Modern Applications of Deep Learning

Michael Pound

Kernel Convolution

- Convolve a kernel across an image or feature map
- At each location, calculate the sum product of the kernel and the input

1	4	3	2	2
2	1	7	4	6
3	4	6	1	8
2	1	5	3	7
1	7	3	5	2

1	2	1
0	0	0
-1	-2	-1

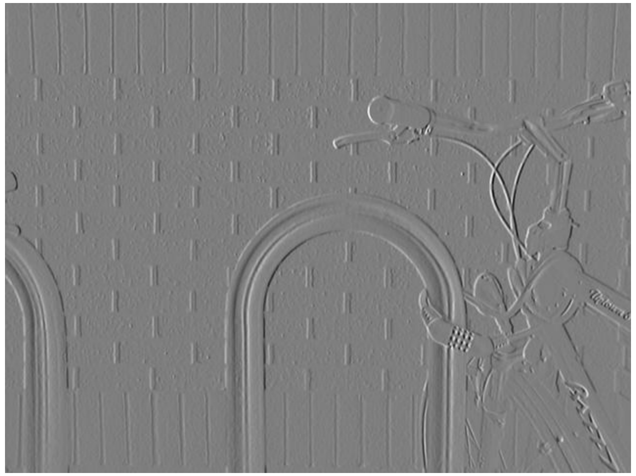
-1	3	-3
2	5	3
-5	-9	-3

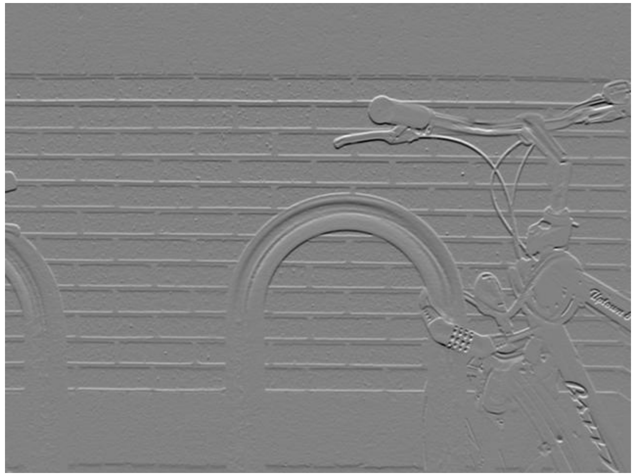
Why are these filters useful?

- The Sobel operator consists of two 3x3 kernels that highlight image edges

1	0	-1
2	0	-2
1	0	-1

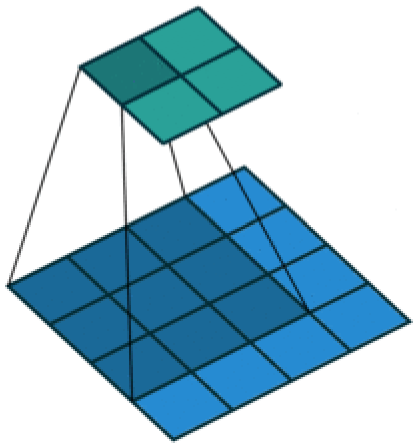
1	2	1
0	0	0
-1	-2	-1



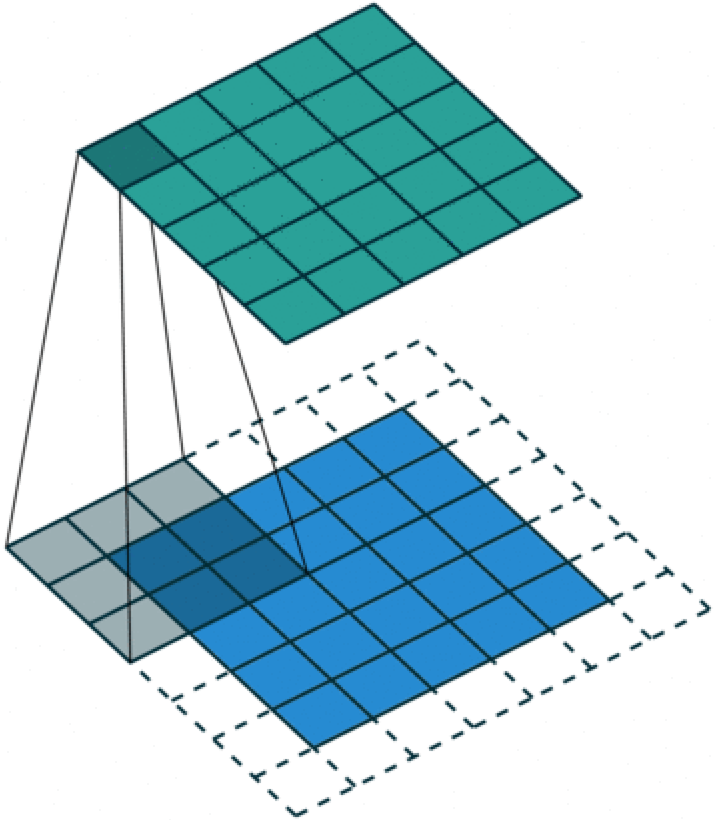


Padding and Stride

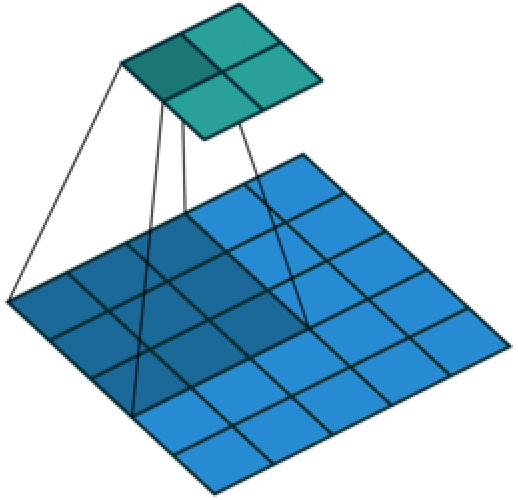
Padding: 0 Stride: 1



Padding: 1 Stride: 1

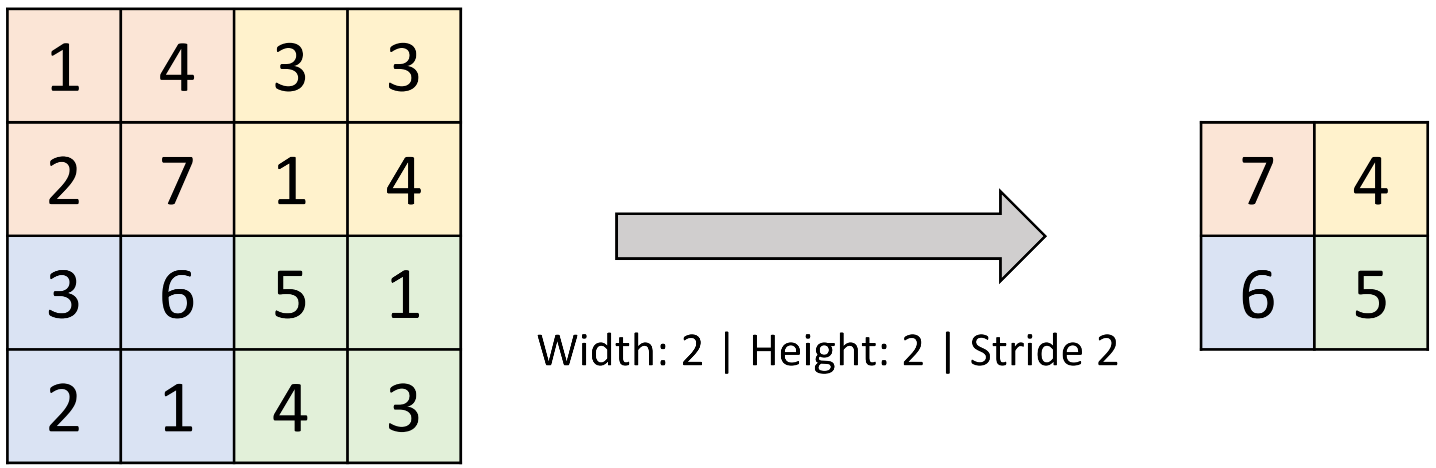


Padding: 0 Stride:2



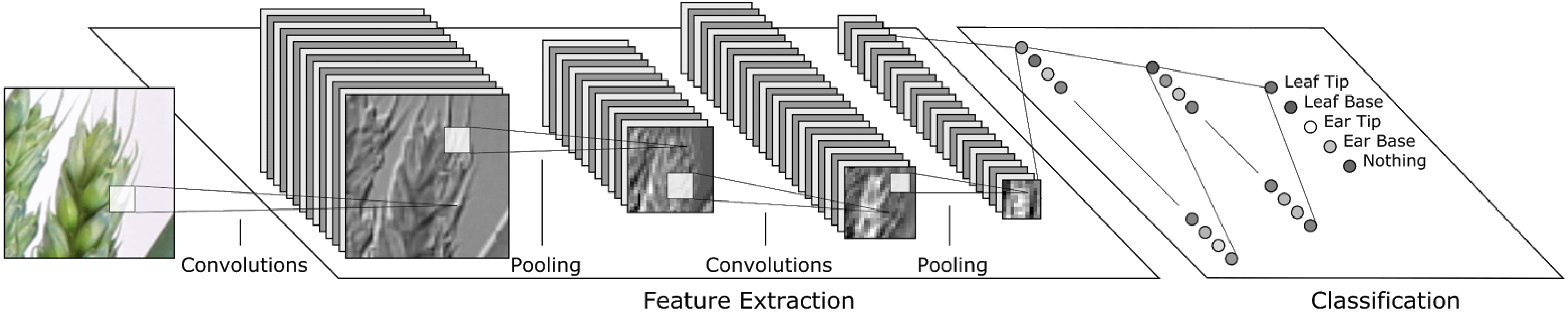
Max Pooling Layers

- Max pooling spatially downsamples feature maps
  - Reduced memory requirements
  - Increased spatial invariance of features



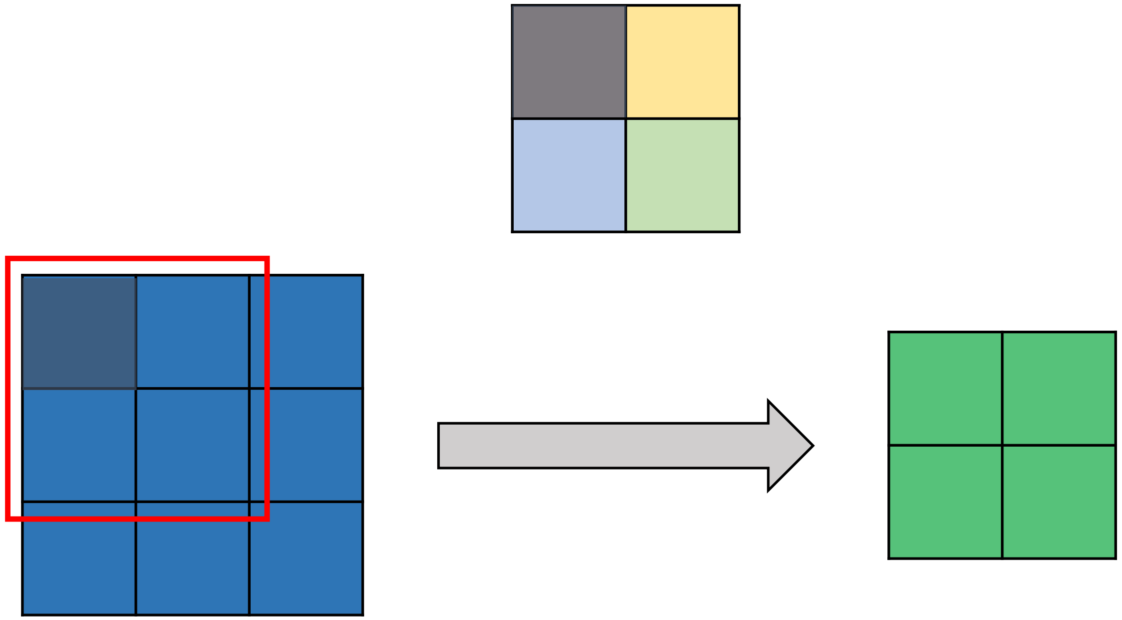
Convolutional Neural Networks

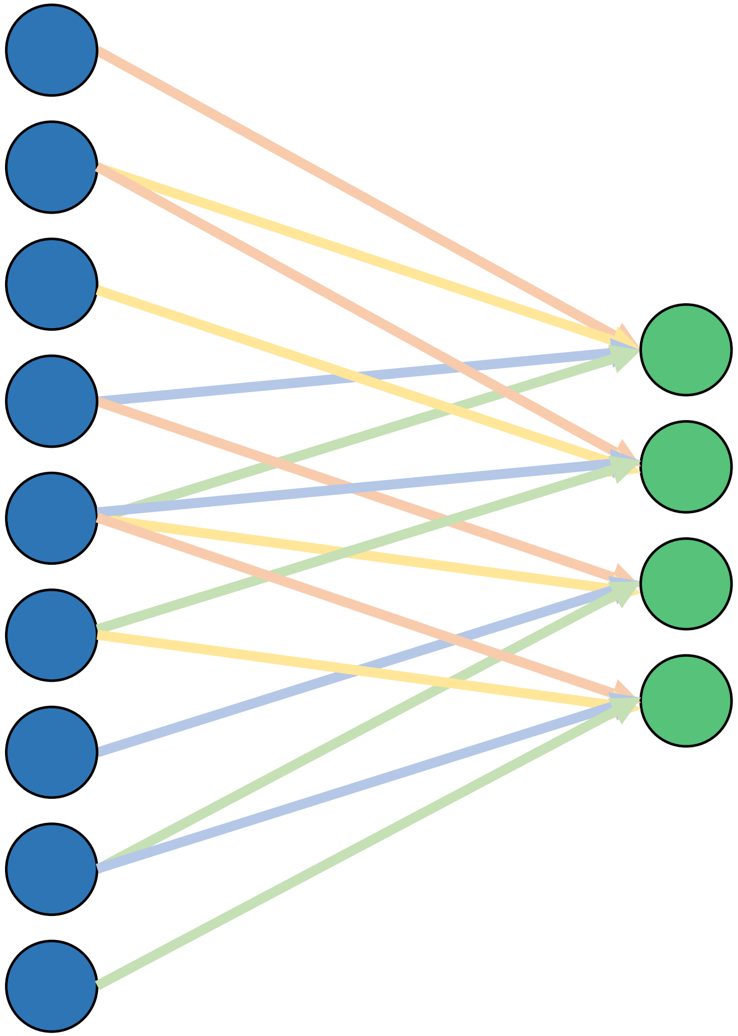
- The majority of deep learning uses Convolutional Neural Networks
  - Usually combine convolution and pooling operations
  - Finish with traditional MLP layers to perform a classification



Convolutional Layers vs MLPs

- Are convolutions and MLPs that different?

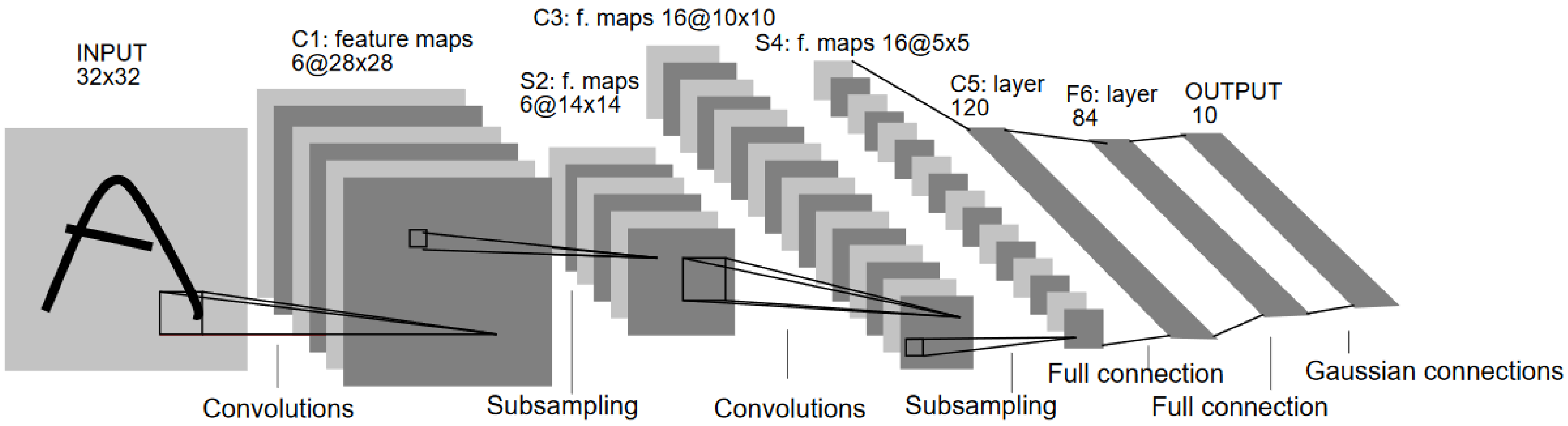




Modern Classification Networks

A Classic Example

- LeNet was the first convolutional network, used for digit classification



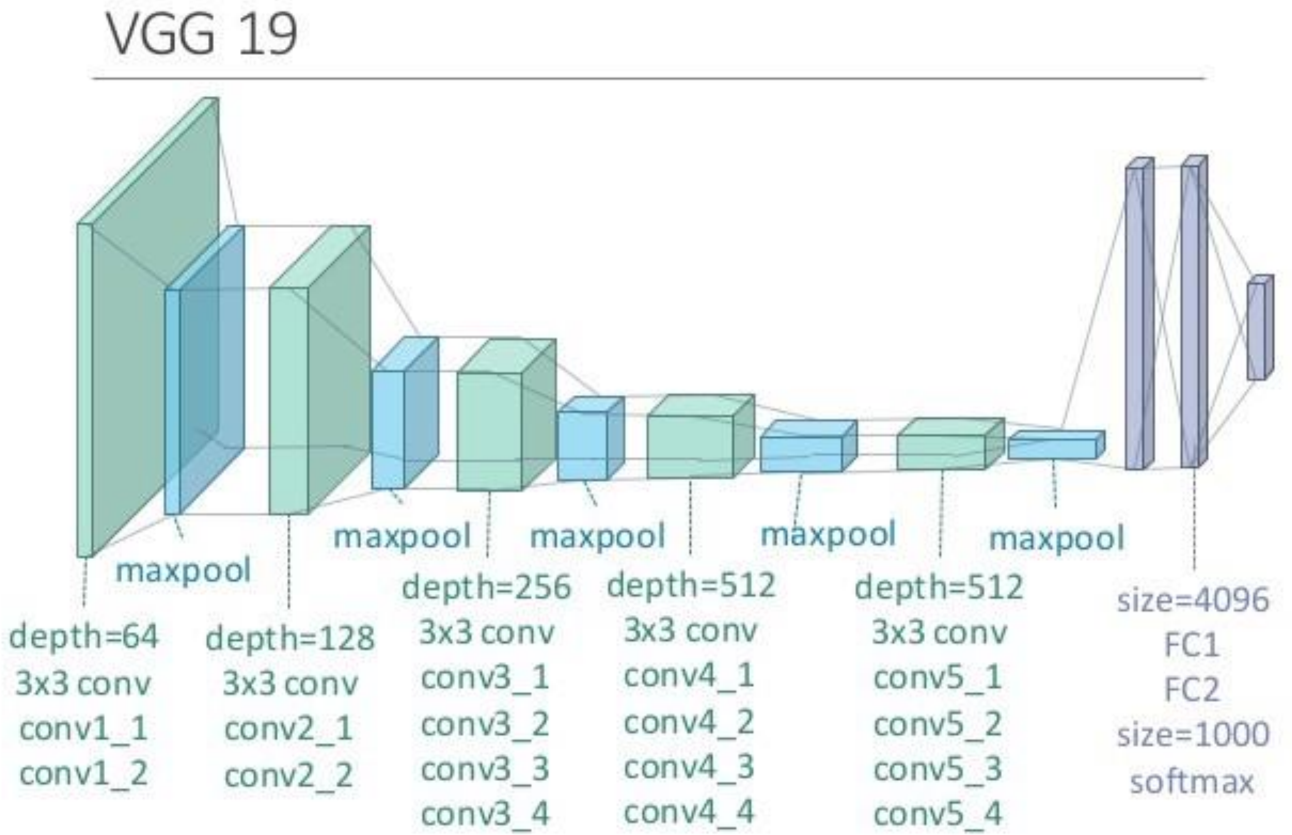
VGG (2014)

- When it was released, VGG was the deepest network so far
- Replaced 7x7 and 11x11 convolutions with chained 3x3
- Padding used to preserve size when using convolutions
- #features increased after each spatial downsampling

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

VGG (2014)

+ Outperformed many existing networks
+ 3x3 convolutions are accurate but efficient
+ Consistent design makes it easy to follow
- Extremely hard to train
- Usually must be used pre-trained



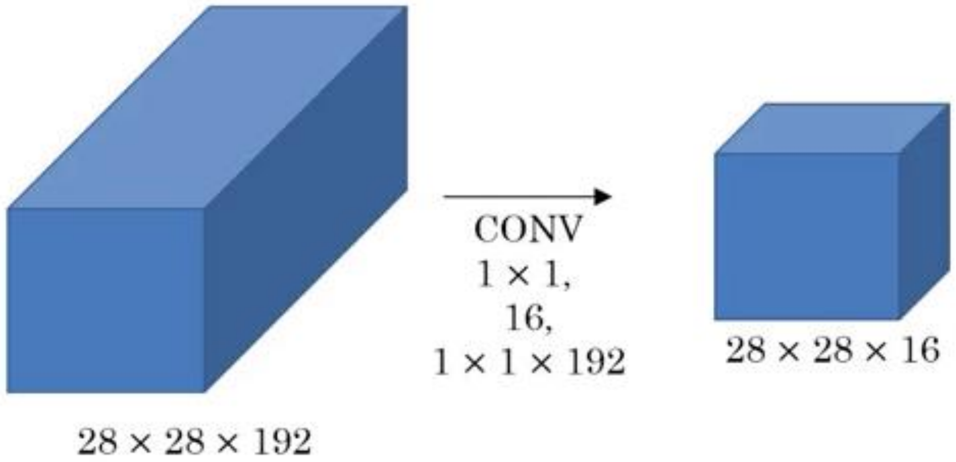
Transfer Learning

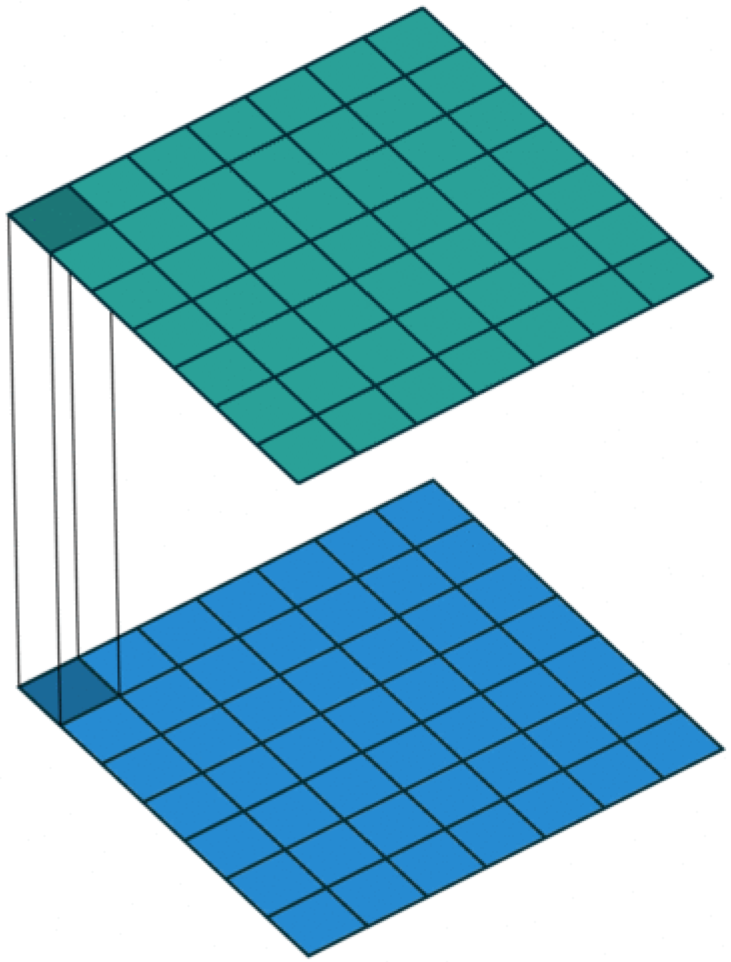
- Training a network like VGG-19 from scratch takes a long time
- It’s common to use pre-trained weights to initialise the network
- Training fine-tunes the network from this start point

```
def vgg19(pretrained=False, **kwargs):
    """VGG 19-layer model (configuration "E")
    Args:
        pretrained (bool): If True, returns a model pre-trained on ImageNet
    """
    if pretrained:
        kwargs['init_weights'] = False
    model = VGG(make_layers(cfg['E']), **kwargs)
    if pretrained:
        model.load_state_dict(model_zoo.load_url(model_urls['vgg19']))
return model
```

1x1 Convolutions

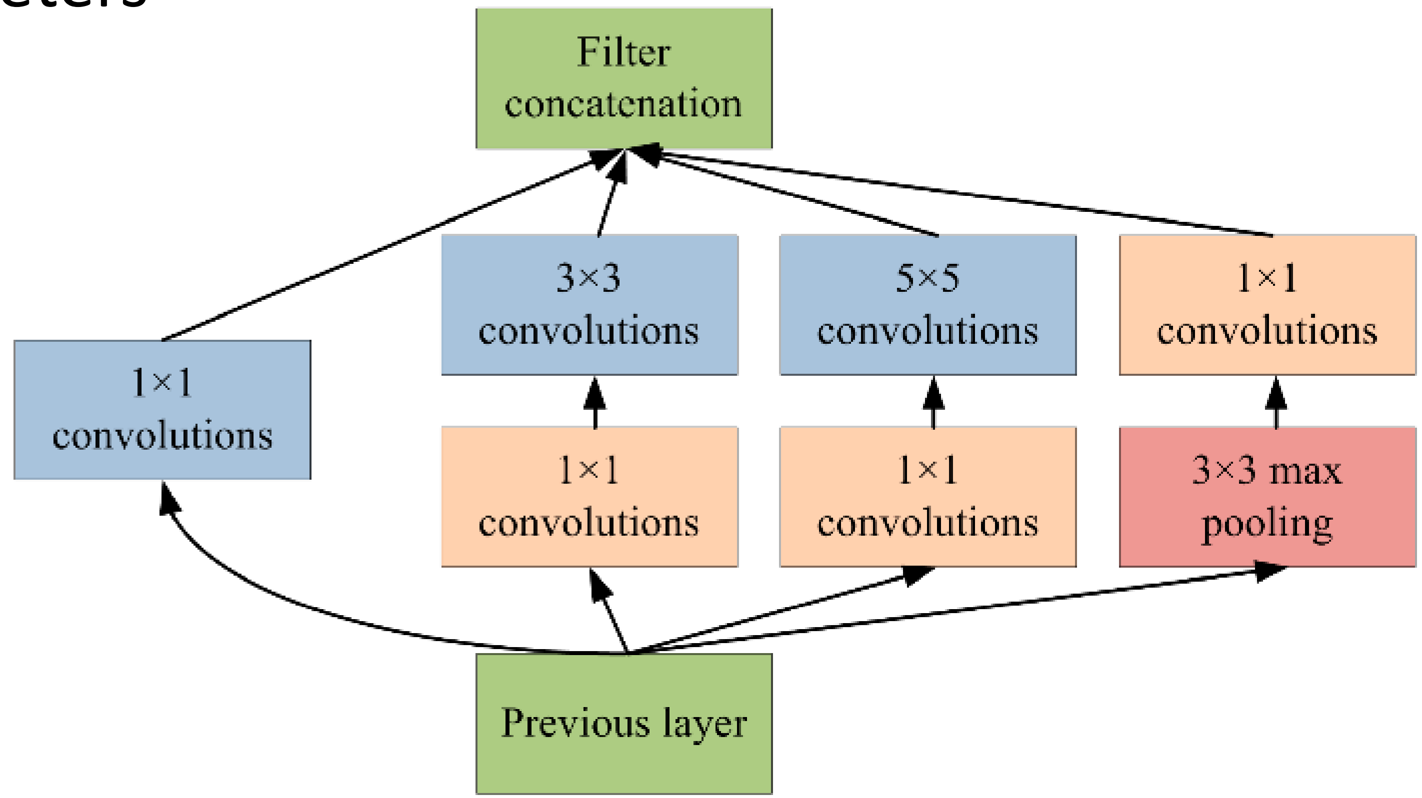
- These might seem entirely pointless, but they have some interesting uses
  - Ignore spatial connections
  - Combine information across feature maps
  - Can increase or decrease feature depth





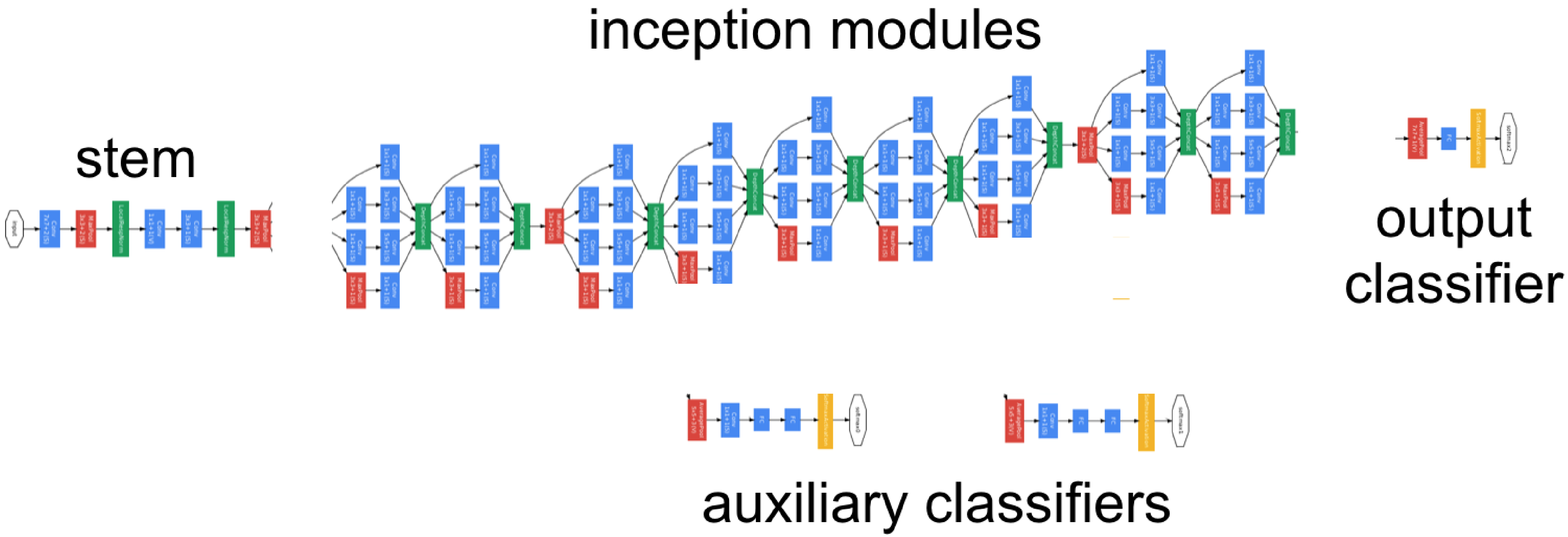
Google's Inception (v1)

- Google argued that multiple paths saves us from choosing kernel size and other parameters



GoogleNet (2015)

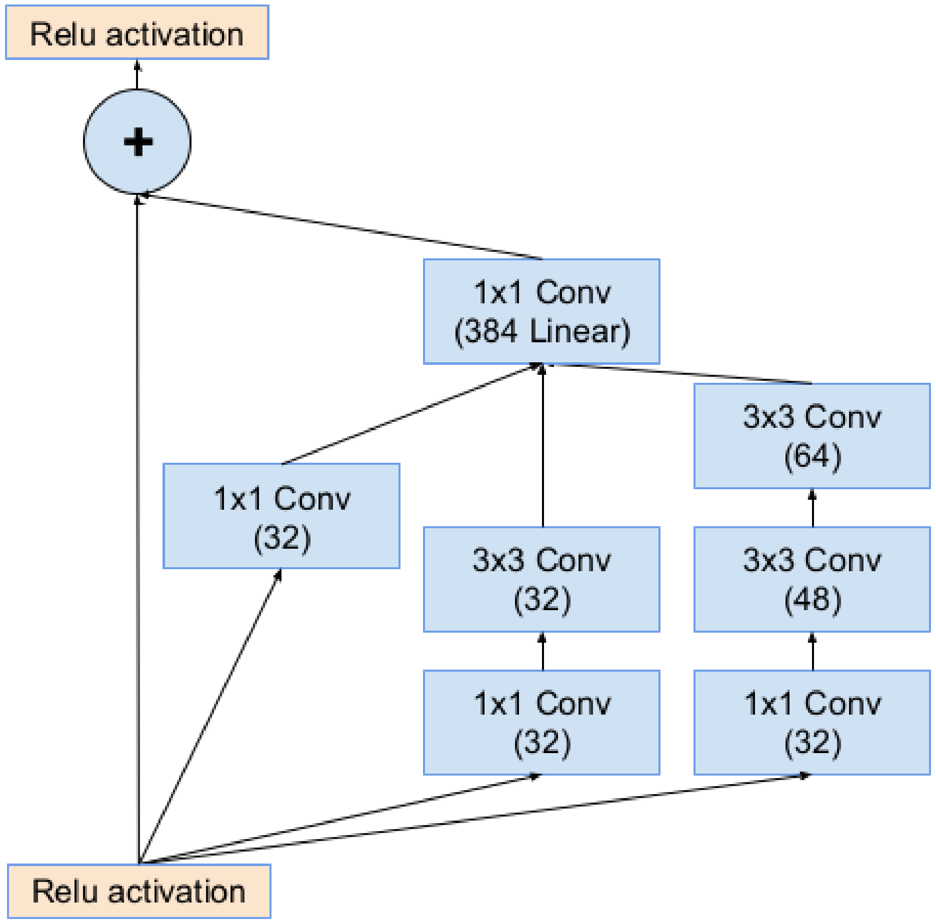
- Uses inception modules throughout
- Currently on v7 (I think)



GoogleNet (2015)

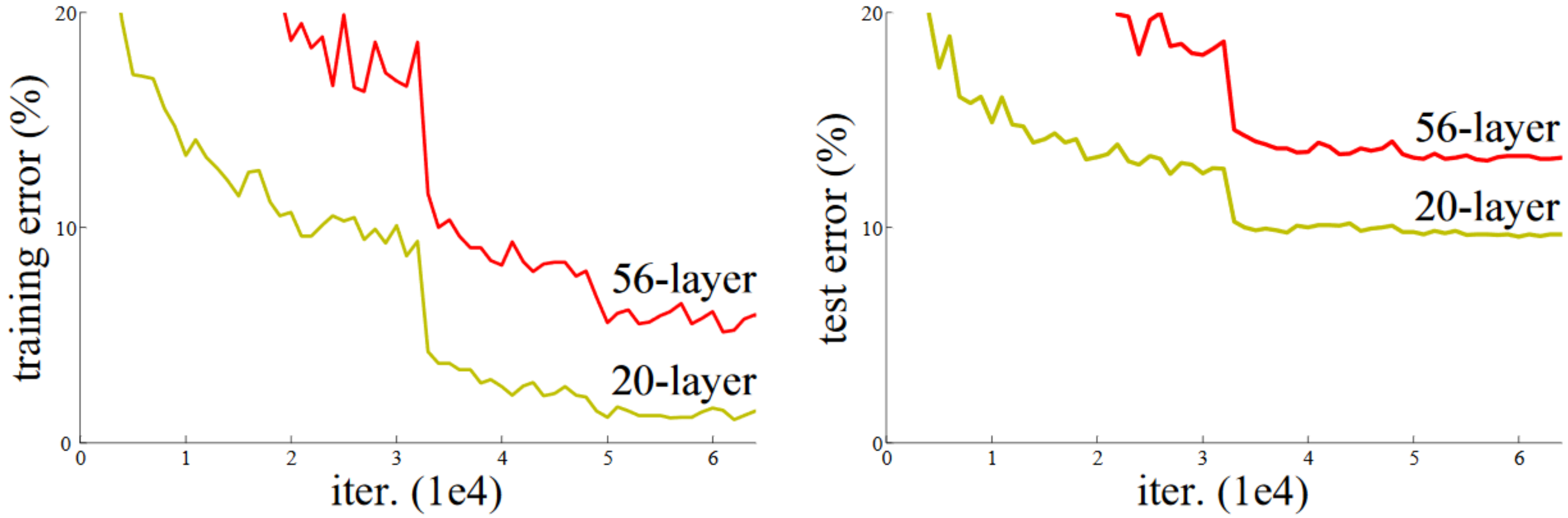
- +Ranked #1 for performance
- +Use of 1x1 convolutions makes it actually quite space efficient
- +Trains much faster than VGG
- \- Pretty convoluted design!

A recent inception block



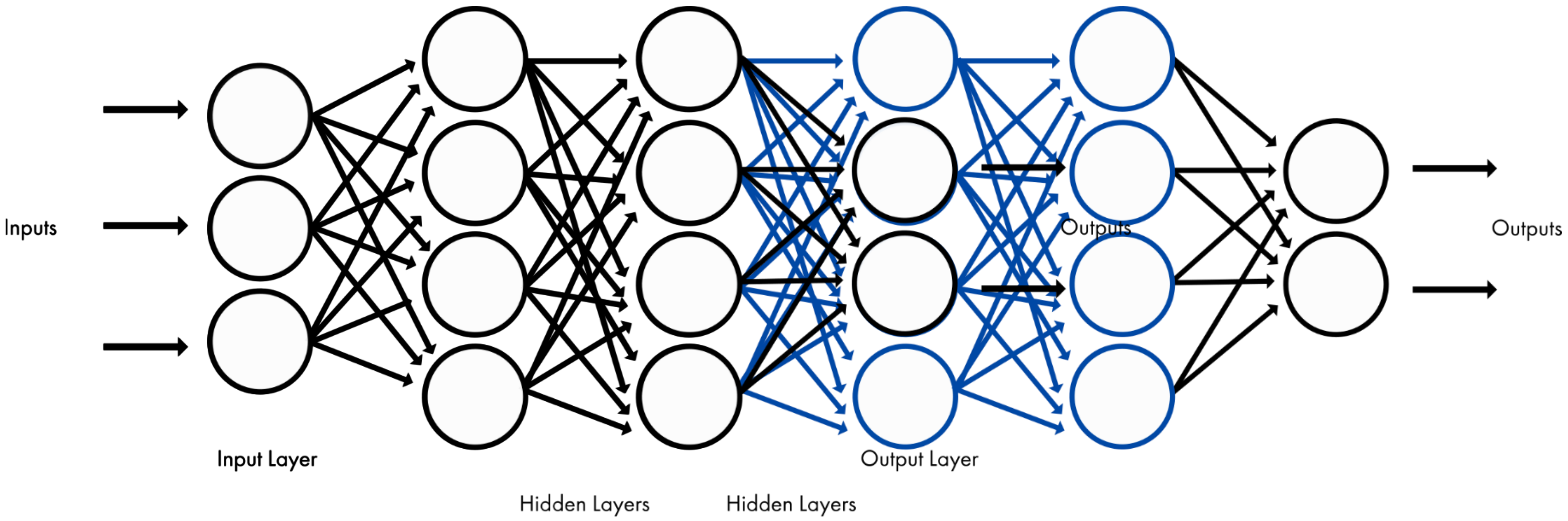
Network Degradation

- Deeper is better, right?



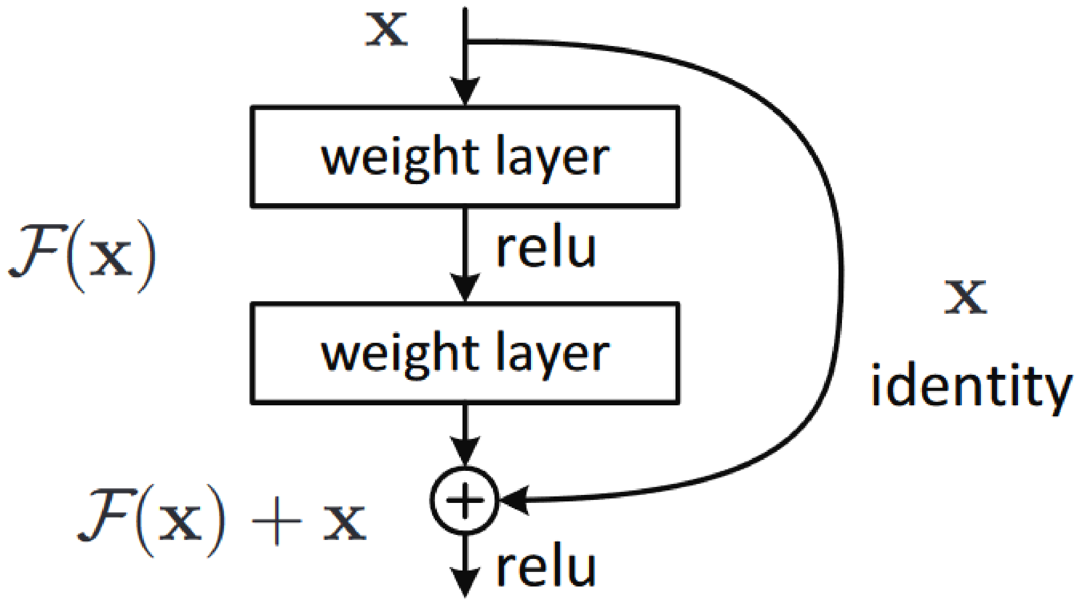
Network Degradation

- Deeper is better, right?



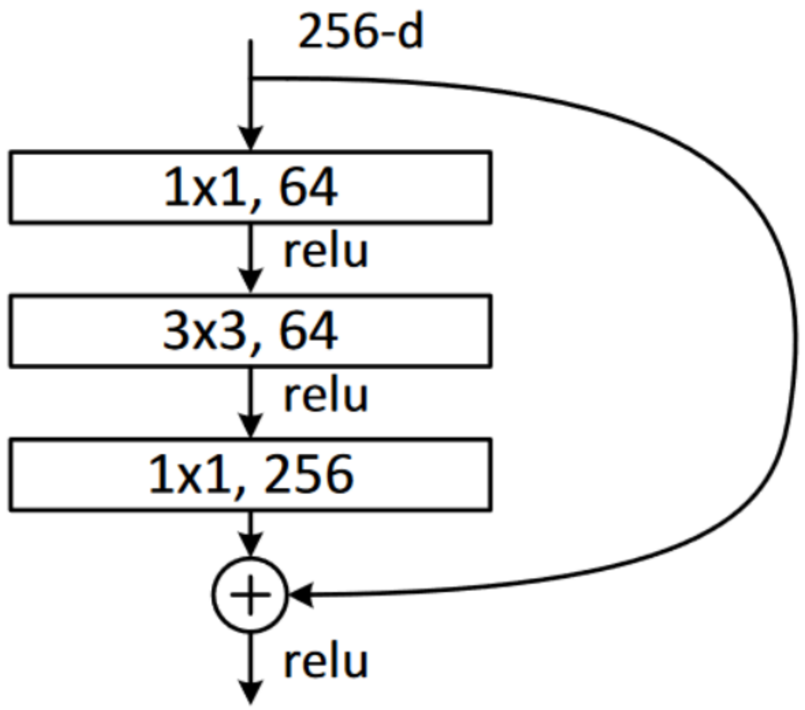
Residual Blocks

- Residual blocks deal with the general challenge of training very deep networks



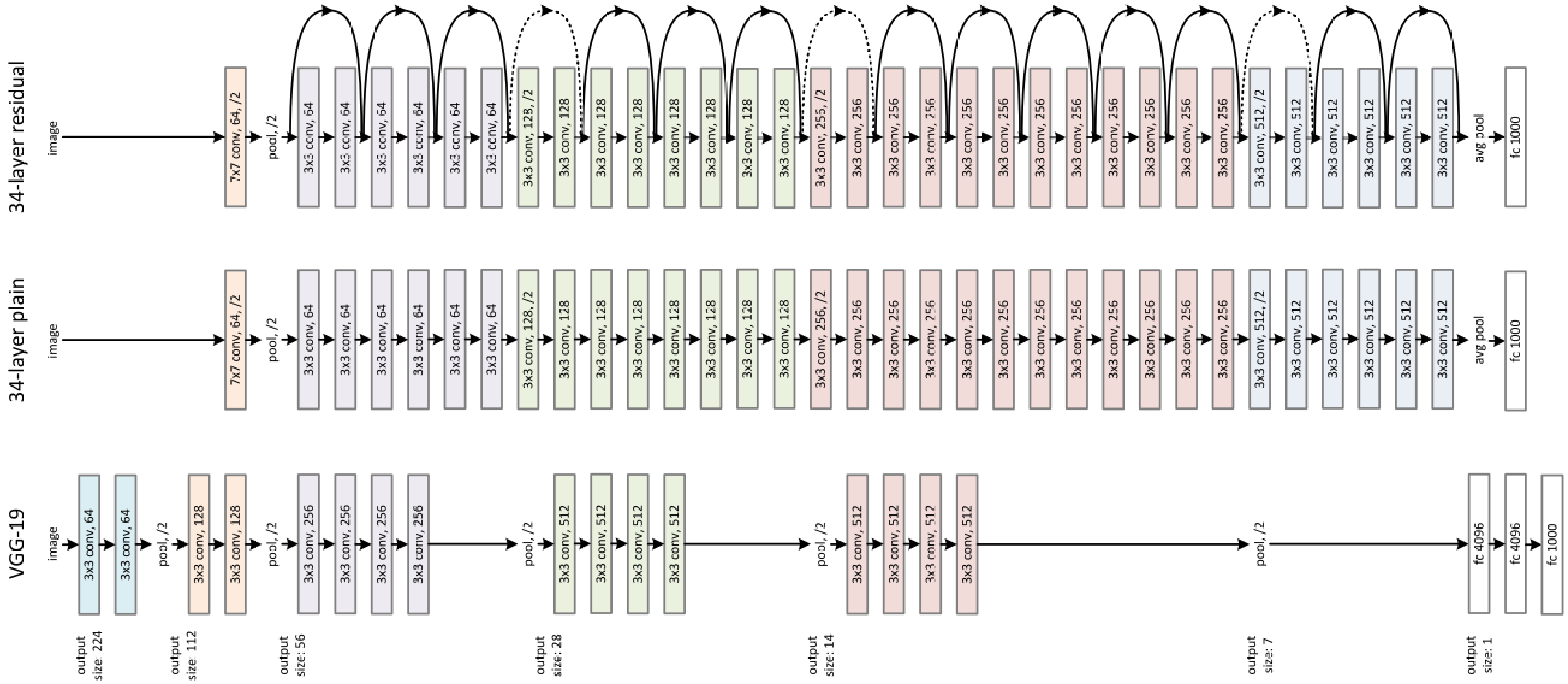
Bottleneck Blocks

- As with inception, briefly reducing the features is more efficient



ResNets (2015)

- ResNets chain blocks into extremely deep networks



Batch Normalisation

- BN normalises activations (channels in the feature map) to a learned mean and variance
- Speeds up training by reducing noise in the input to each layer

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma + \beta$$

ResNet Block Code

- This Pytorch implementation uses a bottleneck as well as batch normalisation

- I've simplified it slightly by assuming:

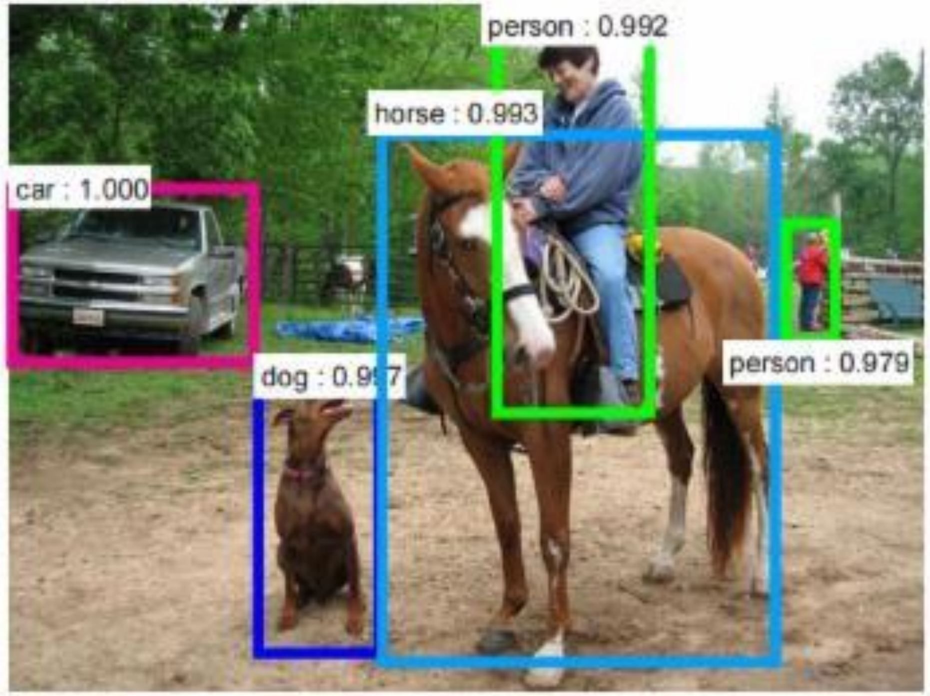
  inplanes == planes

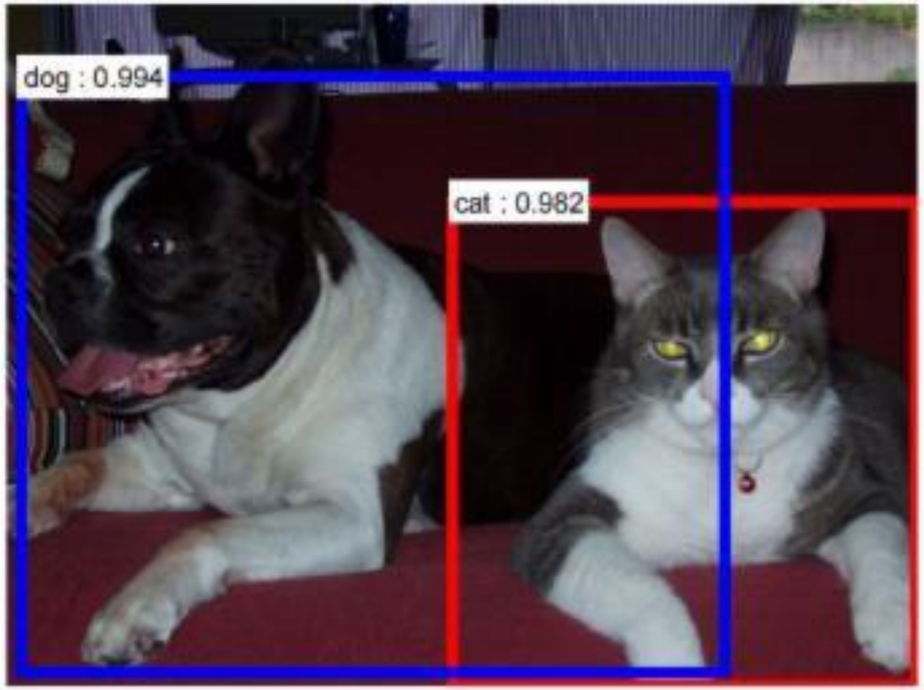
```python
class Bottleneck(nn.Module):
    expansion = 4

    def __init__(self, inplanes, planes, stride=1):
        super(Bottleneck, self).__init__()
        self.conv1 = nn.Conv2d(inplanes, planes, kernel_size=1, bias=False)
        self.bn1 = nn.BatchNorm2d(planes)
        self.conv2 = nn.Conv2d(planes, planes, kernel_size=3, stride=stride,
                               padding=1, bias=False)
        self.bn2 = nn.BatchNorm2d(planes)
        self.conv3 = nn.Conv2d(planes, planes * self.expansion, kernel_size=1, bias=False)
        self.bn3 = nn.BatchNorm2d(planes * self.expansion)
        self.relu = nn.ReLU(inplace=True)

    def forward(self, x):
        residual = x

        out = self.conv1(x)
        out = self.bn1(out)
        out = self.relu(out)

        out = self.conv2(out)
        out = self.bn2(out)
        out = self.relu(out)

        out = self.conv3(out)
        out = self.bn3(out)

        out += residual
        out = self.relu(out)

return out
```

Object Detection

Object Detection

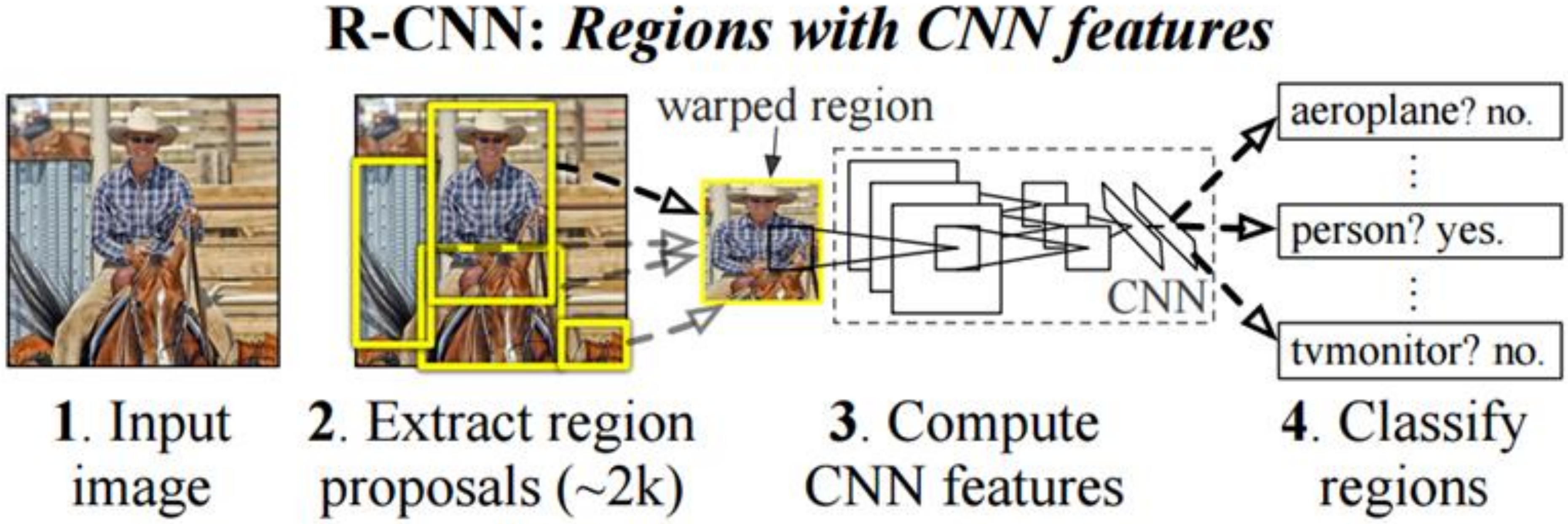
- Image-level classification doesn’t address the issue of multiple objects
- Object detection aims to find bounding boxes of any interesting objects in a scene





R-CNN (2013)

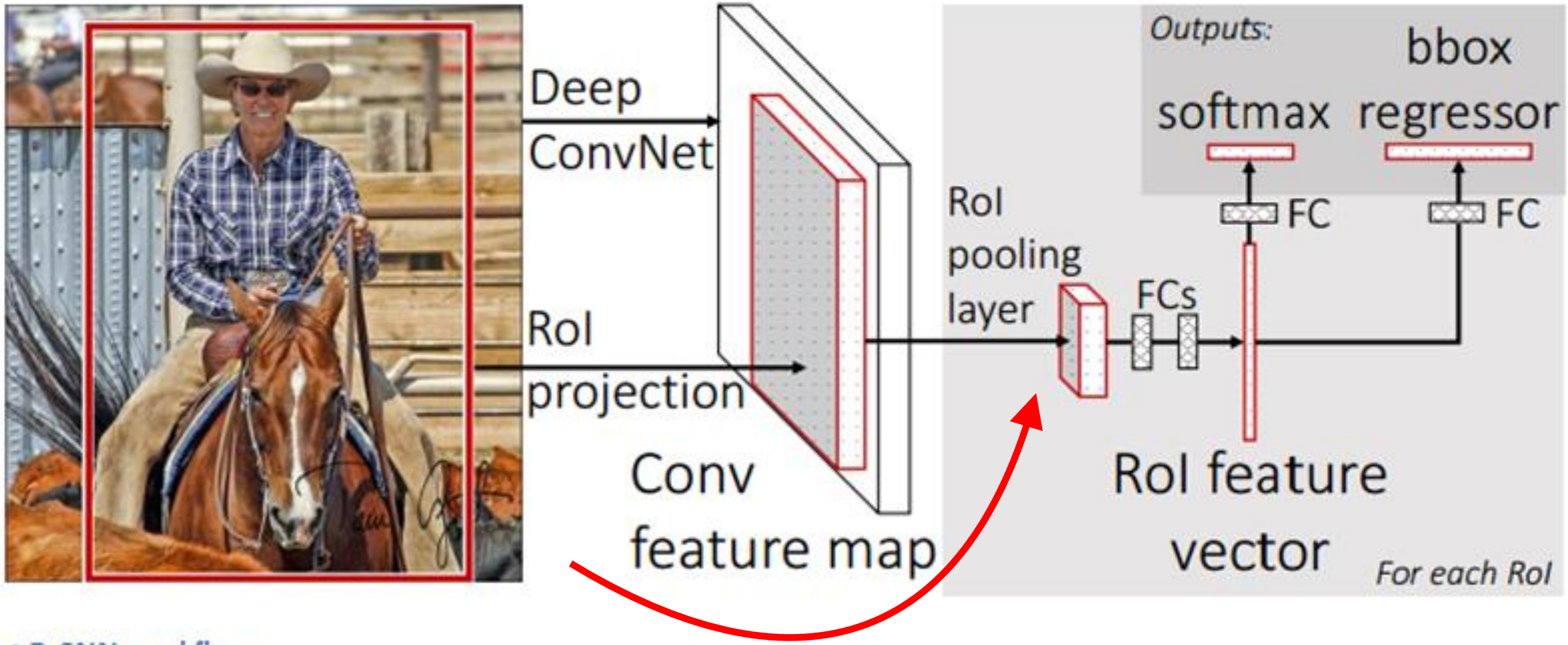
- A very common approach is to obtain candidate bounding boxes for objects, then classify the bounding boxes using a CNN



R-CNN workflow

FastER R-CNN (2015)

- Improves upon its predecessor by sharing the convolutions between ROIS



Fast R-CNN workflow

YOLO (2015-2016)

- YOLO predicts bounding boxes and classes for each cell in a grid

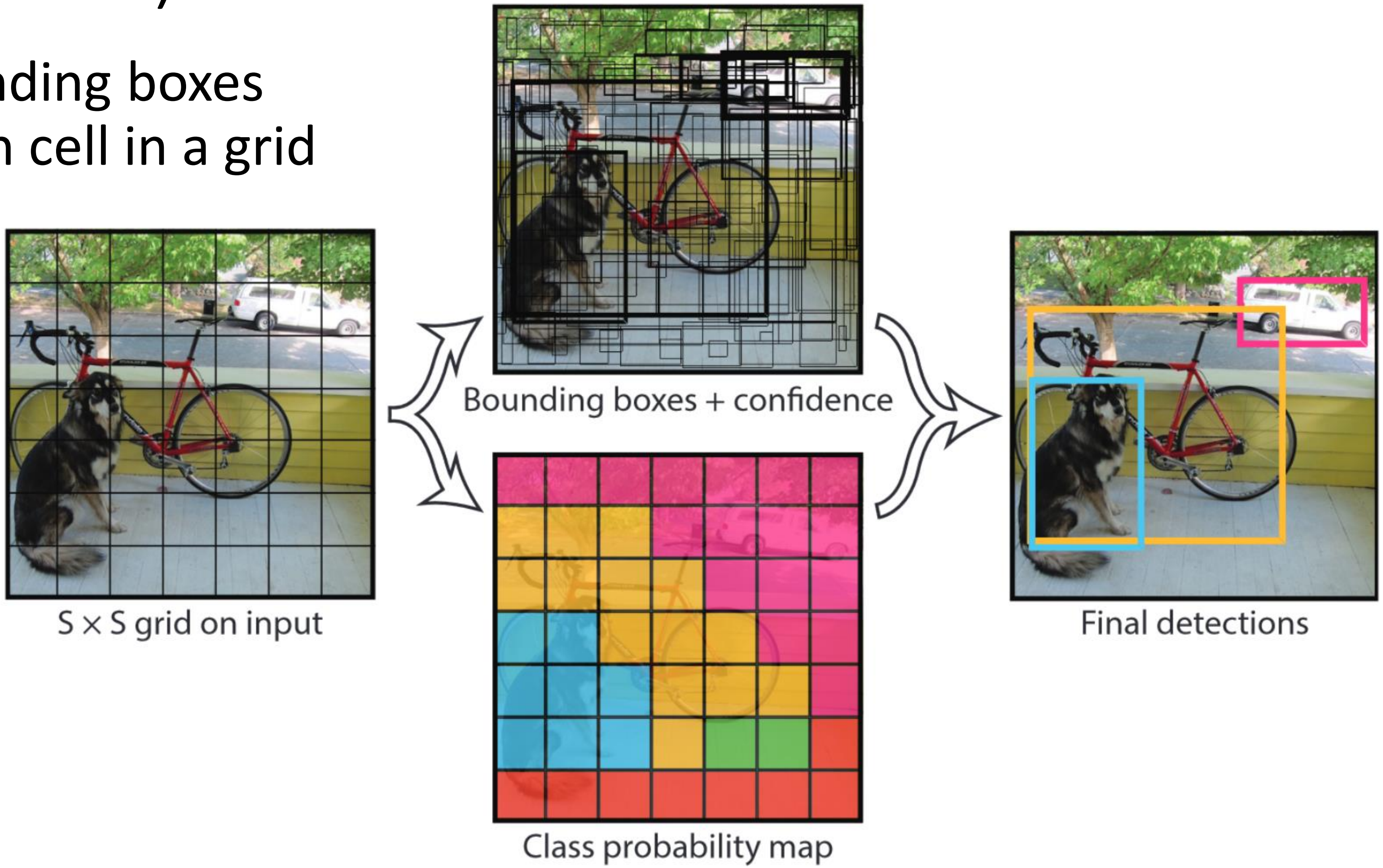
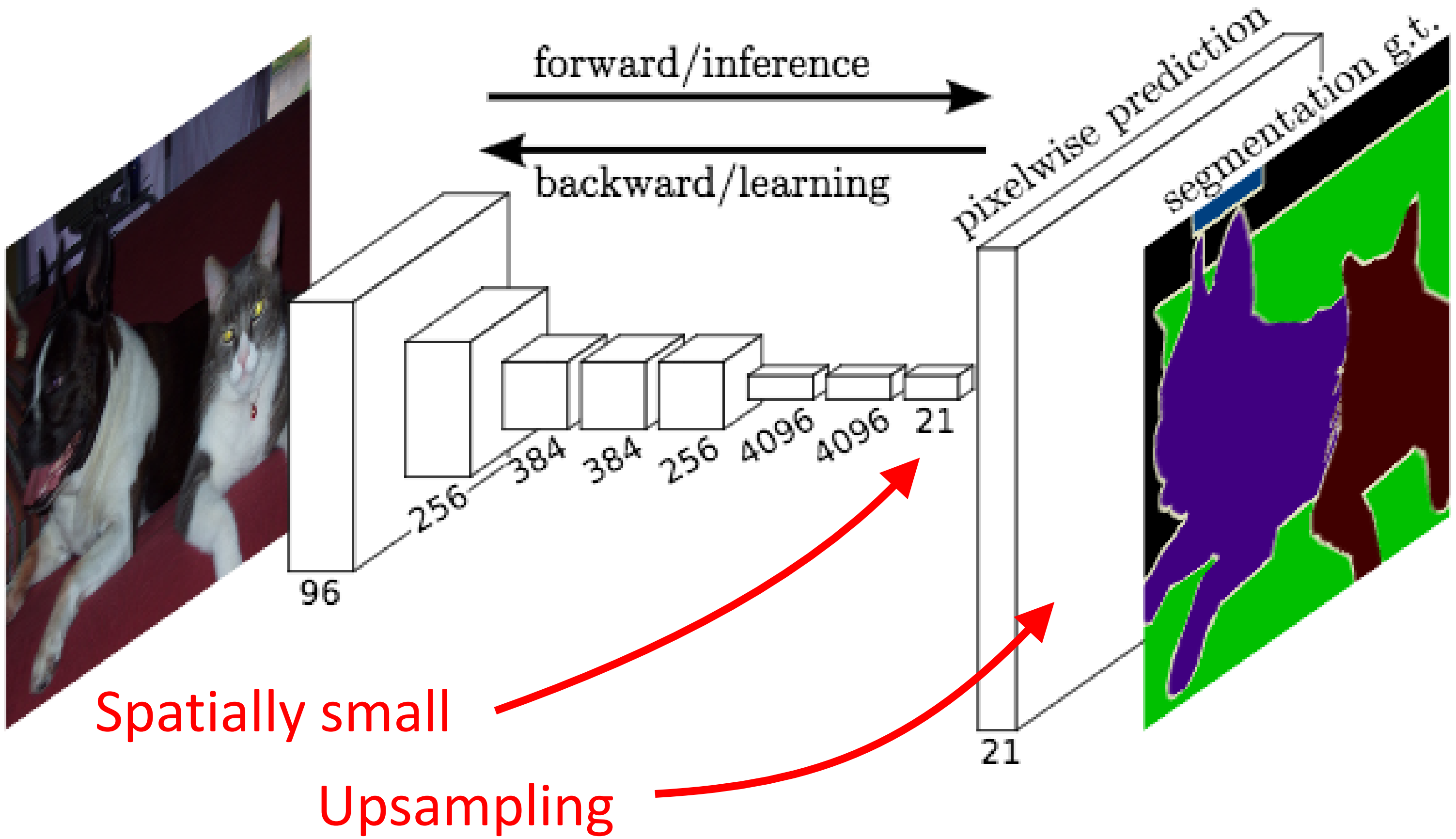


Image Segmentation

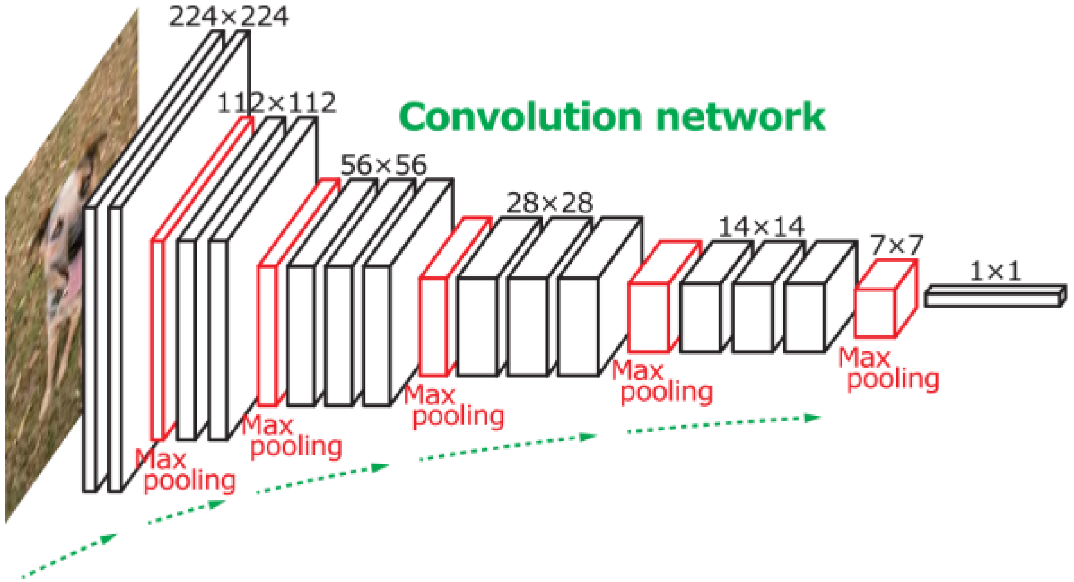
Fully-Connected Networks

- FCNs contain no fully-connected layers, instead they use 1x1 convolutions to predict 2D locations of objects



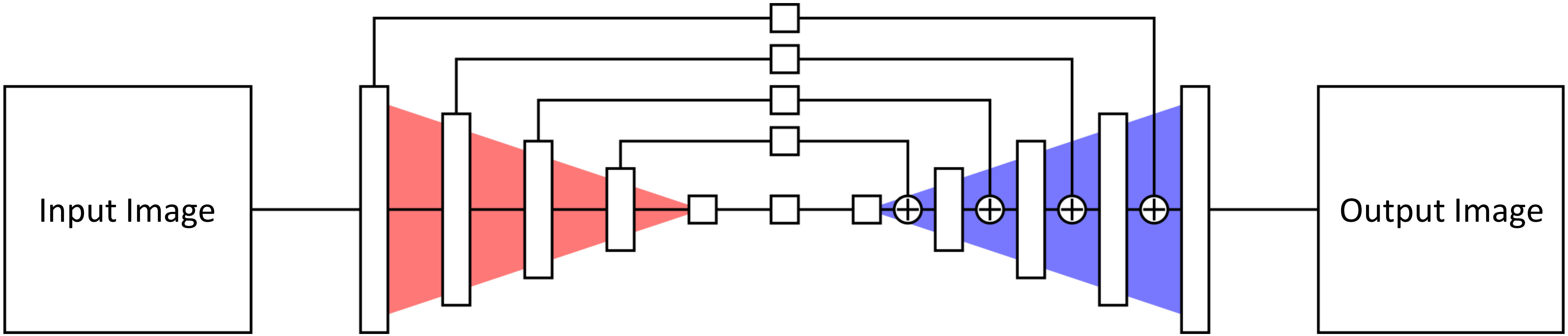
Encoder-Decoders

- Encoder-decoders are the commonly established name for this kind of network



Skip Connections

- Dropping down to low spatial resolution can harm the ability of these networks to recover detail



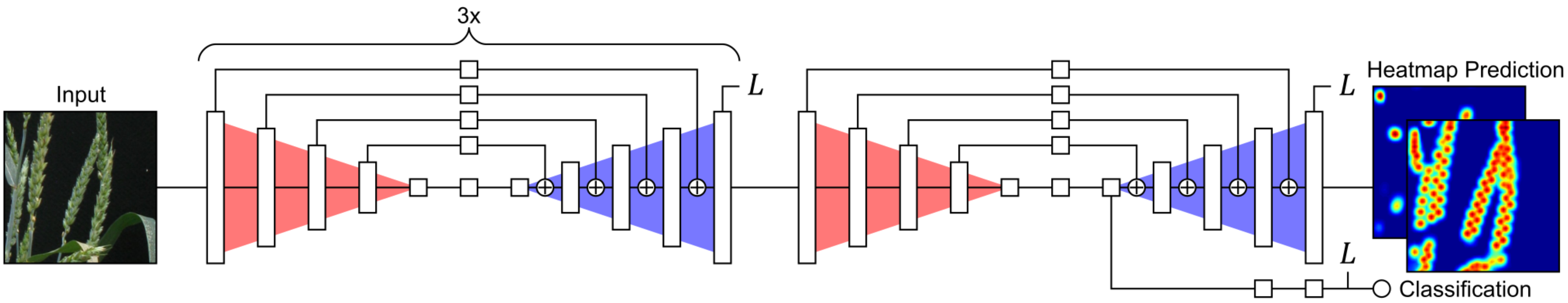
Mask-RCNN

- FasterRCNN + Semantic Segmentation

Heatmap Regression

- We can alter the loss function from BCE to MSE and move from pixel-wise classification (segmentation) to a regression problem

Stacked Hourglass



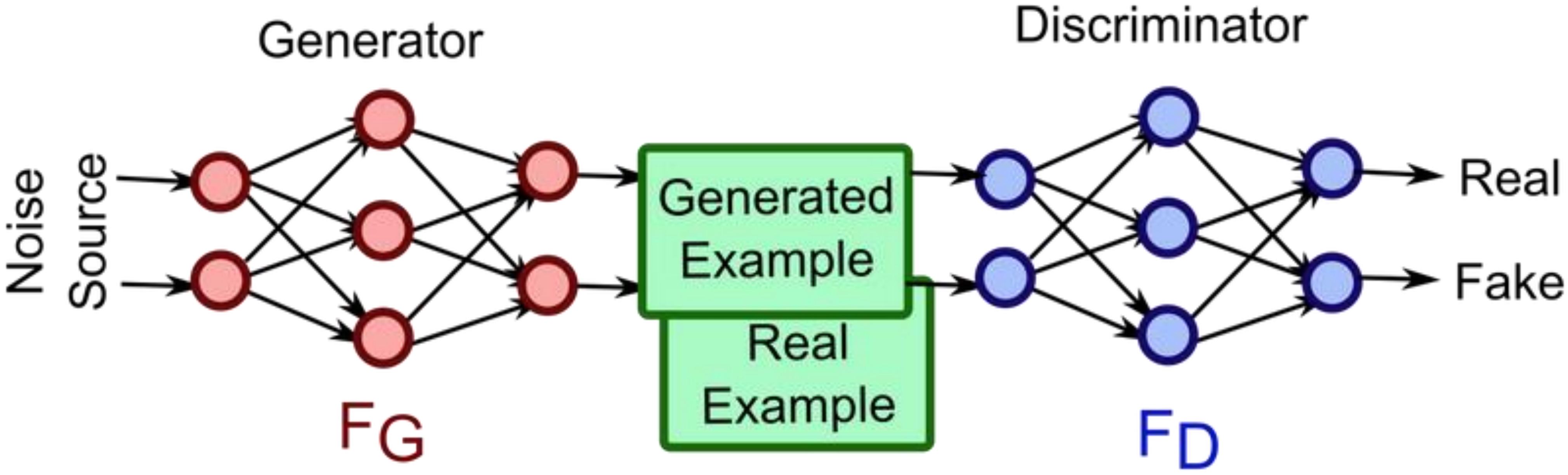
Wheat Feature Localisation

Generative Adversarial Networks

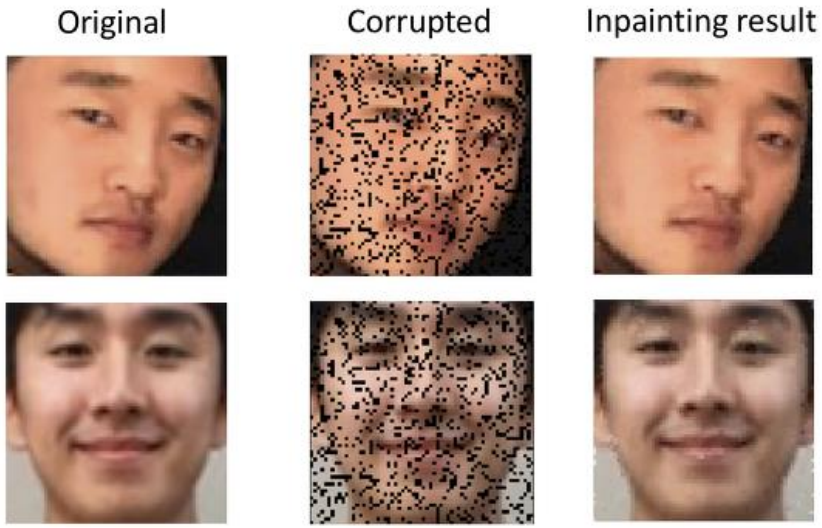
(In two slides..)

Generative Adversarial Networks

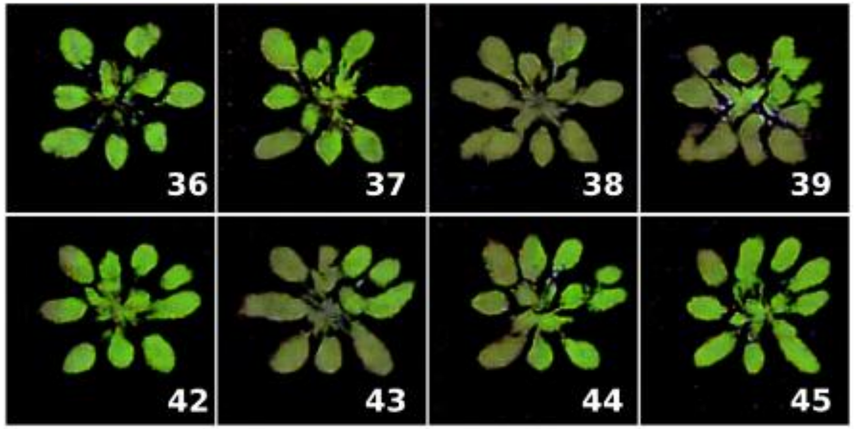
- Two networks trained to beat one another!



Uses of GANS


Original
Corrupted
Inpainting result

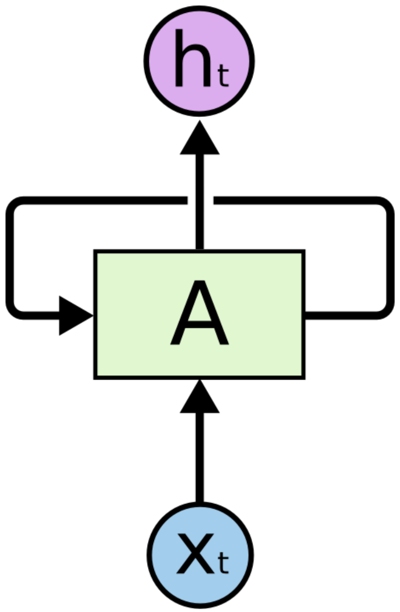

bicubic
(21.59dB/0.6423)
SRResNet
(23.53dB/0.7832)
SRGAN
(21.15dB/0.6868)
original


36
37
38
39
42
43
44
45

Recurrent Networks

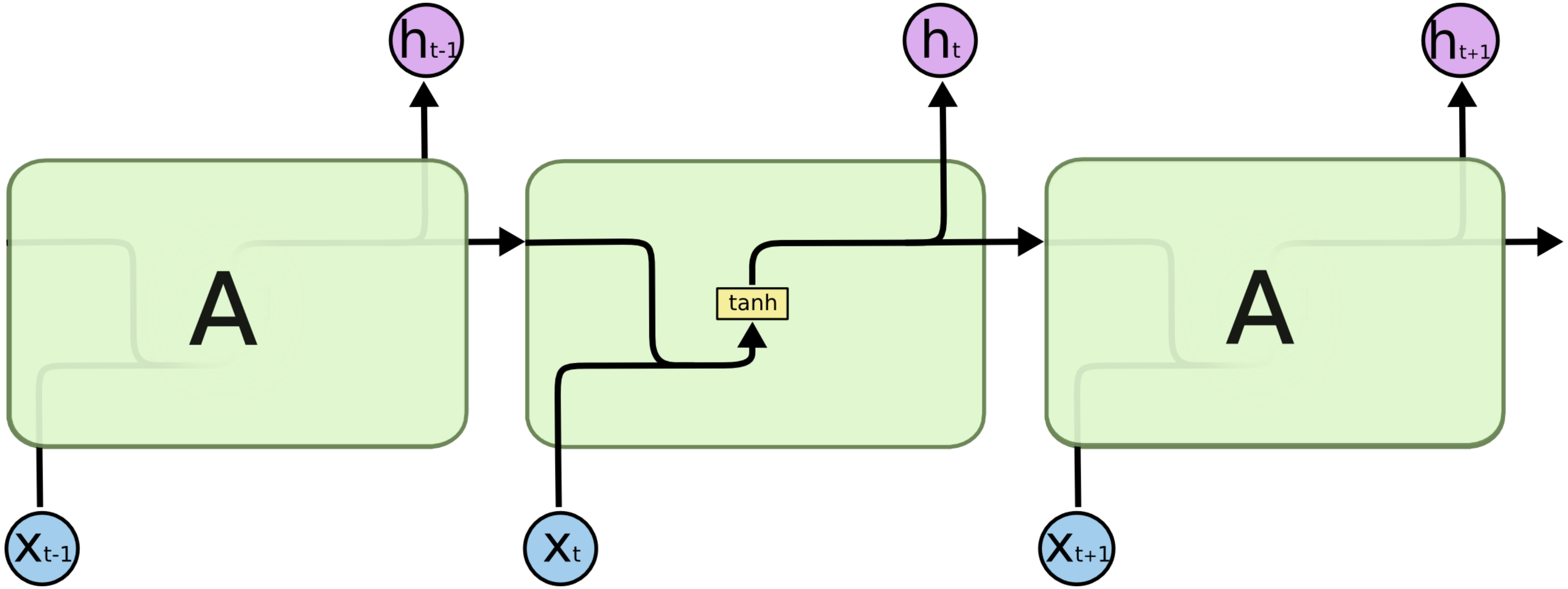
RNNs

- RNNs are neural networks that run on temporal sequences
- The activations of the hidden layer are passed back in at the next time step



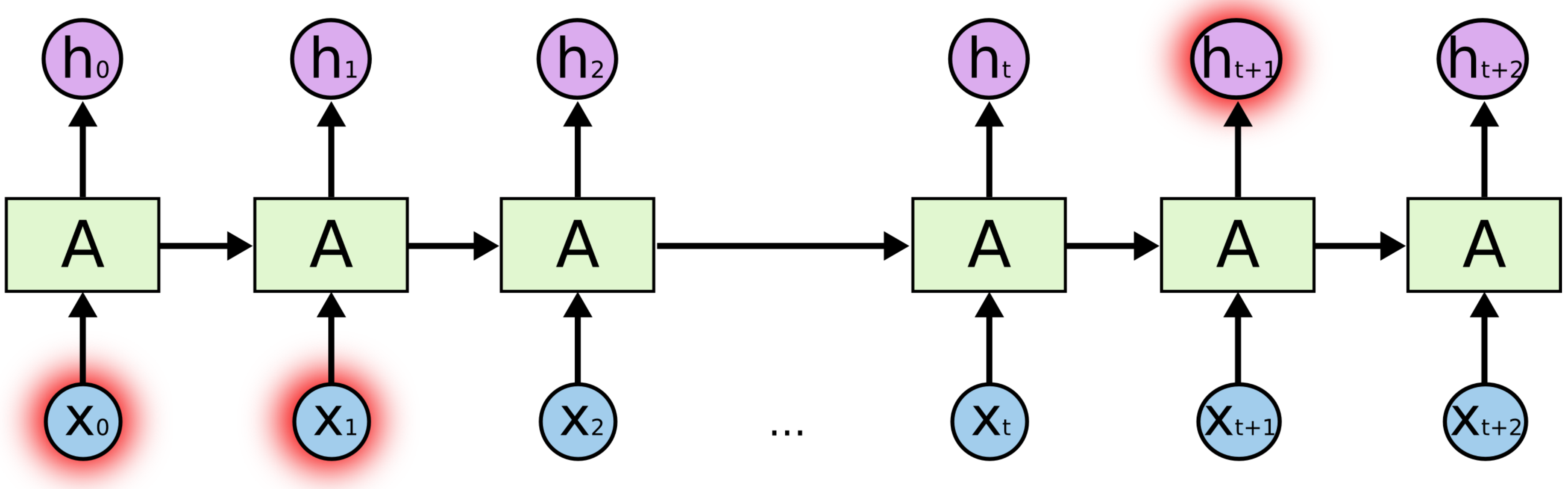
Inside an RNN

- There's really nothing much!



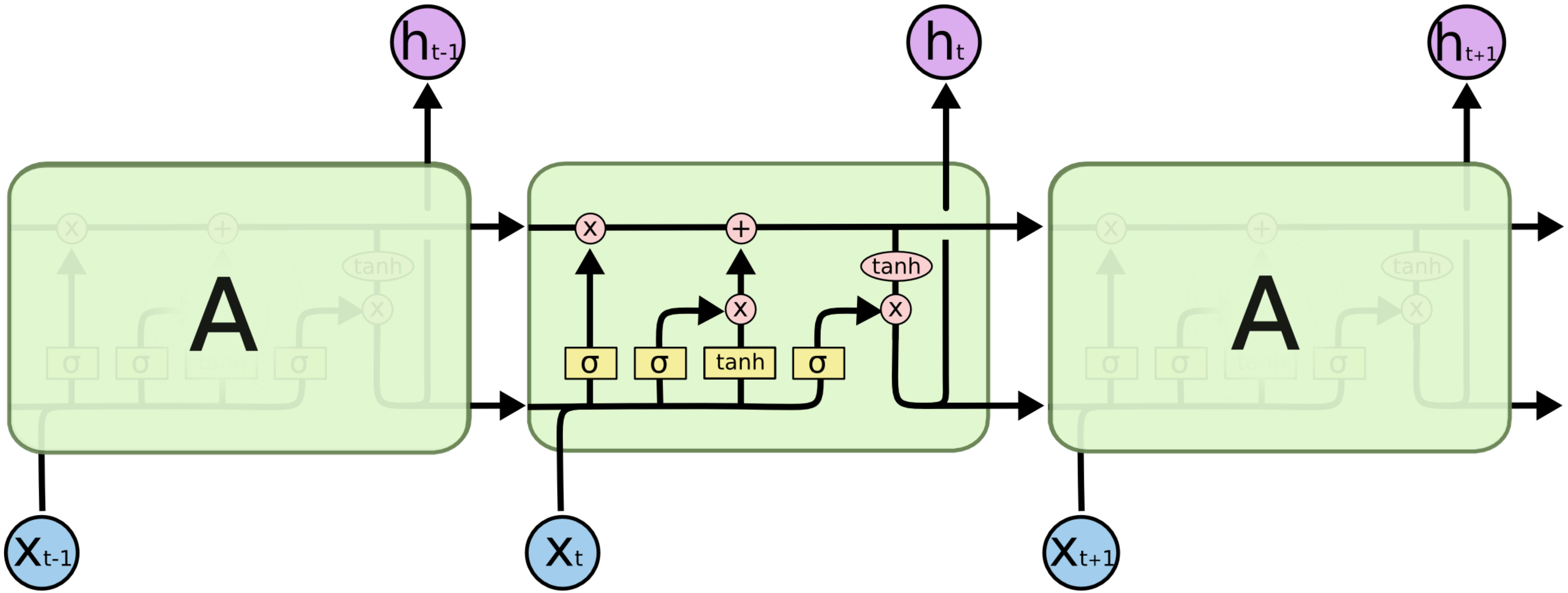
Problems with RNNs

- RNNs don't handle long-term dependencies well due to vanishing gradients



Long Short Term Memory Networks

- LSTMs are a more complex structure, ideally suited to longer sequences



Convolutional LSTMs

- LSTMs can be made 2D using convolution operations
- This example counts and segments leaves one at a time