Key Ideas and Architectures in Deep Learning

Applications that (probably) use DL

Autonomous Driving

Scene understanding

/Segmentation

Applications that (probably) use DL

WordLens

Prisma

Outline of today's talk

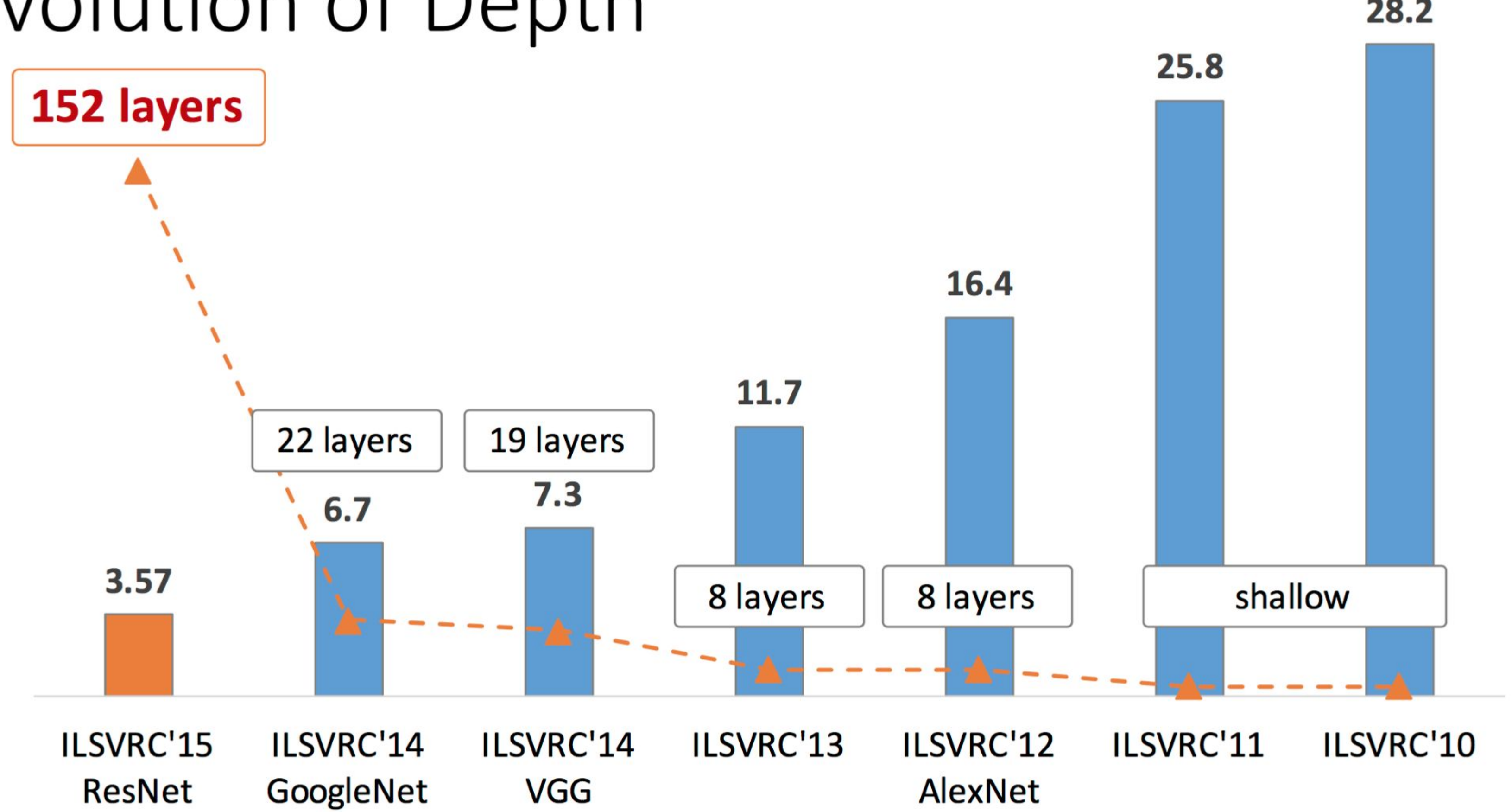
Image Recognition

- LeNet - 1998
- AlexNet - 2012
- VGGNet - 2014
- GoogLeNet - 2014
- ResNet - 2015

Fun application using CNNs

- Image Style Transfer

Revolution of Depth



ImageNet Classification top-5 error (%)

Questions to ask about each architecture/ paper

Special Layers

Loss function

Train faster?

Reduce Overfitting

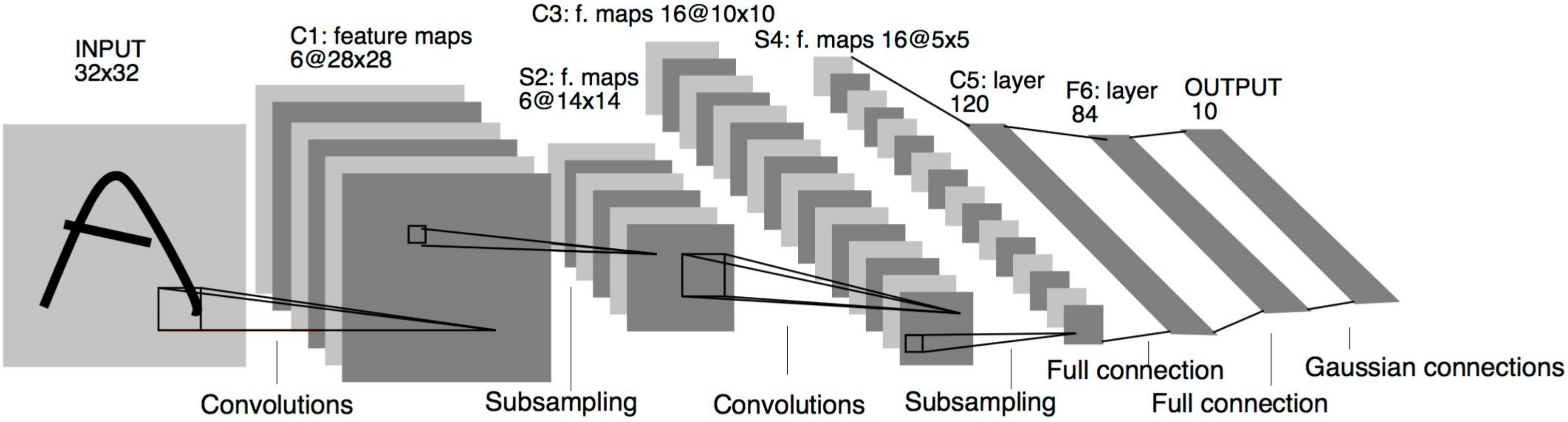
Non-Linearity

Weight-update rule

Reduce parameters

Help you visualize?

LeNet5 - 1998



INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

LeNet5 - Specs

MNIST - 60,000 training, 10,000 testing

Input is 32x32 image

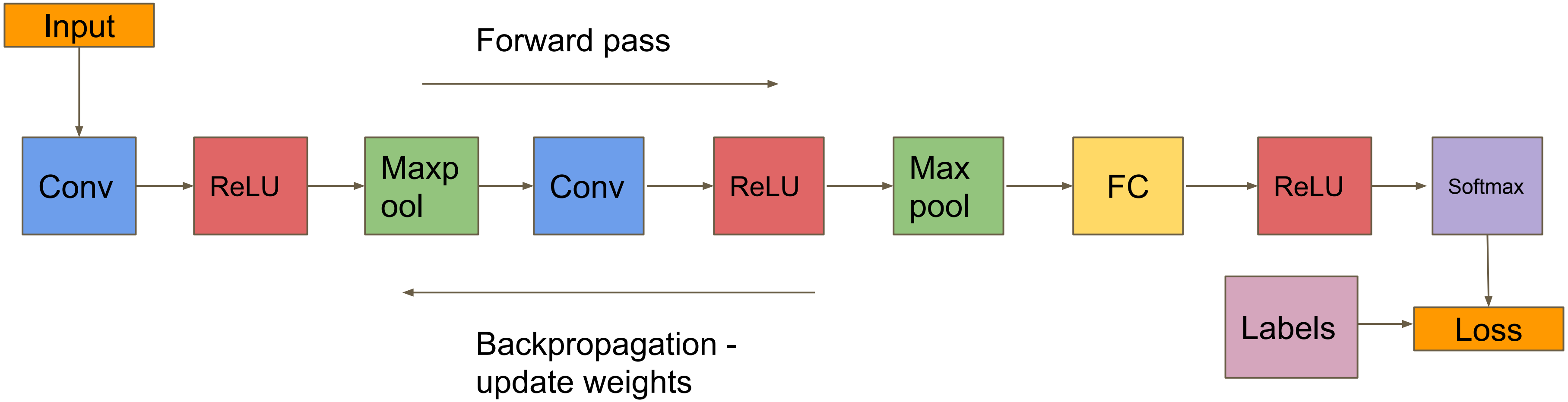
8 layers

60,000 parameters

Few hours to train on a laptop

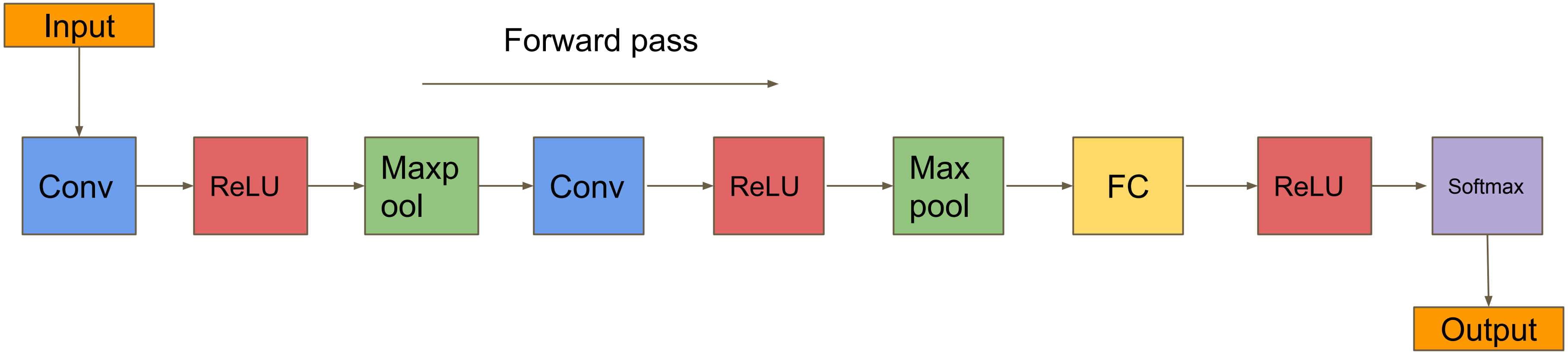
Modified LeNet Architecture - Assignment 3

Training



Modified LeNet Architecture - Assignment 3

Testing



Compare output
with labels

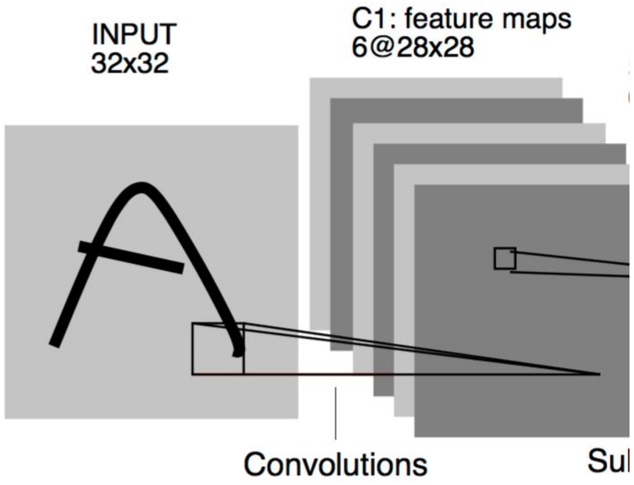
Modified LeNet - CONV Layer 1

Input - 28 x 28

Output - 6 feature maps - each 24 x 24

Convolution filter - 5 x 5 x 1 (convolution) + 1 (bias)

How many parameters in this layer?



Modified LeNet - CONV Layer 1

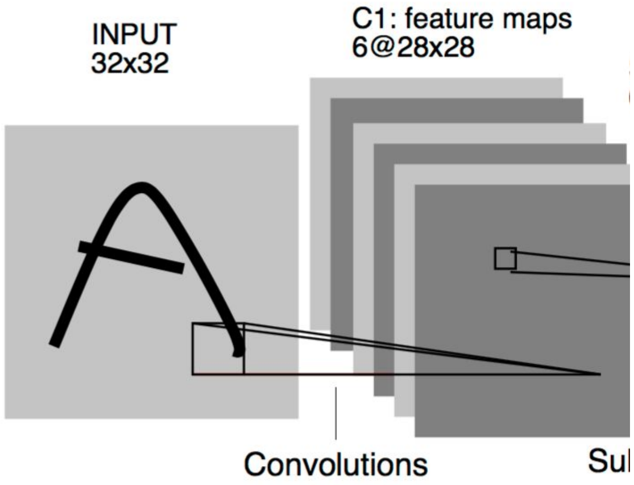
Input - 32 x 32

Output - 6 feature maps - each 28 x 28

Convolution filter - 5 x 5 x 1 (convolution) + 1 (bias)

How many parameters in this layer?

(5x5x1+1)*6 = 156



Modified LeNet - Max-pooling layer

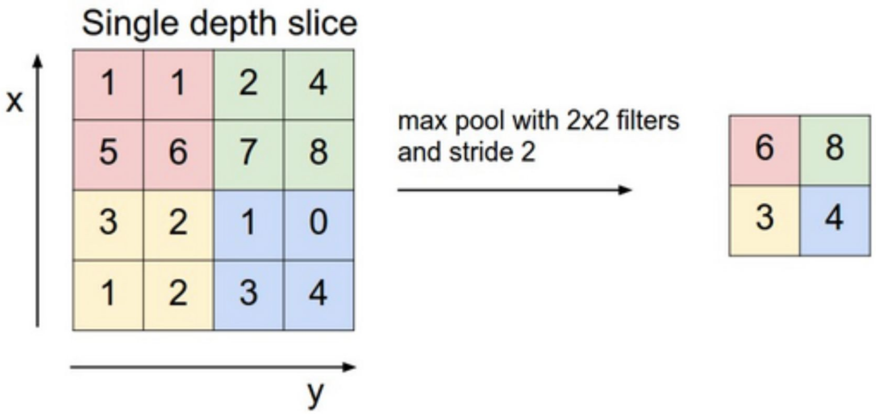
Decreases the spatial extent of the feature maps, makes it translation-invariant

Input - 28 x 28 x 6 volume

Maxpooling with filter size 2 x2

And stride 2

Output - ?



Modified LeNet - Max-pooling layer

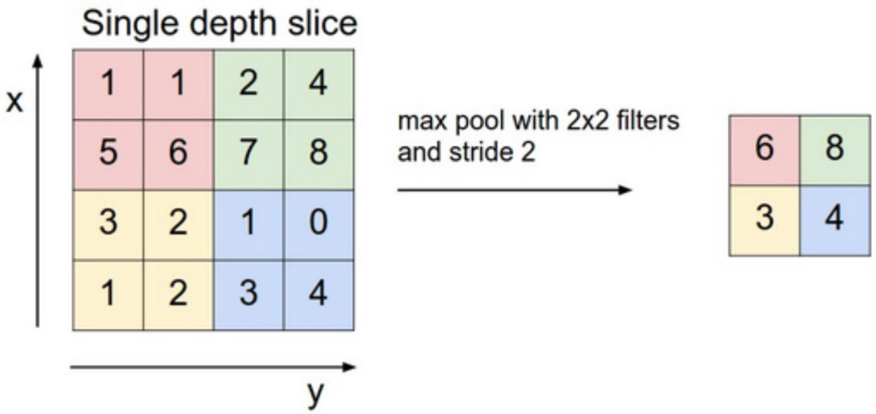
Decreases the spatial extent of the feature maps

Input - 28 x 28 x 6 volume

Maxpooling with filter size 2 x2

And stride 2

Output - 14 x 14 x 6 volume



LeNet5 - Key Ideas

Convolution - extract same features at different spatial locations with few parameters

Spatial averaging - sub-sampling to reduce parameters (we use max-pooling)

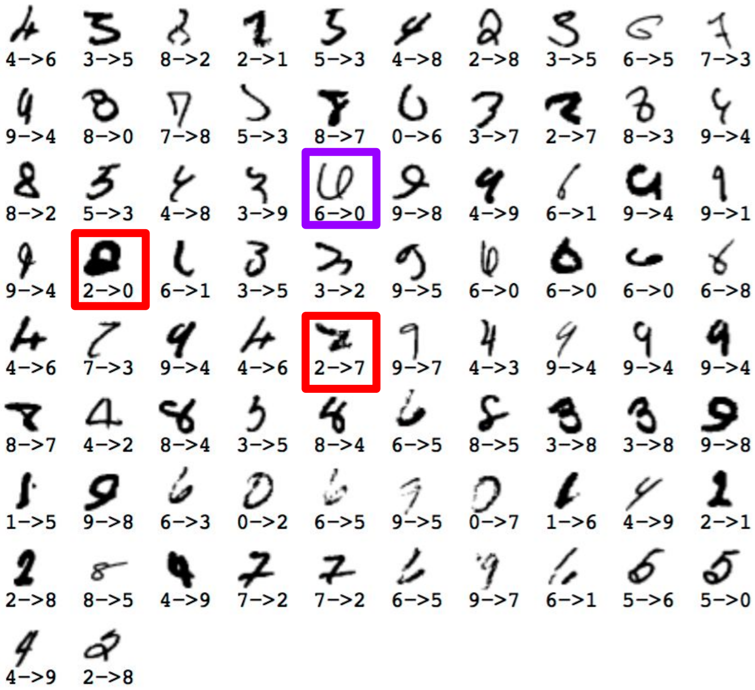
Non-linearity - Sigmoid (but we'll use ReLU)

Multi-layer perceptron in the final layers

Introduced the **Conv -> Non-linearity -> Pooling** unit

LeNet5 Evaluation

Misclassifications



Accuracy

>97%

What happened from 1998-2012?

Neural nets were in incubation

More and more data was available - cheaper digital cameras

And computing power became better - CPUs were becoming faster

GPUs became a general-purpose computing tool (2005-6)

Creation of structured datasets - ImageNet (ILSVRC) 2010 **(super important!)**

A word about datasets - Network inputs

ImageNet (We'll talk about object classification)

CIFAR - Object Classification

Caltech - Pedestrian detection benchmark

KITTI - SLAM, Tracking etc.

Remember : Your algo is only as good as your data!

How are networks evaluated? - Network outputs

Top-5 error

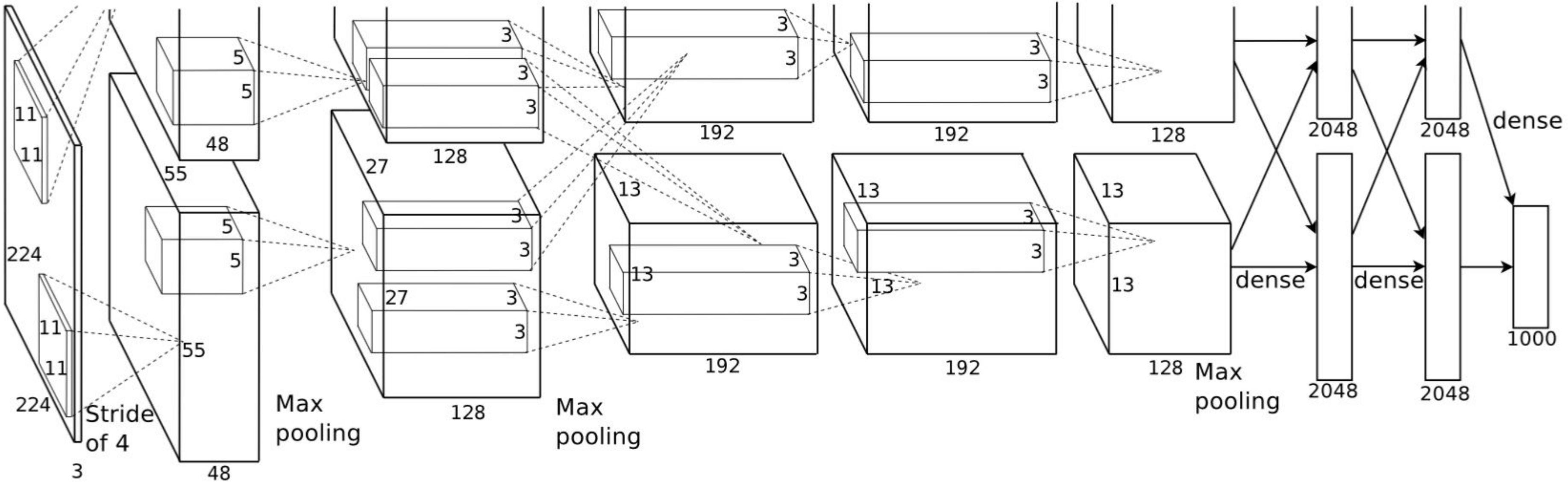
Top-1 error

Accuracy

AlexNet - 2012

Won the 2012 ILSVRC (ImageNet Large-Scale Visual Recognition Challenge)

Achieved a top-5 error rate of 15.4%, next best was 26.2%



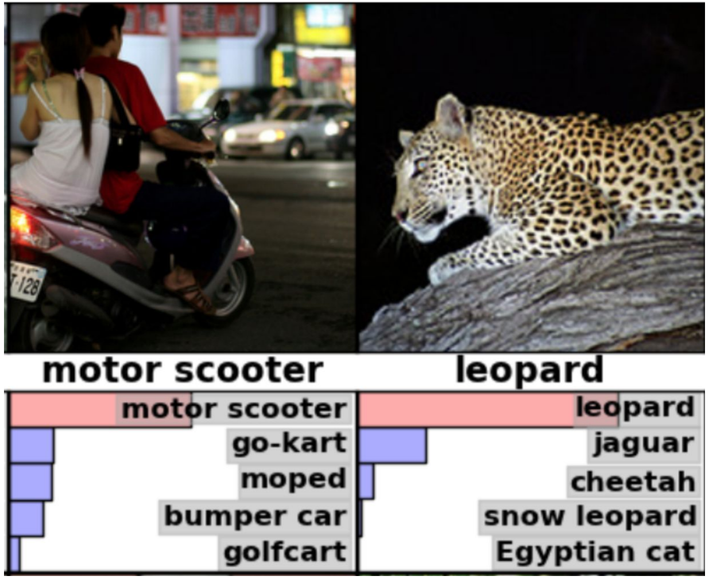
AlexNet - Specs

ImageNet 1000 categories

1.2 million training images

50,000 validation images

150,000 testing images.



60M Parameters

Trained on two GTX 580 GPUs for five to six days.

AlexNet - Key Ideas

Used **ReLU** for the nonlinearity functions - f(x) = max(0,x) - made convergence faster

Used **data augmentation** techniques

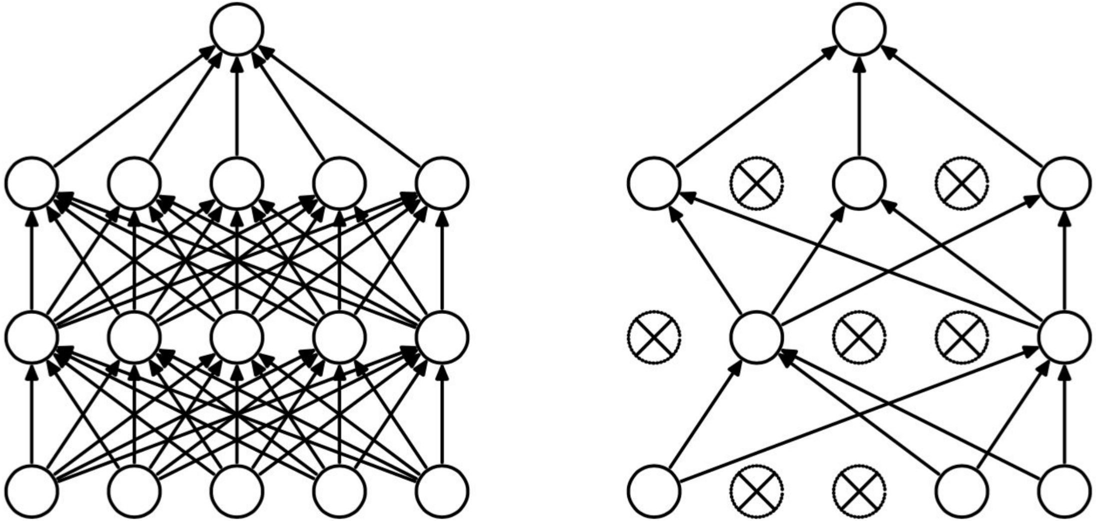
Implemented **dropout** to combat overfitting to the training data.

Trained the model using **batch stochastic gradient descent**

Used **momentum** and **weight decay**

Dropout

Dropout in Neural Networks



(a) Standard Neural Net (b) After applying dropout.

VGG Net - 2014

“Simple and deep”

Top-5 error rate of 7.3% on ImageNet

16 layer CNN - Best result - Conf. D

138 M parameters

Trained on 4 Nvidia Titan Black GPUs

for two to three weeks.

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 Net - Key Ideas

The use of only 3x3 sized filters. Used multiple times = greater receptive fields.

Decrease in spatial dimensions and increase in depth deeper into the network

Used scale jittering as one data augmentation technique during training

Used ReLU layers after each conv layer and trained with batch gradient descent

Reduced number of parameters - $3*(3^2)$ compared to 7^2

Conclusion - Small RFs, deep networks are good. :-)

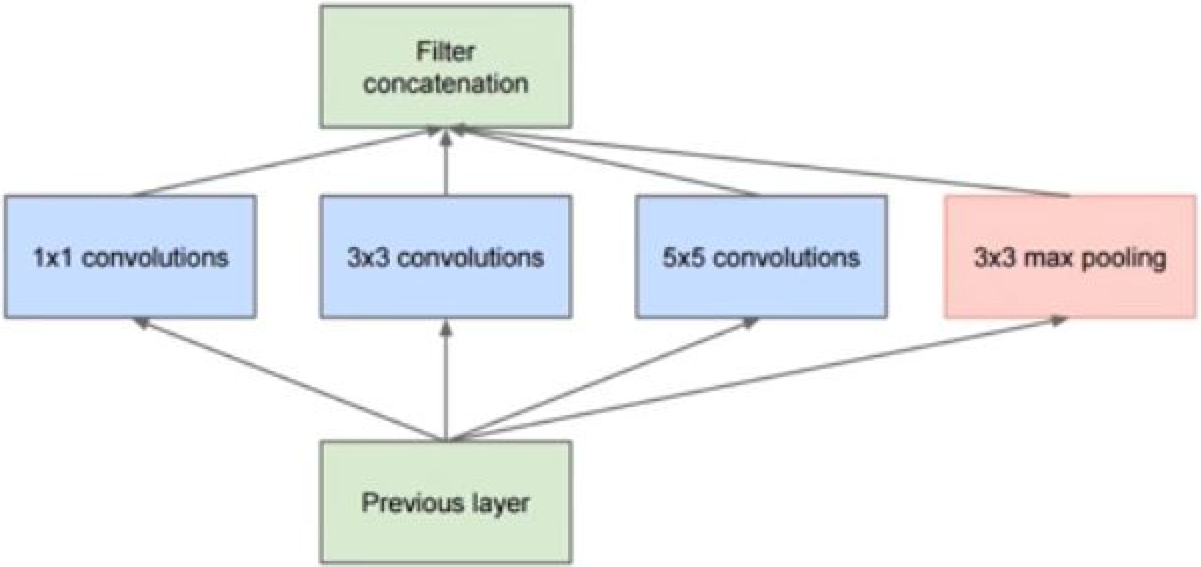
GoogLeNet / Inception - 2014

Winner of ILSVRC 2014 with a top 5 error rate of 6.7% (4M parameters compared to AlexNet’s 60M)

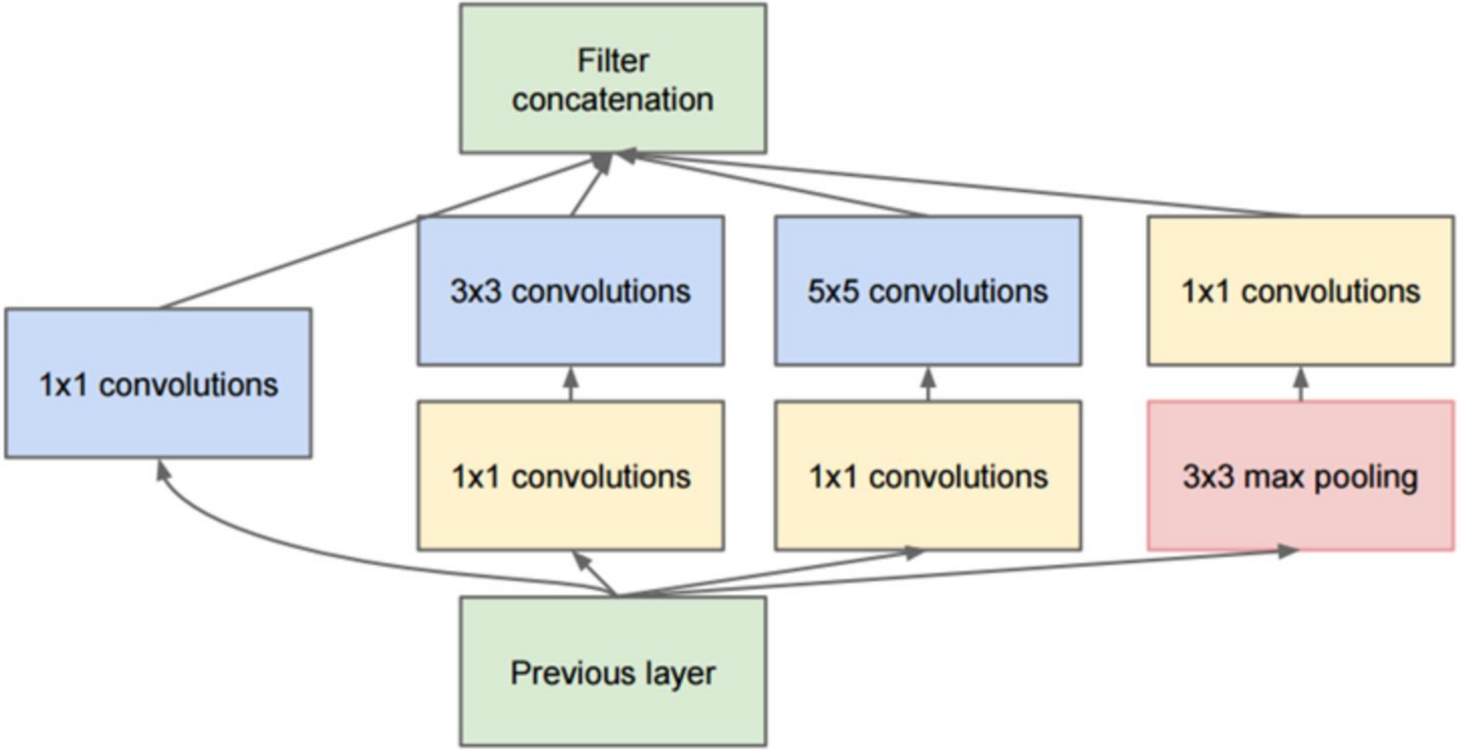
Trained on “a few high-end GPUs within a week”.

The Inception module

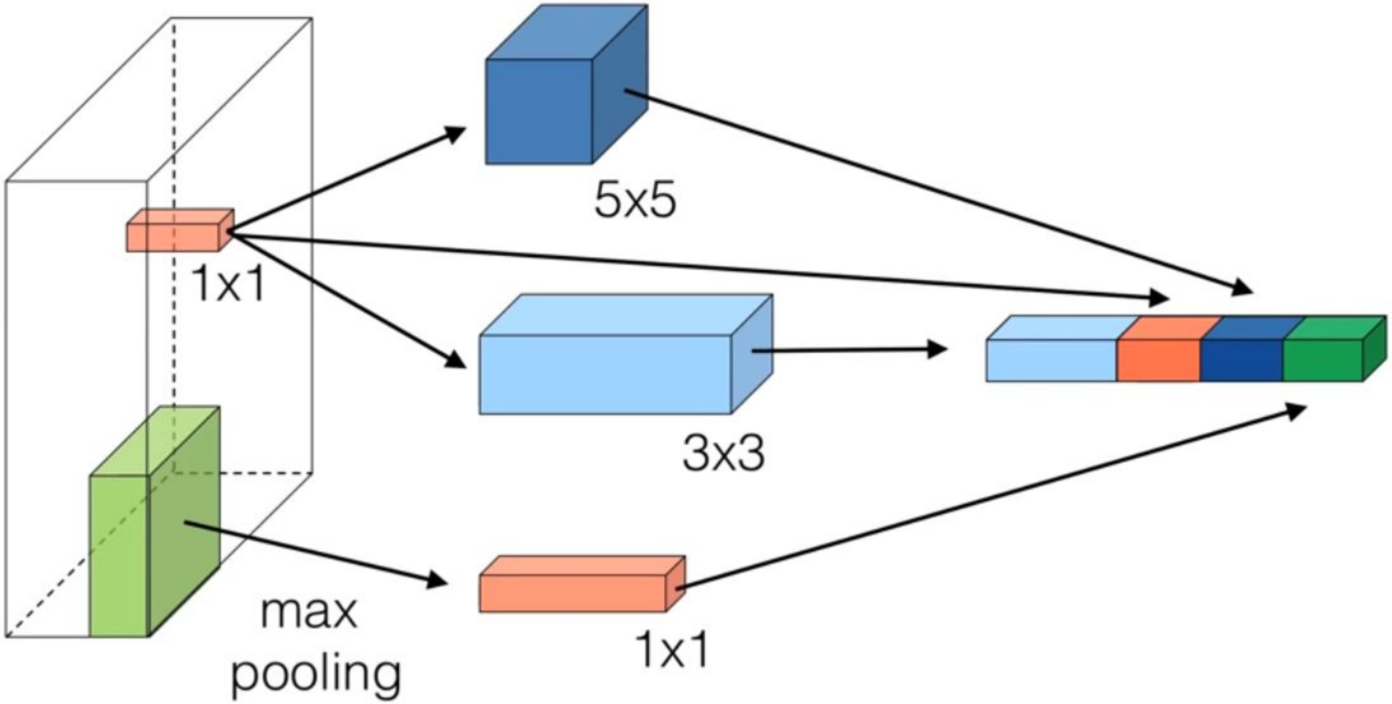
The Inception Module - A closer look



The Inception Module - A closer look

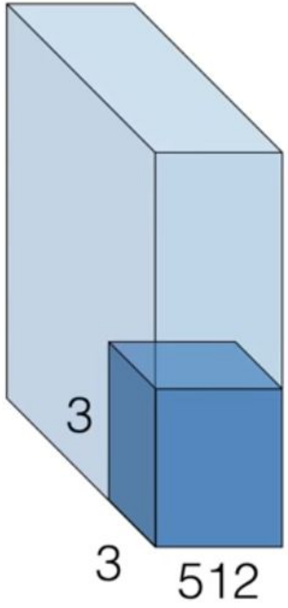


Inception module - Feature Map Concatenation



Inception Parameter count

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M



If we used (3 x 3, 512) convolution:

(3 x 3 x 512 x 512) parameters = 2.359 million parameters

Inception module: 437K parameters

Inception - Key Ideas

Used 9 Inception modules in the whole architecture

No use of fully connected layers! They use an average pool instead, to go from a 7x7x1024 volume to a 1x1x1024 volume - Saves a huge number of parameters.

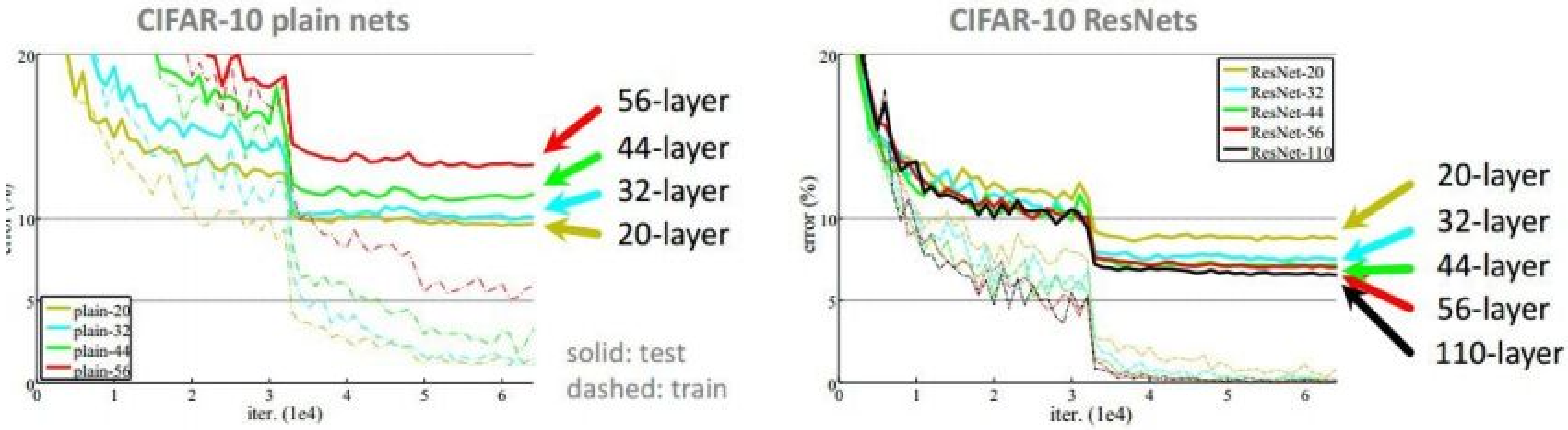
Uses 12x fewer parameters than AlexNet.

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.

Improved performance and efficiency through creatively stacking layers

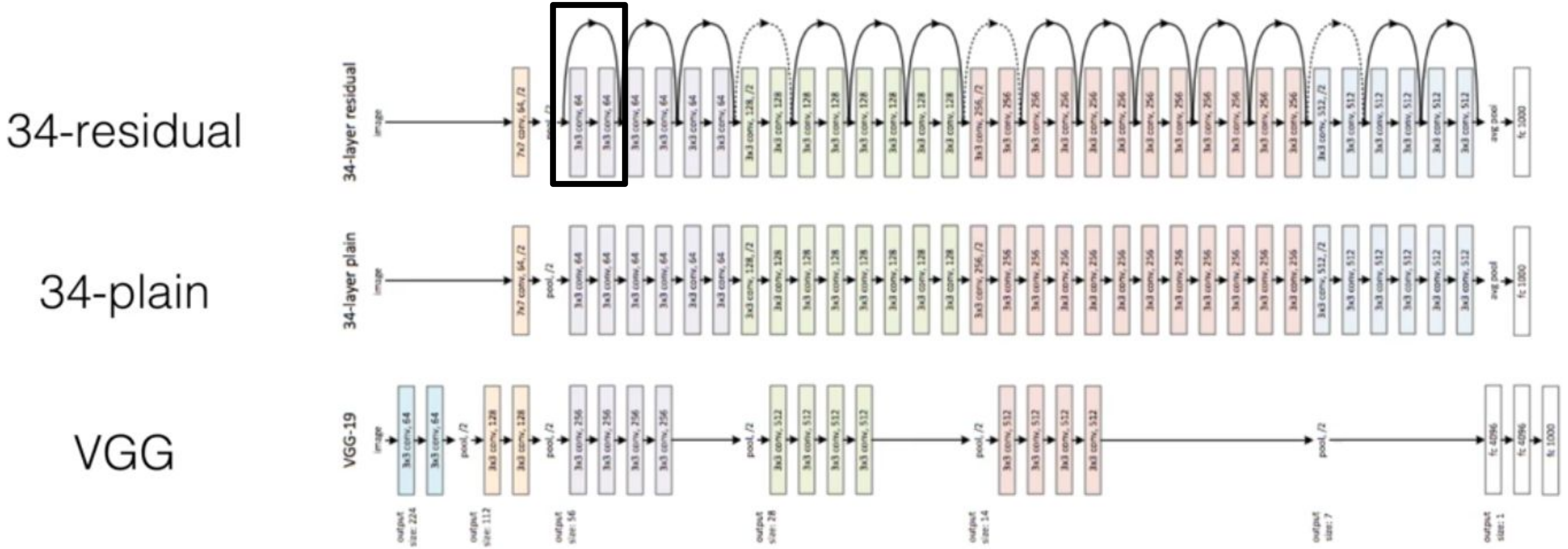
Going deeper

Performance of ResNets versus plain-nets as depth is increased

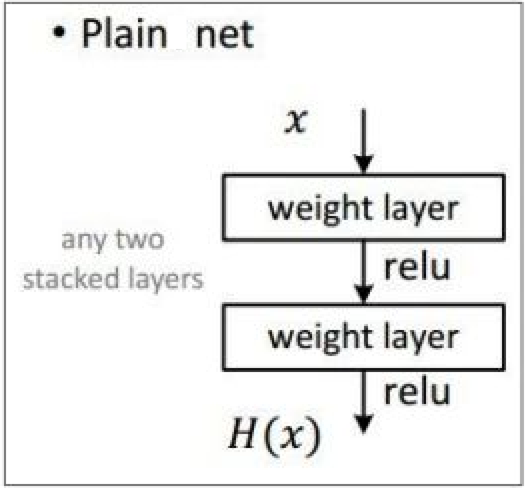


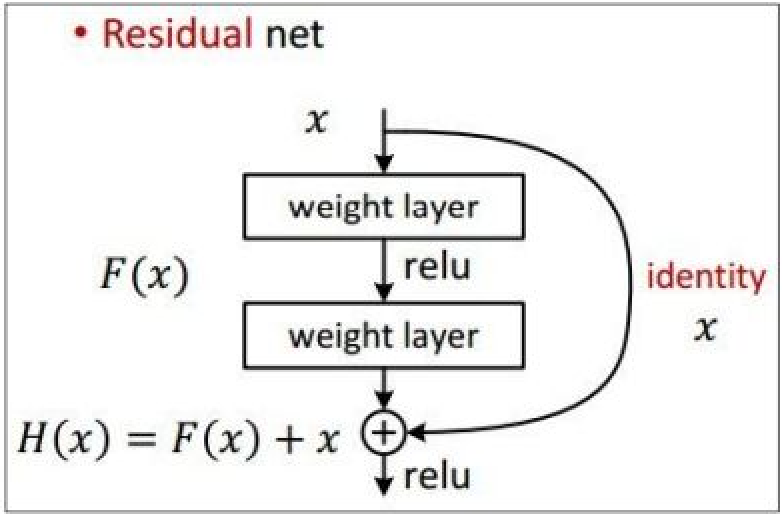
Microsoft ResNet 2015

ResNet won ILSVRC 2015 with an incredible error rate of 3.6%
Humans usually hover around 5-10%
Trained on an 8 GPU machine for two to three weeks.



ResNet - A closer look





ResNets - Key Ideas

Residual learning

Interesting to note that after only the first 2 layers, the spatial size gets compressed from an input volume of 224x224 to a 56x56 volume.

Tried a 1202-layer network, but got a lower test accuracy, presumably due to overfitting.

Do I have to train from scratch every time?

If you have the data, the time and the power you should train from scratch

But since ConvNets can take weeks to train - people make their pre-trained network weights available - Eg. Caffe Model Zoo

Initialize weights only from lower layers

	Do you have a lot of data and compute power?	
Degree of similarity of pretrained data to your own	Low, Less	Low, More
	High, Less	High, More

Train from scratch

Train from scratch

Initialize/ Use weights from a higher layer

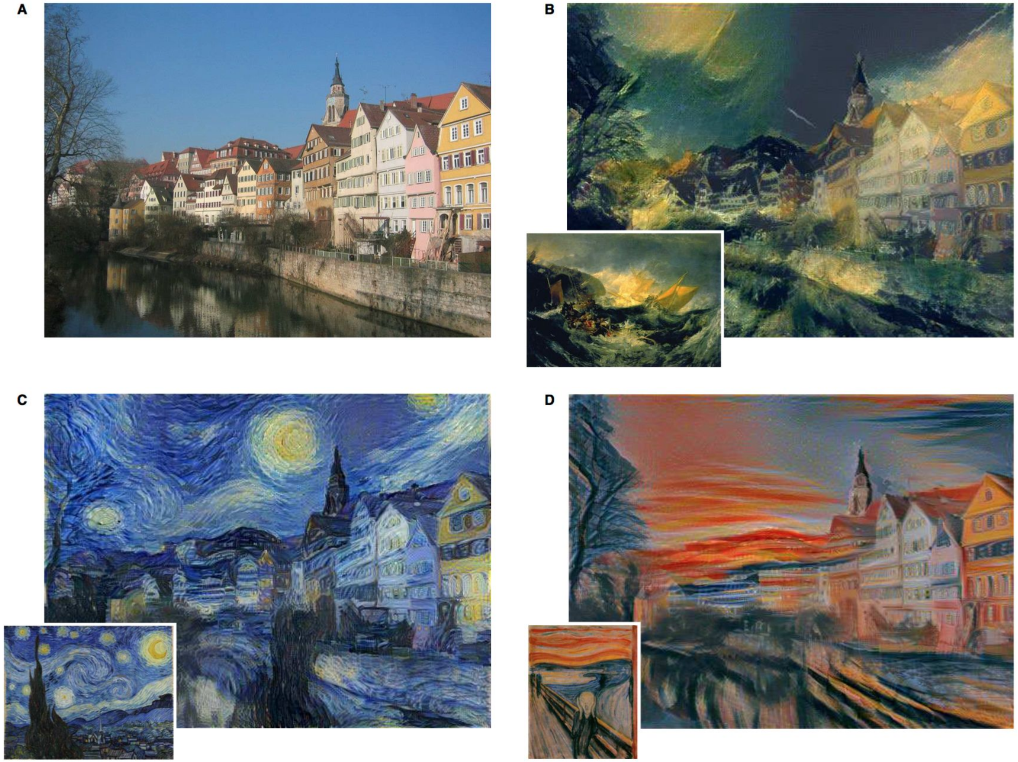
Do I have to train from scratch every time?

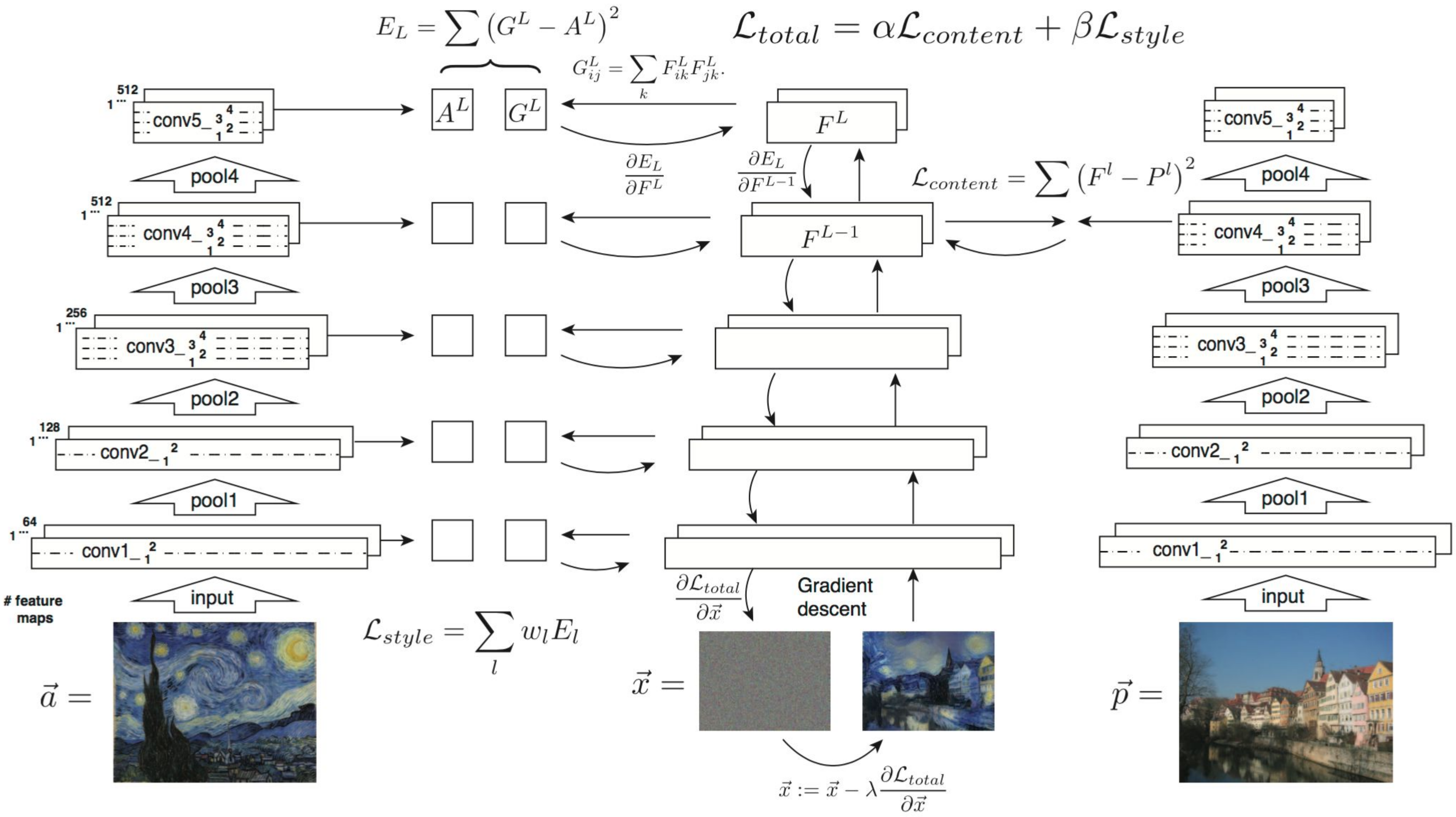
1. Use CNNs weights as initialization for your network - **Assignment 3!**

   Fine-tune the weights using your data + replace and retrain a classifier on top

2. Use CNN as a fixed feature extractor - Build SVM / some other classifier on top of it

A fun application - Style Transfer using ConvNets





$E_L = \sum (G^L - A^L)^2$
$\mathcal{L}_{total} = \alpha\mathcal{L}_{content} + \beta\mathcal{L}_{style}$
$G^L_{ij} = \sum_k F^L_{ik}F^L_{jk}.$
A^L
G^L
F^L
F^{L-1}
$\frac{\partial E_L}{\partial F^L}$
$\frac{\partial E_L}{\partial F^{L-1}}$
$\mathcal{L}_{content} = \sum (F^l - P^l)^2$
512
256
128
64
conv5_3
conv4_3
conv3_3
conv2_2
conv1_2
pool4
pool3
pool2
pool1
input
feature maps
$\mathcal{L}_{style} = \sum_l w_l E_l$
$\frac{\partial \mathcal{L}_{total}}{\partial \vec{x}}$
Gradient descent
$\vec{a} =$
$\vec{x} =$
$\vec{p} =$
$\vec{x} := \vec{x} - \lambda\frac{\partial \mathcal{L}_{total}}{\partial \vec{x}}$

Slide Credits and References

A brief overview of DL papers
https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html

http://iamaaditya.github.io

A course on CNNs http://cs231n.github.io/

LeNet paper - http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

Style transfer -
http://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Gatys_Image_Style_Transfer_CVPR_2016_paper.pdf

Slide Credits and References

Dropout (Recommended read)

http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

ResNet Tutorial

http://kaiminghe.com/icml16tutorial/icml2016_tutorial_deep_residual_networks_kaiminghe.pdf

Backpropagation Refresher (Useful read)

http://arunmallya.github.io/writeups/nn/backprop.html

Thank you!