Introduction to Deep Learning

Lecture 5
Modern CNNs

Maria Vakalopoulou & Stergios Christodoulidis

MICS Laboratory

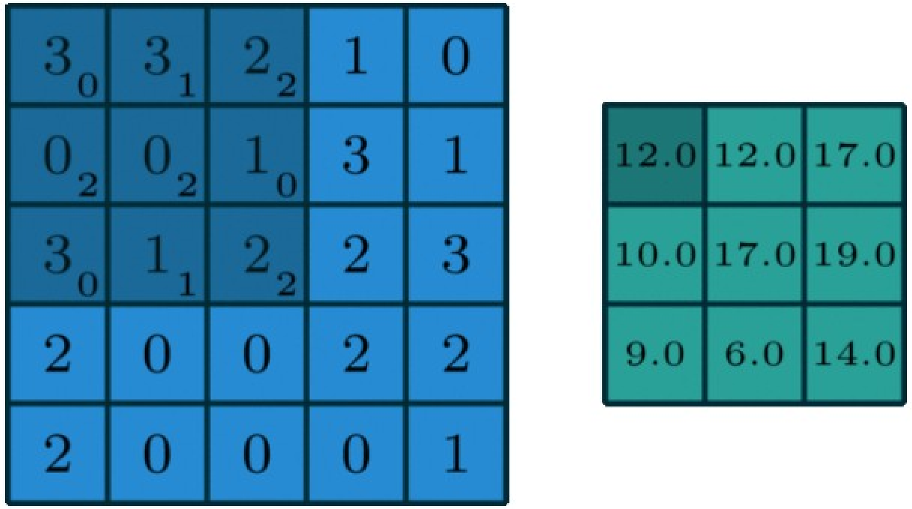
CentraleSupélec
Université Paris-Saclay

Wednesday, November 24, 2021

Last Lecture

2D Discrete Convolution

$$C[i,j] \triangleq \sum_{m=0}^{(M_a-1)} \sum_{n=0}^{(N_a-1)} f[m,n] \cdot g[i-m, j-n]$$



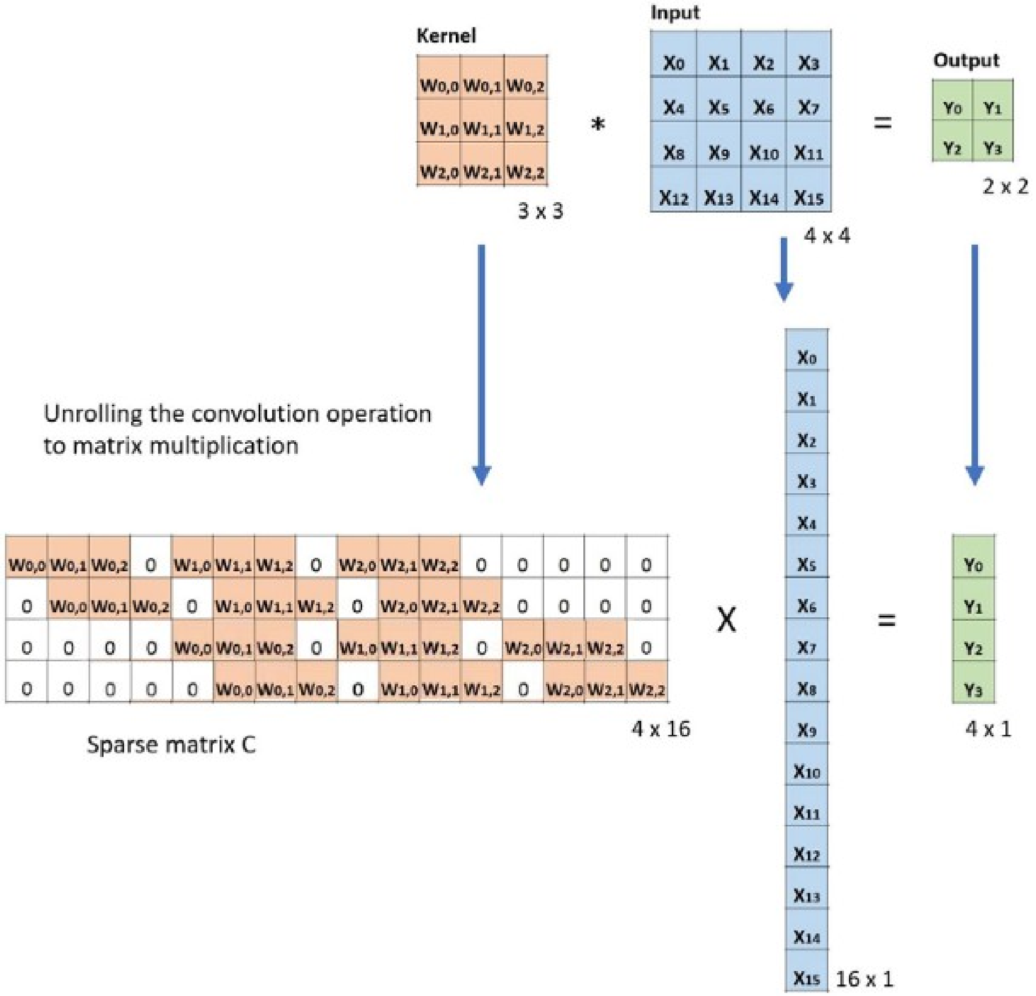
[Dumoulin & Visin, A guide to convolution arithmetic for deep learning (2018)]

How can convolutions be used in Deep Learning?

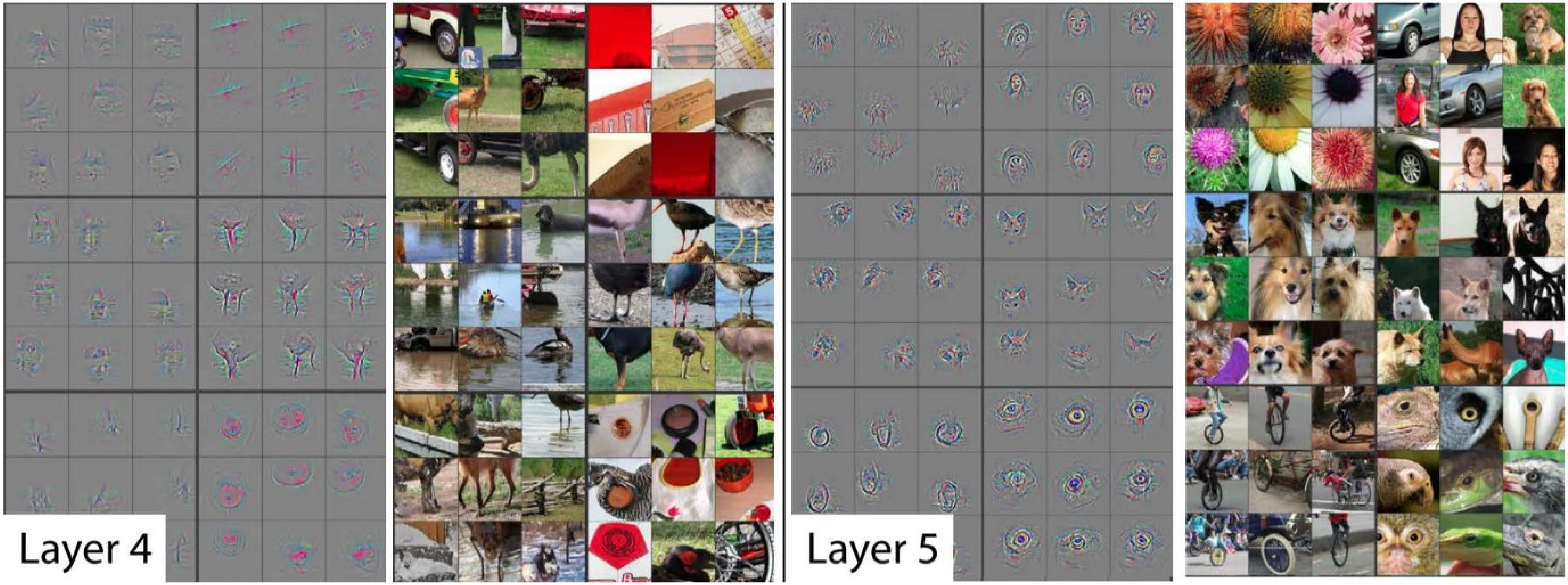
- **Convolution kernels can be trainable.**
- Essentially it can be performed by applying a dot product (Toepliz matrix transformation).
- The convolution operation can be applied in any number of dimensions (1D, 2D, 3D, … etc.)
- The gradient w.r.t. it's parameters and inputs:

$$\frac{\partial a_{rc}}{\partial w_{ij}} = x_{r-i,c-j}$$

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \sum_{r} \sum_{c} \frac{\partial \mathcal{L}}{\partial a_{rc}} x_{r-i,c-j}$$



What Input Maximizes Feature Map Outputs?



[M. Zeiler & Fergus, Visualizing and Understanding Convolutional Networks, ECCV, (2014)]

Transfer Learning

- Assume two datasets S and T
- Dataset S (source) is
  - Fully annotated, plenty of images
  - We can build a model h_S
- Dataset T (target) is
  - Not a much annotated, or much fewer images
  - The annotations of S do not need to overlap with T
- We can use the model h_S to learn a better h_T
- This is called transfer learning

(Source, e.g. ImageNet 1M samples)

(Target, 1K samples)

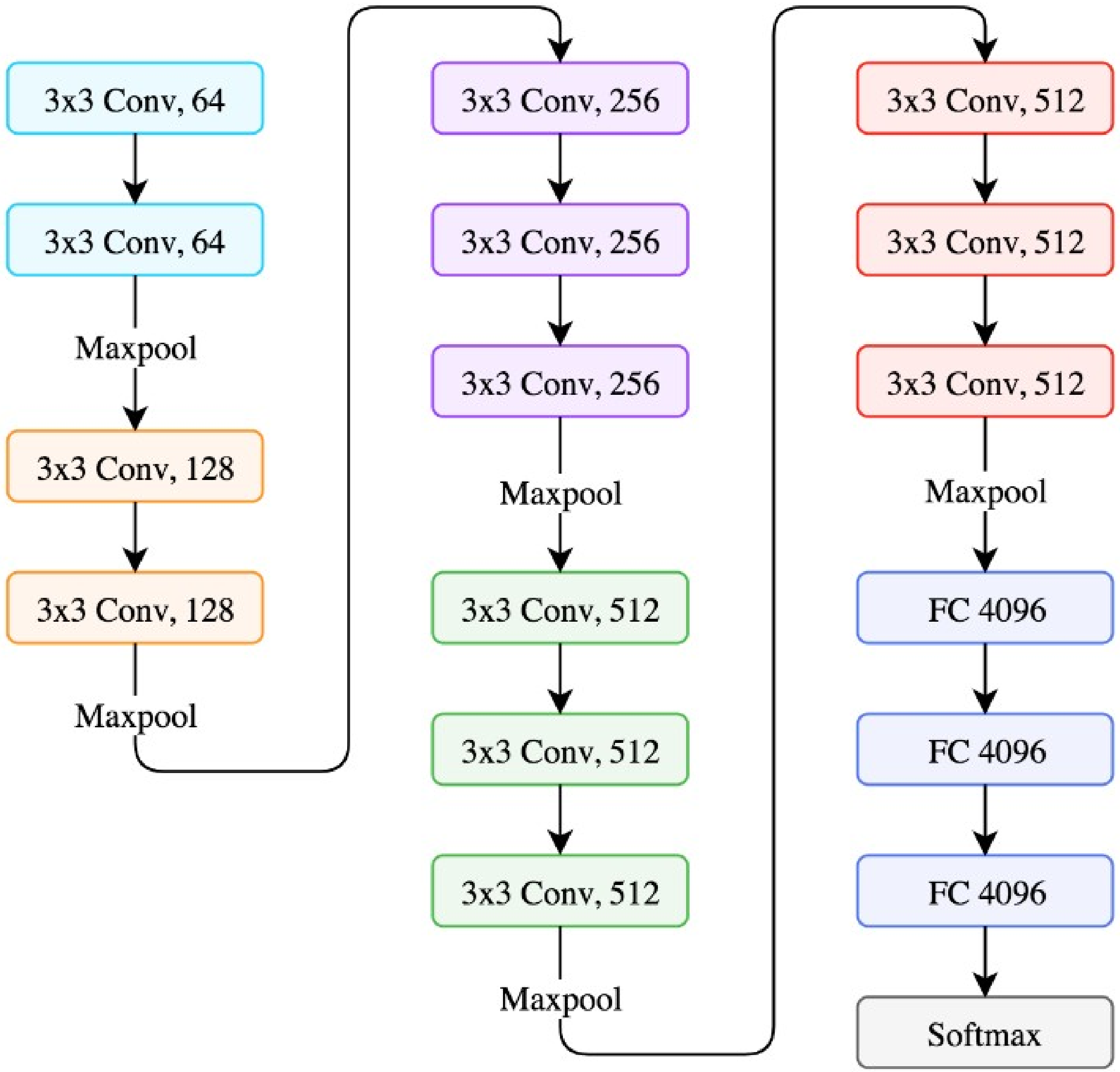
Today’s Lecture

Today’s Lecture

- Modern/ Popular CNN architectures
- Go deeper on what makes them tick
  - What makes them different

VGG16

- 7.3% error rate in ImageNet
- Compared to 18.2% of Alexnet



VGG16

- Input size: 224 x 224
- Filter sizes: 3 x 3
- Convolution stride: 1
  - Spatial resolution preserved
- Padding: 1
- Max pooling: 2 x 2 with a stride of 2
- ReLU activations
- No fancy input normalizations
  - No local response normalizations
- Although deeper, number of weights in not exploding

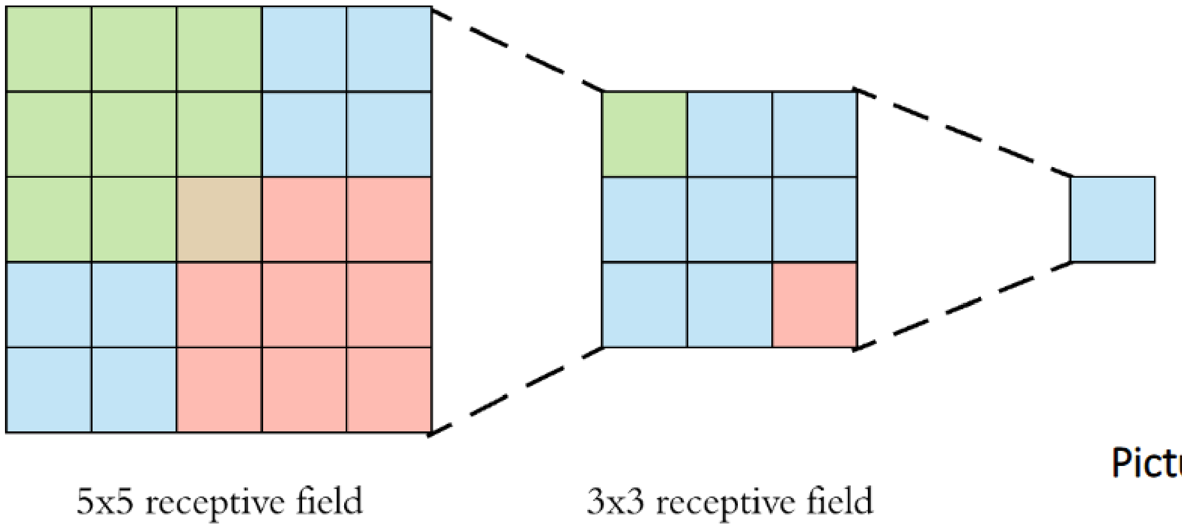
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	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					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Why 3x3 filters?

- The smallest possible filter to capture the “up”, “down”, “left”, “right”
- Two 3x3 filters have the receptive field of one 5 x 5
- Three 3x3 filters have the receptive field of ...



Picture credit: [Arden Dertat]

Why 3x3 filters?

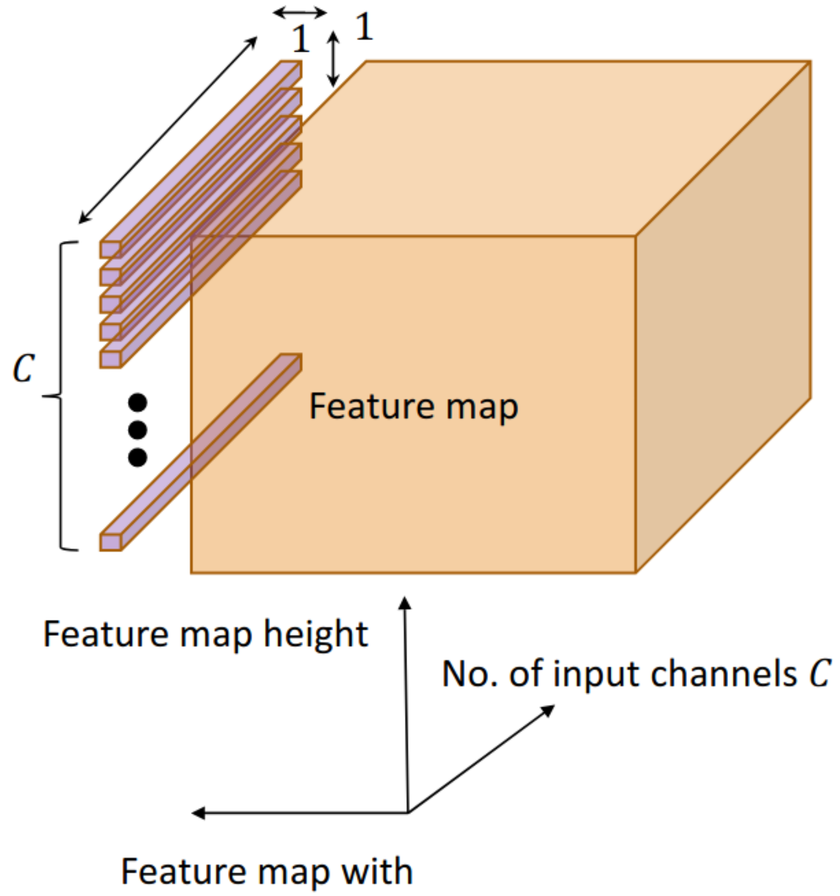
- The smallest possible filter to capture the “up”, “down”, “left”, “right”
- Two 3x3 filters have the receptive field of one 5 x 5
- Three 3x3 filters have the receptive field of 7 x 7
- 1 large filter can be replaced by a deeper stack of successive smaller filters
- Benefit?

Why 3x3 filters?

- The smallest possible filter to capture the “up”, “down”, “left”, “right”
- Two 3x3 filters have the receptive field of one 5 x 5
- Three 3x3 filters have the receptive field of 7 x 7
- 1 large filter can be replaced by a deeper stack of successive smaller filters
- Benefit?
- Three more nonlinearities for the same “size” of pattern learning
- Also fewer parameters and regularization
  - (3x3xC)x3 = 27 C ,   (7x7xC)x1 = 49C
- Conclusion: 1 large filter can be replaced by a deeper, potentially more powerful, stack of successive smaller filters

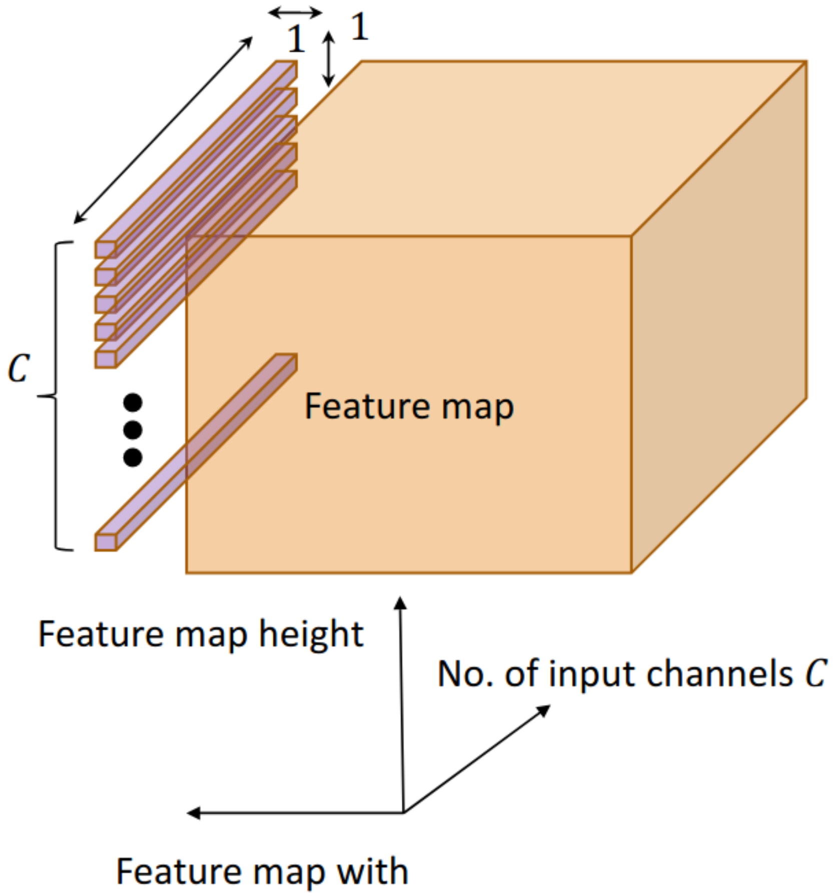
Even smaller filters?

- Also 1x1 filters are used
- Followed by a nonlinearity
- Why?



Even smaller filters?

- Also 1x1 filters are used
- Followed by a nonlinearity
- Why?
- Increasing nonlinearities without affecting receptive field sizes
  - Linear transformation of the input channels



Training

- Batch size: 256
- SGD with momentum= 0.9
- Weight decay λ = 5 10^{-4}
- Dropout on first two fully connected layers
- Learning rare η_0=10^{-2}, then decreased by factor of 10 when validation accuracy stopped improving
  - Three times this learning rate decrease

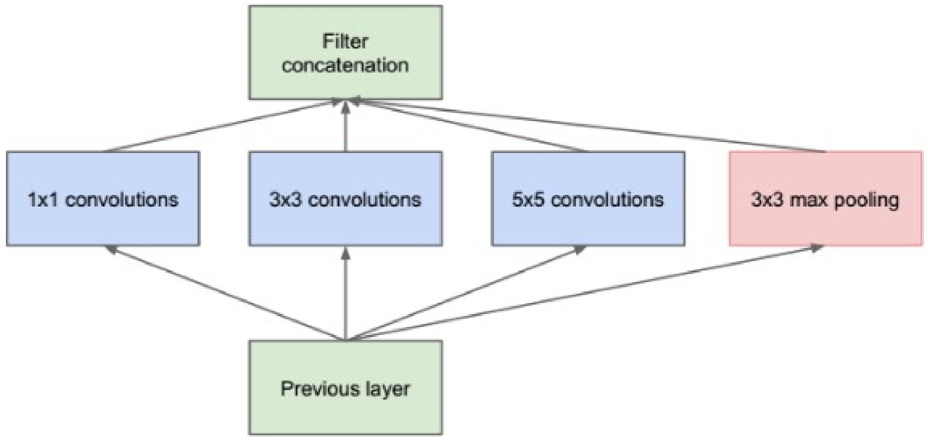
Inception

- Basic idea
  - Salient parts have great variation in sizes
  - Naively stacking convolutional operations is expensive
  - Very deep nets are prone to overfitting

Picture credit: Bharath Raj

Inception

- Module
  - Multiple kernel filters of different sizes (1x1,3x3,5x5)
    - Naive version
  - Problem?

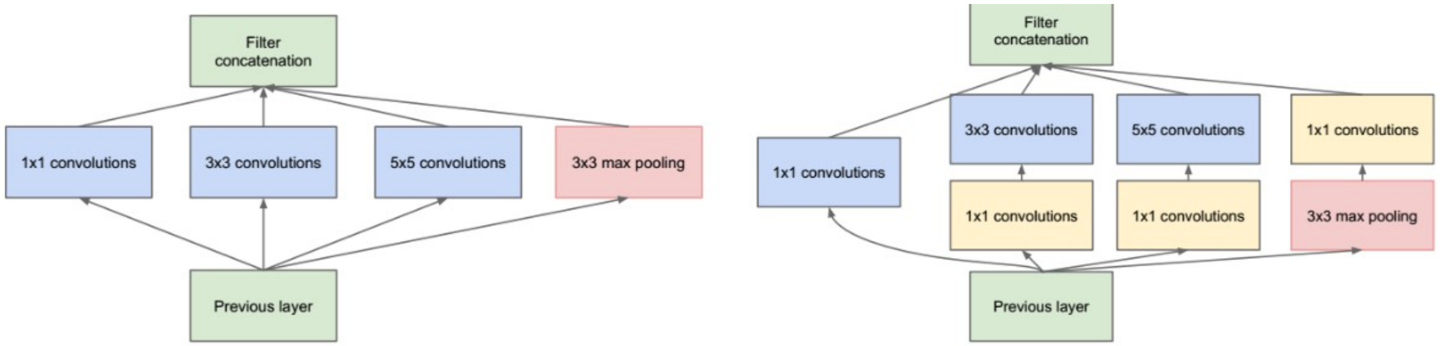


(a) Inception module, naïve version

Picture credit: Bharath Raj

Inception

- Module
  - Multiple kernel filters of different sizes (1x1,3x3,5x5)
    - Naive version
  - Problem?
    - Very expensive!
  - Add intermediate 1x1 convolutions



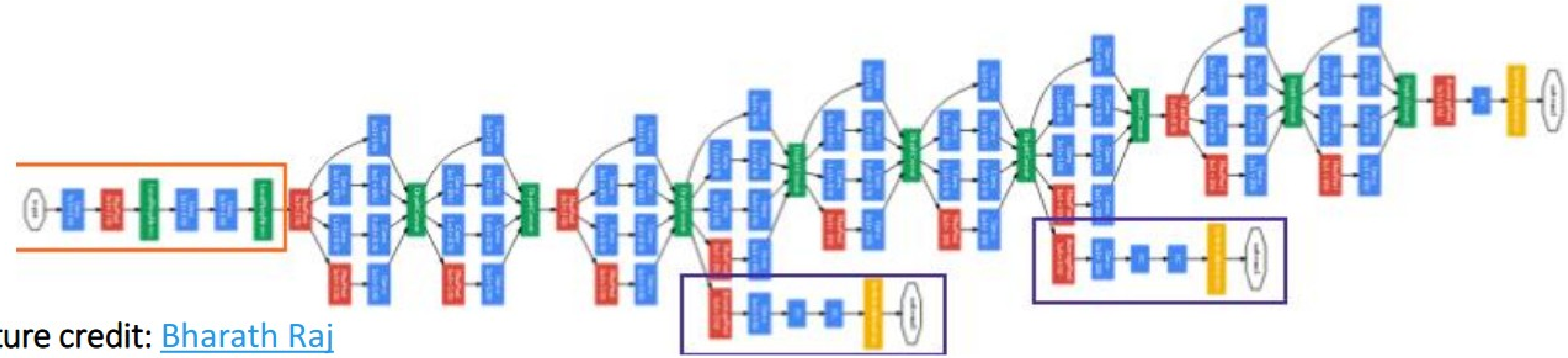
Picture credit: Bharath Raj

(a) Inception module, naïve version

(b) Inception module with dimension reductions

Architecture

- 9 Inception Modules
- 22 layers deep (27 with the pooling layers)
- Global average pooling at the end of last Inception Module
- 6.67% Imagenet error, compared to 18.2% of Alexnet



Picture credit: [Bharath Raj]

Main Problem: Vanishing gradients

- The network was too deep (at the time)
- Roughly speaking, backprop is lots of matrix multiplications

$$\frac{\partial \mathcal{L}}{\partial w^l} = \frac{\partial \mathcal{L}}{\partial a^L} \cdot \frac{\partial a^L}{\partial a^{L-1}} \cdot \frac{\partial a^{L-1}}{\partial a^{L-2}} \cdot \ \dots \cdot \frac{\partial a^l}{\partial w^l}$$

- Many of intermediate terms <1 → the final $\frac{\partial \mathcal{L}}{\partial w^l}$ gets extremely small
- Extremely small gradient → ?

Main Problem: Vanishing gradients

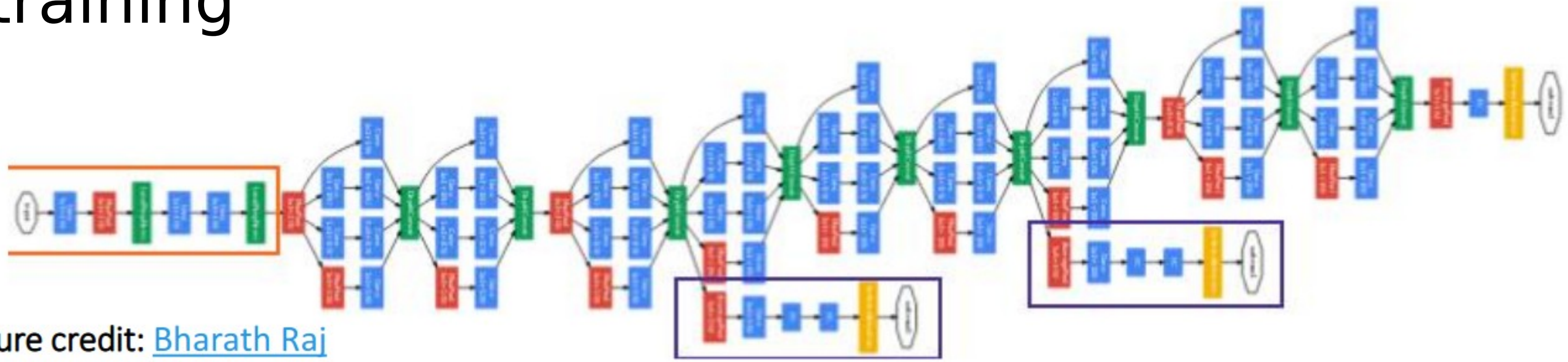
- The network was too deep (at the time)
- Roughly speaking, backprop is lots of matrix multiplications

$$\frac{\partial \mathcal{L}}{\partial w^l} = \frac{\partial \mathcal{L}}{\partial a^L} \cdot \frac{\partial a^L}{\partial a^{L-1}} \cdot \frac{\partial a^{L-1}}{\partial a^{L-2}} \cdot \ \dots \cdot \frac{\partial a^l}{\partial w^l}$$

- Many of intermediate terms <1 → the final $\frac{\partial \mathcal{L}}{\partial w^l}$ gets extremely small
- Extremely small gradient → Extremely slow learning

Architecture

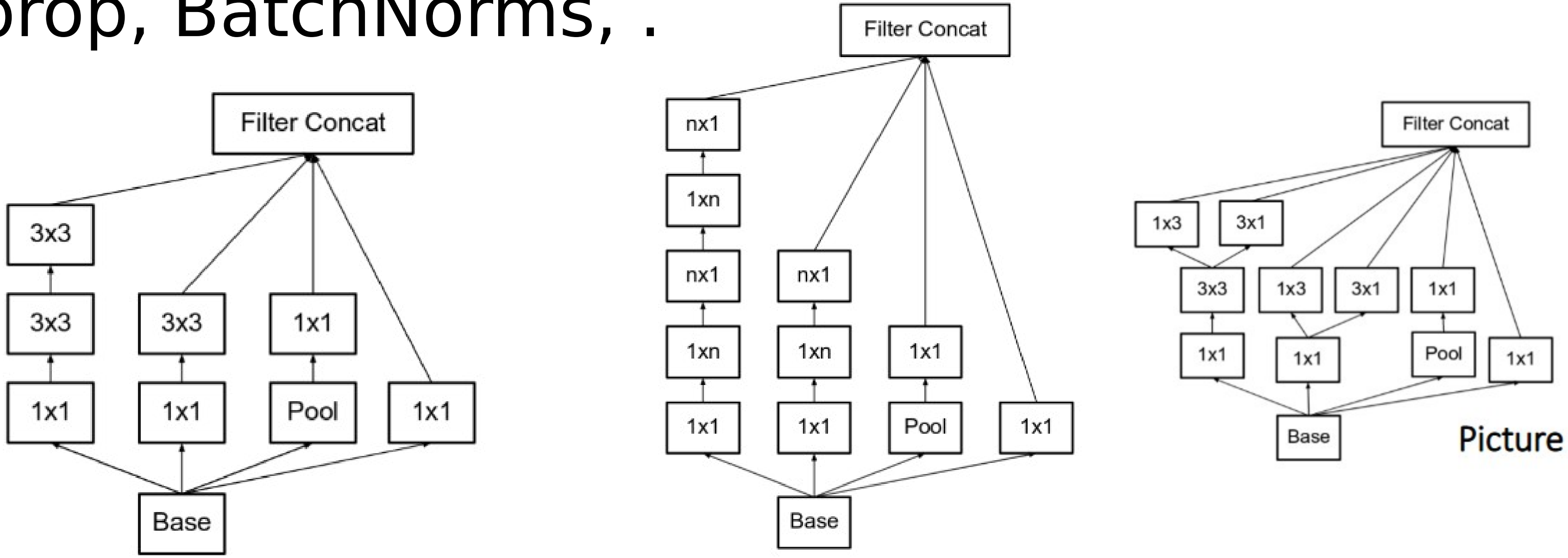
- 9 Inception Modules
- 22 layers deep (27 with the pooling layers)
- Global average pooling at the end of last Inception Module
- Because of the increased depth → Vanishing gradients
- Inception solution to vanishing gradients: intermediate classifiers
    - Removed after training



Picture credit: [Bharath Raj]

Inceptions v2, v3, v4

- Factorize 5x5 in two 3x3 filters
- Factorize nxn in two nx1 and 1xn filters (quite a lot cheaper)
- Make nets wider
- RMSprop, BatchNorms, .



Picture credit: [Bharath Raj]

ResNets, DenseNets, HighwayNets

Some facts

- The first truly Deep Network, going deeper than 1000 layers
- More importantly the first Deep Architecture that proposed a novel concept on how to gracefully go deeper than a few dozen layers
  - Not simply getting more GPUs, more training time etc
- Smashed Imagenet with a 3.57% error (in ensembles)
- Won all object classification, detection, segmentation, etc. challenges

Hypothesis

- Hypothesis: Is it possible to have a very deep network at least as accurate as averagely deep networks?
- Thought experiment: Let's assume two Convnets A, B. They are almost identical, in that B is same as A, with extra "identity" layers. Since identity layers pass the information unchanged, the errors of the two networks should...

Hypothesis

- Hypothesis: Is it possible to have a very deep network at least as accurate as averagely deep networks?
- Thought experiment: Let's assume two Convnets A, B. They are almost identical, in that B is same as A, with extra "identity" layers. Since identity layers pass the information unchanged, the errors of the two networks should be similar. Thus, there is a Convnet B, which is at least as good as Convnet A w.r.t. training error

Testing the hypothesis

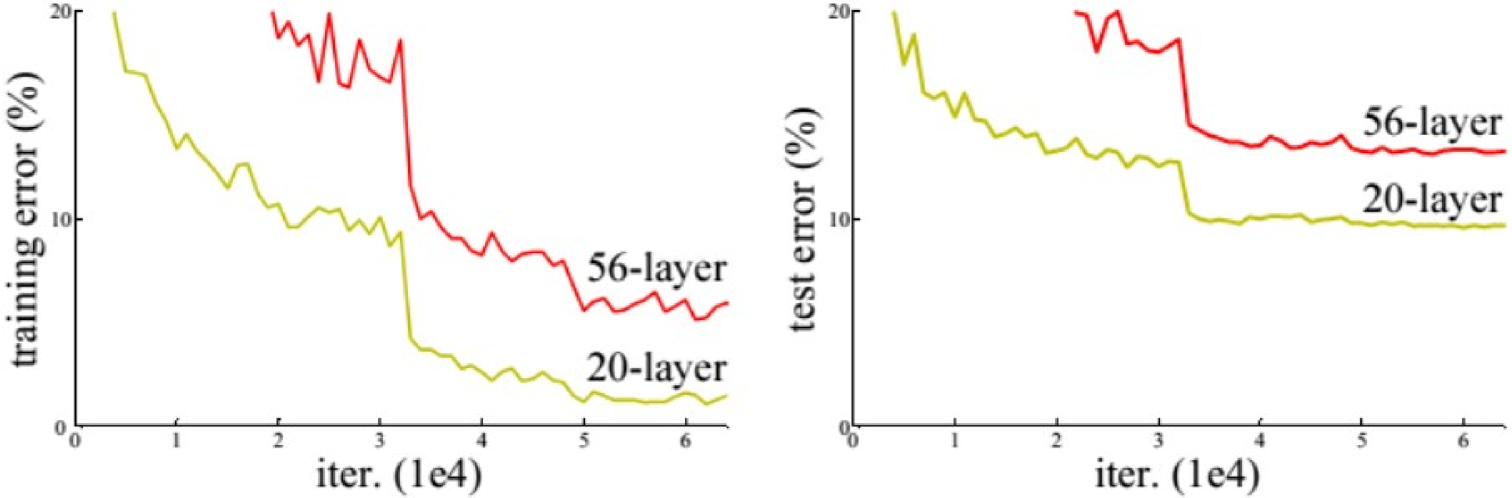


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

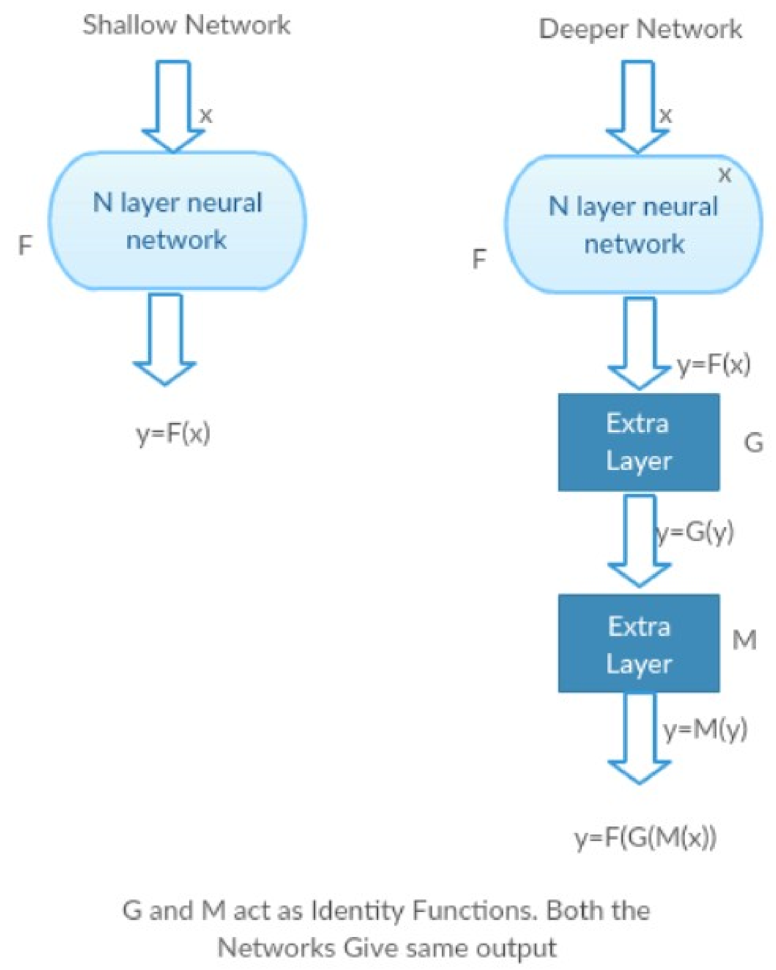
Testing the hypothesis

- Adding identity layers increases training error!!
  - Training error, not testing error
- Performance degradation not caused by overfitting
  - Just the optimization task is harder
- Assuming optimizers are doing their job fine, it appears that not all networks are the same as easy to optimize

$\mathcal{L}_A^{train} < \mathcal{L}_B^{train}$!!!

What is the problem?

- Very deep networks stop learning after a bit
  - An accuracy is reached, then the network saturates
- Signal gets lost through so many layers



Basic Idea (Residual idea, intuitively)

- Let’s say we have the neural network nonlinear $\mathrm{a} = F(x)$
- Easier to learn a function $\mathrm{a} = F(x)$ to model differences $\mathrm{a} \sim \delta \mathrm{y}$ than to model absolutes $\mathrm{a} \sim \mathrm{y}$
  - Think of it like in input normalization → you normalize around 0
  - Think of it like in regression → you model differences around the mean value
- So, ask the neural network to explicitly model different mapping

$$F(x) = H(x) - x \Rightarrow H(x) = F(x) + x$$

- F(x) are the stacked nonlinearities
- X is the input to the nonlinear layer

ResNet block

- $H(x) = F(x) + x$
- If dimensions don't match
  - Either zero padding
  - Or a projection layer to match dimensions

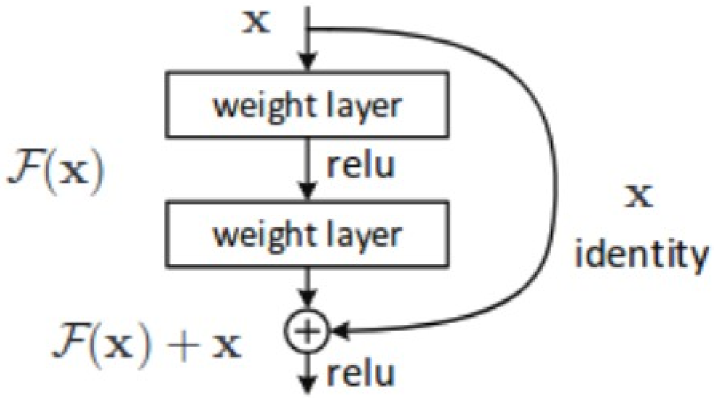
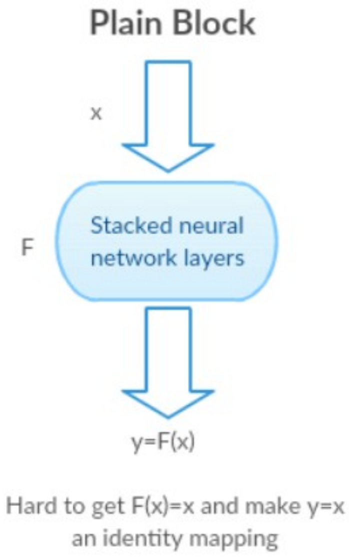
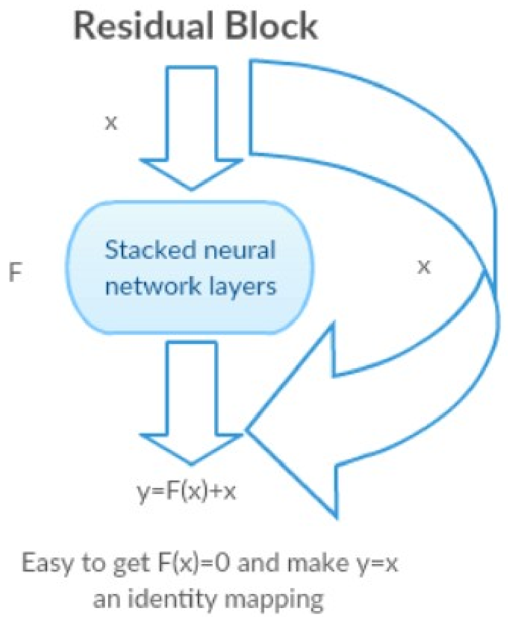


Figure 2. Residual learning: a building block.





Testing Hypothesis

- Without the residual connections deeper networks attain worse scores

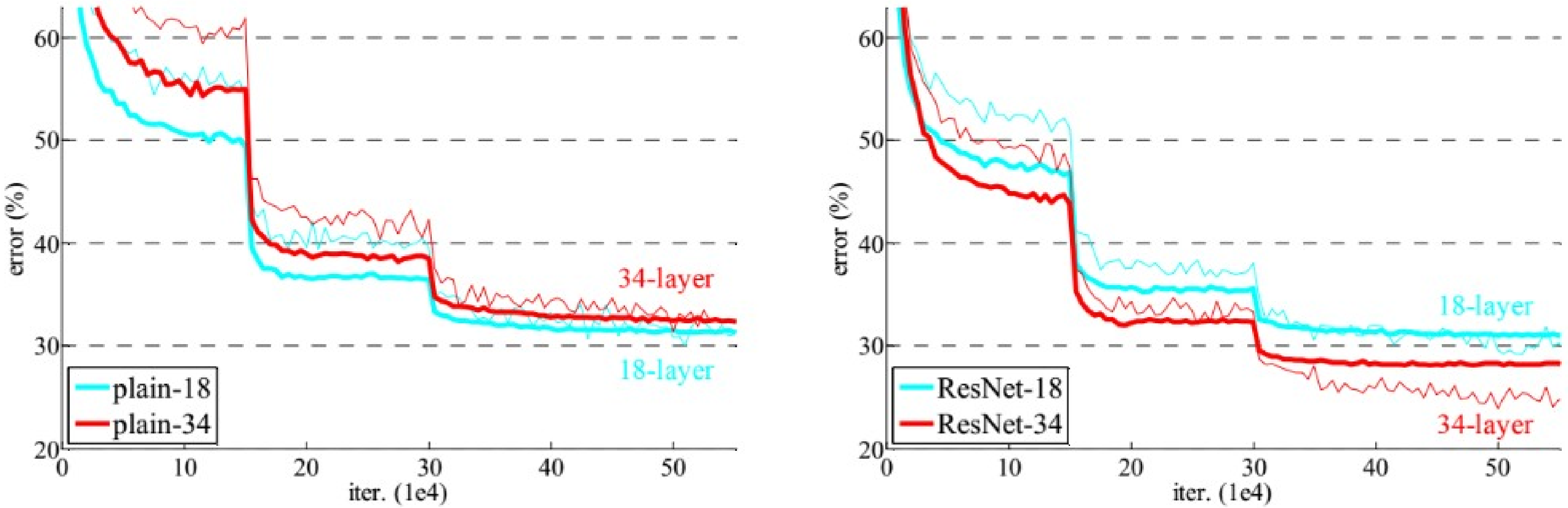


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

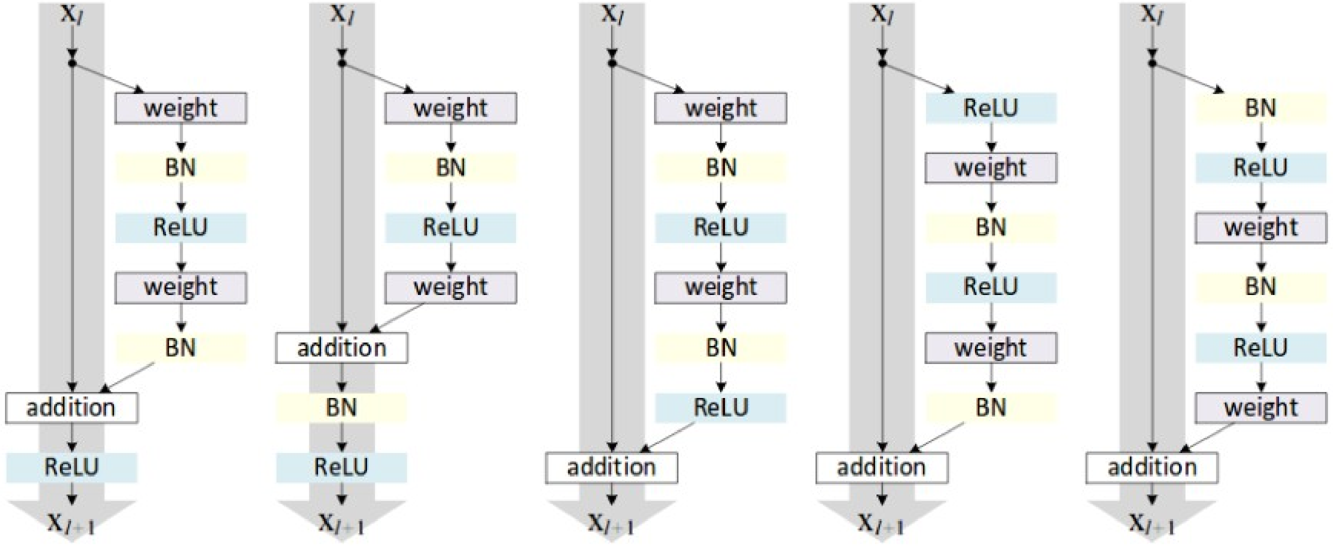
ResNet breaks records

- Ridiculously low error in ImageNet
- Up to 1000 layers ResNets trained
  - Previous deepest network ~30-40 layers on simple datasets

method	top-5 err. (**test**)
VGG [41] (ILSVRC'14)	7.32
GoogLeNet [44] (ILSVRC'14)	6.66
VGG [41] (v5)	6.8
PReLU-net [13]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

Table 5. Error rates (%) of **ensembles**. The top-5 error is on the test set of ImageNet and reported by the test server.

ResNet architecture & ResNeXt



(a) original (b) BN after addition (c) ReLU before addition (d) ReLU-only pre-activation (e) **full pre-activation**

case	Fig.	ResNet-110	ResNet-164
original Residual Unit [1]	Fig. 4(a)	6.61	5.93
BN after addition	Fig. 4(b)	8.17	6.50
ReLU before addition	Fig. 4(c)	7.84	6.14
ReLU-only pre-activation	Fig. 4(d)	6.71	5.91
full pre-activation	Fig. 4(e)	**6.37**	**5.46**

ResNeXt

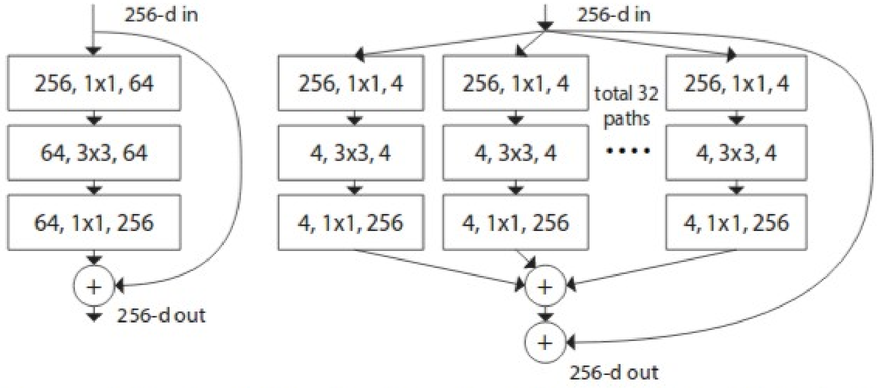


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

	setting	top-1 err (%)	top-5 err (%)
1× complexity references:			
ResNet-101	1 × 64d	22.0	6.0
ResNeXt-101	32 × 4d	21.2	5.6
2× complexity models follow:			
ResNet-**200** [15]	1 × 64d	21.7	5.8
ResNet-101, wider	1 × **100**d	21.3	5.7
ResNeXt-101	**2** × 64d	20.7	5.5
ResNeXt-101	**64** × 4d	**20.4**	**5.3**

Table 4. Comparisons on ImageNet-1K when the number of FLOPs is increased to 2× of ResNet-101's. The error rate is evaluated on the single crop of 224×224 pixels. The highlighted factors are the factors that increase complexity.

Some observations

- BatchNorms absolutely necessary because of vanishing gradients
- Networks with skip connections (like ResNets) converge faster than the same network without skip connections
- Identity shortcuts cheaper and almost equal to project shortcuts

HighwayNet

- Similar to ResNets, only introducing a gate with learnable parameters on the importance of each skip connection

$$y = H(x, W_H) \cdot T(x, W_T) + x \cdot (1 - T(x, W_T))$$

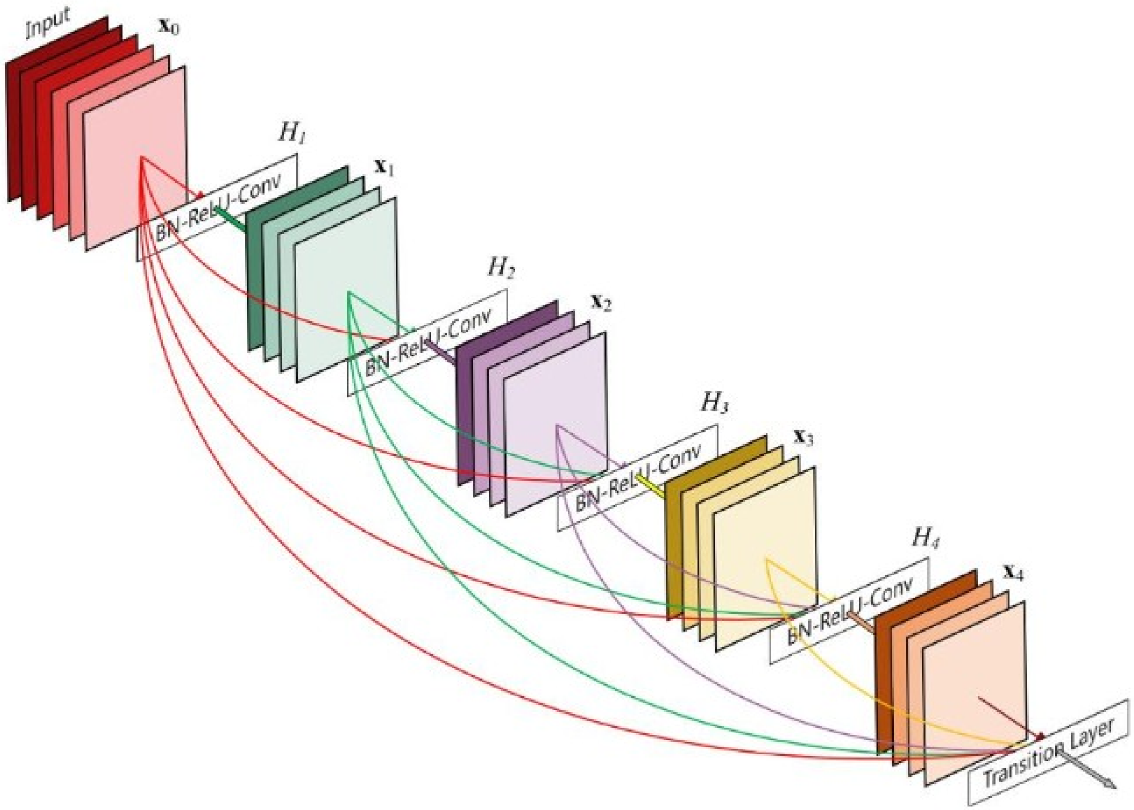
- Similar to…. LSTMS as we will see later

DenseNet

- Add skip connections to multiple forward layers

$$y = h(x_l, x_{l-1}, \dots, x_{l-n})$$

- Why?

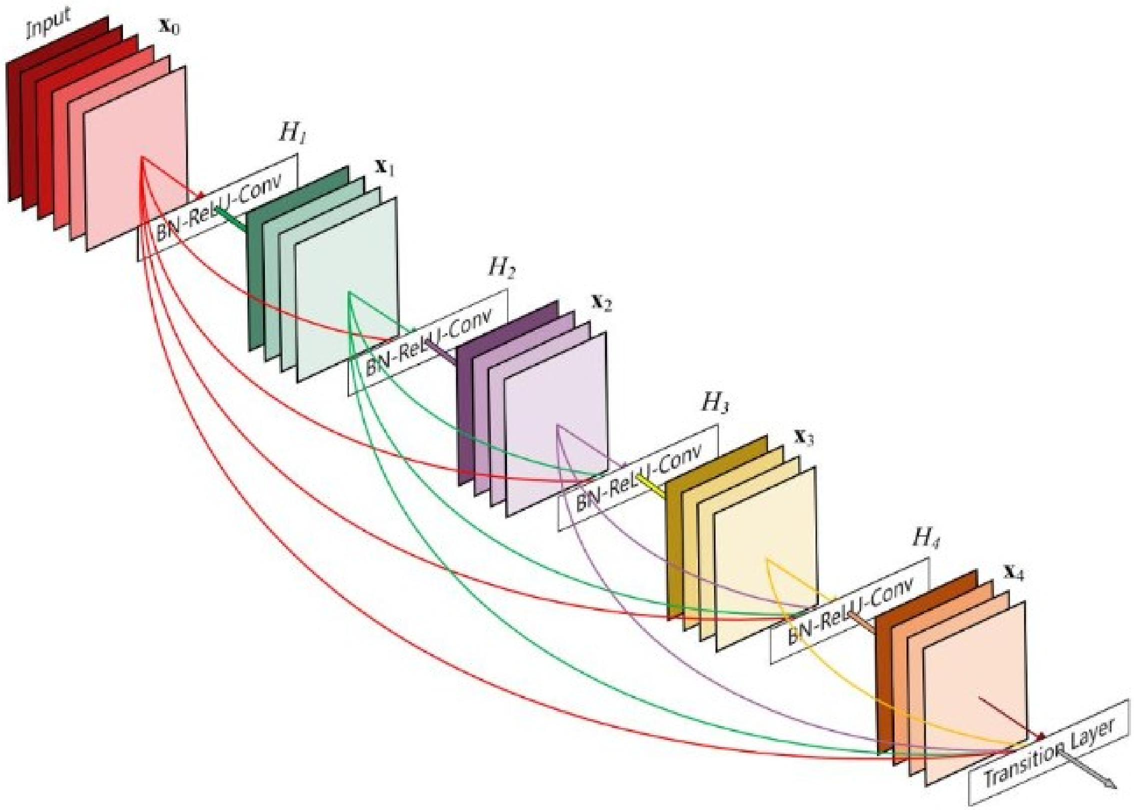


DenseNet

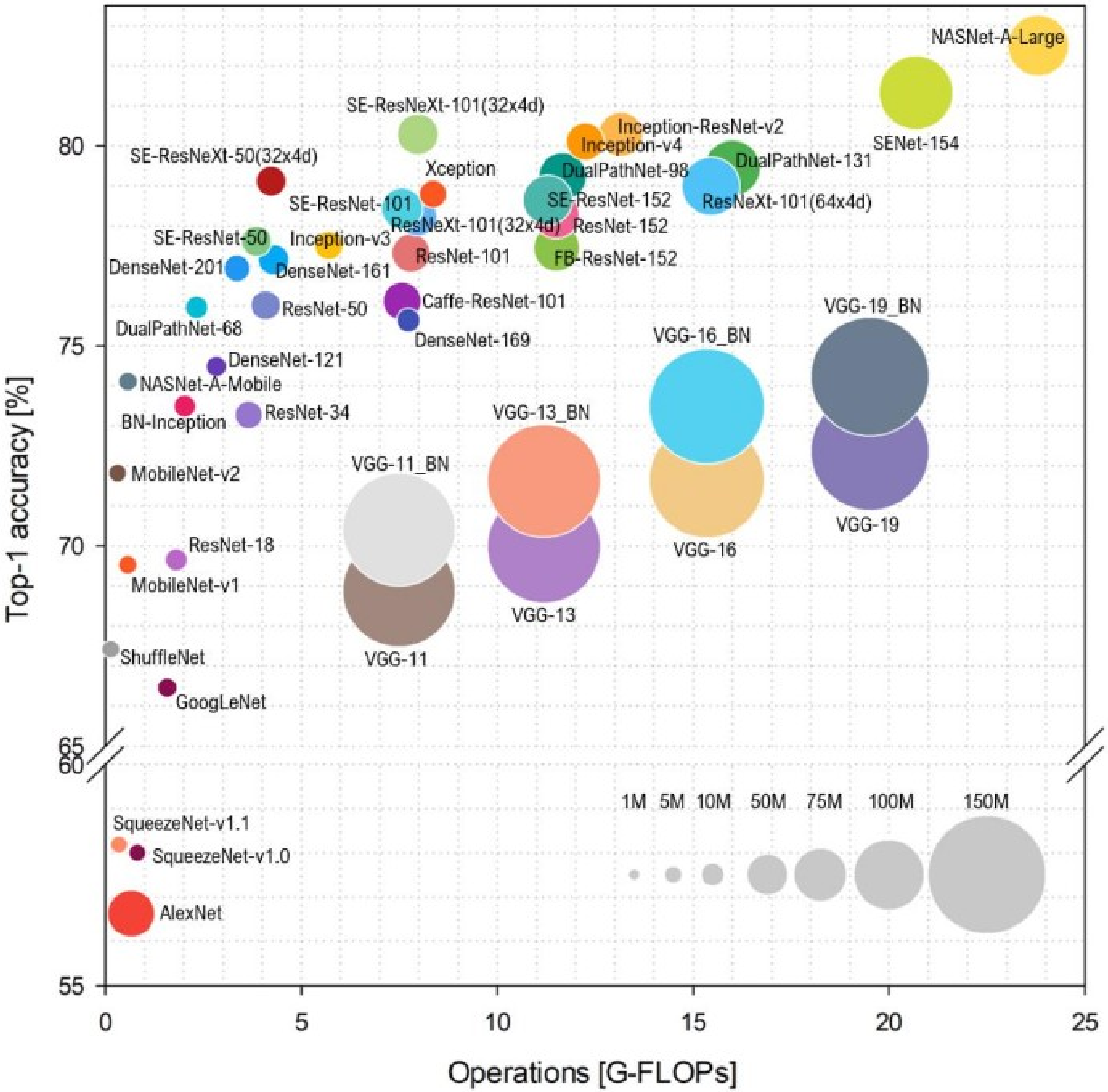
- Add skip connections to multiple forward layers

$$y = h(x_l, x_{l-1}, \ldots, x_{l-n})$$

- Assume layer 1 captures edges, while layer 5 captures faces (and other stuff)
- Why not have a layer that combines both faces and edges (e.g. to model a scarred face)
- Standard ConvNets do not allow for this
  - Layer 6 combines only layer 5 patterns, not lower

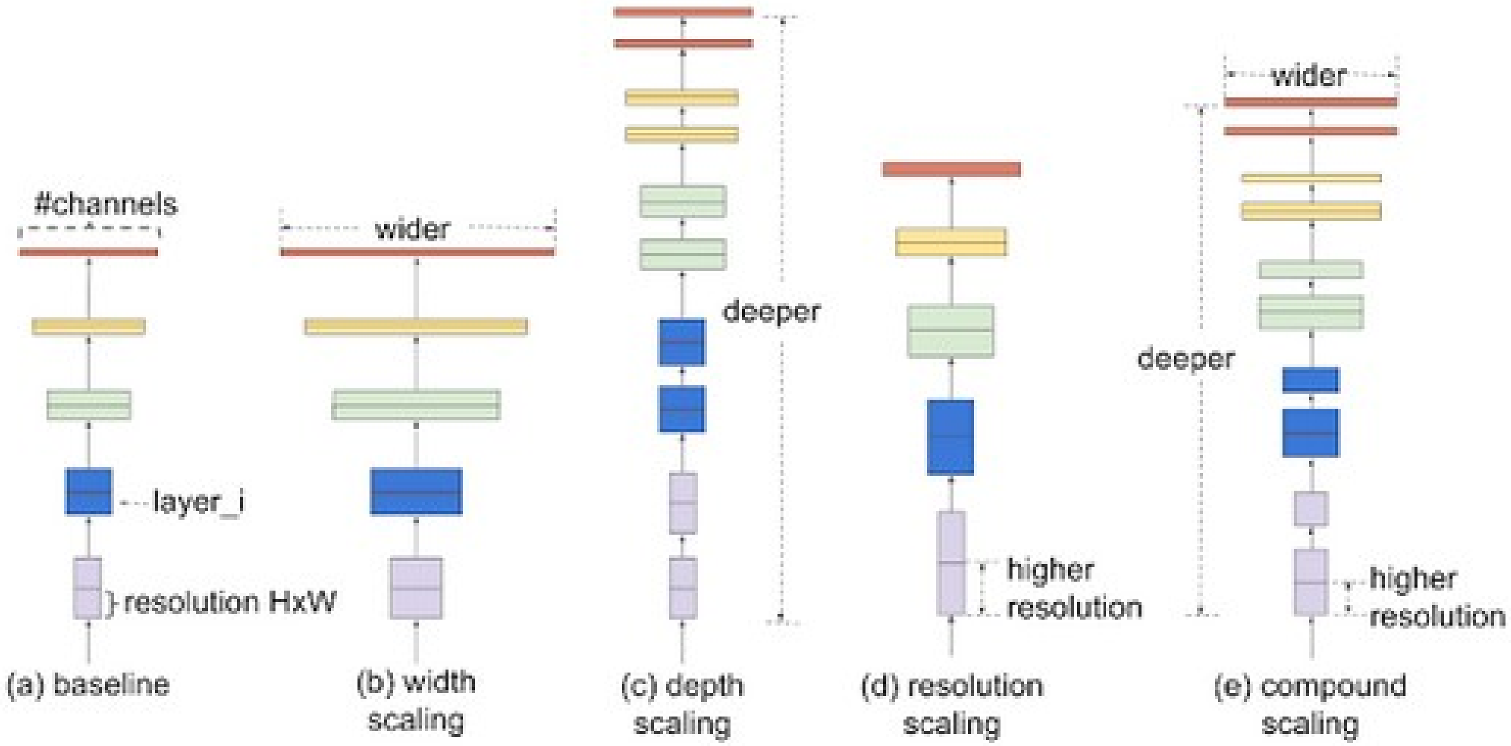


State of the art



EfficientNet

- Automatic hyperparameter definition (AutoML)
- Scaling up CNNs. however many many ways to do it



EfficientNet

- Automatic hyperparameter definition (AutoML)
- Scaling up CNNs. however many many ways to do it
- Balance dimensions of width/depth/resolution by scaling with a constant ratio
  - a,b, γ are constant coefficients determined by a small grid search on the original small model
  - Fix φ=1 and do a small grid search of the re
  - EfficientNet-B0 is with φ=1
  - Fixing afterward the rest we obtain
    - EfficientNet-B1 to B7

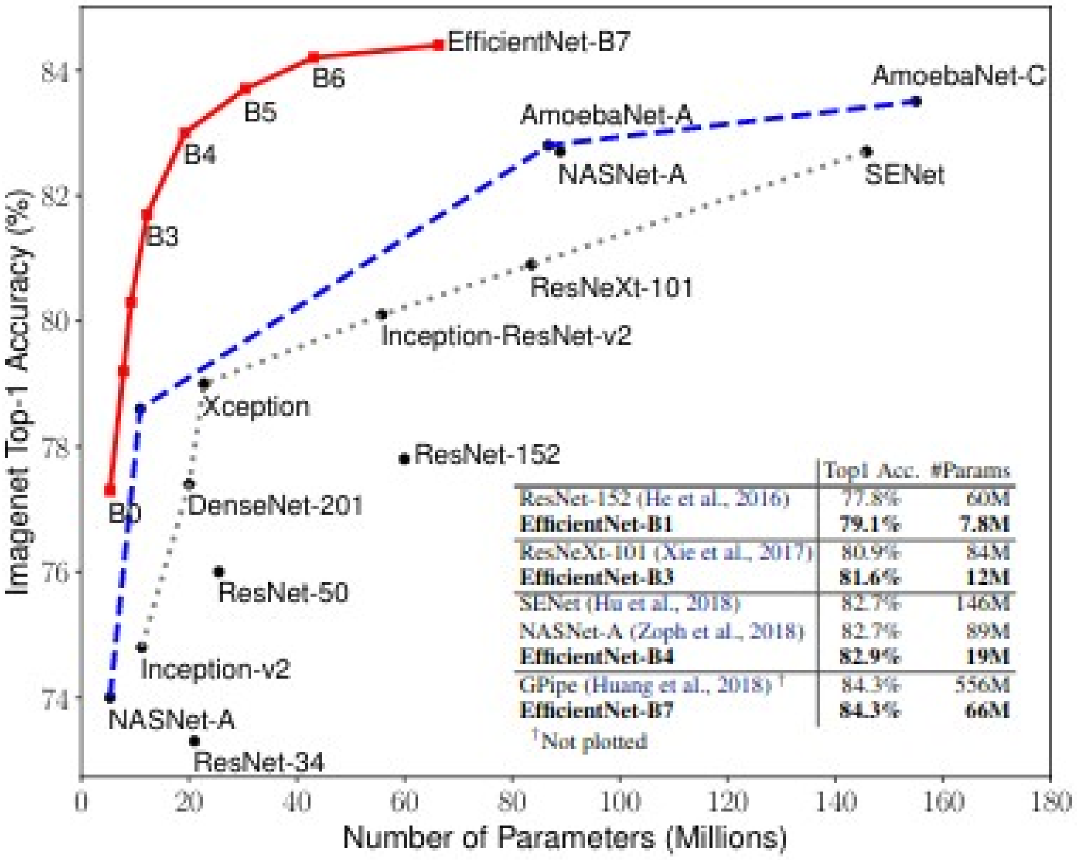
$$\begin{aligned}
\text{depth: } & d = \alpha^{\phi} \\
\text{width: } & w = \beta^{\phi} \\
\text{resolution: } & r = \gamma^{\phi} \\
\text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\
& \alpha \geq 1, \beta \geq 1, \gamma \geq 1
\end{aligned}$$

EfficientNet

- m denotes accuracy; T denotes target FLOPs
- Search space from NAS

$$ACC(m) \times [FLOPS(m)/T]^w$$

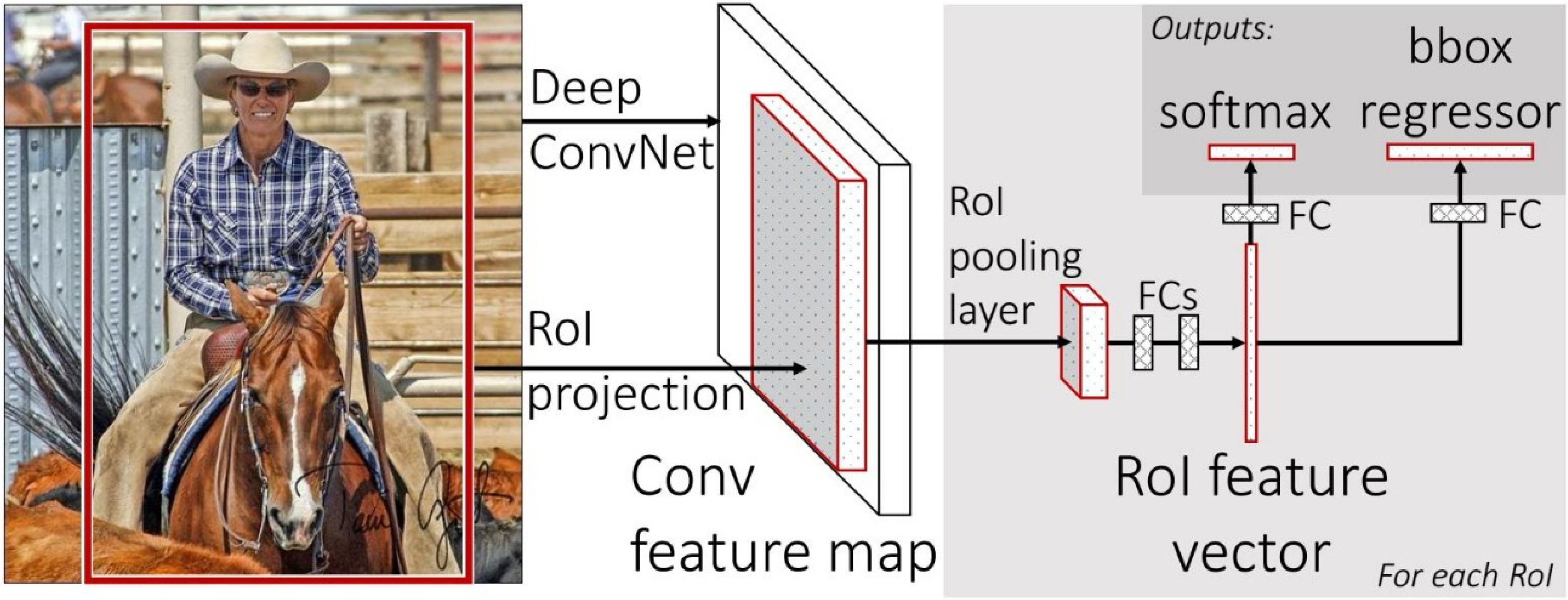
Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1



R-CNNs, Fully Convolutional Nets

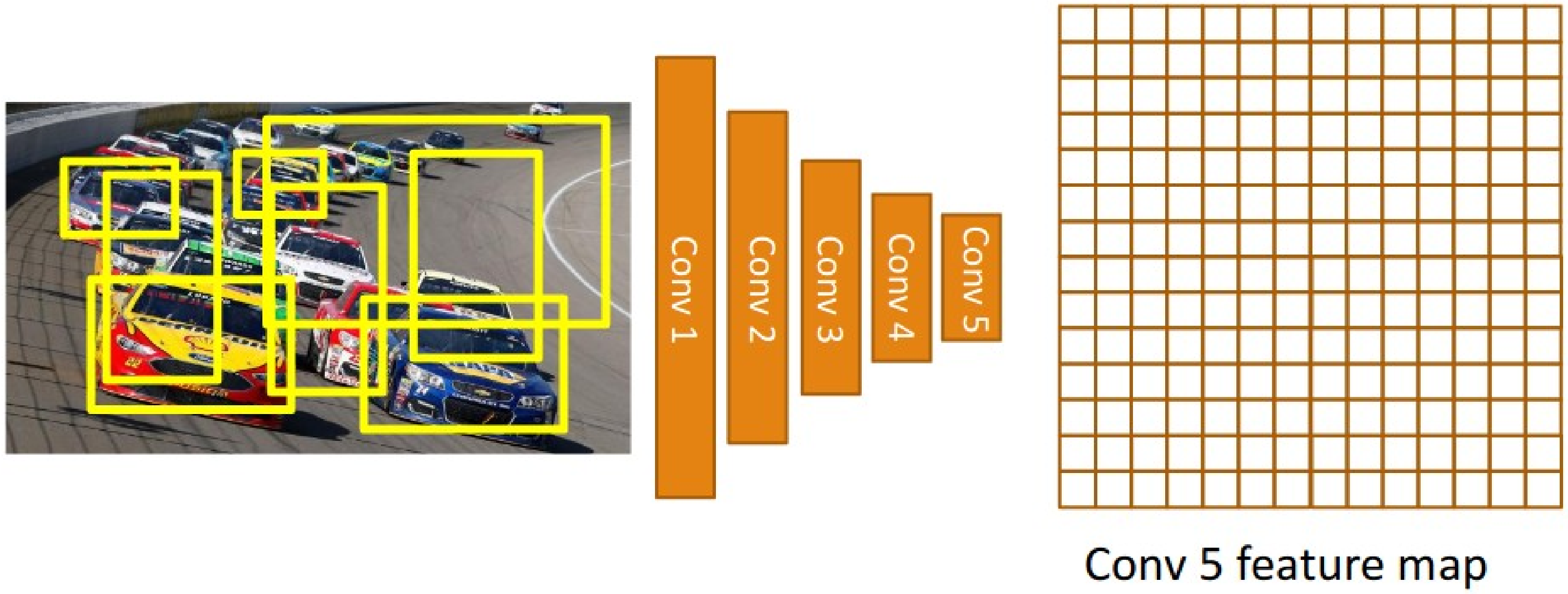
Sliding window on feature maps

- SPPnet [He2014]
- Fast R-CNN [Girshick2015]



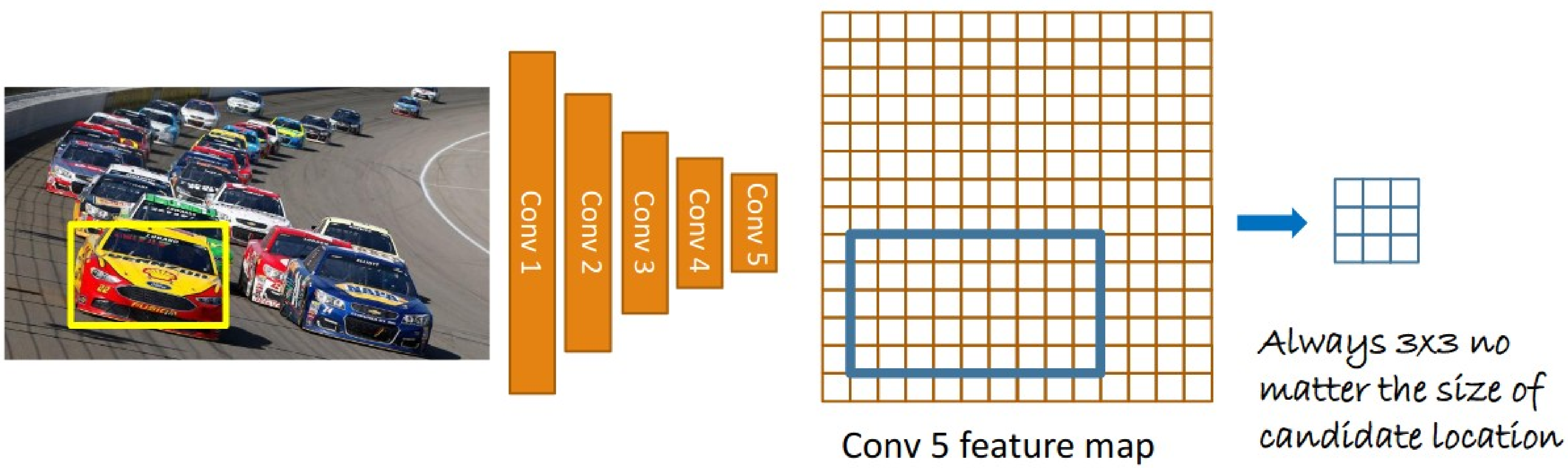
Fast R-CNN: Steps

- Process the whole image up to conv5
- Compute possible locations for objects (some correct, most wrong)



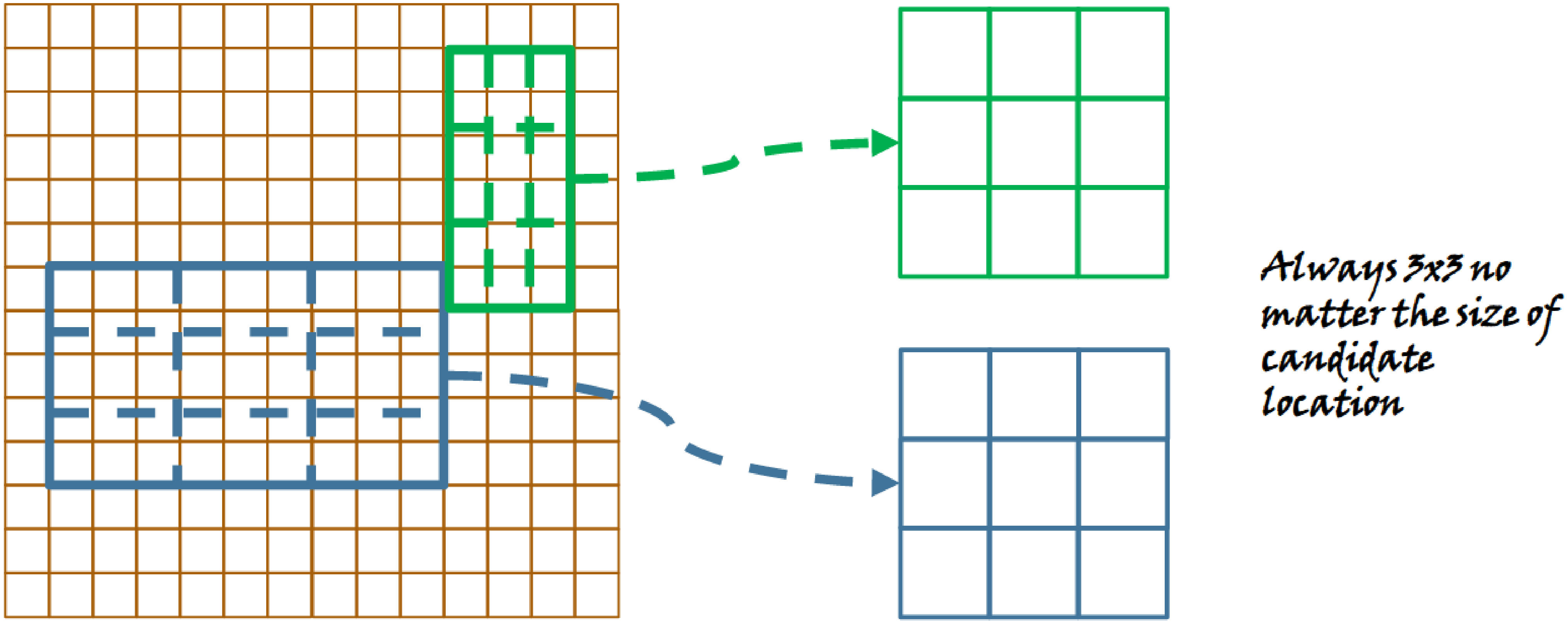
Fast R-CNN: Steps

- Process the whole image up to conv5
- Compute possible locations for objects (some correct, most wrong)
- Given single location → ROI pooling module extracts fixed length feature



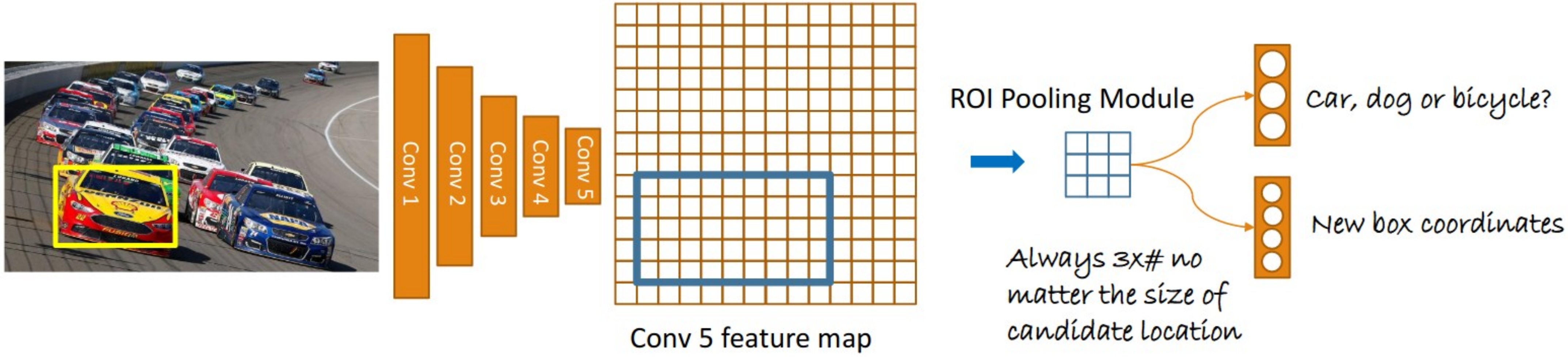
Region-of-Interest (ROI) Pooling Module

- Divide feature map in TxT cells
  - The cell size will change depending on the size of the candidate location

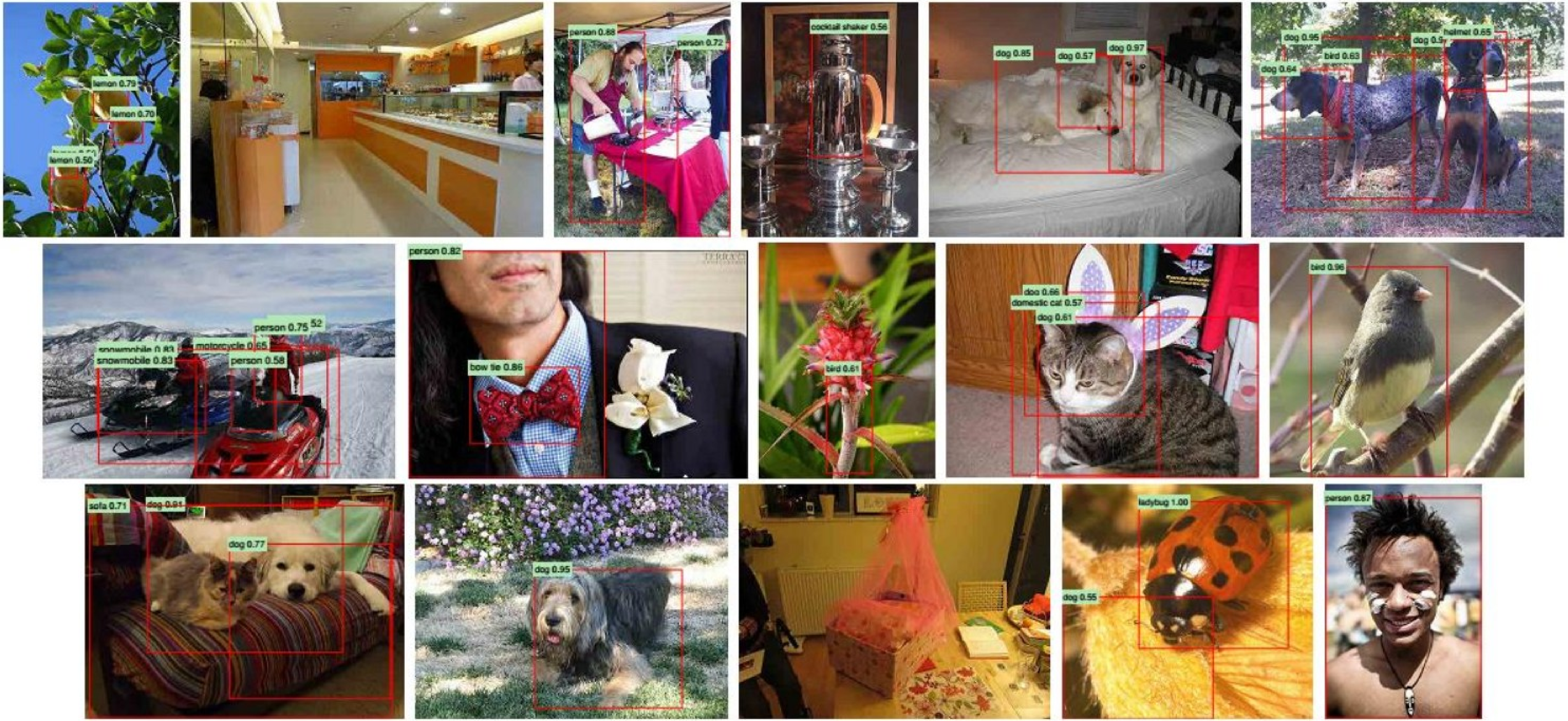


Fast R-CNN: Steps

- Process the whole image up to conv5
- Compute possible locations for objects (some correct, most wrong)
- Given single location → ROI pooling module extracts fixed length feature
- Connect to two final layers, 1 for classification, 1 for box refinement



Some Results

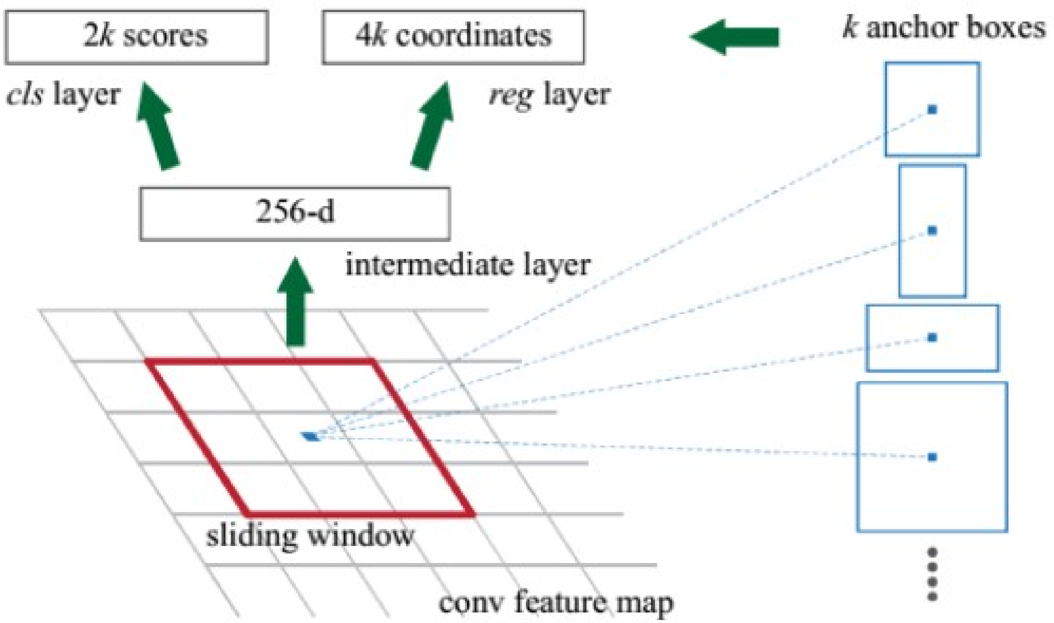


Fast-RCNN

- Reuse convolutions for different candidate boxes
  - Compute feature maps only once
- Region-of-Interest pooling
  - Define stride relatively → box width divided by predefined number of “poolings” T
  - Fixed length vector
- End-to-end training
- (Very) accurate object detection
- (Very) Faster
  - Less than a second per image
- But: External box proposals needed

Faster R-CNN [Girshick2016]

- Fast R-CNN → External candidate locations
- Faster R-CNN → deep network proposes candidate locations
- Slide the feature map → k anchor boxes per



Region Proposal Network

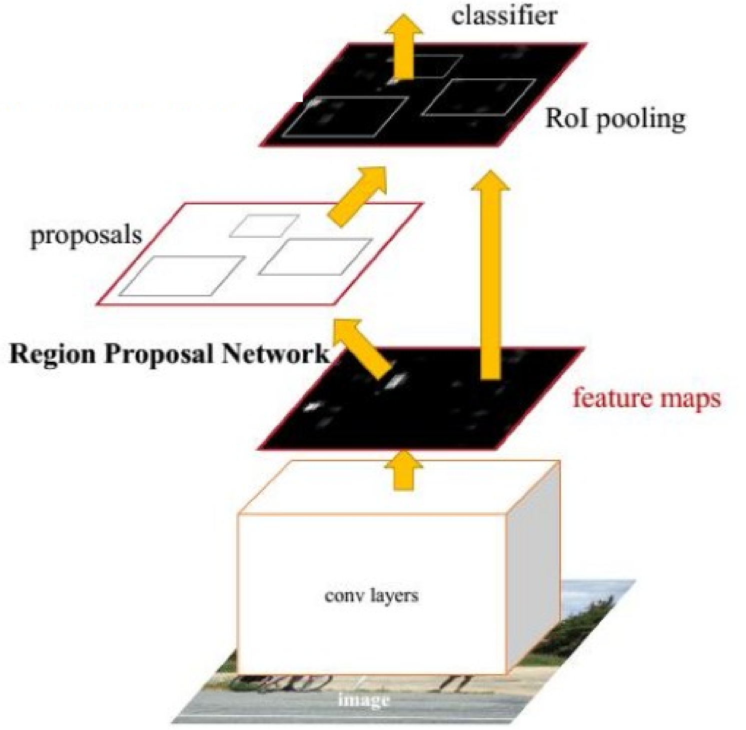
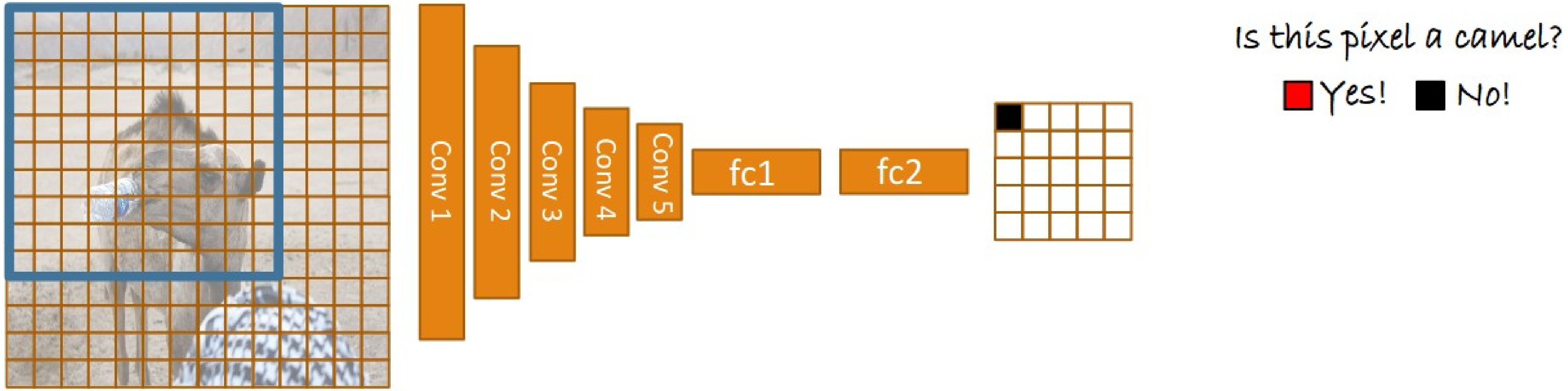


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

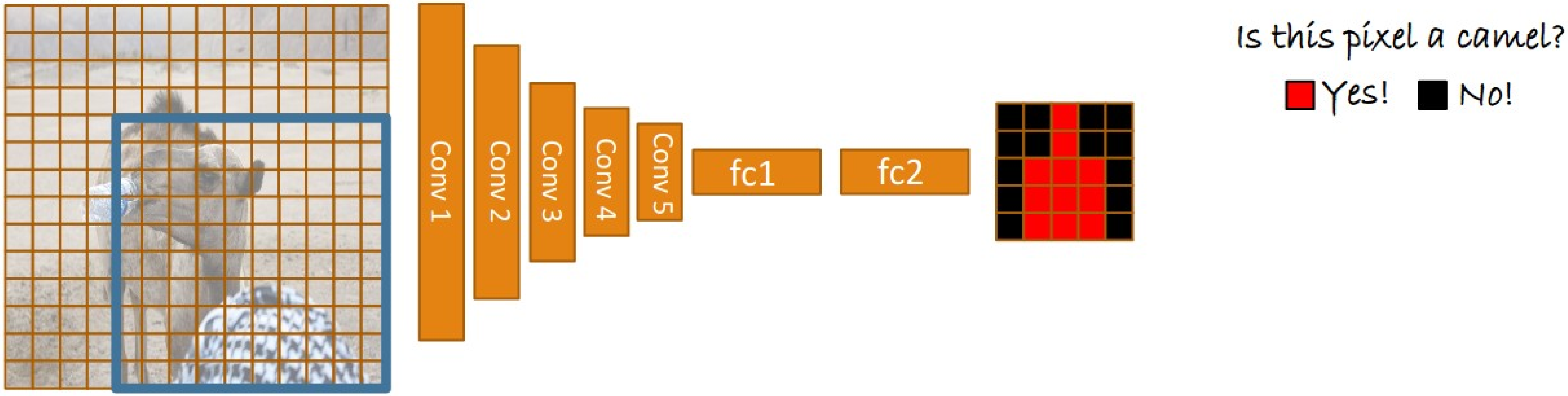
Going Fully Convolutional [LongCVPR2014]

- Image larger than network input → slide the network

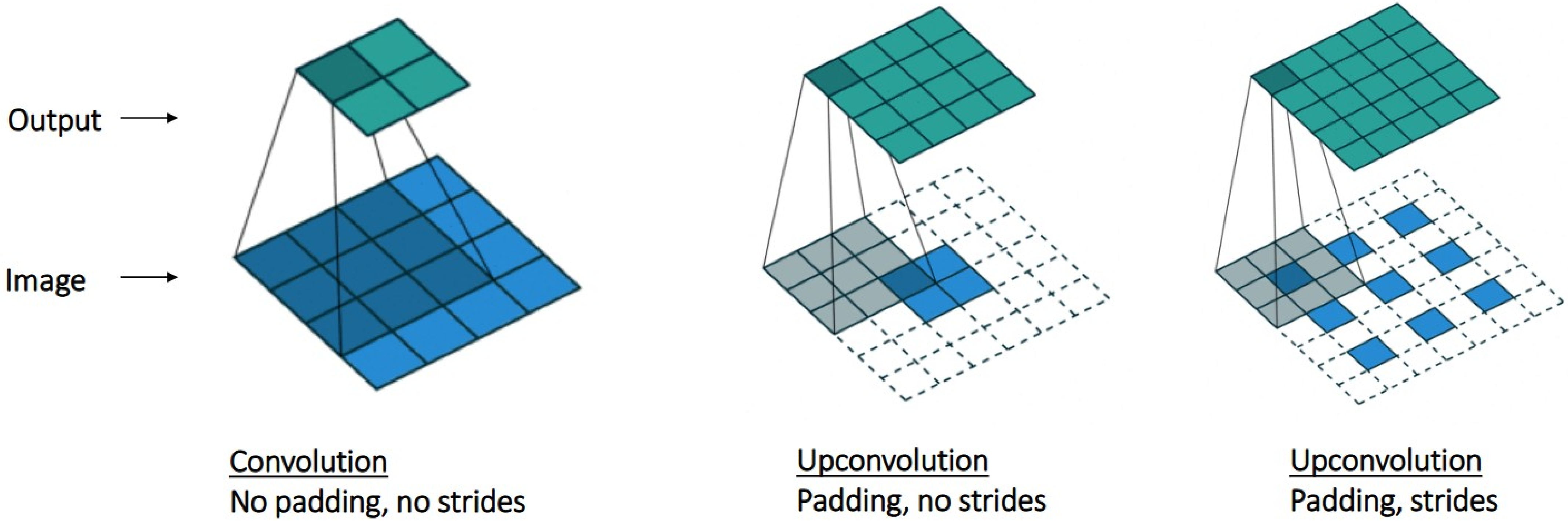


Going Fully Convolutional [LongCVPR2014]

- Image larger than network input → slide the network

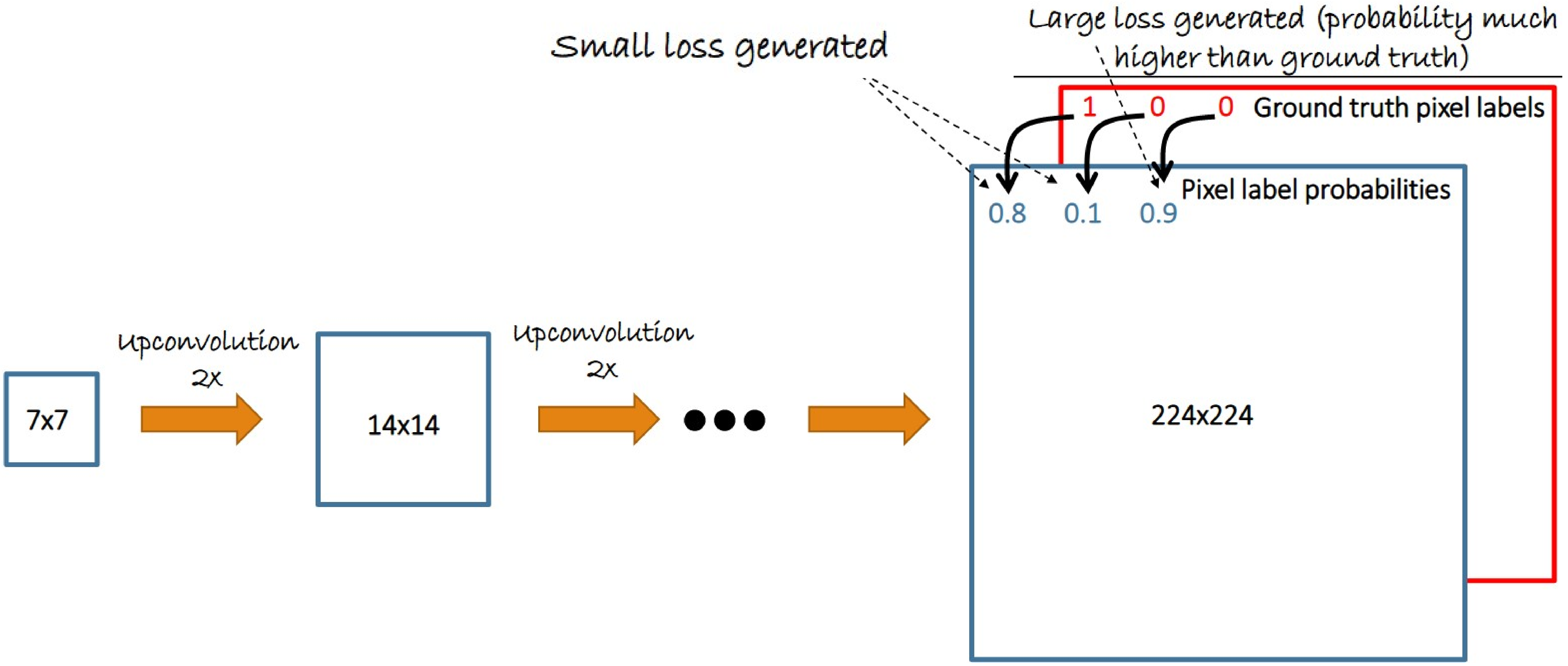


Deconvolutional modules

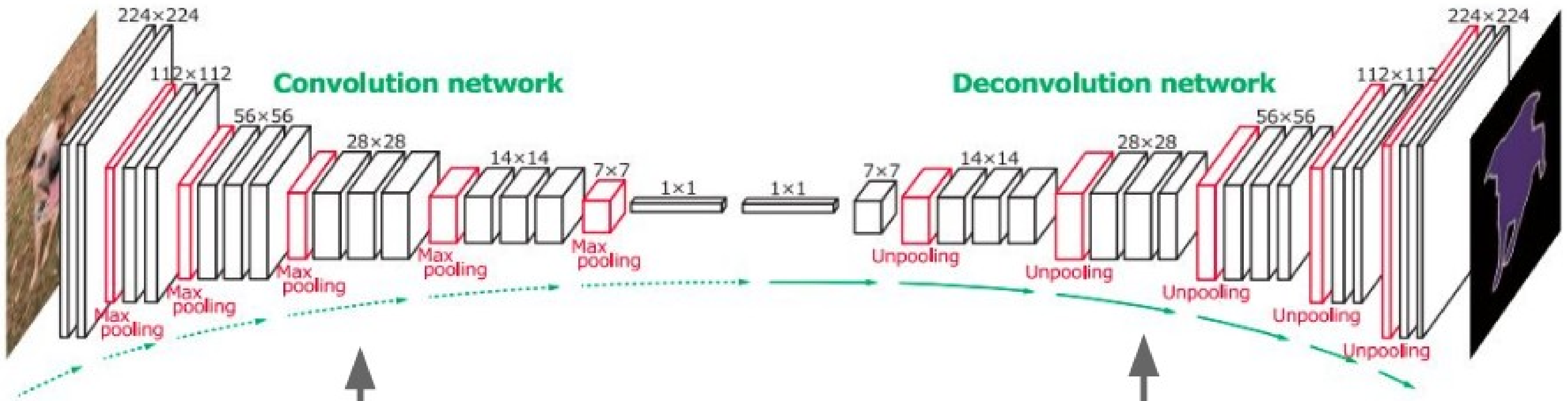


More visualizations: https://github.com/vdumoulin/conv_arithmetic

Coarse → Fine Output



Deconvolutional modules



Normal VGG

"Upside down" VGG

Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Agony of Choice

- Architecture: depth, width, scales, residuals, ….
- Loss function: cross entropy, focal loss, MSE, …
- Optimization: optimizer, learning rate, momentum, …
- Data normalization
- Modelization of the problem

Kaggle Competition is out!!

https://www.kaggle.com/c/fdl21-fdl-dsba

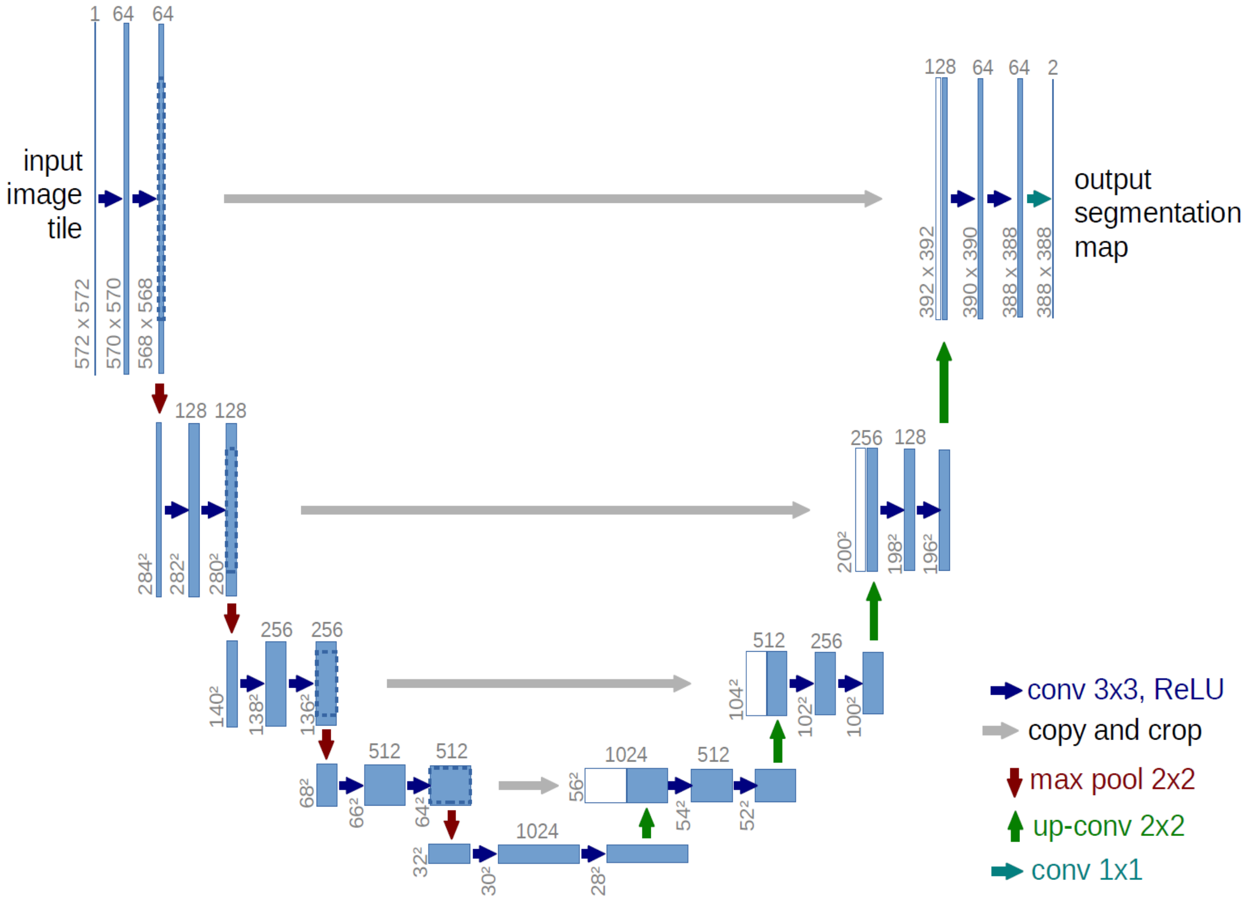
- Segmentation of remote sensing data
- Images acquired from UAVs
- 25 different land cover classes

Kaggle Competition is out!! (50% grade)

- https://www.kaggle.com/c/fdl21-fdl-dsba
- Evaluation Dice Score

$$Dice = \frac{2 \times TP}{(TP + FP) + (TP + FN)}$$

- Baseline U-Net
  - Resize to 1024x1024
  - Crop 512x512 patches
  - Batch size:8, epochs:100, cross entropy, SDG, lr=0.1
- Score 0.62310



Kaggle Competition is out!! (50% grade)

- https://www.kaggle.com/c/fdl21-fdl-dsba

- Teams of 2-3 people
- You will have to submit to
  fdl.dsba@gmail.com
  - Your results on the leaderboard
  - A report with your method
  - The code to reproduce your results

  - **Deadline 20/12**