On the importance of depth and skip connections

Deep Learning PhD Course
2017/2018

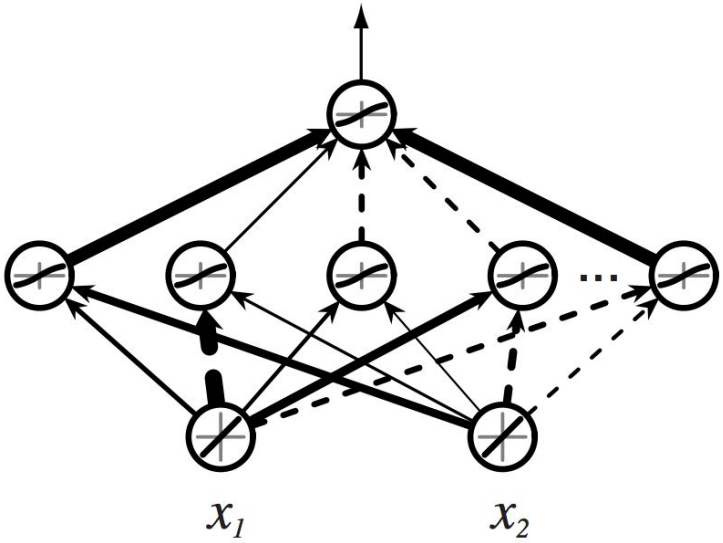
Marco Ciccone

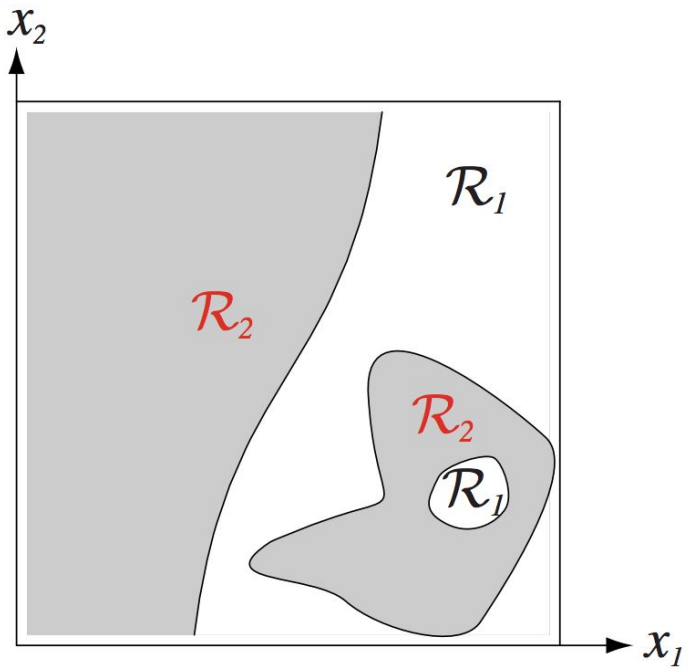
Dipartimento di Informatica Elettronica e Bioingegneria
Politecnico di Milano

Universal approximation theorem (Hornik, 1991):

“ A single hidden layer feedforward neural network can approximate any measurable function to any desired degree of accuracy on a compact set ”

NNs as universal approximators





Images from Hugo Larochelle's DL Summer School Tutorial

NNs as universal approximators

What does it mean?

- Regardless of what function we are trying to learn, a large enough MLP will be able to represent it.
- The theorem holds for linear, sigmoid, tanh and many other hidden layer activation functions.

This is a good result, but it doesn't mean there is a learning algorithm that can find the necessary parameter values!

NNs as universal approximators

In the worse case, an *exponential number of hidden units* may be required.

In summary, a feedforward network with a single layer is sufficient to represent any function, but the layer may have to be *unfeasibly large* and may fail to learn and generalize correctly.

And Deep Learning save us all…

Deep Learning

- Deep learning aims at learning models with multilayer representations
  - Multilayer (feedforward) neural network
  - Multilayer graphical model (deep belief network, deep Boltzmann machine)
  - (...)

- Each layer corresponds to a ‘‘*distributed representation*’’
  - Units in layer are not mutually exclusive
    - each unit is a separate feature of the input
    - two units can be ‘‘active’’ at the same time
  - they do not correspond to a partitioning (clustering) of the inputs
    - in clustering, an input can only belong to a single cluster

Distributed Representation I

- It is possible to represent exponential number of regions with a linear number of parameters.
- In non-distributed representations, the number of parameters are linear to the number of regions.
- Here, the number of regions potentially grow exponentially with the number of parameters and number of examples.

Deep Learning - Theoretical justification

A deep architecture can represent certain functions (exponentially) more compactly

Instead of growing our network wider, we grow it deeper

References

- "Learning Deep Architectures for AI", Yoshua Bengio, 2009
- "Exploring Strategies for Training Deep Neural Networks", Larochelle et Al, 2009
- "Shallow vs. Deep Sum-Product Networks", Delalleau & bengio, 2011
- "On the number of response regions of deep feed forward networks with piece-wise linear activations", Pascanu et Al, 2013

Distributed Representation II

- Features are individually meaningful. They remain meaningful despite the other features. There maybe some interactions but most features are learned independent of each other.
- We don’t need to see all configurations to make a meaningful statement.
- Non-mutually exclusive features create a combinatorially large set of distinguishable configurations.

Deep Learning - Theoretical justification II

- Using deep architectures expresses a useful prior over the space of functions the model learns.

- Encodes a very general belief that the function we want to learn should involve *composition of several simpler functions.*

- We can interpret the learning problem as discovering a set of underlying factors of variation that can in turn be described in terms of other, simpler underlying factors of variation.

Deep Learning - Example

Boolean functions

- A Boolean circuit is a sort of feed-forward network where hidden units are logic gates (i.e. AND, OR or NOT functions of their arguments)

- Any Boolean function can be represented by a ‘‘single hidden layer’’ Boolean circuit
  - however, it might require an exponential number of hidden units

- It can be shown that there are Boolean functions which
  - require an exponential number of hidden units in the single layer case
  - require a polynomial number of hidden units if we can adapt the number of layers

If the function we are trying to learn has a particular characteristic obtained through composition of many operations,

then it is better to approximate these functions with a deep neural network.

Remark

A deeper network does not correspond to a higher capacity.

Deeper doesn’t mean we can represent more functions.

Plain Deep Neural Networks

Neural Networks, Vol. 1, pp. 119–130, 1988
Printed in the USA. All rights reserved.

0893-6080/88 $3.00 + .00

ORIGINAL CONTRIBUTION

Neocognitron: A Hierarchical Neural Network Capable of Visual Pattern Recognition

KUNIHIKO FUKUSHIMA

NHK Science and Technical Research Laboratories

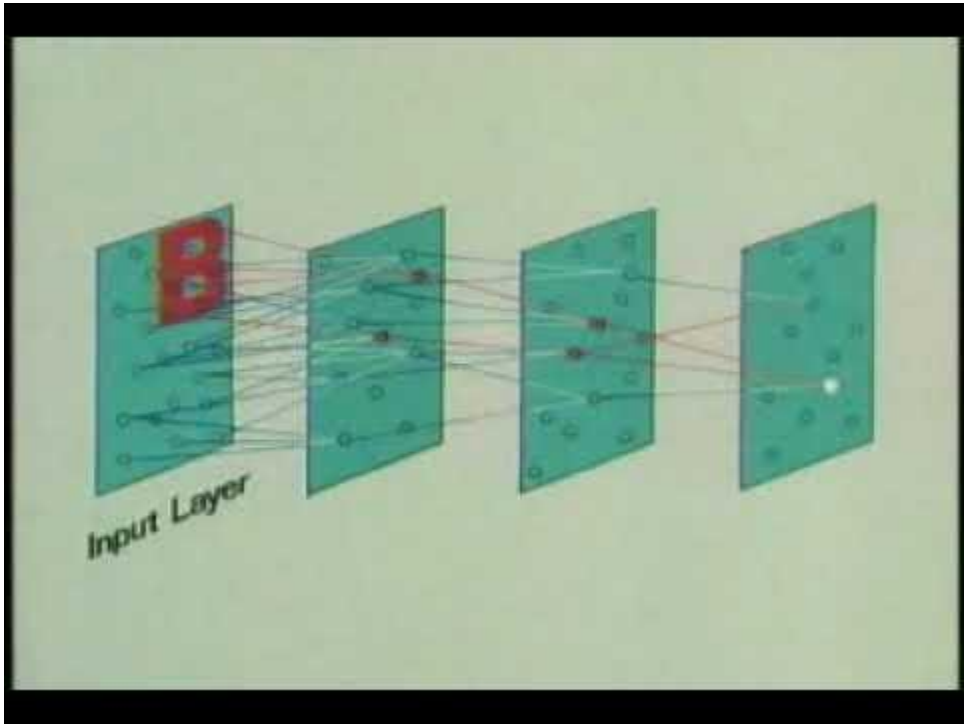
(*Received and accepted* 15 *September* 1987)

Abstract—*A neural network model for visual pattern recognition, called the "neocognitron," was previously proposed by the author. In this paper, we discuss the mechanism of the model in detail. In order to demonstrate the ability of the neocognitron, we also discuss a pattern-recognition system which works with the mechanism of the neocognitron. The system has been implemented on a minicomputer and has been trained to recognize handwritten numerals.*

The neocognitron is a hierarchical network consisting of many layers of cells, and has variable connections between the cells in adjoining layers. It can acquire the ability to recognize patterns by learning, and can be trained to recognize any set of patterns. After finishing the process of learning, pattern recognition is performed on the basis of similarity in shape between patterns, and is not affected by deformation, nor by changes in size, nor by shifts in the position of the input patterns.

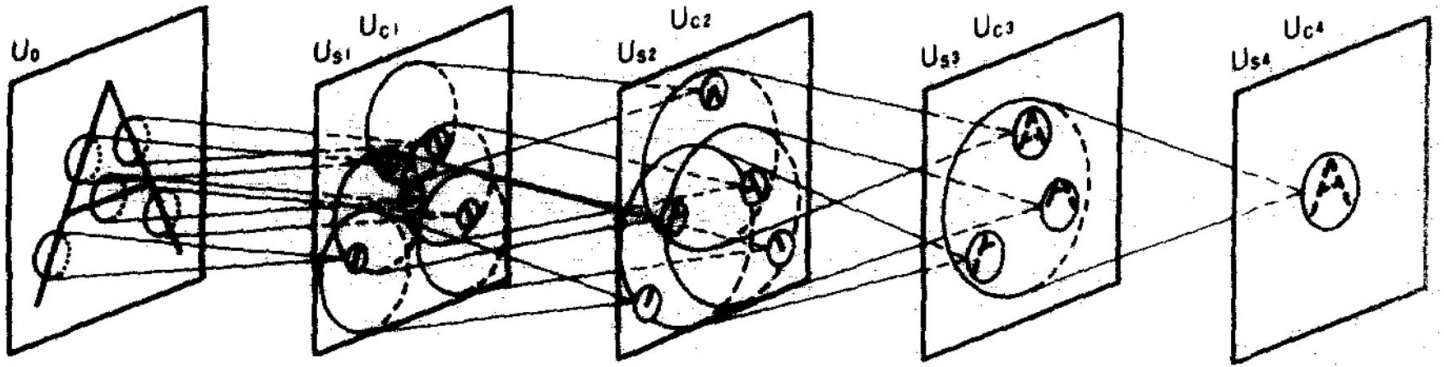
In the hierarchical network of the neocognitron, local features of the input pattern are extracted by the cells of a lower stage, and they are gradually integrated into more global features. Finally, each cell of the highest stage integrates all the information of the input pattern, and responds only to one specific pattern. Thus, the response of the cells of the highest stage shows the final result of the pattern-recognition of the network. During this process of extracting and integrating features, errors in the relative position of local features are gradually tolerated. The operation of tolerating positional error a little at a time at each stage, rather than all in one step, plays an important role in endowing the network with an ability to recognize even distorted patterns.

Neucognitron



Neucognitron: a Hierarchical Neural Network Capable of Visual Pattern Recognition, Fukushima 1988

Neucognitron



Several “cells” (S-cells and C-cells) extract and combine features at different levels.

S-CELL PROCESSING

The output of an S-cell is given by

$$u_{Sl}(\boldsymbol{n}, k) = r_l \cdot \varphi\left[\frac{\sigma_l + \sum_{\kappa=1}^{K_{Cl-1}} \sum_{\nu \in A_l} a_l(\nu, \kappa, k) \cdot u_{Cl-1}(\boldsymbol{n}+\nu, \kappa)}{\sigma_l + \frac{r_l}{1+r_l} \cdot b_l(k) \cdot u_{Vl}(\boldsymbol{n})} - 1\right] \tag{1}$$

where

$$\varphi[x] = \begin{cases} x & \text{if} \quad x \geqq 0 \\ 0 & \text{if} \quad x < 0. \end{cases} \tag{2}$$

RELU in 1988!

LeNet

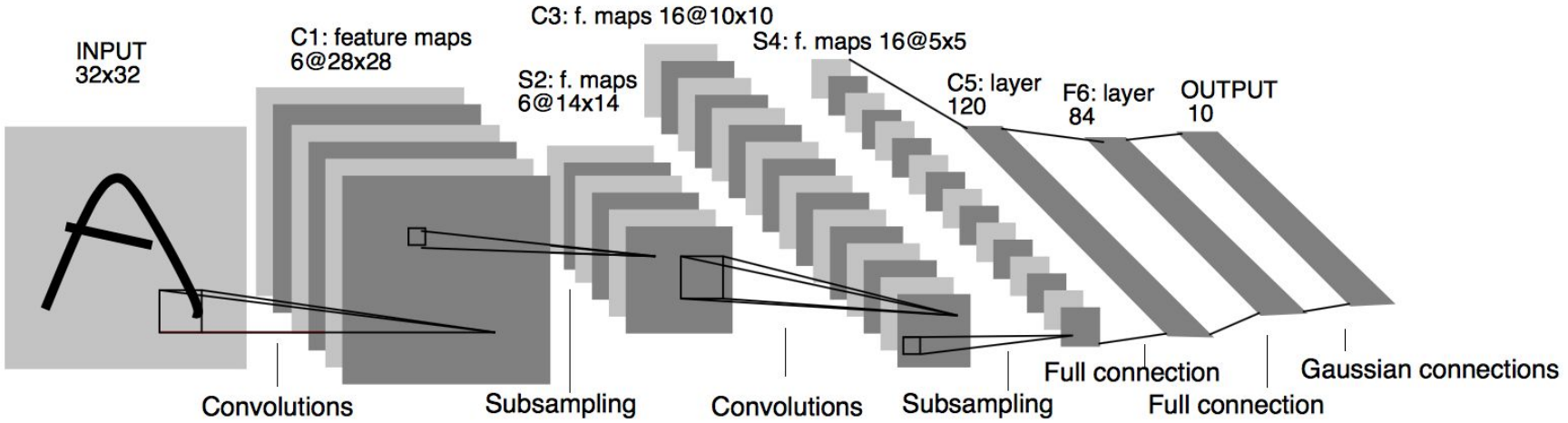
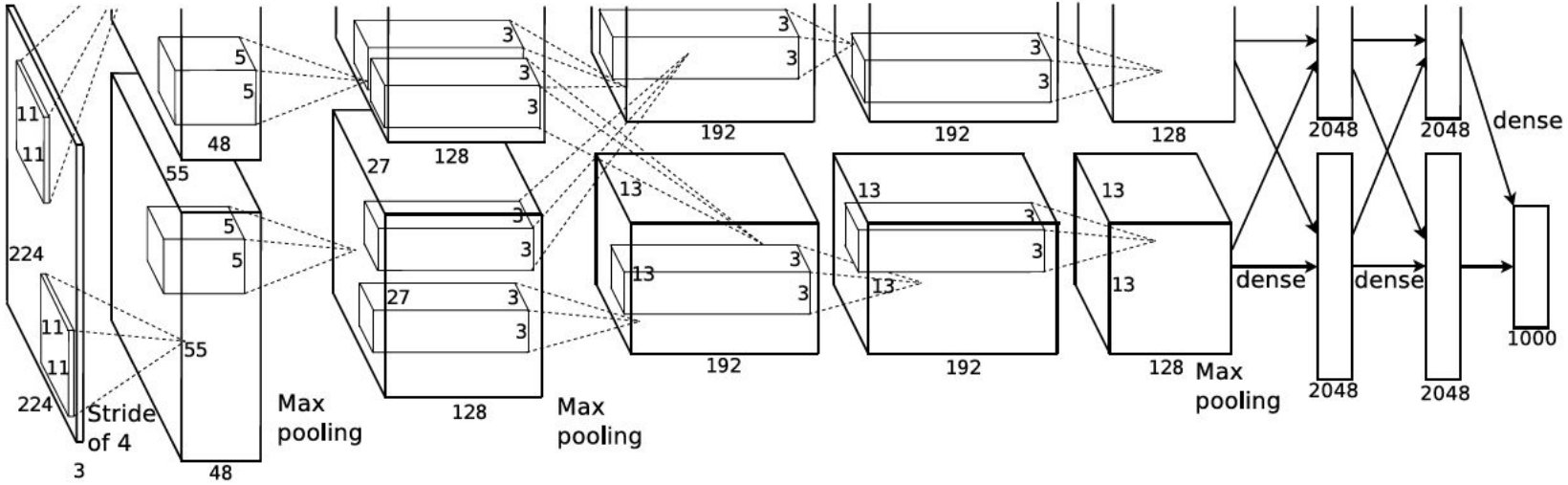


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

AlexNet (2012)

First Deep CNN actually working

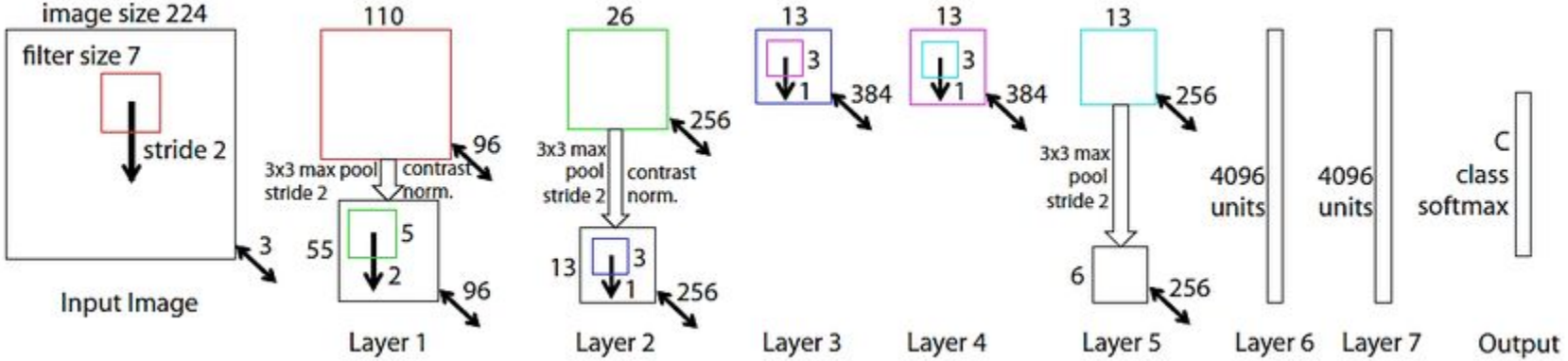
- 60 Millions of parameters
- Much of them are in the FC layers (classifier)
- Use of Dropout
- Use of RELU
- Minibatch Gradient Descent with Momentum
- Data Augmentation
- Trained for about a week on NVIDIA GTX 580



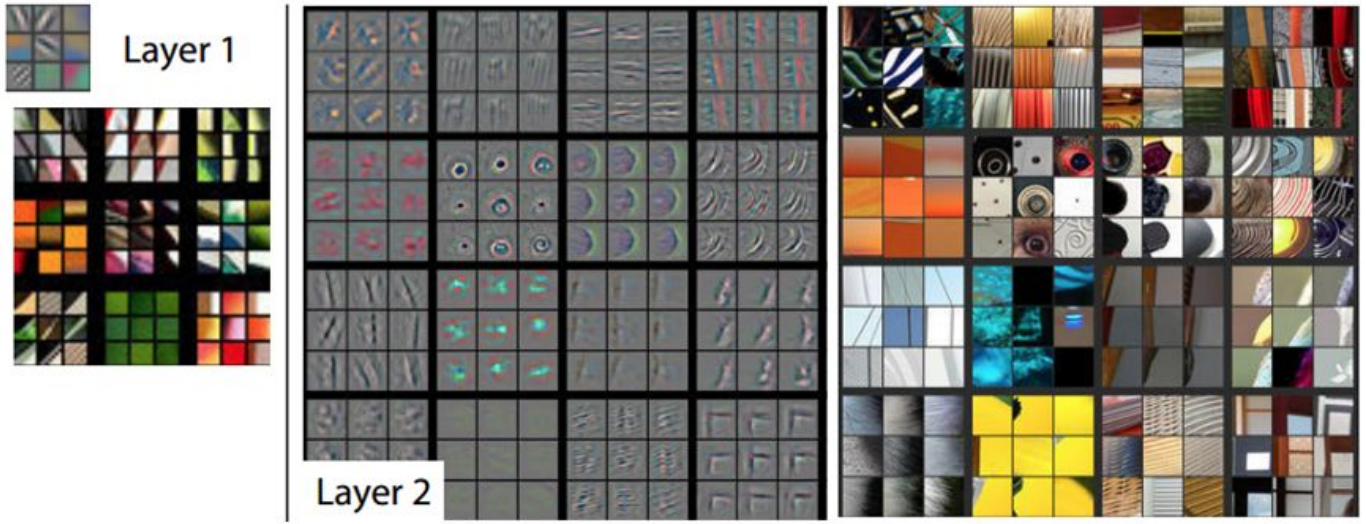
AlexNet architecture may look weird because there are two different “streams”. This is because the training process was so computationally expensive that they had to split the training onto 2 GPUs.

ZF-Net (2013)

- Improvement of AlexNet with accurate tuning of the hyperparameters
- In the same paper Zeiler and Fergus present a method to visualize the filter learned by the Network (*DeconvNet*).

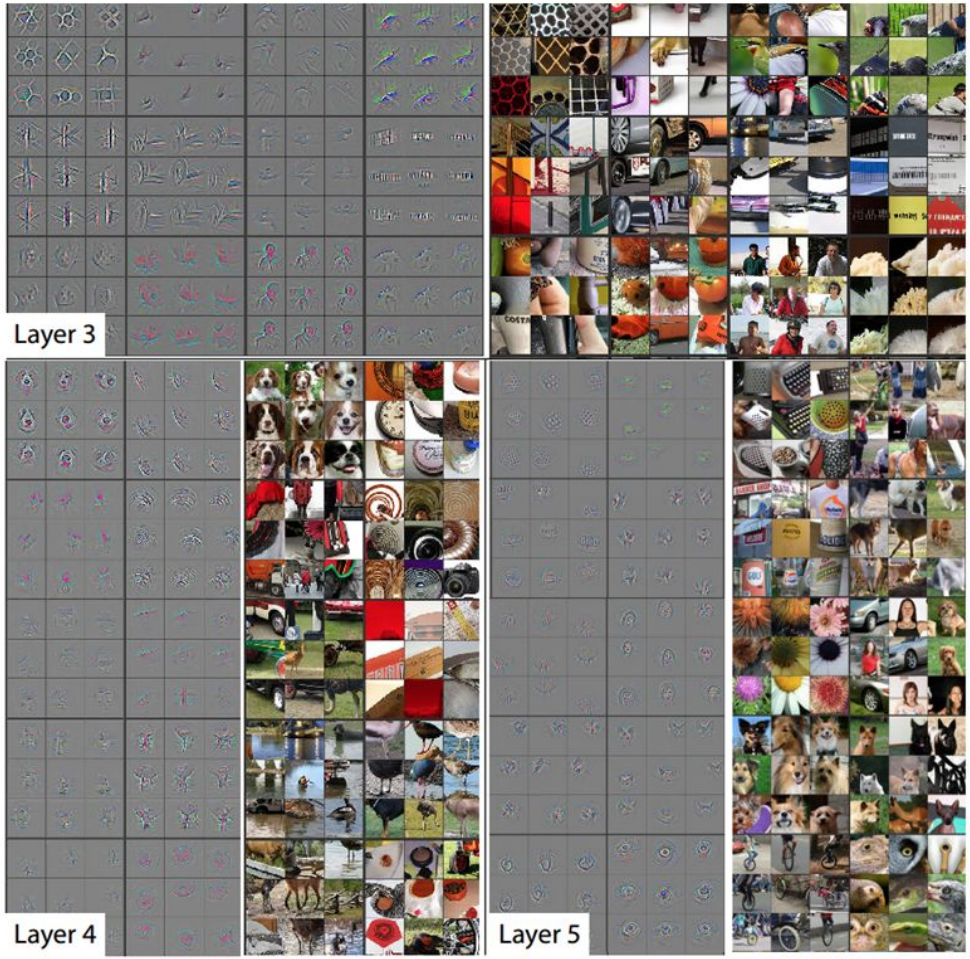


DeconvNet



Visualizations of Layer 1 and 2. Each layer illustrates 2 pictures, one which shows the filters themselves and one that shows what part of the image are most strongly activated by the given filter. For example, in the space labled Layer 2, we have representations of the 16 different filters (on the left)

DeconvNet



Visualizations of Layers 3, 4, and 5

VGG-16 (2014)

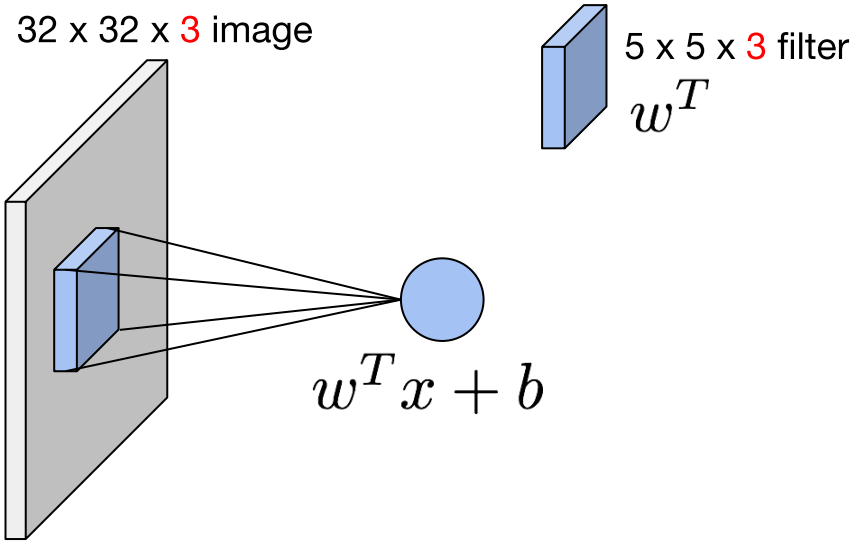
- Use 3x3 kernels
- Block of stacked 3 conv layers have an effective receptive field of 7x7 but less parameters (+3 nonlinearities)
- Today we have a nice efficient 3x3 convolution (Winograd)
- Lots of parameters **140 M** (Most of them are in the FC Layers)
- Extracts very general and transferable features
- FC Layers are redundant and can be removed (without loss of accuracy)
- The structure can be used for other tasks such as Semantic Segmentation

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

Recall on CNNs

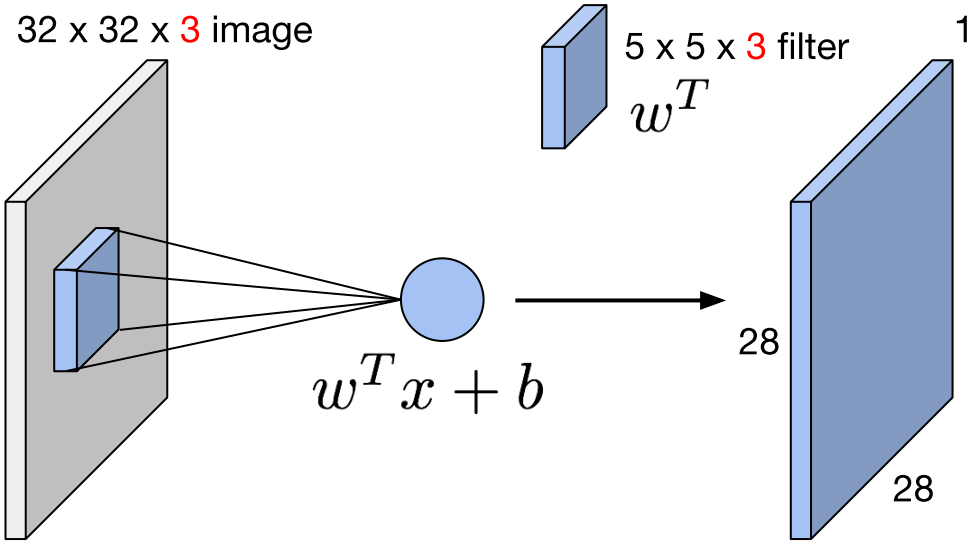


The kernel is convolved with the image i.e. slide the filter over all the input locations and compute the dot product

The result of the dot product between the elements of the kernel (weights) and the small region of the input is one number.

This is a local linear combination of the input features.

Recall on CNNs

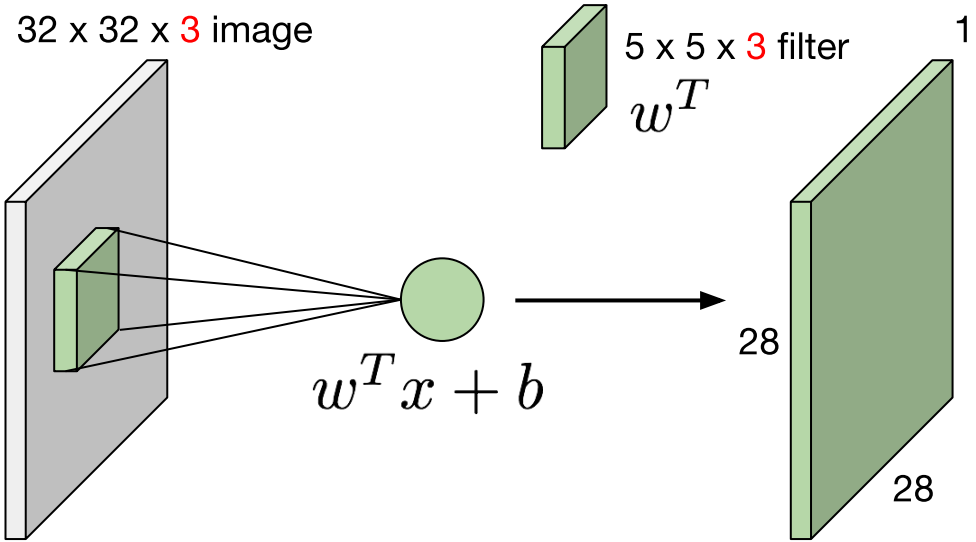


The result of the convolution between the input and the kernel is called **“feature map”** or **“activation map”**

Note:
Depending on the type of convolution, the feature maps have a different size:

- VALID (n - k + 1)
- SAME
- FULL (n + k -1)

Recall on CNNs

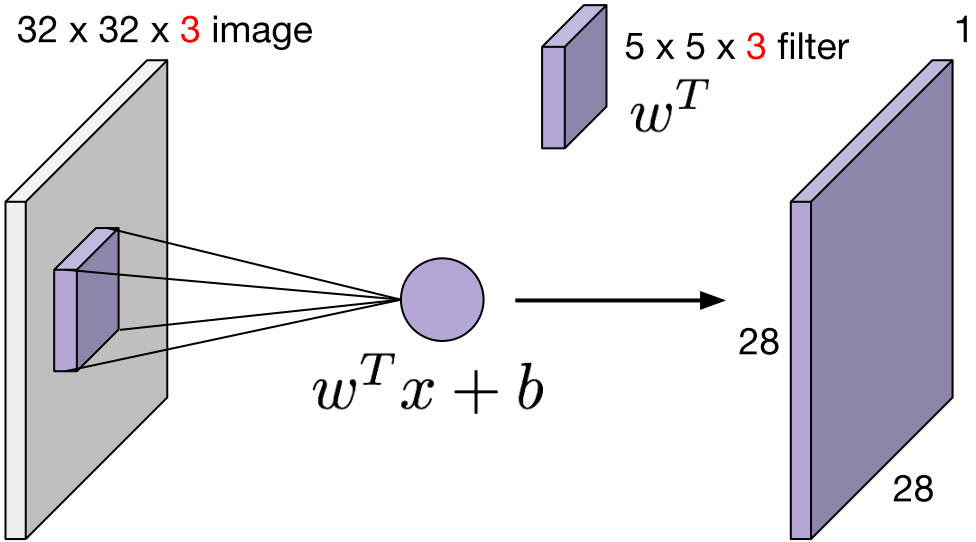


Each color corresponds to a different filter that:

- Shares its parameters among all the locations of the input
- Extracts a different feature independently on the location in which is applied

The result of the convolution between the input and a kernel gives a **feature map.**

Recall on CNNs

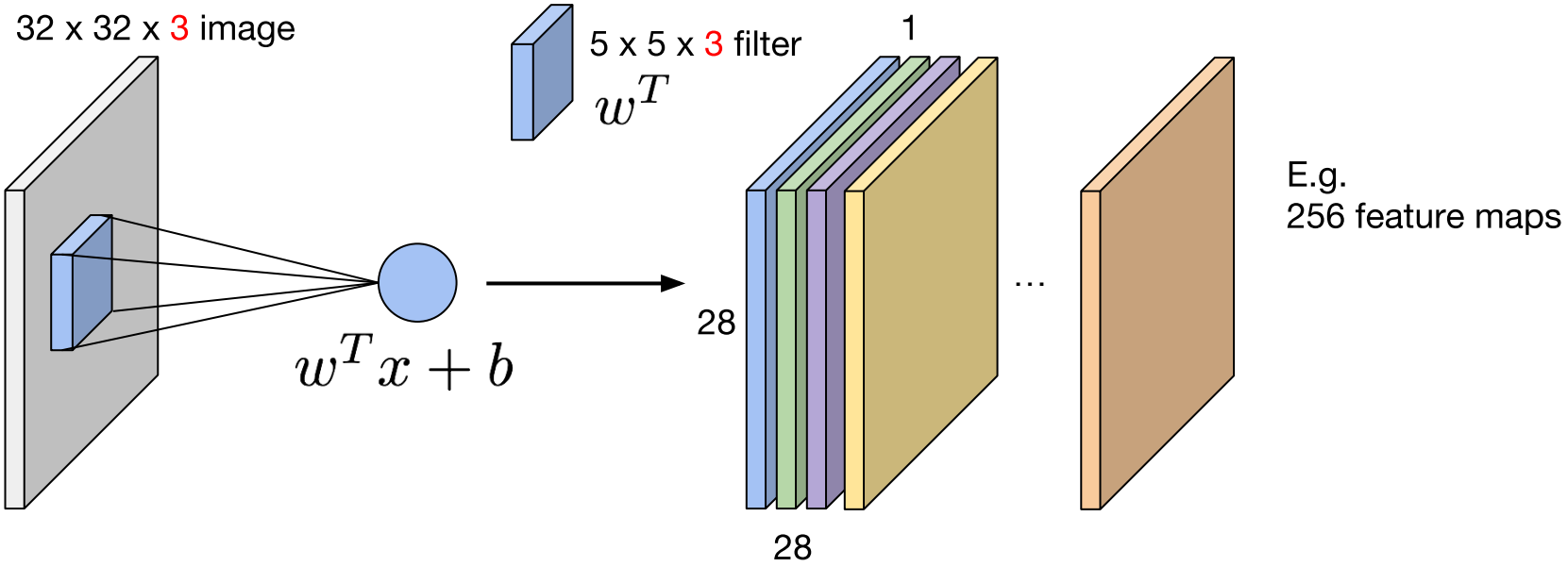


Each color corresponds to a different filter that:

- Shares its parameters among all the locations of the input
- And extract a different feature independently on the location in which is applied

The result of the convolution between the input and a kernel gives a **feature map.**

Recall on CNNs

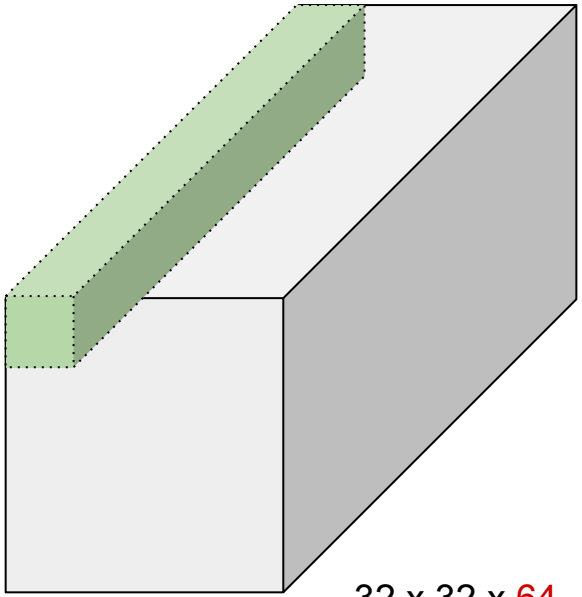


CNNs learn many different filters:
Each one is specialized in extracting a specific feature in the image.

Each multidimensional filter linearly combines the input (locally) and produce a feature map.

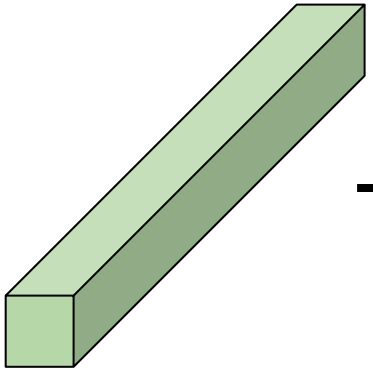
Network in Network (NiN)

Use 1x1 Convolutions



32 x 32 x 64

*

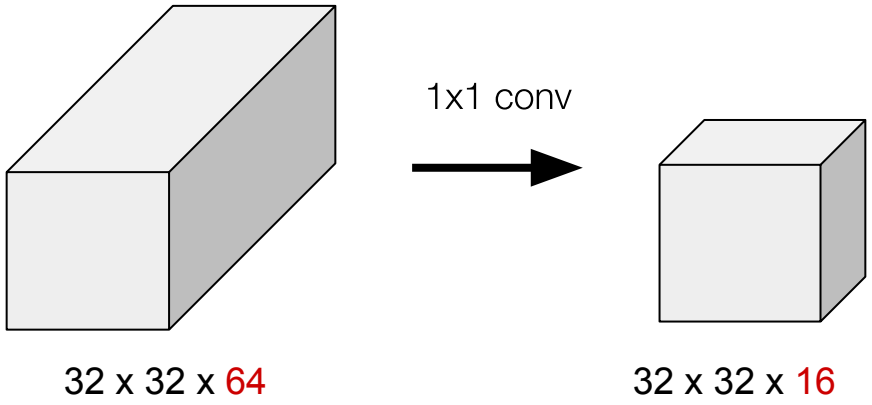


1 x 1 x 64

Activation

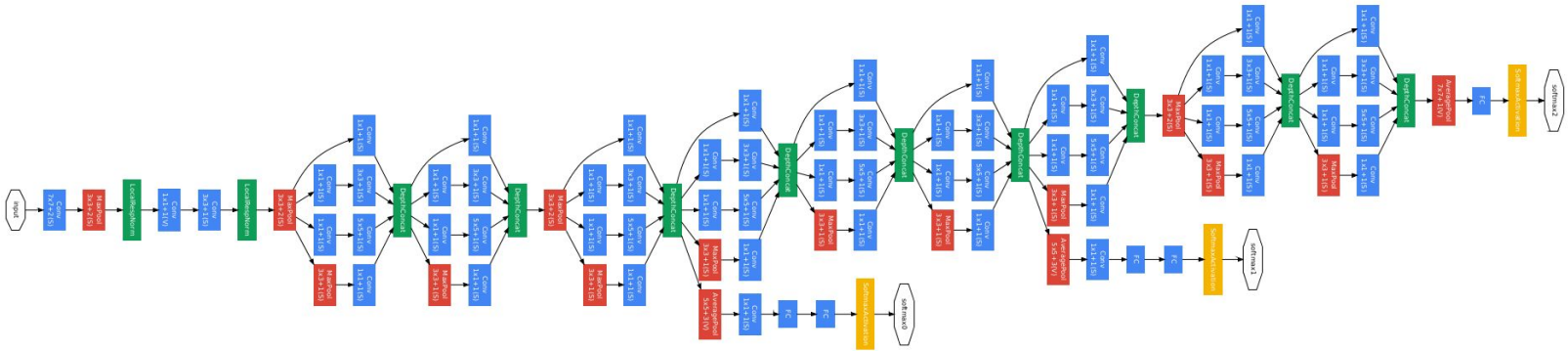
1x1 Convolution

- It combines features over the channel dimension
- It's a fully connected over the channel dimension applied in convolutional way
- Useful for (nonlinear) dimensionality reduction: apply convolution on all the input channels, but have less filters producing less feature maps.



GoogleNet (2014)

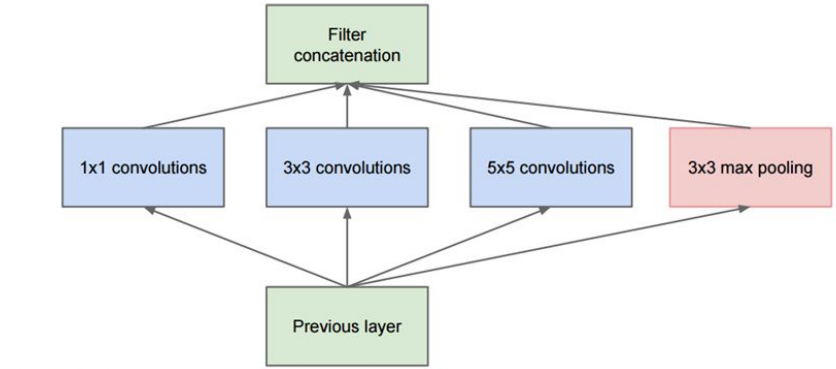
- Used 9 Inception modules in the whole architecture, with over 100 layers in total! Now that is deep…
- No use of fully connected layers! They use an average pool instead, to go from a 7x7x1024 volume to a 1x1x1024 volume. This saves a huge number of parameters.
- Uses 12x fewer parameters than AlexNet (4 Millions)
- At test time, multiple crops of the same image were created, fed into the network, and the softmax probabilities were averaged to give us the final solution.
- There are several updated versions to the Inception module (Versions 6 and 7).



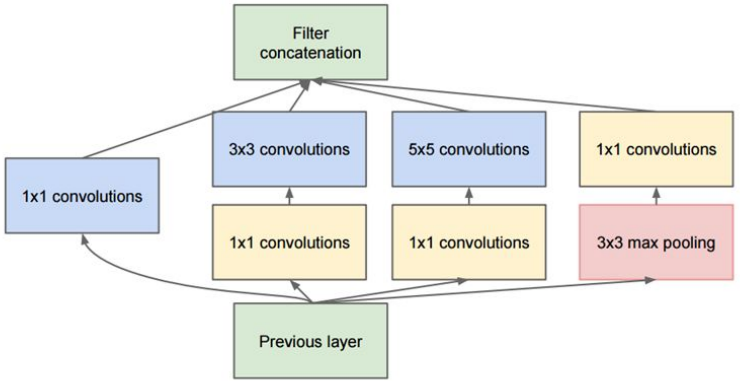
Inception module

GoogLeNet 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.



Naïve idea of an Inception module



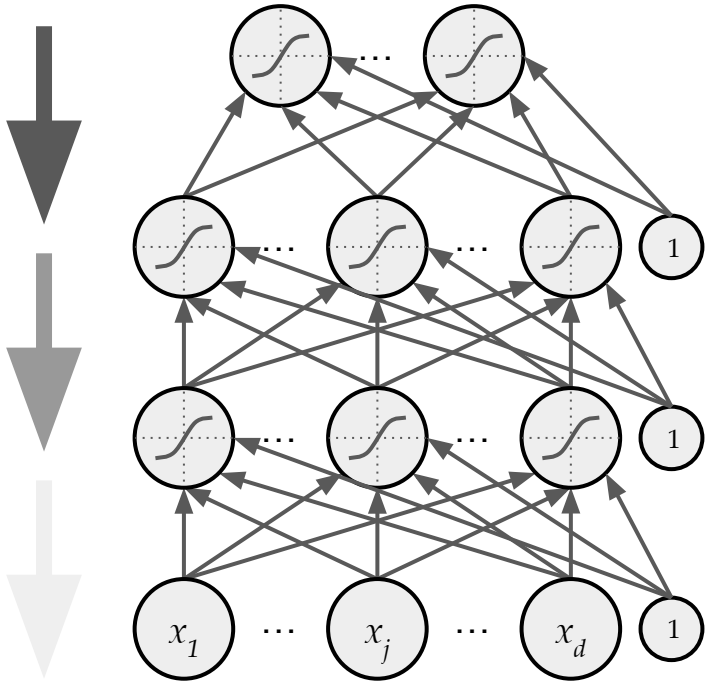
Full Inception module

Training a Deep Neural Network is hard

Optimization is harder (underfitting)

- *Vanishing gradient problem*
- Saturated units block gradient propagation

This is a well known problem in *recurrent neural networks* (we'll see in a few lectures)



Skip Connections

U-net

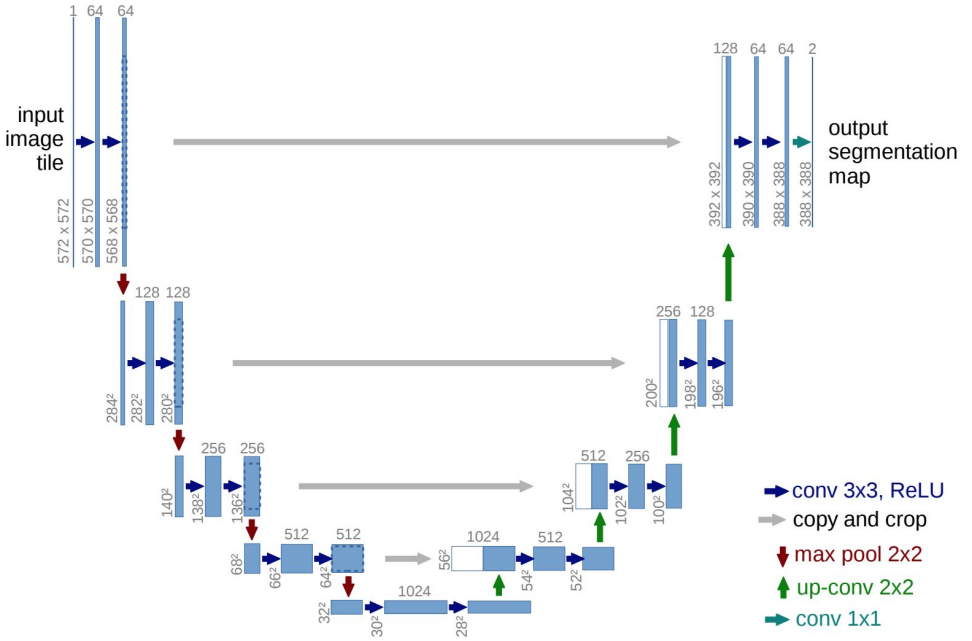


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Additive Compositional Layers

Highway Networks

T: Transform Gate C: Carry Gate

$$\mathbf{y} = H(\mathbf{x}, \mathbf{W_H}) \cdot T(\mathbf{x}, \mathbf{W_T}) + \mathbf{x} \cdot C(\mathbf{x}, \mathbf{W_C}).$$

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

Coupled gates C = 1 - T

$$\mathbf{y} = \begin{cases} \mathbf{x}, & \text{if } T(\mathbf{x}, \mathbf{W_T}) = \mathbf{0}, \\ H(\mathbf{x}, \mathbf{W_H}), & \text{if } T(\mathbf{x}, \mathbf{W_T}) = \mathbf{1}. \end{cases} \qquad \frac{d\mathbf{y}}{d\mathbf{x}} = \begin{cases} \mathbf{I}, & \text{if } T(\mathbf{x}, \mathbf{W_T}) = \mathbf{0}, \\ H'(\mathbf{x}, \mathbf{W_H}), & \text{if } T(\mathbf{x}, \mathbf{W_T}) = \mathbf{1}. \end{cases}$$

Highway Networks, Srivastava et al (2015)
Training Very Deep Networks, Srivastava et al (2015)

Highway Networks

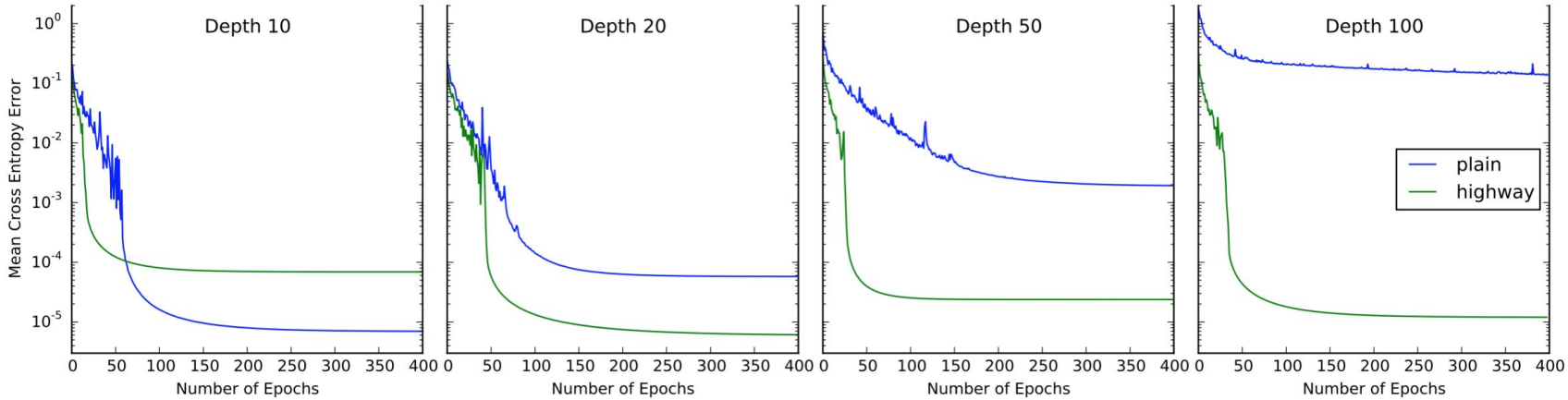


Figure 1. Comparison of optimization of plain networks and highway networks of various depths. All networks were optimized using SGD with momentum. The curves shown are for the best hyperparameter settings obtained for each configuration using a random search. Plain networks become much harder to optimize with increasing depth, while highway networks with up to 100 layers can still be optimized well.

From http://people.idsia.ch/~rupesh/very_deep_learning/

[…]

"This project presents a different take on the problem.

We simply redesign neural networks in a way that makes them ***easier to optimize even for very large depths.***"

Residual Networks (ResNet)

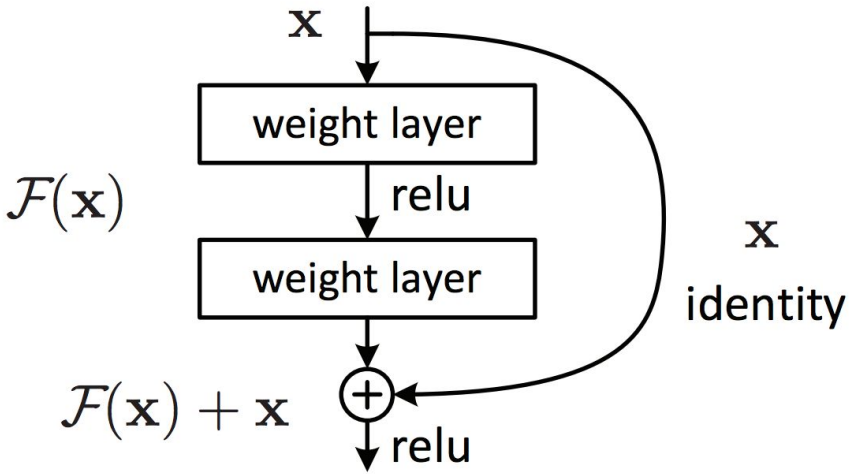
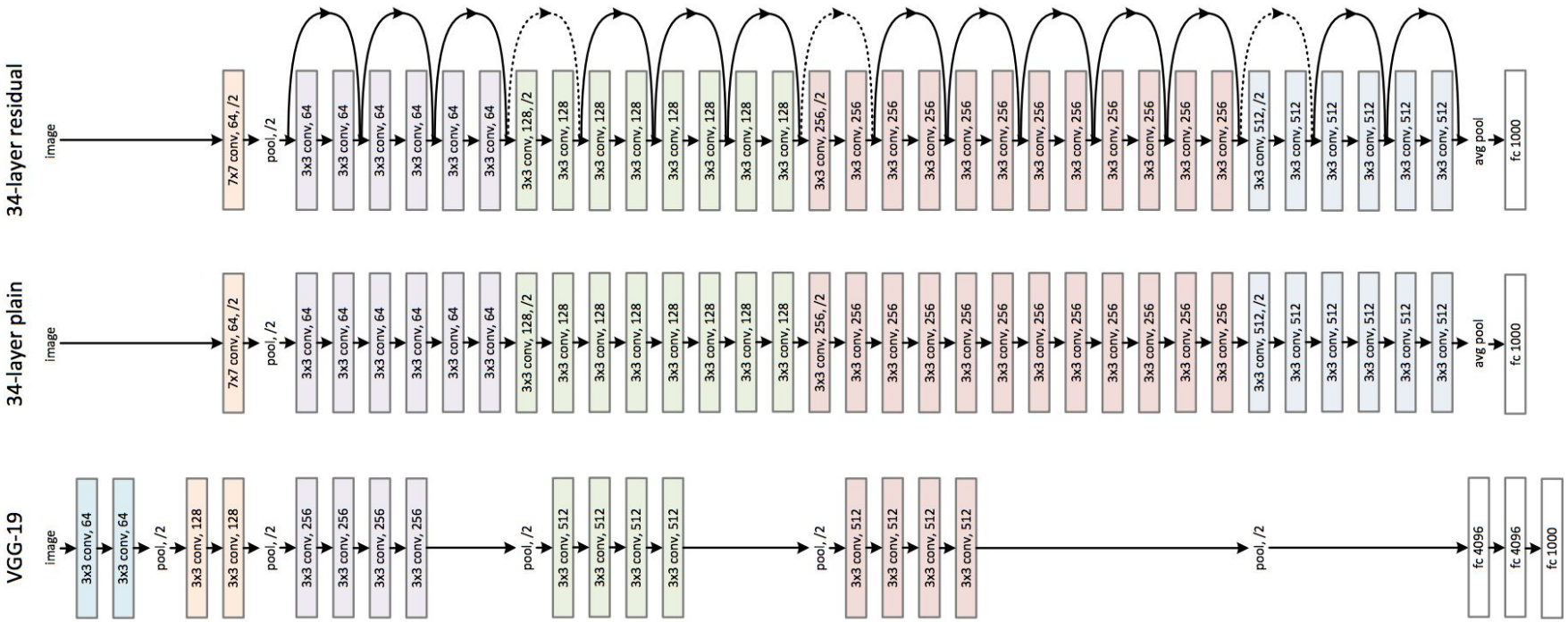


Figure 2. Residual learning: a building block.

Deep Residual Learning for Image Recognition, He et al (2015)

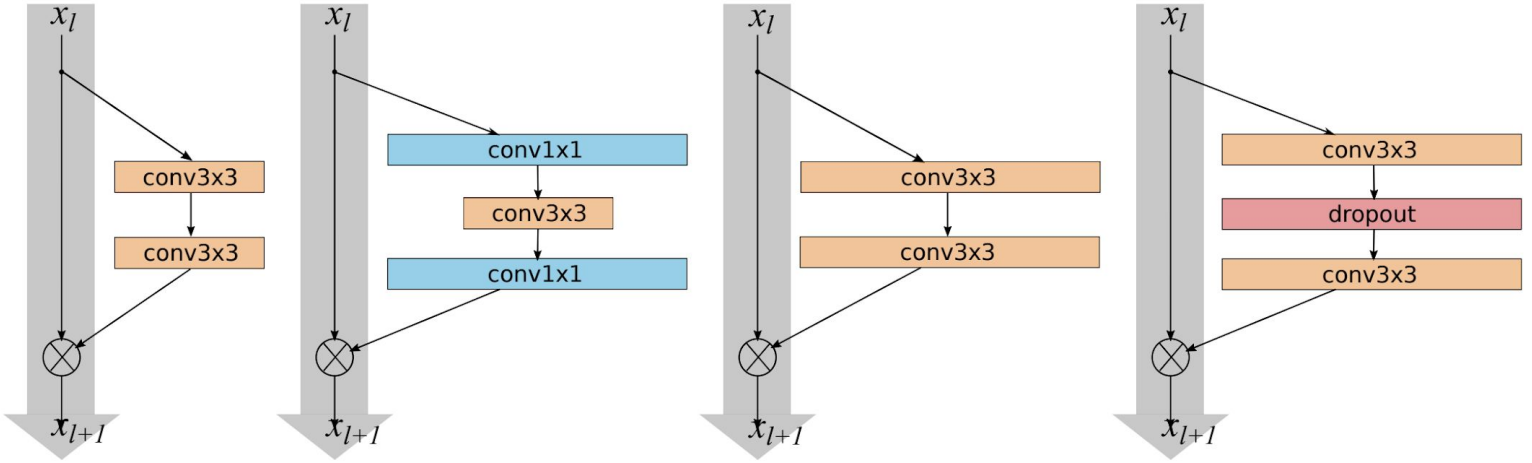
Residual Networks (ResNet)



Remember:

Deeper doesn’t mean better!

Wide ResNet



(a) basic (b) bottleneck (c) basic-wide (d) wide-dropout

Figure 1: Various residual blocks used in the paper. Batch normalization and ReLU precede each convolution (omitted for clarity)

Wide Residual Networks, Zagoruyko ad Komodakis 2016

Wide ResNet

group name	output size	block type = $B(3,3)$
conv1	32×32	[3×3, 16]
conv2	32×32	$\begin{bmatrix} 3\times3, 16\times k \\ 3\times3, 16\times k \end{bmatrix} \times N$
conv3	16×16	$\begin{bmatrix} 3\times3, 32\times k \\ 3\times3, 32\times k \end{bmatrix} \times N$
conv4	8×8	$\begin{bmatrix} 3\times3, 64\times k \\ 3\times3, 64\times k \end{bmatrix} \times N$
avg-pool	1×1	$[8 \times 8]$

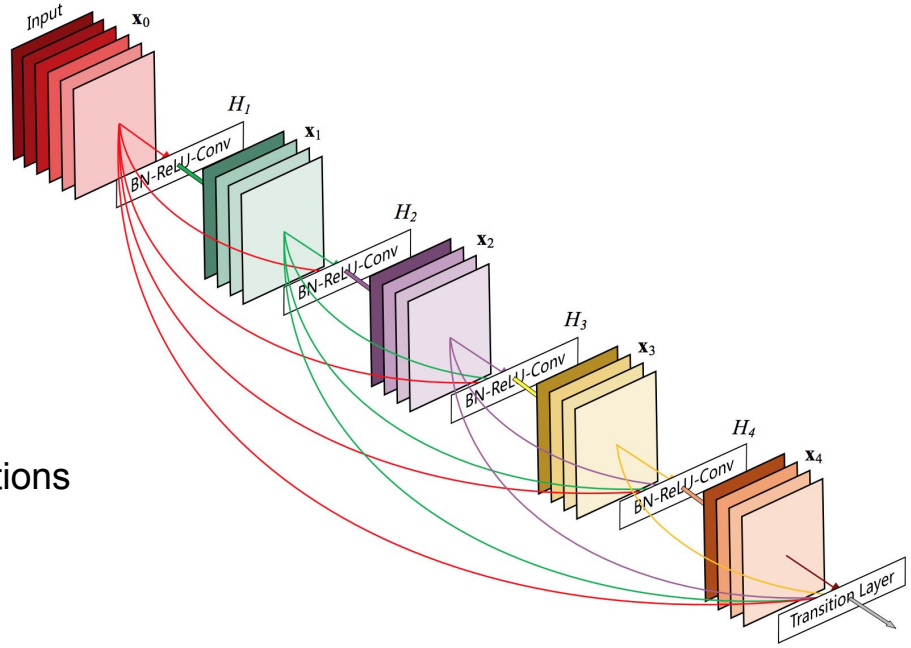
Increase the number of filters by a widening factor k, rather than depth

Wide ResNet

k: widen factor

	depth-k	# params	CIFAR-10	CIFAR-100
NIN [20]			8.81	35.67
DSN [19]			8.22	34.57
FitNet [24]			8.39	35.04
Highway [28]			7.72	32.39
ELU [6]			6.55	24.28
original-ResNet[11]	110	1.7M	6.43	25.16
	1202	10.2M	7.93	27.82
stoc-depth[14]	110	1.7M	5.23	24.58
	1202	10.2M	4.91	-
pre-act-ResNet[13]	110	1.7M	6.37	-
	164	1.7M	5.46	24.33
	1001	10.2M	4.92(4.64)	22.71
WRN (ours)	40-4	8.9M	4.53	21.18
	16-8	11.0M	4.27	20.43
	28-10	36.5M	**4.00**	**19.25**

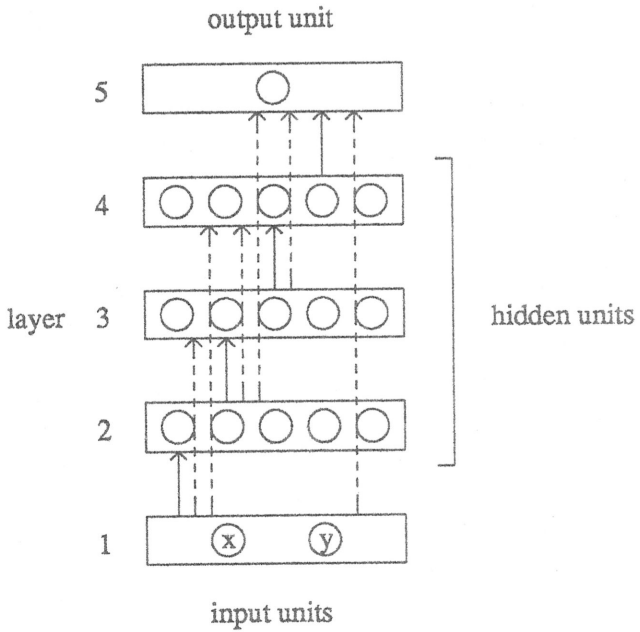
DenseNet



L(L+1) / 2 direct connections

[Densely Connected Convolutional Networks, Huang et al 2016]

DenseNet (1988)



Lang, K. J., and Witbrock. M. (1988). “Learning to Tell Two Spirals Apart.” In Proc. of 1988 Connectionist Models Summer School

Stochastic Depth Net

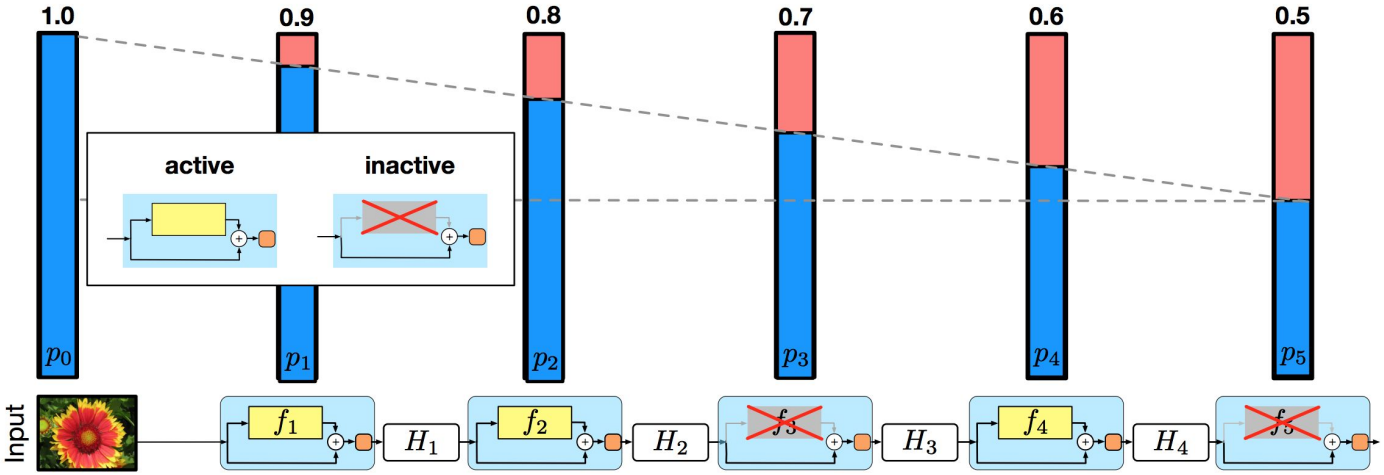


Fig. 2. The linear decay of p_ℓ illustrated on a ResNet with stochastic depth for $p_0 = 1$ and $p_L = 0.5$. Conceptually, we treat the input to the first ResBlock as H_0, which is always active.

Deep Networks with Stochastic Depth, Huang et al (2015)

But what are this networks learning?

Highway Network (Lesioning)

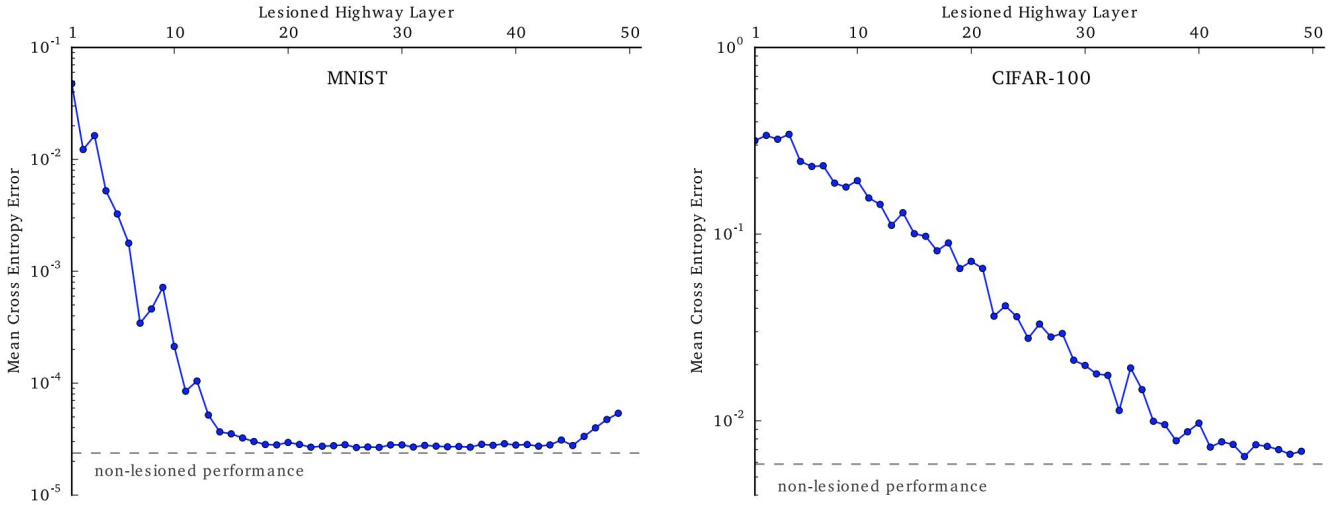
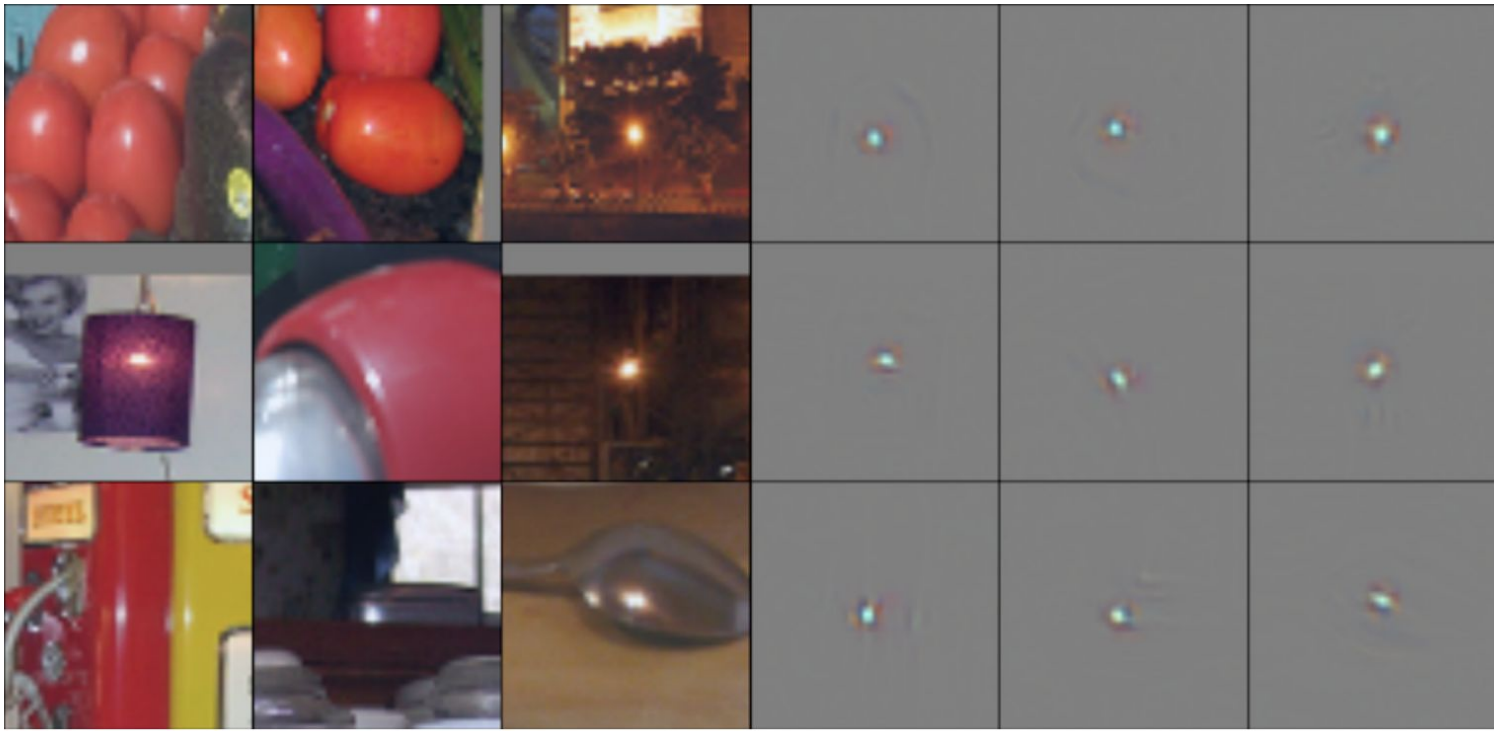


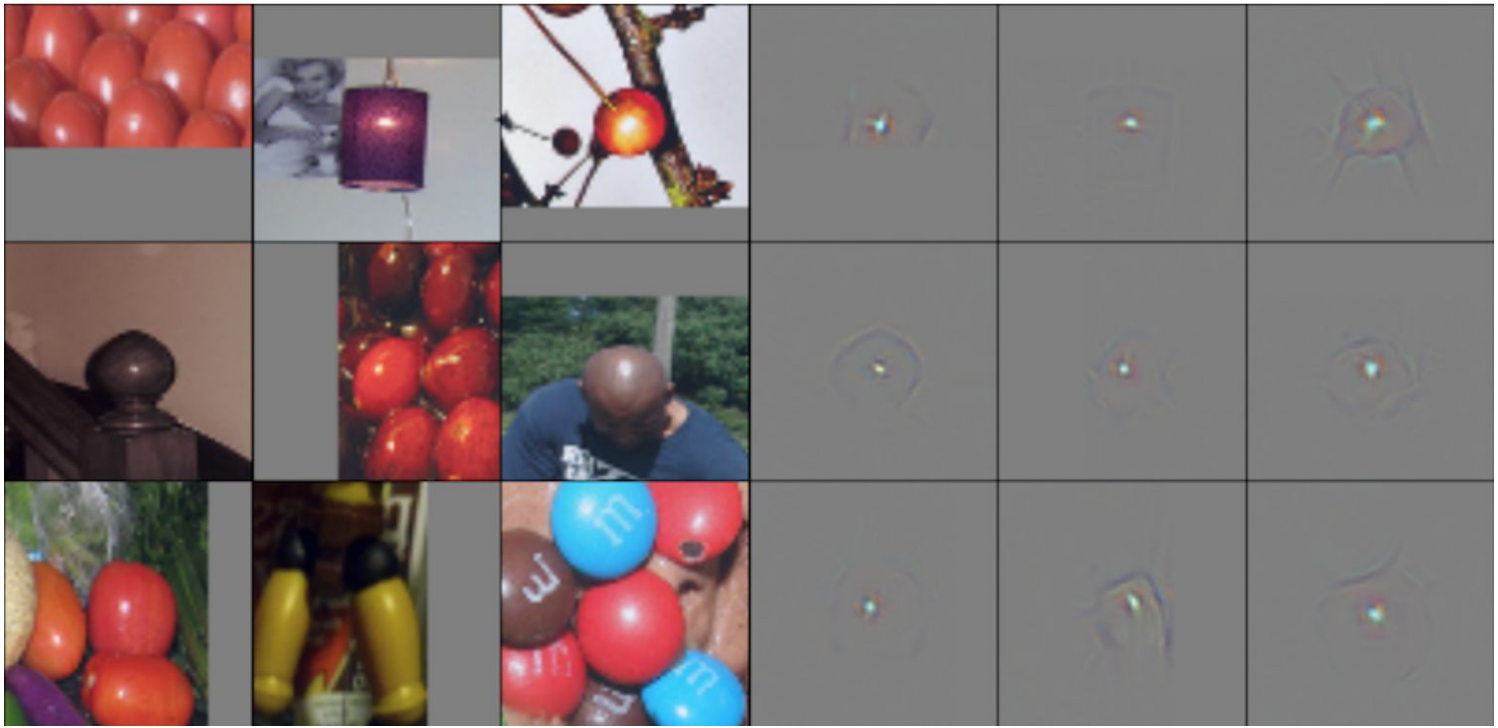
Figure 4: Lesioned training set performance (y-axis) of the best 50-layer highway networks on MNIST (left) and CIFAR-100 (right), as a function of the lesioned layer (x-axis). Evaluated on the full training set while forcefully closing all the transform gates of a single layer at a time. The non-lesioned performance is indicated as a dashed line at the bottom.

Feature Refinement (ResNet block a)



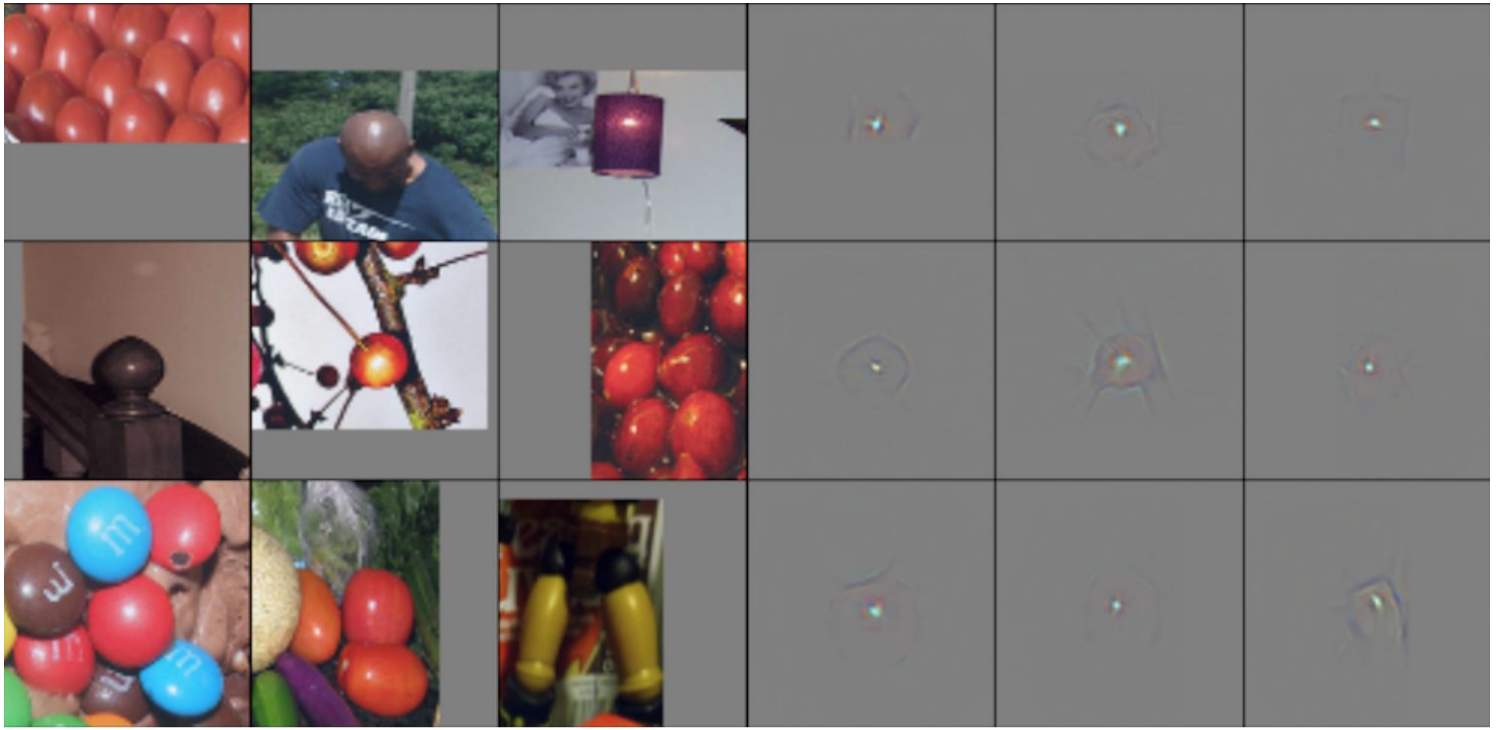
It shows how the response of a single filter (unit) evolves over the three blocks of stage 1 in a 50-layer ResNet trained on ImageNet.

Feature Refinement (ResNet block b)



On the left of each visualization are the top 9 patches from the ImageNet validation set that maximally activated that filter. To the right the corresponding guided backpropagation (Springenberg et al., 2014) visualizations are shown.

Feature Refinement (ResNet block c)



Chu et al. (2017) observed: “[. . .] residual layers of the same dimensionality learn features that get refined and sharpened

Iterative Refinement (Loss)

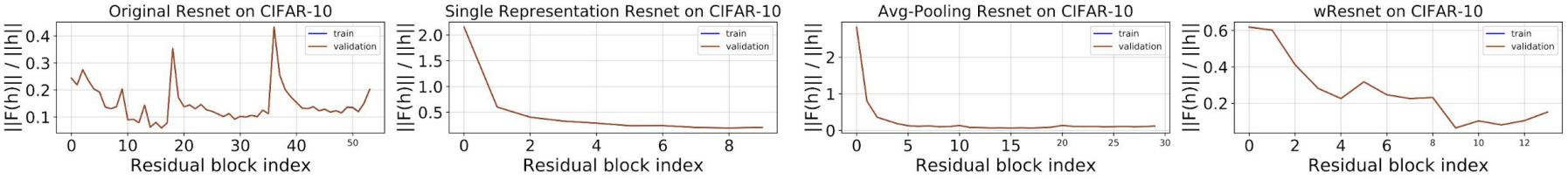


Figure 2: Average ratio of ℓ^2 norm of output of residual block to the norm of the input of residual block for (left to right) original Resnet, single representation Resnet, avg-pooling Resnet, and wideResnet on CIFAR-10. (Train and validation curves are overlapping.)

[Residual Connections Encourage Iterative Inference, Jastrzebski et al (2017)]

Recap

- Plain Deep Networks are hard to train due to gradient issues
- Skip connections are very good way of dealing with those problems
- This comes with interesting properties (and priors) that has to be studied

Thanks for your attention.