center of its associated closest true bounding box.
- The width ratio between anchor and its associated closest true bounding box. (log scale)
- The height ratio between anchor and its associated closest true bounding box. (log scale)

This four-element vector is the target value of the regression head, it describes how an anchor should be transformed to its closest bounding box. Note that this value is only computed when the anchor contains an object. If it doesn't contain an object, the regression loss would not be computed and included in the back-propagation.

Approach - Classification Head Target

If the max IoU score for an anchor is greater than the threshold mentioned in the previous slide, then this anchor actually contains an object, so the classification target value for this anchor would be **either 1 or 2, depending on whether the object in its closest true bounding box is manipulated or original (The innovative part of our project!)**.

Moreover, if the max IoU score for that anchor is less than another threshold, (in our case 0.3), then we say this anchor certainly does not contain any object. The classification target value for this value would be 0.

If the max IoU score is between the two thresholds, then we say this anchor is a bad anchor. It will not have a classification label and its classification loss would not be computed and included in the back-propagation.

Approach - Back Propagation

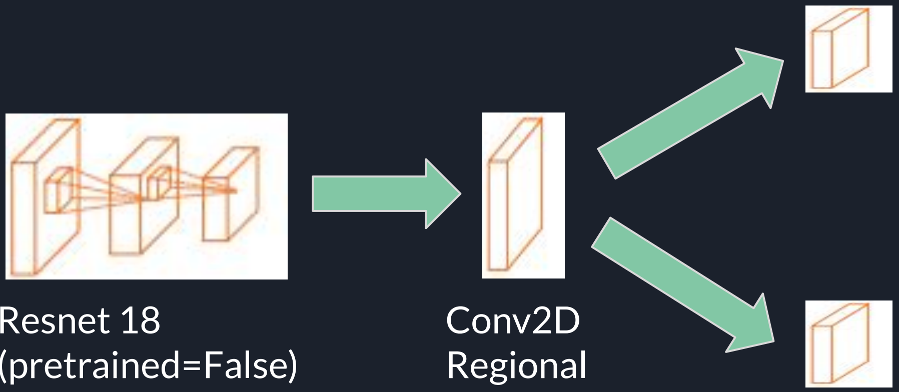
- For the regression head, compute smooth L1 loss for the output.
- For the classification head, compute cross-entropy loss (instead of the binary cross entropy loss in original implementation) since we now have three classes.
- Then for each output head, sum the two losses up and do back-propagation.

Approach - Prediction

During the prediction stage, for each image we got a 3xN array, which are the raw softmax scores for three classes for each of the N anchors. Using softmax function, we could recover the probabilities of each class from raw logit outputs. If the maximum probability is for class 1 (tampered object in anchor) or class 2 (authentic object in anchor), and that maximum probability is greater than a certain threshold (in our case 0.7), then we use its associated regression-head output to transform this anchor and add this transformed bounding box to a list of candidates.

Next, we perform a non-maximum suppression: NMS iteratively removes lower scoring boxes which have an IoU greater than a specific threshold with another (higher scoring) box to get a set of final predictions with limited overlapping.

Solution diagram & architecture



Implementation Details

- Input image size: 128x128
- Number of anchors per image:
  - 70 anchors per position: 10 scales * 7 ratios
  - Since we use 8x8 feature map, the number of anchor positions will be 64. Therefore, there will be 70 * 64 = 4480 proposed anchors in total for each image, and the stride for anchor positions is 128/8 = 16
- Optimizer: SGD optimizer with lr = 0.1 and momentum = 0.9
- IoU thresholds between anchor and true bounding boxes:
  - > 0.7: contains object
  - < 0.3: does not contain object
  - between 0.3 and 0.7: bad anchor
- Prediction probability threshold: 0.7
- IoU threshold between bounding box candidates for NMS: 0.3

Demo & Experiment design flow

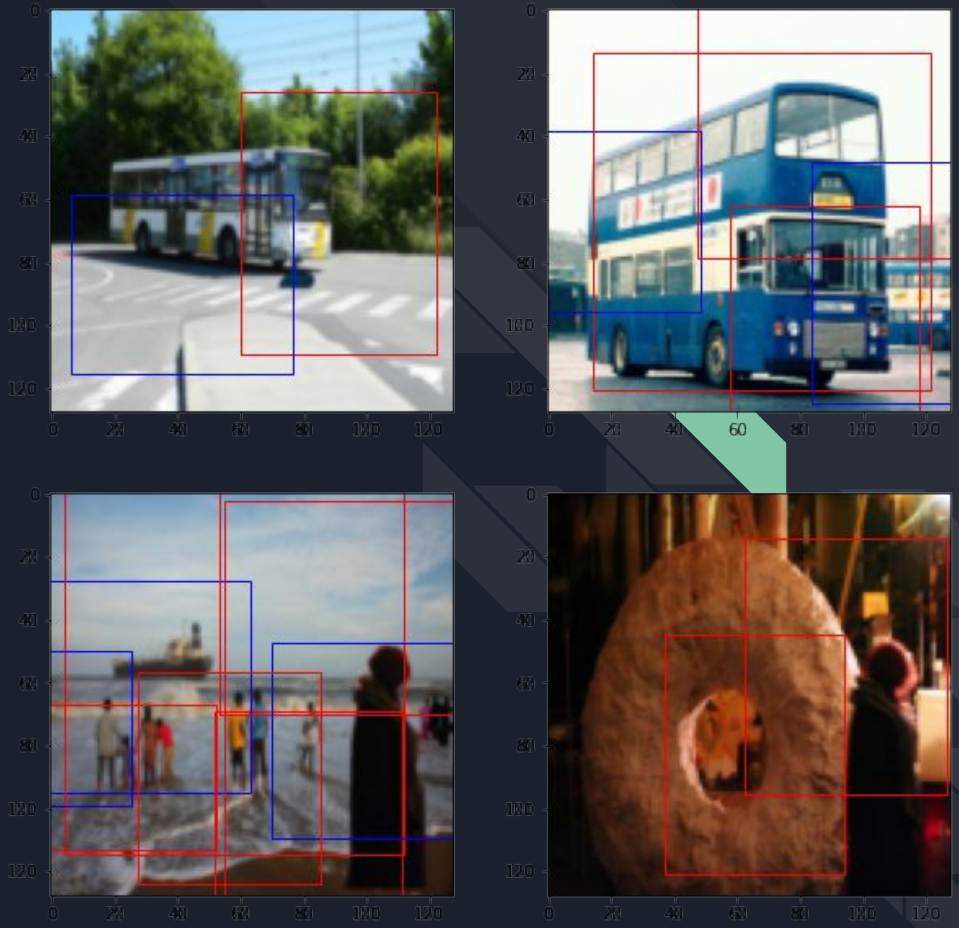
Demo:

- For some example images, visualize the images as well as their ground truth bounding box, as well as whether objects in those boxes are tampered. Then put those images into our model and compare the model prediction and ground truth.

Experimental Evaluation:

- Loss Curve
- Hyperband: Model performance under different set of hyperparameters

Live Demo



Experimental Evaluation - Hyperparameters

Hyperparameter Tuning with Hyperband

Search Space:

- Base ResNet: 18, 34, 50
- Initial Learning Rate: 0.01, 0.1, 1.0
- Pretrained: True, False

We train each models for 40 epochs and decay learning rate with gamma = 0.1 every 15 epochs.

Model with Minimum Test Classification Loss: Pretrained ResNet50 + Initial Learning Rate = 1.0

Model with Minimum Test Regression Loss: Pretrained ResNet34 + Initial Learning Rate = 0.1

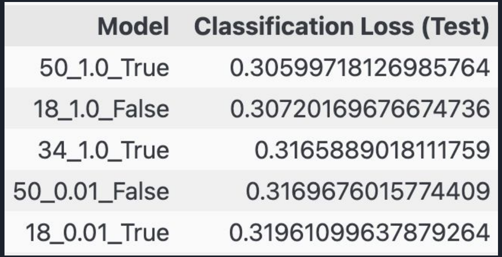
Experimental Evaluation - Observations

Key Observations:

- Complex model generally performs better
- When the model is pretrained and the initial learning rate is high, the model performs badly
- The regression head and the classification head may have different optimal learning rate: instead of simply adding them up like loss = clf_loss + reg_loss, we may achieve better result by searching for the optimal ratio between these two losses

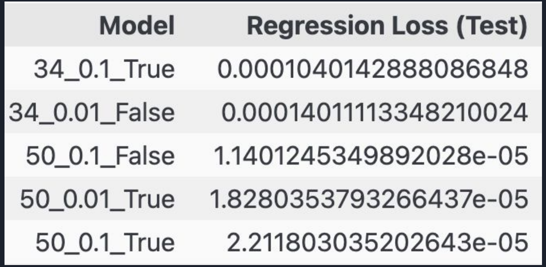
  (i.e. loss = w x clf_loss + reg_loss, search for the best w)

5 Best Classification Loss Values:



Model	Classification Loss (Test)
50_1.0_True	0.30599718126985764
18_1.0_False	0.30720169676674736
34_1.0_True	0.3165889018111759
50_0.01_False	0.3169676015774409
18_0.01_True	0.31961099637879264

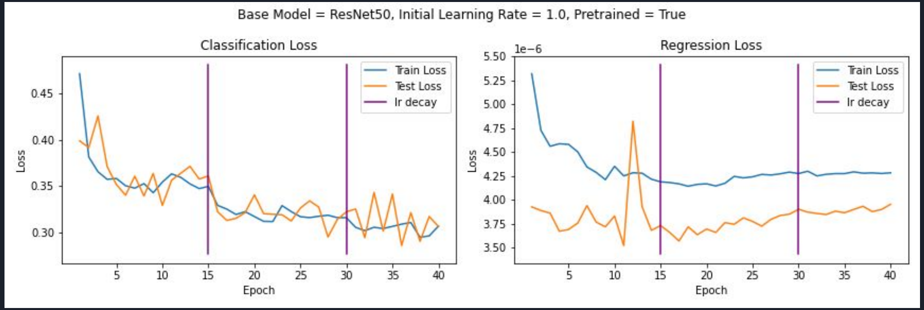
5 Best Regression Loss Values:



Model	Regression Loss (Test)
34_0.1_True	0.0001040142888086848
34_0.01_False	0.00014011113348210024
50_0.1_False	1.1401245349892028e-05
50_0.01_True	1.8280353793266437e-05
50_0.1_True	2.211803035202643e-05

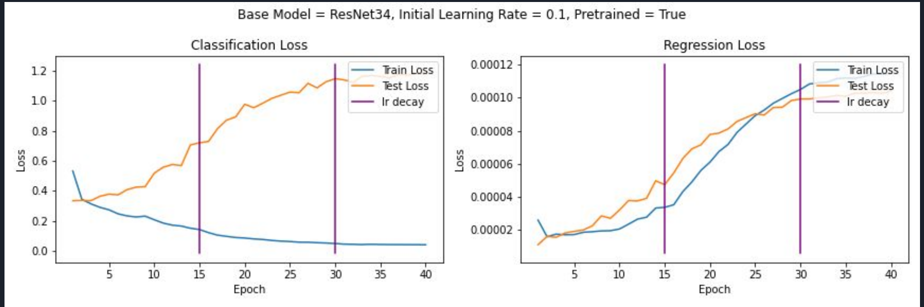
Experimental Evaluation - Loss Curves

Pretrained ResNet50 + Initial Learning Rate = 1.0 (Best Classification Model)



Pretrained ResNet34 + Initial Learning Rate = 0.1 (Best Regression Model)

(This one is probably experiencing overfitting, but still gives the best result)



Conclusion

Advantages:

- Accomplishing object detection and manipulation detection
- Straightforward implementation
- Stable and moderate Performance

Limitations:

- Limited manipulation techniques are considered
- Manual input of anchors
- Regardless of whether the object in the input is tampered or not, there is only one ground truth bounding box in each image while in reality an image could include multiple original and manipulated objects
- Fragile to adversarial attacks
- Still, visually, the predictions are not accurate enough (the reality is ugly)

Possible Improvements:

- Adopt Faster R-CNN for more efficient performance
- Add noise features to detect manipulation
- Create more complex synthetic datasets

Github Repository

Thanks!