Week 10

What has been done this week

I have mainly worked on writing the report + setting up scripts to retrain the detection networks for accurate comparison.

- [ ] Report
  - [x] MobileNet
  - [x] ResNet
  - [ ] SSD
  - [ ] IOU
  - [x] Faster RCNN
  - [ ] Baseball
  - [ ] Golf Club Terminology
  - [ ] SVHN trained network
  - [ ] Tesseract
  - [ ] Docker
- [x] Read MobileNet Paper
- [x] Read ResNet Paper
- [x] Read Faster RCNN Paper
- [x] Retrain Detection step **TODAY** NOTE TO SELF: USE TF 1.10.0
  - [x] Create a training_old.tfrecord of the original data
  - [x] Proper folder structure
    - [x] pretrained models
    - [x] experiments
  - [x] rcnn_resnet50_init
  - [x] rcnn_resnet50_alldata
  - [x] rcnn_resnet50_50proposals
  - [x] rcnn_resnet50_10proposals
  - [x] ssd_mobilenetv1_alldata
- [x] Script for calculating IOU Metric on a validation set

```
python evaluation_detection.py --model_path=models/research/object_detection/OUTGRAPH_mobilenet_v1_coco
python evaluation_detection.py --model_path=models/research/object_detection/OUTGRAPH_mobilenet_v1_coco_21000_EPOCHS
python evaluation_detection.py --model_path=models/research/object_detection/OUTGRAPH_rcnn_resnet50_cpkt10008
python evaluation_detection.py --model_path=models/research/object_detection/OUTGRAPH_rcnn_resnet50_fullvid
python evaluation_detection.py --model_path=models/research/object_detection/OUTGRAPH_rcnn_resnet50_reduced_alldata
```

"OLD Models":

Model	IOU	FPS	Complete Miss (IOU = 0)
OUTGRAPH_rcnn_resnet50_cpkt10008	0.582	2.48	26.8%
OUTGRAPH_rcnn_resnet50_fullvid	0.768	2.59	8.7%
OUTGRAPH_mobilenet_v1_coco_21000_EPOCHS	0.228	**25.40**	70.7%
OUTGRAPH_rcnn_resnet50_reduced_alldata	**0.788**	6.95	**3.7**%

"Retrained Models": (SSD is quite slow on the cloud for some reason...)

Model	Test IOU	Train IOU	FPS	Complete Miss (IOU = 0)
rcnn_resnet50_init	0.666	0.726	2.56	18.9%
rcnn_resnet50_alldata	**0.786**	**0.814**	2.55	**4.3**%
rcnn_resnet50_50proposals_alldata	0.756	0.774	5.61	7.1%
rcnn_resnet50_10proposals_alldata	0.672	0.724	6.99	18.9%
ssd_movilenetv1_alldata	0.401	0.407	**10.86**(?)	47%

Literature

Below notes have been written into the report in a more complete version.

Resnet: Deep Residual Learning for Image Recognition https://arxiv.org/abs/1512.03385

In short: We experience problems with the training error becomming larger for deep networks. In reality, we should be able to make a network deeper by simply adding "identity" mapping layers. I.e layers that just pass the output without doing anything without a hit to training performance. This is however not the case.

Vanishing gradients are are a problem for deep networks which in parts is combatted with batch normalization. However, the idea from the authors of Residual Networks is to learn the residual instead of the mapping. I.e, instead of learning f in $y = f(x)$, which can be hard when we initialize the weights from zero, we try to just learn the error/residual f in $y = x + f(x)$. I.e the difference between the identity and the target. This has shown to help greatly in learning. The problem should be the same, since we should be able to approximate the target in either case, however the latter seems to be easier to optimize. If the identity mapping is indeed the best for a layer, the weights will be turned to 0, else they will be tuned to correct the error.

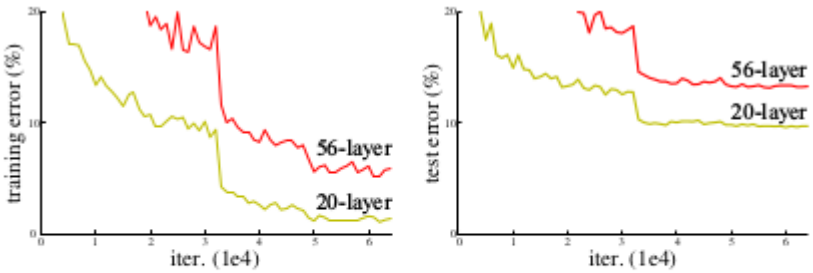


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

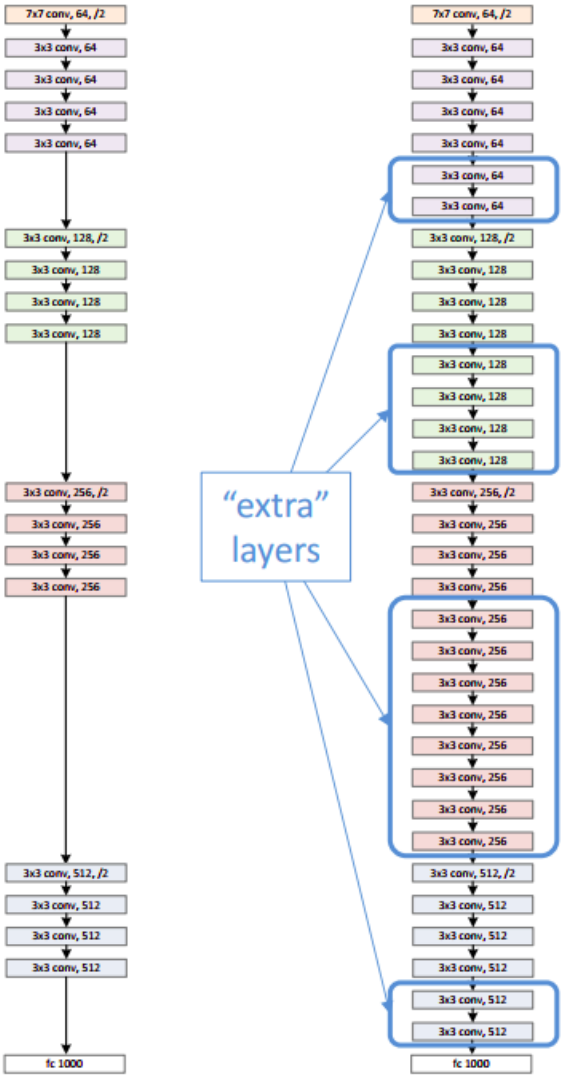
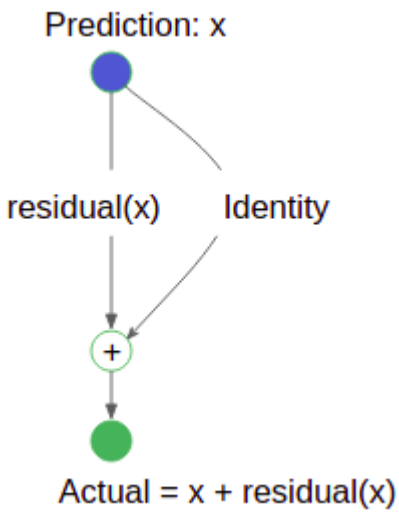


Fig: A Shallow network (left) and a deeper network (right) constructed by taking the layers of the shallow network and adding identity layers between them. Img Credit: Kaiming He.



In the diagram, `x` is our prediction and we want it to be equal to the `Actual`. However, if is it off by a margin, our residual function `residual()` will kick in and produce the residual of the operation so as to correct our prediction to match the actual. If `x == Actual`, `residual(x)` will be `0`. The `Identity` function just copies `x`.

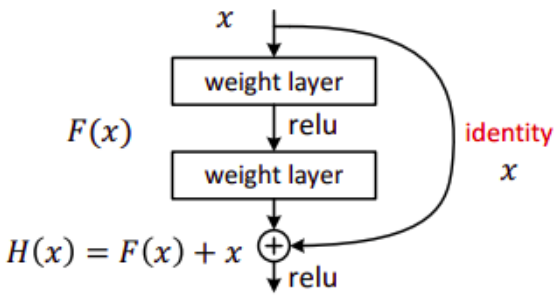
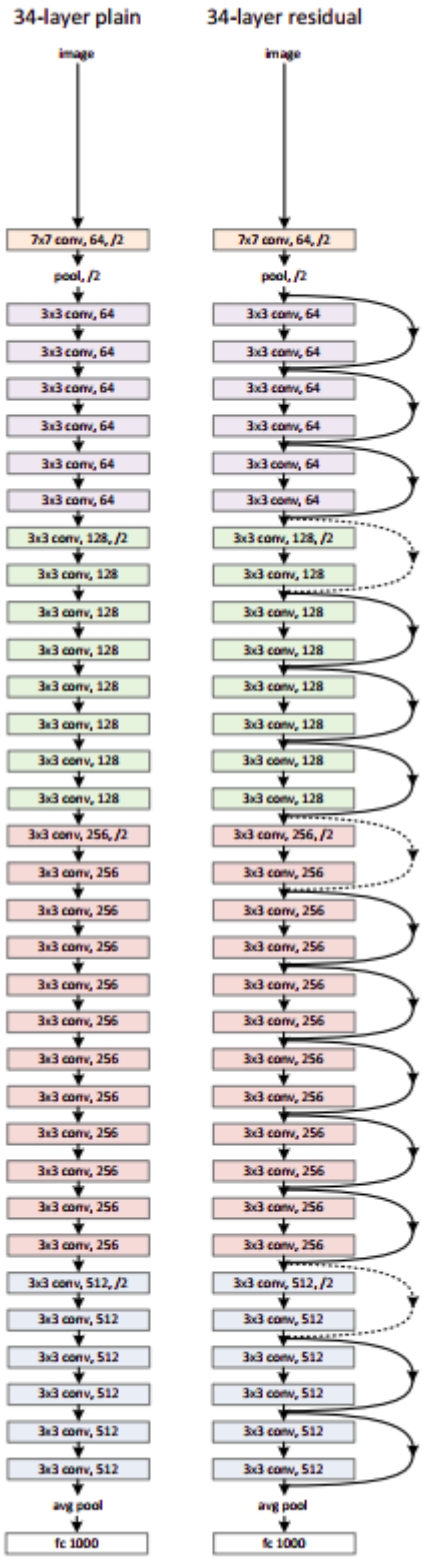


Fig.: The reusable residual network. (Img credit: https://arxiv.org/abs/1512.03385)

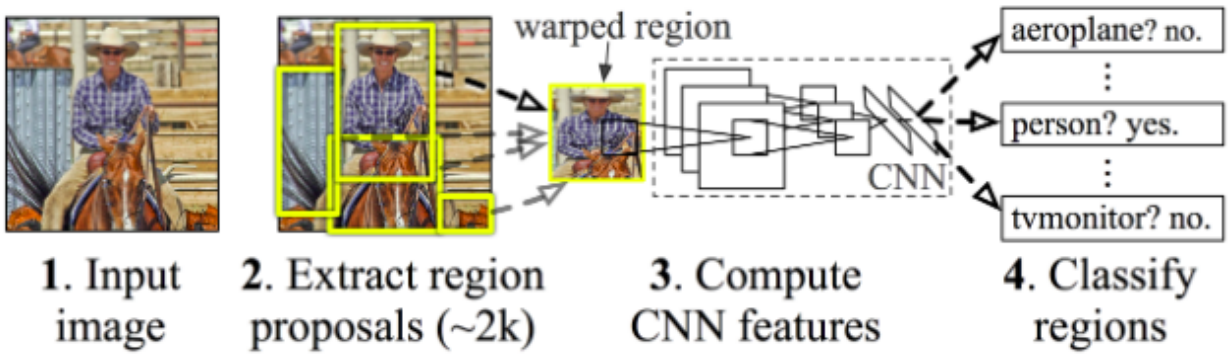
The network can be mathematically depicted as:

`H(x) = F(x) + x`, where `F(x) = W2*relu(W1*x+b1)+b2`



RCNN / Fast RCNN / Faster RCNN

RCNN: https://arxiv.org/abs/1311.2524



1. Extract region proposals with selective search to extract 2000 regional proposals
2. For each regional proposal:
    1. warp the region to input size for a CNN
    2. Compute cnn features
    3. Using the computed features, do classification with SVM (support vector machine) and bbox

$$\hat{G}_x = P_w d_x(P) + P_x \quad (1)$$
$$\hat{G}_y = P_h d_y(P) + P_y \quad (2)$$
$$\hat{G}_w = P_w \exp(d_w(P)) \quad (3)$$
$$\hat{G}_h = P_h \exp(d_h(P)). \quad (4)$$

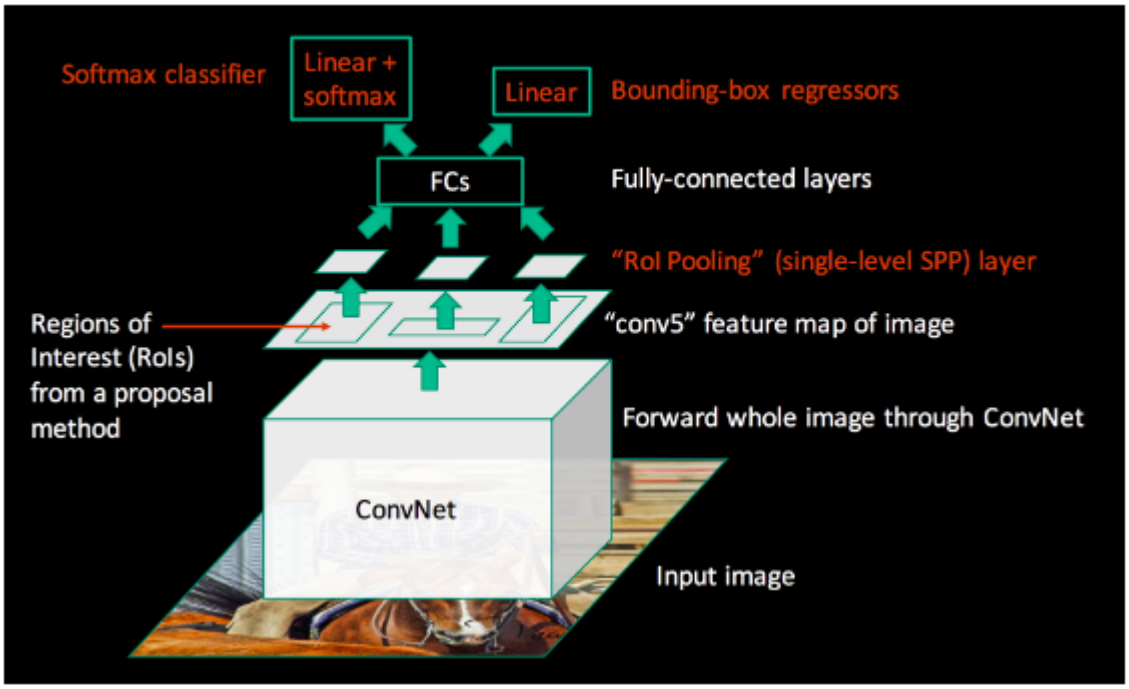
regressionG

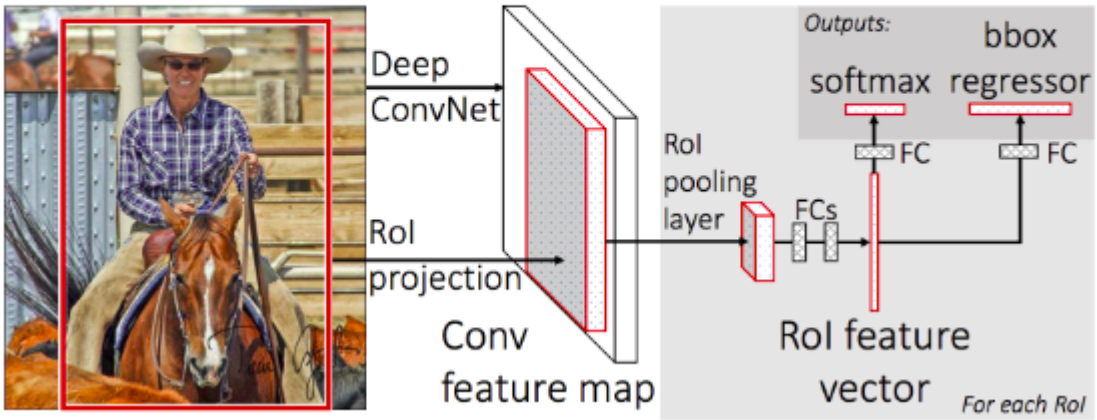
Each function $d_\star(P)$ (where $\star$ is one of x, y, h, w) is modeled as a linear function of the pool_5 features of proposal P, denoted by $\phi_5(P)$. (The dependence of $\phi_5(P)$ on the image data is implicitly assumed.) Thus we have $d_\star(P) = \mathbf{w}_\star^{\mathrm{T}} \phi_5(P)$, where $\mathbf{w}_\star$ is a vector of learnable model parameters. We learn $\mathbf{w}_\star$ by optimizing the regularized least squares objective (ridge regression):

$$\mathbf{w}_\star = \underset{\hat{\mathbf{w}}_\star}{\operatorname{argmin}} \sum_i^N (t_\star^i - \hat{\mathbf{w}}_\star^{\mathrm{T}} \phi_5(P^i))^2 + \lambda \|\hat{\mathbf{w}}_\star\|^2. \quad (5)$$

Super slow because we run a wrapped image through a CNN 2000 times.

Fast RCNN: https://arxiv.org/abs/1504.08083

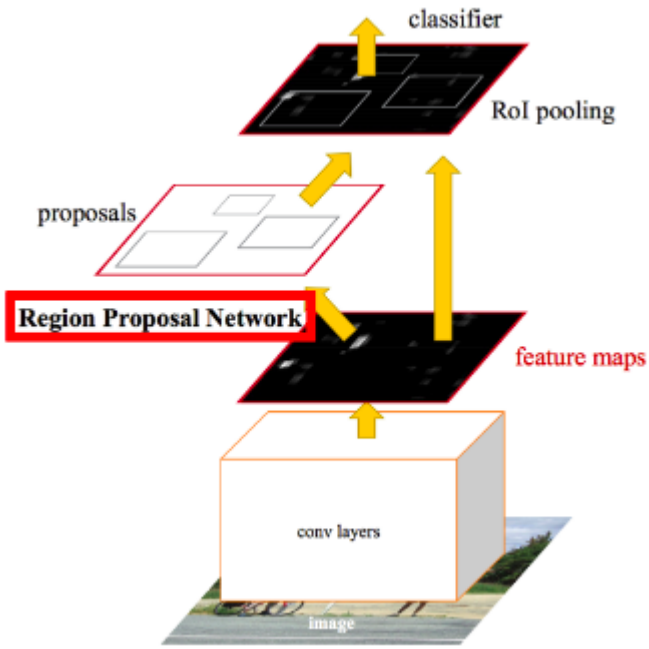




1. Extract region proposals with (for instance) selective search to generate n regional proposals
2. Run whole image through a CNN once, to generate a conv feature map
3. For each regional proposal:
4. Use ROI pooling layer to extract features from the conv feature map corrosponding to the region
5. Pass this new feature map through some FC layers to generate a RoI feature vector
6. Use a new FC layer to do softmax classification of the object in the region
7. Use a regression layer to do bbox regression

Much faster because we only run the image through a CNN once. We train the network in a single stage instead of SVM and linear regression, now we use softmax + linear layer for bbox regression instead.

Faster RCNN: https://arxiv.org/abs/1506.01497



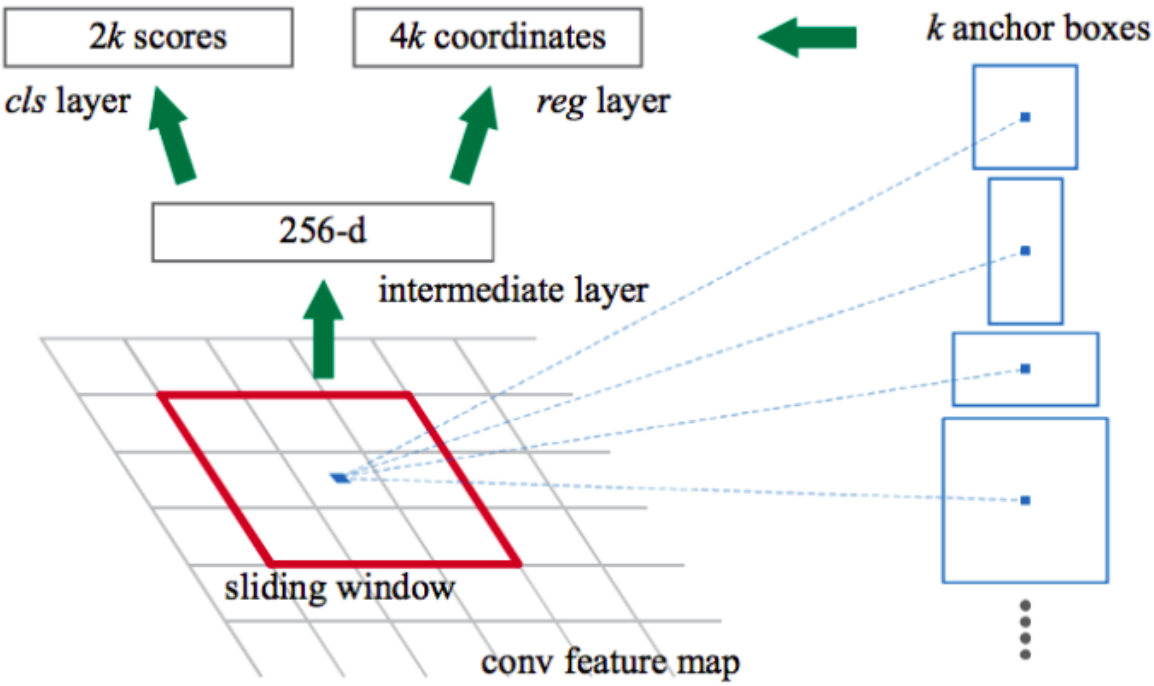
1. Run whole image through a CNN to generate a conv feature map
2. Generate region proposals using a **region proposal network (RPN)**:
    1. Using conv5-layer from the CNN as input, slide a 3x3 spatial window over the featres map to compute 256 features
    2. Using these 256 features, use 2 FC layers to compute two things: scores (how likely it is an object or background), and a boundary box.

       Use anchor boxes (standard boxes that emperically fits common objects quite well), as a prior to the boxes and use regression to finetune their coordinates.

       Input: CNN feature map

       Output: A bounding box per anchor and a score representing how likely it is, that this bounding box contains an object.

       What is objectiveness?: "We assign a positive label to two kinds of anchors: (i) the anchor/anchors with the highest Intersection-over Union (IoU) overlap with a ground-truth box, or (ii) an anchor that has an IoU overlap higher than 0.7 with 5 any ground-truth box" For training, using non-max-suppresion (NMS), around 2k proposals are used. In test-time, they use 300 proposals to feed into the classifier. This can be reduced for the application to get faster inference



3. Repeat the process from fast RCNN

Loss:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*)$$

Fast and more accurate because we don't have to do selective search anymore. We can also adjust the number of proposals (by taking the top-n proposals). We can do this now, because we have a score for each proposal and can thus select the n best ones

What to do next week

- Finish reporting my work. Will have to get this done before I can reconsider changing project.

- [ ] Train FaceNet-ish network
- [ ] STN-OCR
- [ ] OpenCV object tracker speed
- [ ] Read SSD Paper (again)