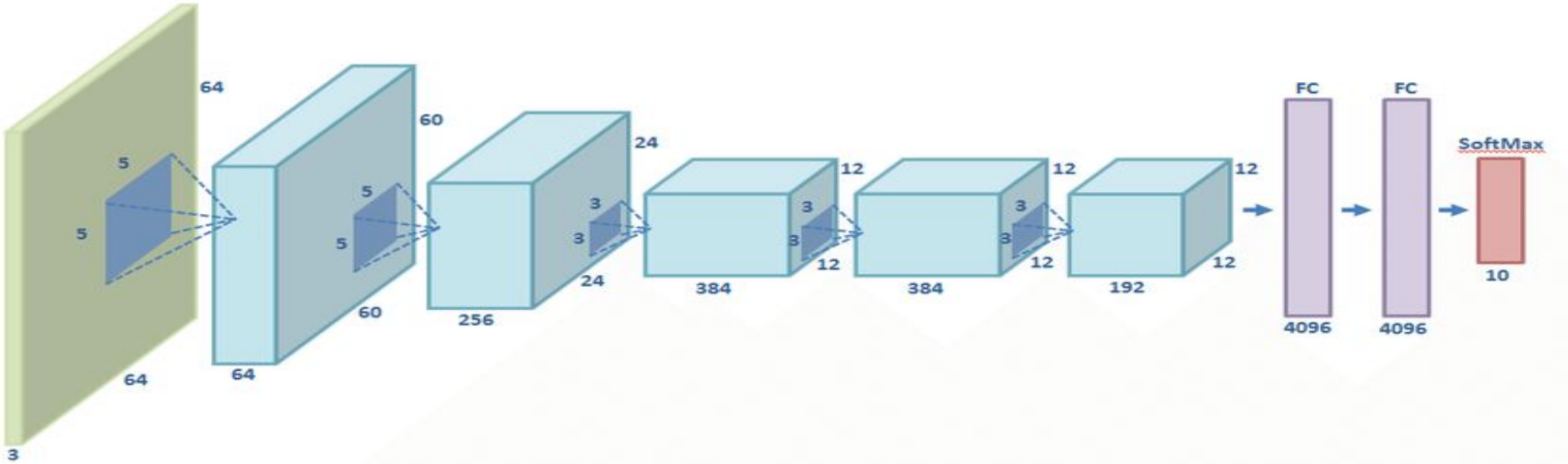
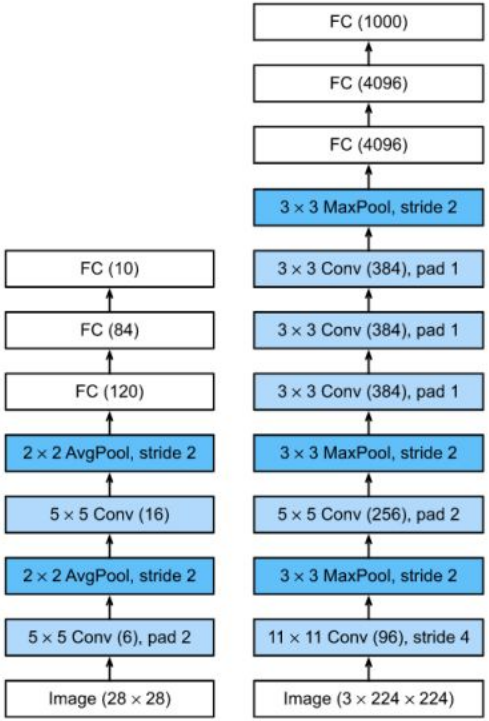
ALEXNET(2012)



- 5 convolutional, 3 fully-connected layers
- 1000 different classes (e.g. cats, dogs etc.)

ALEXNET(2012)



- Trained the network on ImageNet data, which contained over 15 million annotated images from a total of over 22,000 categories.
- Used ReLU for the nonlinearity functions (Found to decrease training time as ReLUs are several times faster than the conventional tanh function).
- Used data augmentation techniques that consisted of image translations, horizontal reflections, and patch extractions.
- Implemented dropout layers in order to combat the problem of overfitting to the training data.
- Trained the model using batch stochastic gradient descent, with specific values for momentum and weight decay.
- Trained on two GTX 580 GPUs for **five to six days**.

ZF NET (2013)

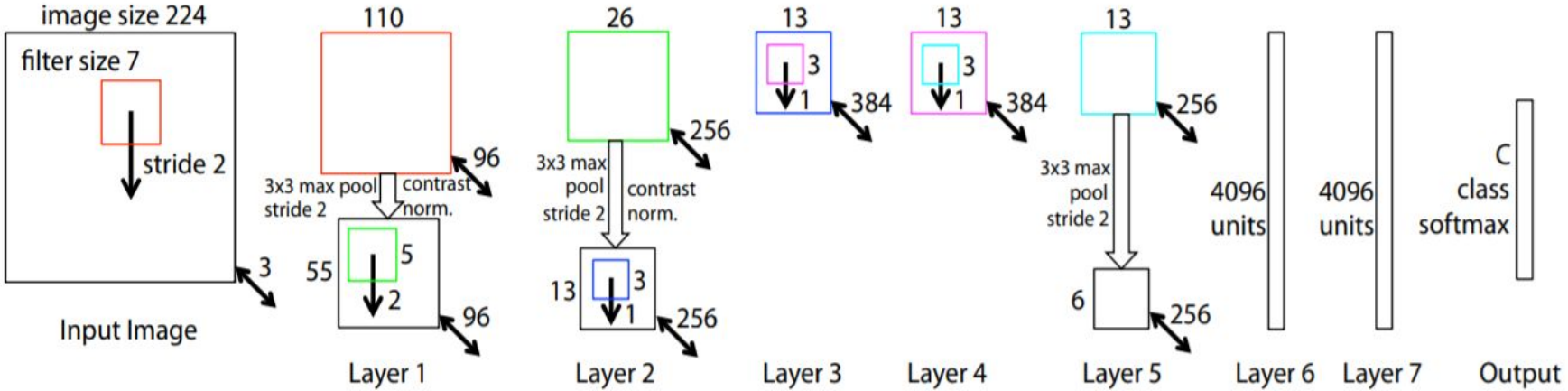


image size 224
filter size 7
stride 2
3
Input Image
110
96
3x3 max pool
stride 2
contrast
norm.
55
5
2
96
Layer 1
26
256
3x3 max
pool
stride 2
contrast
norm.
13
3
1
256
Layer 2
13
3
1
384
Layer 3
13
3
1
384
Layer 4
13
256
3x3 max
pool
stride 2
6
256
Layer 5
4096
units
Layer 6
4096
units
Layer 7
C
class
softmax
Output

ZF NET (2013)

- Very similar architecture to AlexNet, except for a few minor modifications.
- AlexNet trained on 15 million images, while ZF Net trained on only 1.3 million images.
- Instead of using 11x11 sized filters in the first layer (which is what AlexNet implemented), ZF Net used filters of size **7x7** and a **decreased** stride value. The reasoning behind this modification is that a smaller filter size in the first conv layer helps **retain** a lot of original pixel information in the input volume. A filtering of size 11x11 proved to be skipping a lot of relevant information, especially as this is the first conv layer.
- As the network grows, we also see a rise in the number of filters used.
- Used ReLUs for their activation functions, cross-entropy loss for the error function, and trained using batch stochastic gradient descent.
- Trained on a GTX 580 GPU for **twelve days**.
- Developed a visualization technique named **Deconvolutional Network**, which helps to examine different feature activations and their relation to the input space. Called “deconvnet” because it maps features to pixels (the opposite of what a convolutional layer does).

VGG NET (2014)

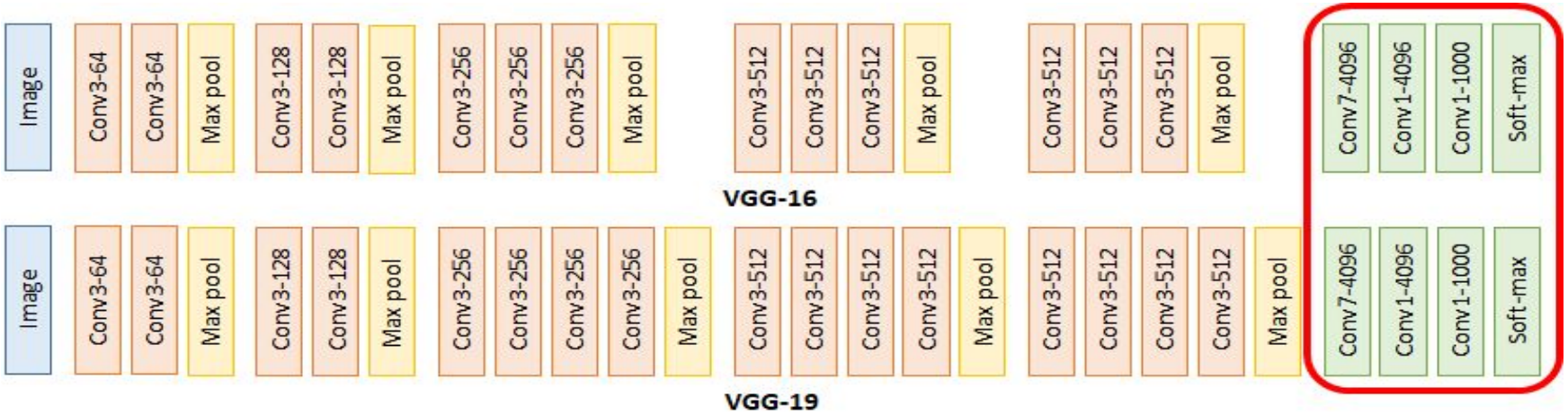
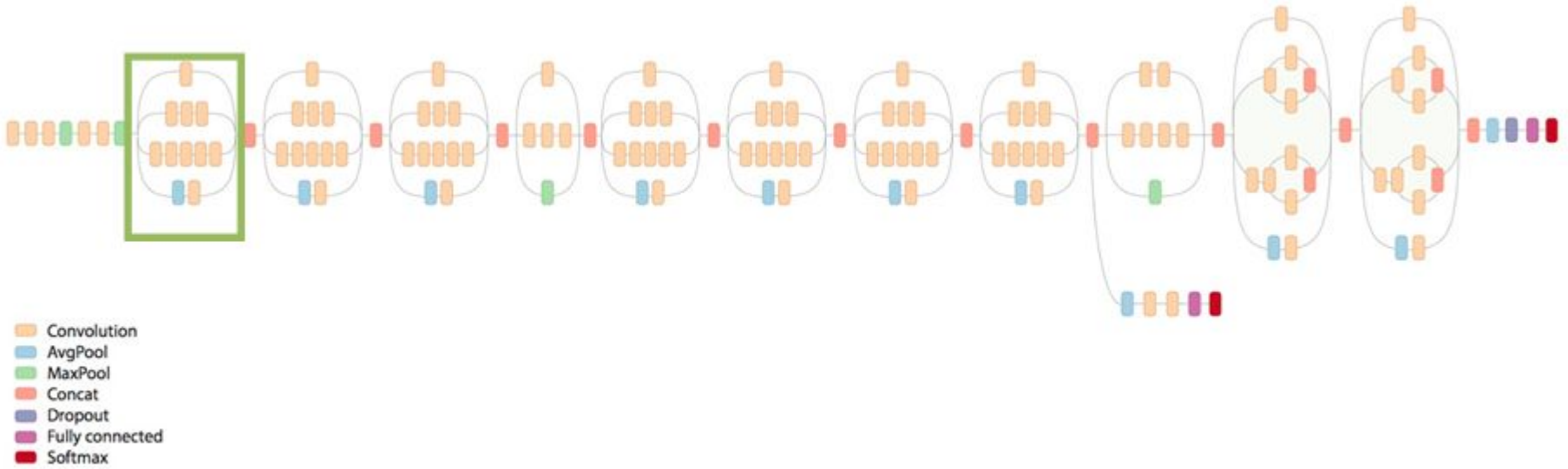


Image
Conv3-64
Conv3-64
Max pool
Conv3-128
Conv3-128
Max pool
Conv3-256
Conv3-256
Conv3-256
Max pool
Conv3-512
Conv3-512
Conv3-512
Max pool
Conv3-512
Conv3-512
Conv3-512
Max pool
Conv7-4096
Conv1-4096
Conv1-1000
Soft-max
VGG-16
Image
Conv3-64
Conv3-64
Max pool
Conv3-128
Conv3-128
Max pool
Conv3-256
Conv3-256
Conv3-256
Conv3-256
Max pool
Conv3-512
Conv3-512
Conv3-512
Conv3-512
Max pool
Conv3-512
Conv3-512
Conv3-512
Conv3-512
Max pool
Conv7-4096
Conv1-4096
Conv1-1000
Soft-max
VGG-19

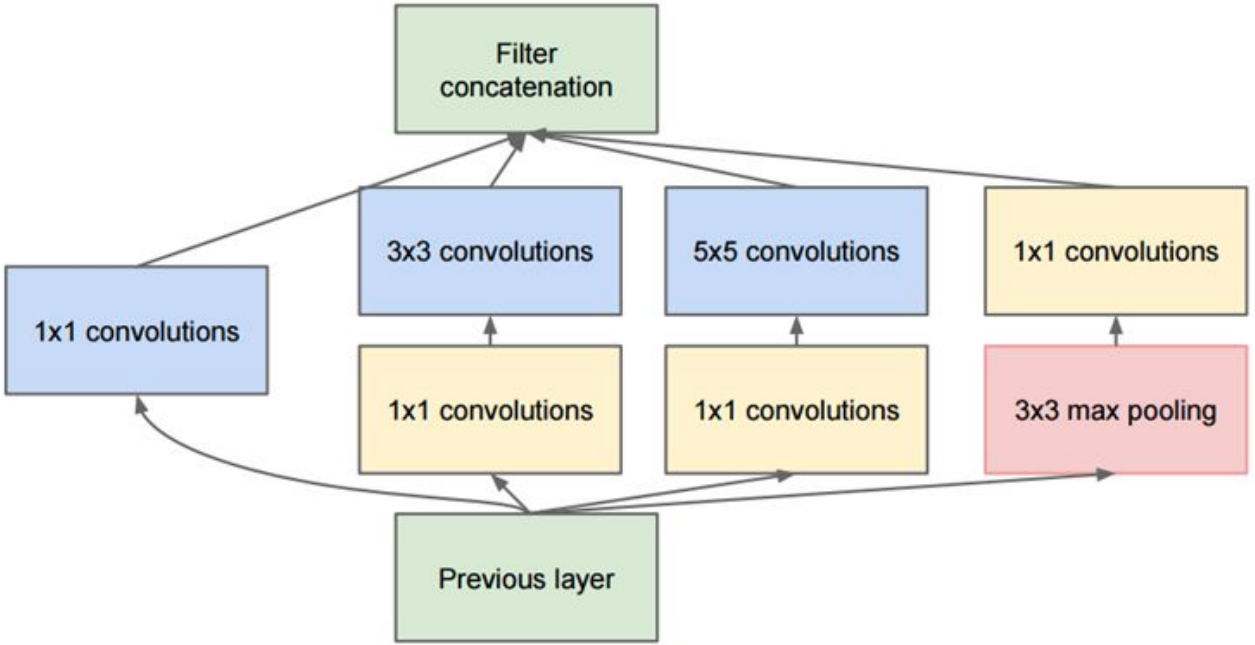
VGG NET (2014)

- Simplicity and depth
- The use of only 3x3 sized filters is quite different from AlexNet's 11x11 filters in the first layer and ZF Net's 7x7 filters. The authors' reasoning is that the combination of two 3x3 conv layers has an effective receptive field of 5x5. This in turn simulates a larger filter while keeping the benefits of smaller filter sizes. One of the benefits is a decrease in the number of parameters. Also, with two conv layers, we're able to use two ReLU layers instead of one.
- 3 conv layers back to back have an effective receptive field of 7x7.
- As the spatial size of the input volumes at each layer decrease (result of the conv and pool layers), the depth of the volumes increase due to the increased number of filters as you go down the network.
- Interesting to notice that the number of filters doubles after each maxpool layer. This reinforces the idea of shrinking spatial dimensions, but growing depth.
- Worked well on both image classification and localization tasks. The authors used a form of localization as regression.
- Built model with the Caffe toolbox.
- Used scale jittering as one data augmentation technique during training.
- Used ReLU layers after each conv layer and trained with batch gradient descent.
- Trained on 4 Nvidia Titan Black GPUs for **two to three weeks**.

GOOGLENET (2015)



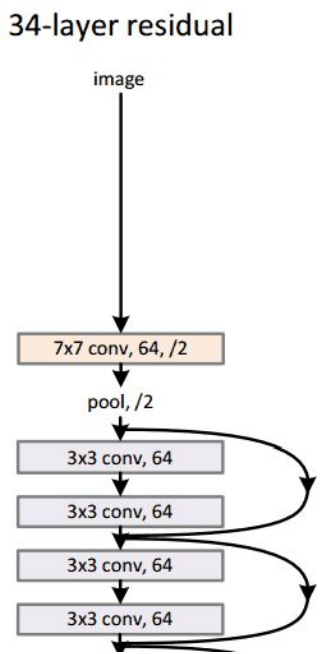
GOOGLENET (2015)



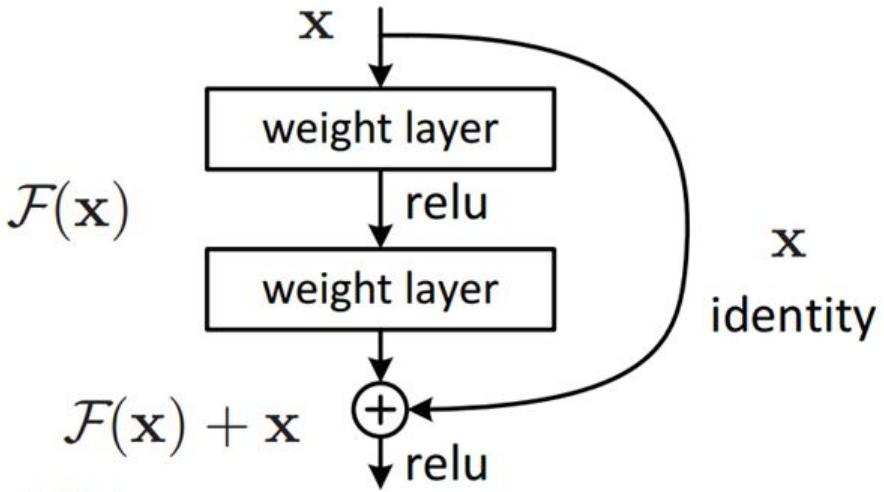
GOOGLENET (2015)

- Used 9 Inception modules in the whole architecture, with over 100 layers in total! Now that is deep…
- No use of fully connected layers! They use an average pool instead, to go from a 7x7x1024 volume to a 1x1x1024 volume. This saves a huge number of parameters.
- Uses 12x fewer parameters than AlexNet.
- During testing, multiple crops of the same image were created, fed into the network, and the softmax probabilities were averaged to give us the final solution.
- Utilized concepts from R-CNN (a network we'll discuss later) for their detection model.
- There are updated versions to the Inception module (Versions 6 and 7).
- Trained on "a few high-end GPUs **within a week**".

RESNET (2015)


34-layer residual
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64

RESNET (2015)

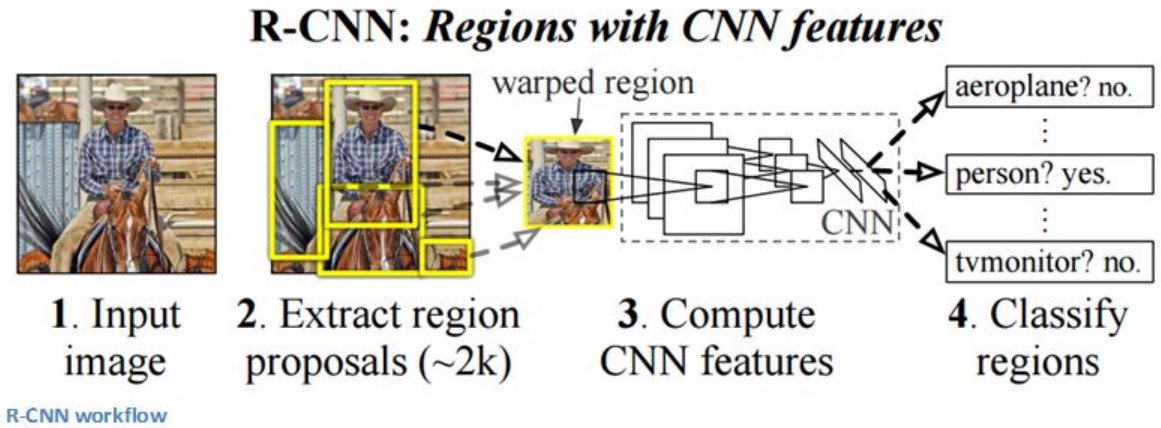


A residual block

RESNET (2015)

- “Ultra-deep” – Yann LeCun.
- 152 layers…
- Interesting note that after only the *first 2* layers, the spatial size gets compressed from an input volume of 224x224 to a 56x56 volume.
- Authors claim that a naïve increase of layers in plain nets result in higher training and test error.
- The group tried a 1202-layer network, but got a lower test accuracy, presumably due to overfitting.
- Trained on an 8 GPU machine for **two to three weeks**.

REGION BASED CNNS (R-CNN - 2013, FAST R-CNN - 2015, FASTER R-CNN - 2015)

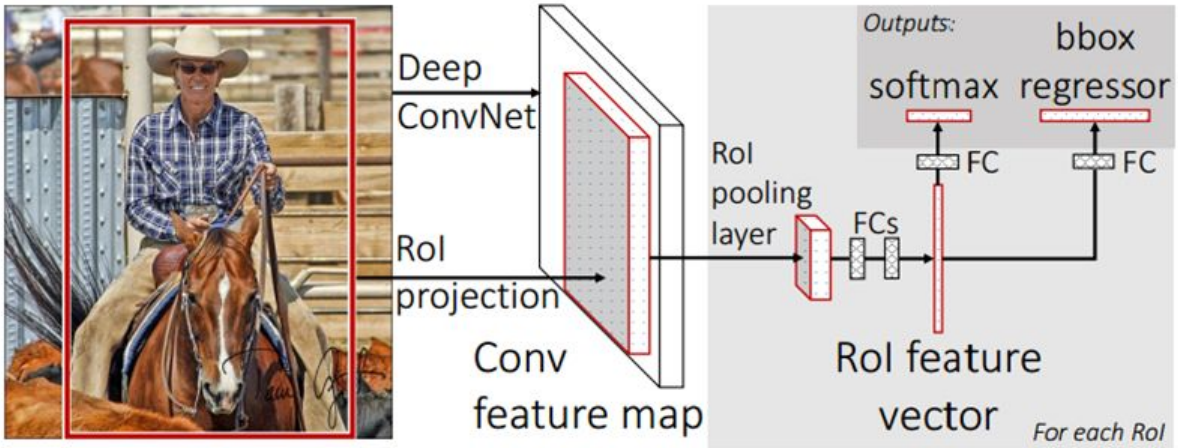


R-CNN workflow

REGION BASED CNNS (R-CNN - 2013)

- The purpose of R-CNNs is to solve the problem of object detection. Given a certain image, we want to be able to draw bounding boxes over all of the objects. The process can be split into two general components, the region proposal step and the classification step.
- Fast R-CNN was able to solve the problem of speed by basically sharing computation of the conv layers between different proposals and swapping the order of generating region proposals and running the CNN.

FAST R-CNN - 2015



Fast R-CNN workflow

Major Merits

- AlexNet: This was the first time a model performed so well on a historically difficult ImageNet dataset.
- ZFNet: Provided great intuition as to the workings on CNNs and illustrated more ways to improve performance, introduced Deconvnet.
- VGG Net: It is one of the most influential papers in my mind because it reinforced the notion that convolutional neural networks have to have a deep network of layers in order for this hierarchical representation of visual data to work.
- GoogLeNet: It was one of the first models that introduced the idea that CNN layers didn't always have to be stacked up sequentially. Coming up with the Inception module, the authors showed that a creative structuring of layers can lead to improved performance and computationally efficiency.
- ResNet: 3.6% error rate.
- Region-based CNNs: Being able to determine that a specific object is in an image is one thing, but being able to determine that object's exact location is a huge jump in knowledge for the computer. Faster R-CNN has become the standard for object detection programs today.

Selected References

- https://adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html
- http://mrl.cs.vsb.cz/data/ano2/ano2_ang/AlexNet_ZFNet_VGGNet_GoogLeNet_ResNet_DenseNet_RCNN.pdf
- https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf