MIT CSAIL

MIT COMPUTER VISION

6.869: Advances in Computer Vision

William T. Freeman, Antonio Torralba, 2017

Lecture 8

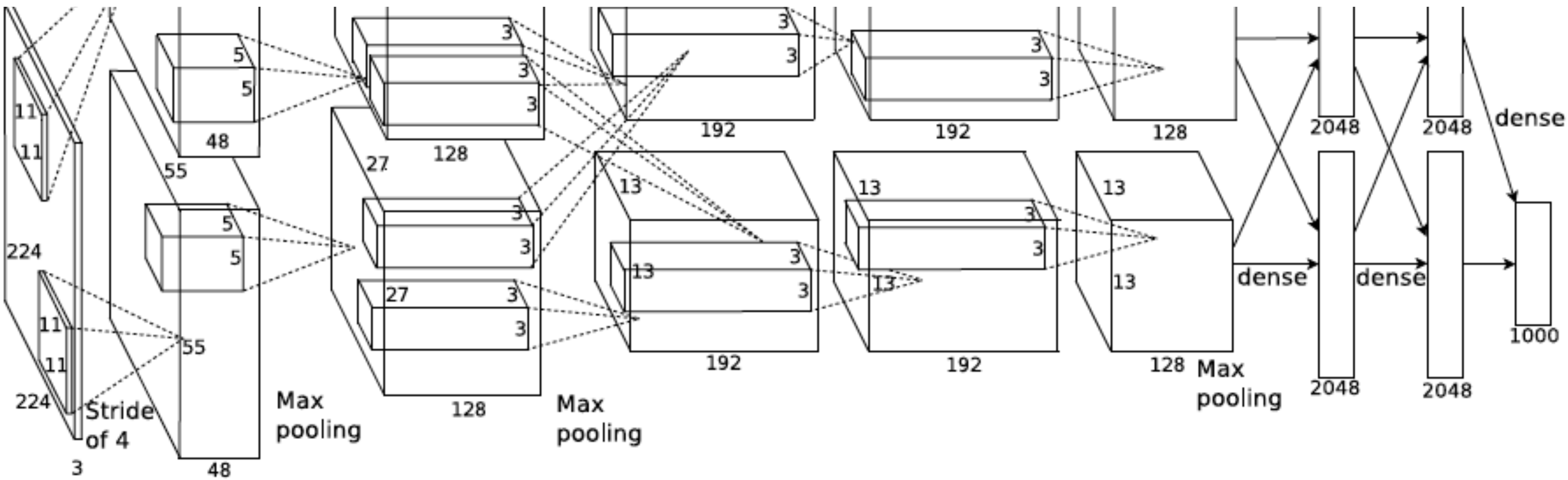
Learned feedforward visual processing
Neural Networks, Deep learning, ConvNets

How convnets work

- Operations in each layer
- Architecture
- Training
- Results

Krizhevsky et al. [NIPS2012]

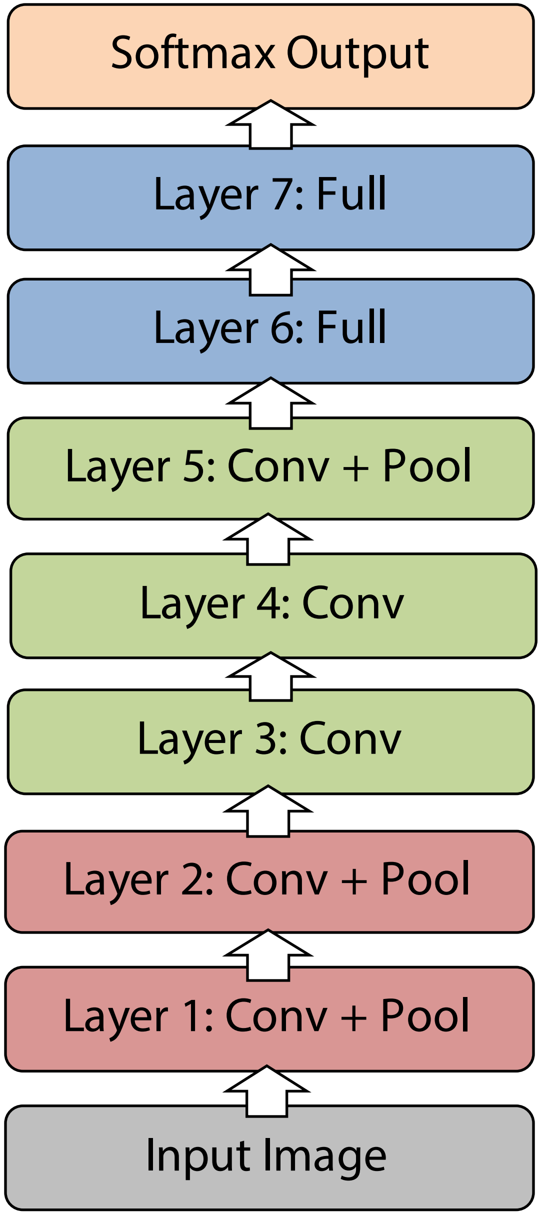
- Same model as LeCun'98 but:
  - Bigger model (8 layers)
  - More data (10^6 vs 10^3 images)
  - GPU implementation (50x speedup over CPU)
  - Better regularization (DropOut)



- 7 hidden layers, 650,000 neurons, 60,000,000 parameters
- Trained on 2 GPUs for a week

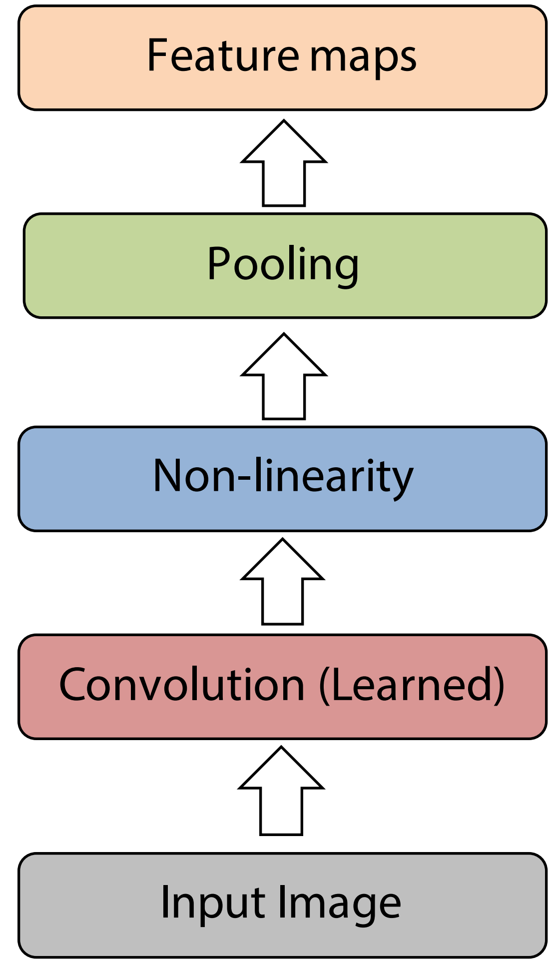
Architecture of Krizhevsky et al.

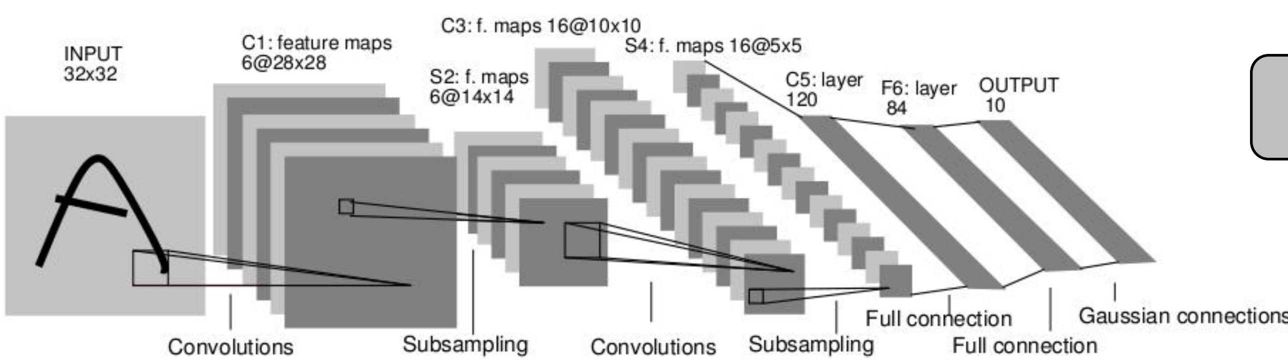
- 8 layers total



Overview of Convnets

- Feed-forward:
  - Convolve input
  - Non-linearity (rectified linear)
  - Pooling (local max)
- Supervised
- Train convolutional filters by back-propagating classification error



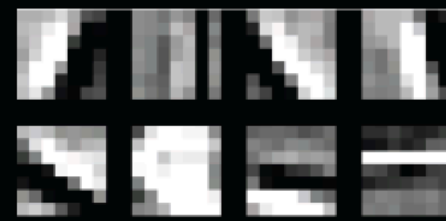


LeCun et al. 1998

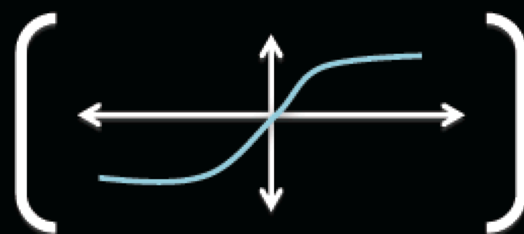
Components of Each Layer

Pixels / Features

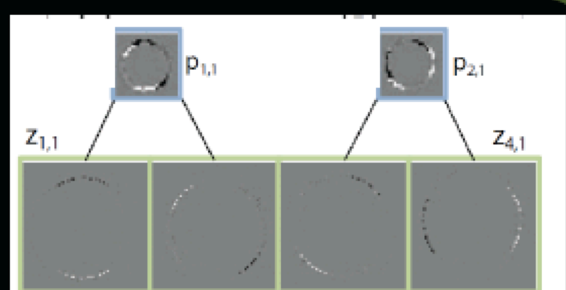
Filter with learned dictionary



Non-linearity



Spatial local max pooling



[Optional] Normalization across data/features

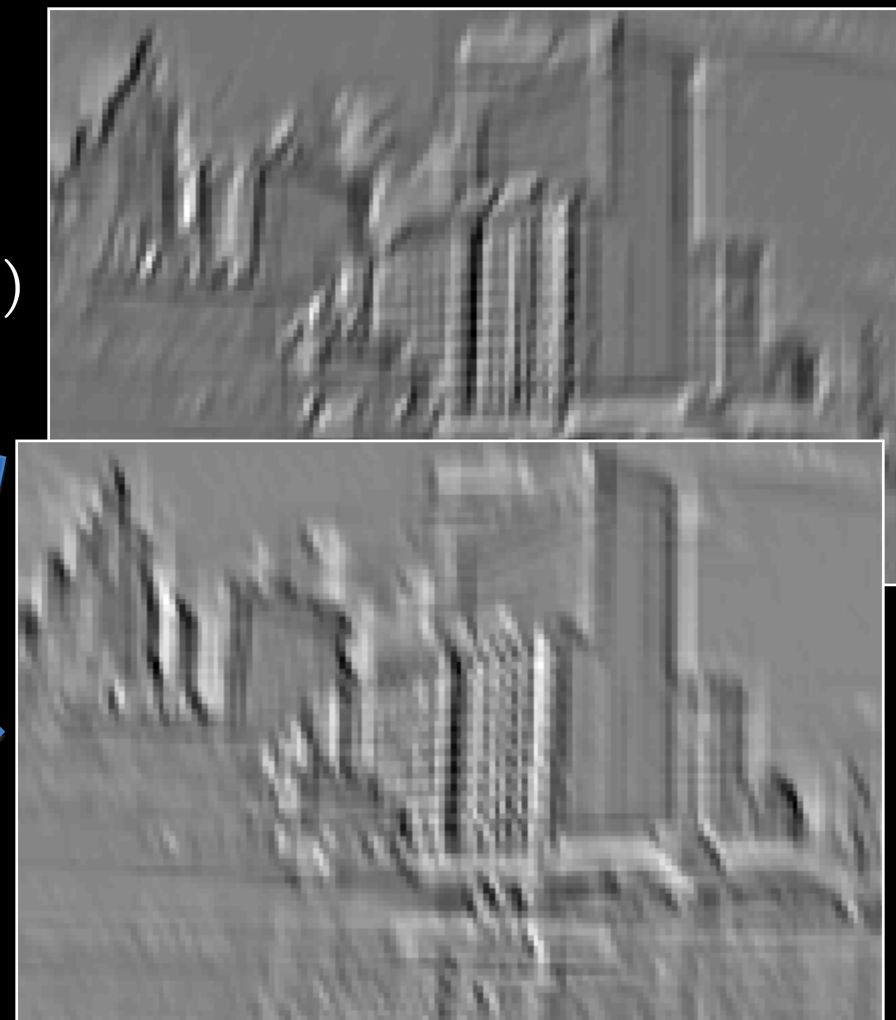
Output Features

Slide: R. Fergus

Filtering

- Convolutional
  - Dependencies are local
  - Translation invariance
  - Tied filter weights (few params)

Input

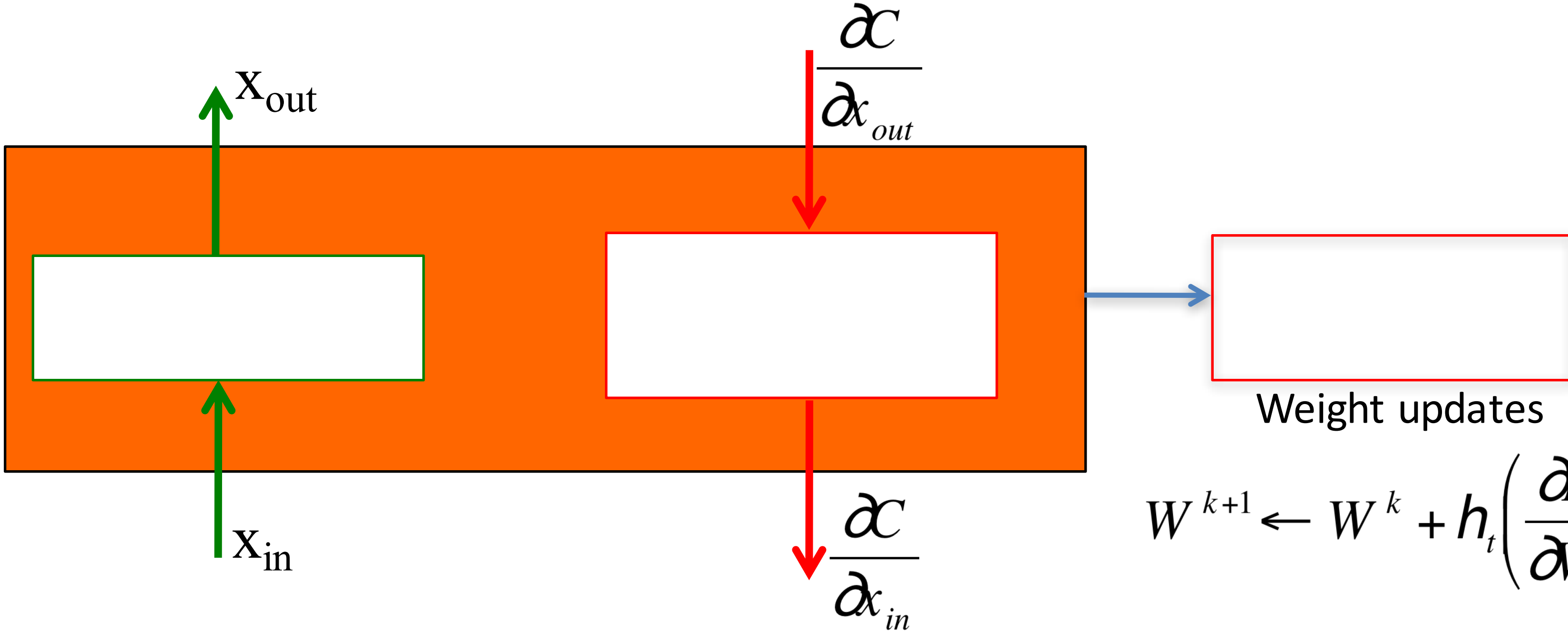


Feature Map

Slide: R. Fergus

Pset: Convolution Module

Assume the input x_{in} and output x_{out} are 1D signals of the same length N.
The convolution kernel is w_i, and has length $M < N$



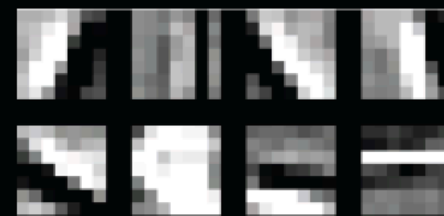
$$W^{k+1} \leftarrow W^{k} + h_t \left(\frac{\partial E}{\partial W} \right)^T$$

Derive the equations that go inside each box.
Discuss how you handle the boundaries.

Components of Each Layer

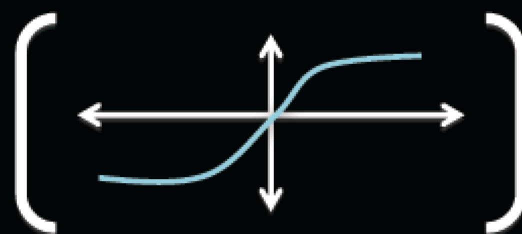
Pixels / Features →

Filter with learned dictionary



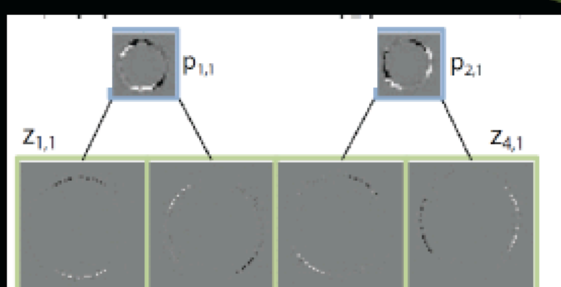
↓

Non-linearity



↓

Spatial local max pooling

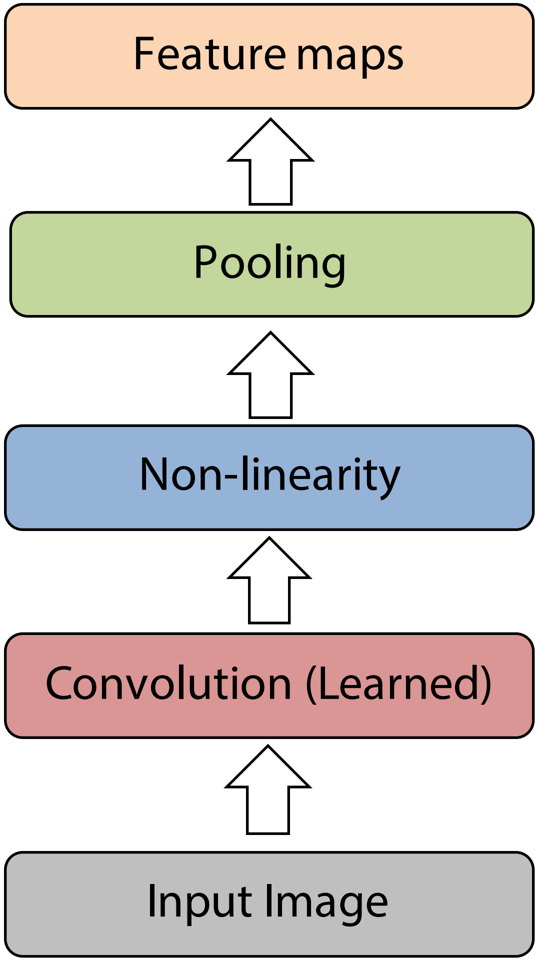


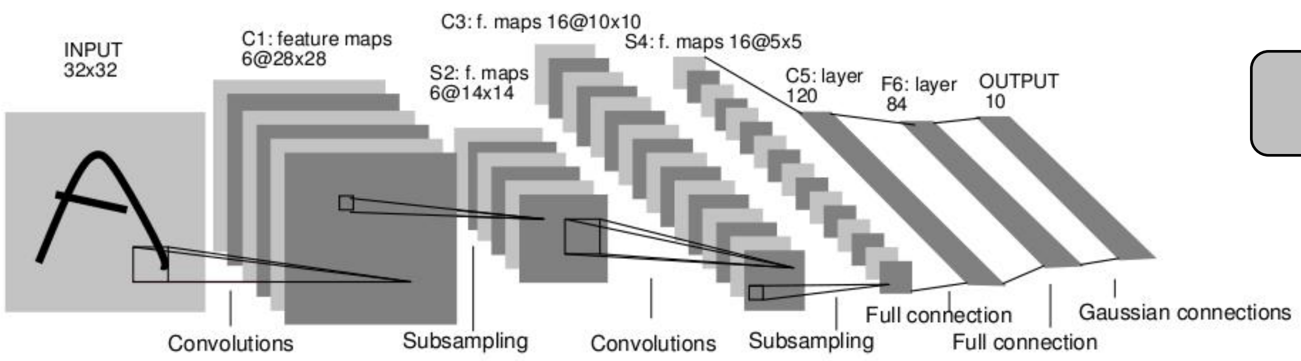
→ [Optional] Normalization across data/features → Output Features

Slide: R. Fergus

Overview of Convnets

- Feed-forward:
  - Convolve input
  - Non-linearity (rectified linear)
  - Pooling (local max)
- Supervised
- Train convolutional filters by back-propagating classification error

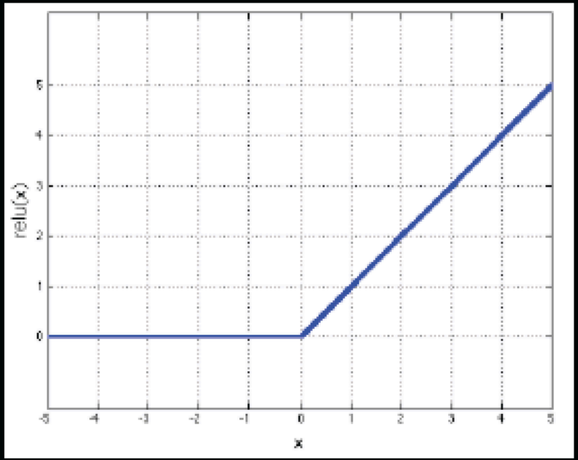




LeCun et al. 1998

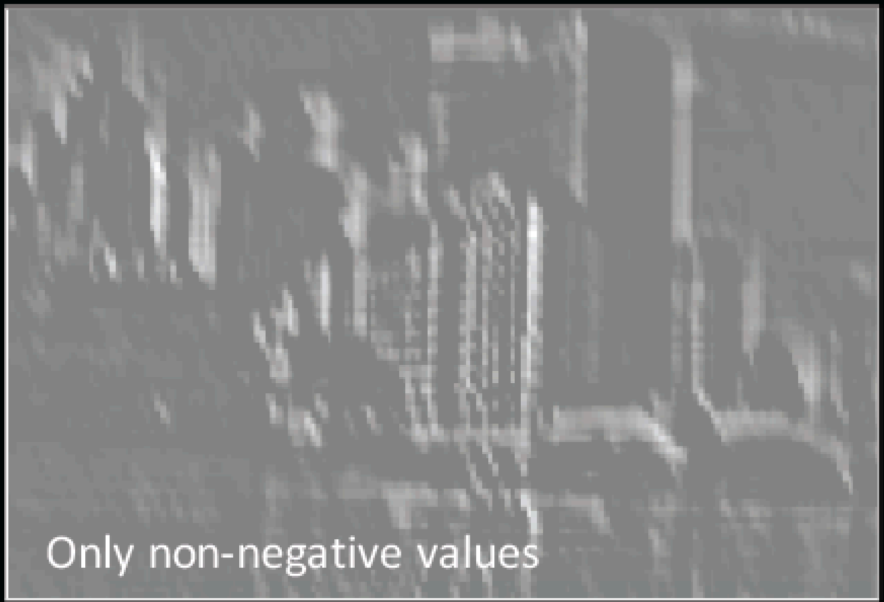
Non-Linearity

- Rectified linear function
  - Applied per-pixel
  - output = max(0,input)



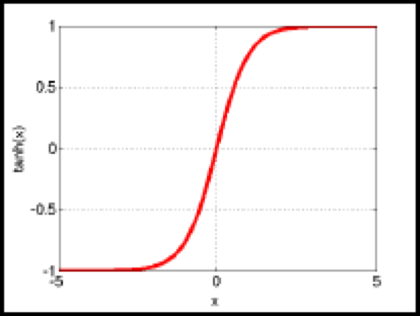
Input feature map

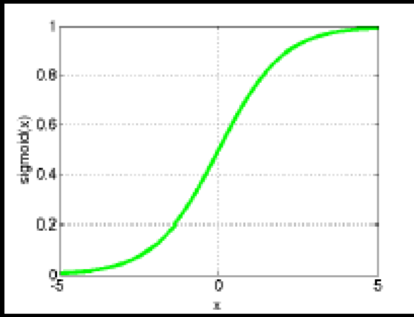
Output feature map



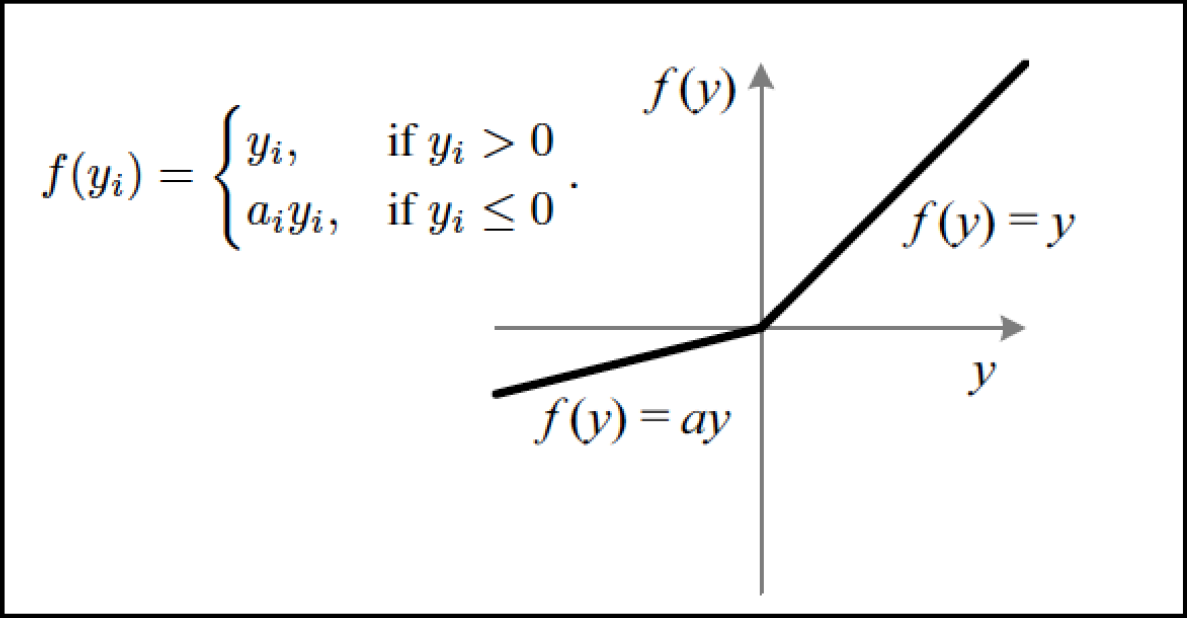
Non-Linearity

- Other choices:
  - Tanh
  - Sigmoid: 1/(1+exp(-x))
  - PReLU





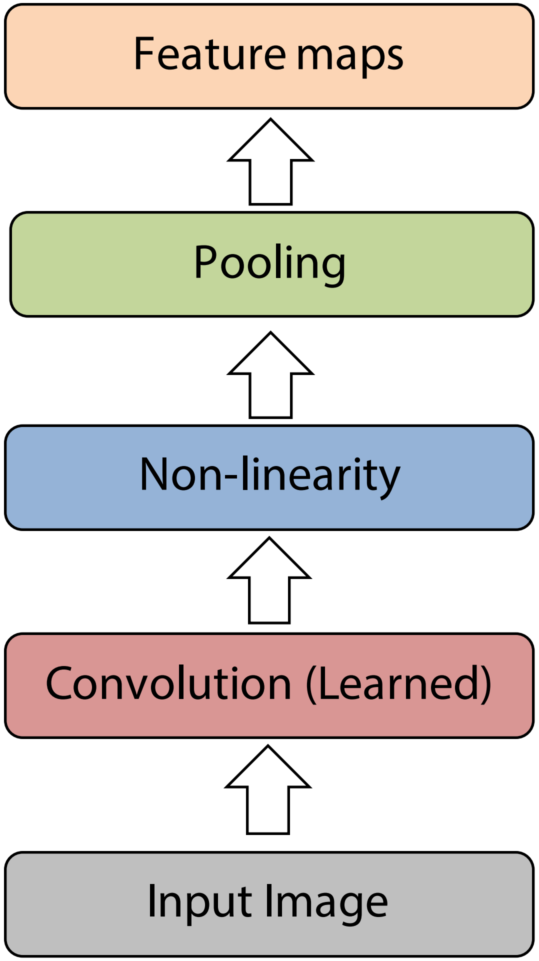
[Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, Kaiming He et al. arXiv:1502.01852v1.pdf, Feb 2015]

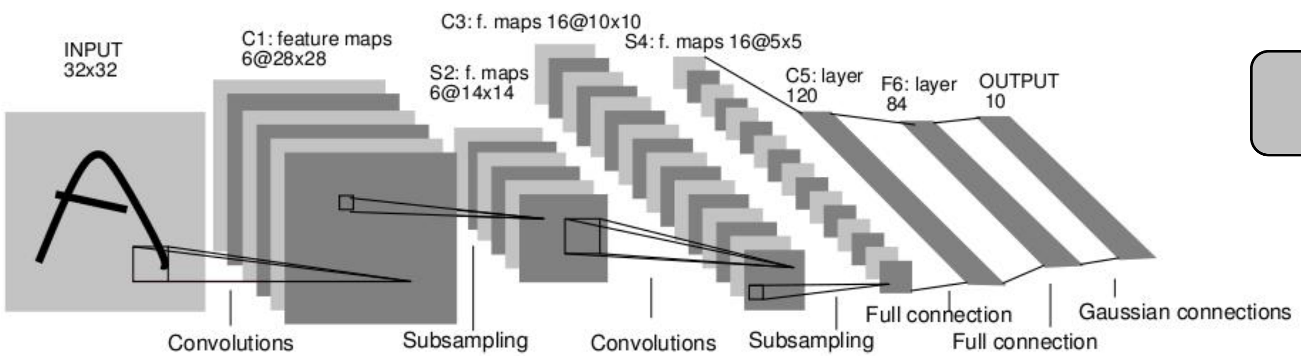


Slide: R. Fergus

Overview of Convnets

- Feed-forward:
  - Convolve input
  - Non-linearity (rectified linear)
  - Pooling (local max)
- Supervised
- Train convolutional filters by back-propagating classification error



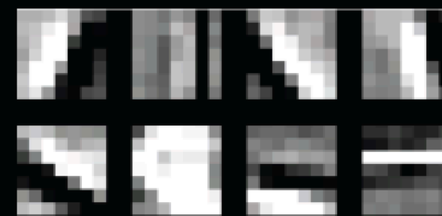


LeCun et al. 1998

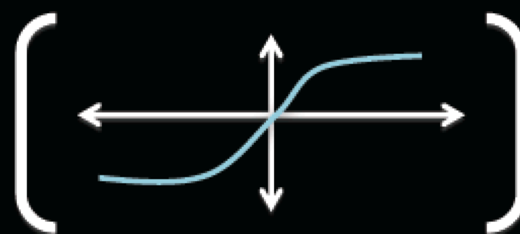
Components of Each Layer

Pixels / Features

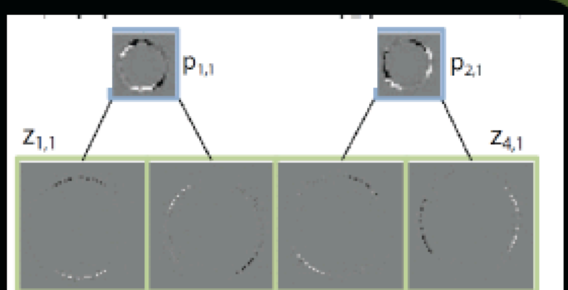
Filter with learned dictionary



Non-linearity



Spatial local max pooling



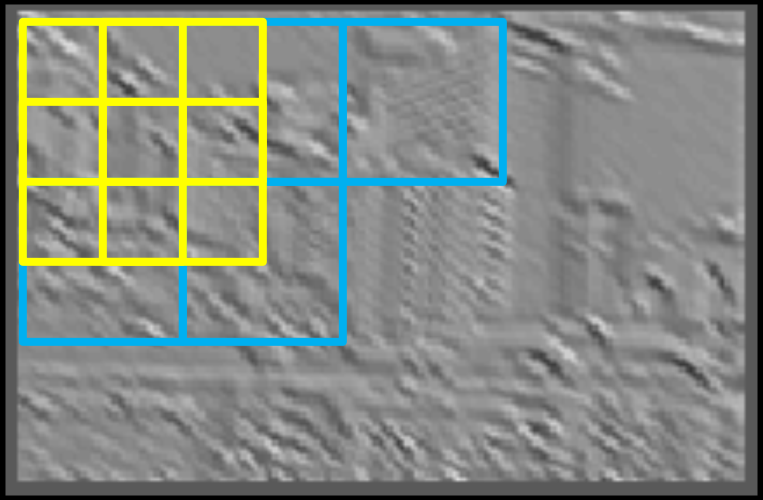
[Optional] Normalization across data/features

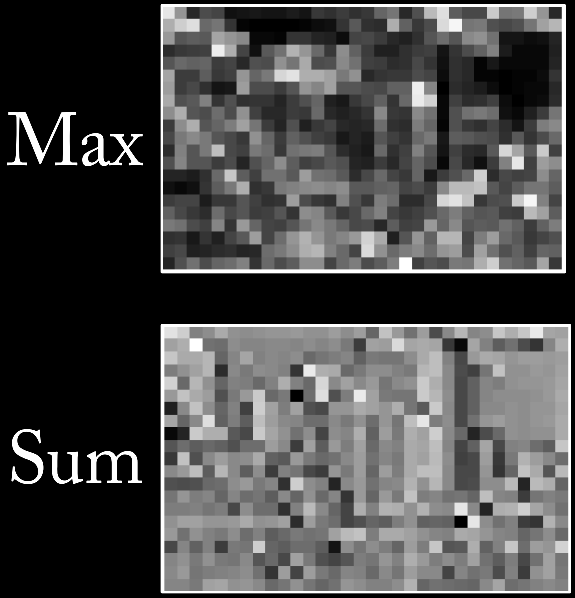
Output Features

Slide: R. Fergus

Pooling

- Spatial Pooling
  - Non-overlapping / overlapping regions
  - Sum or max
  - Boureau et al. ICML'10 for theoretical analysis

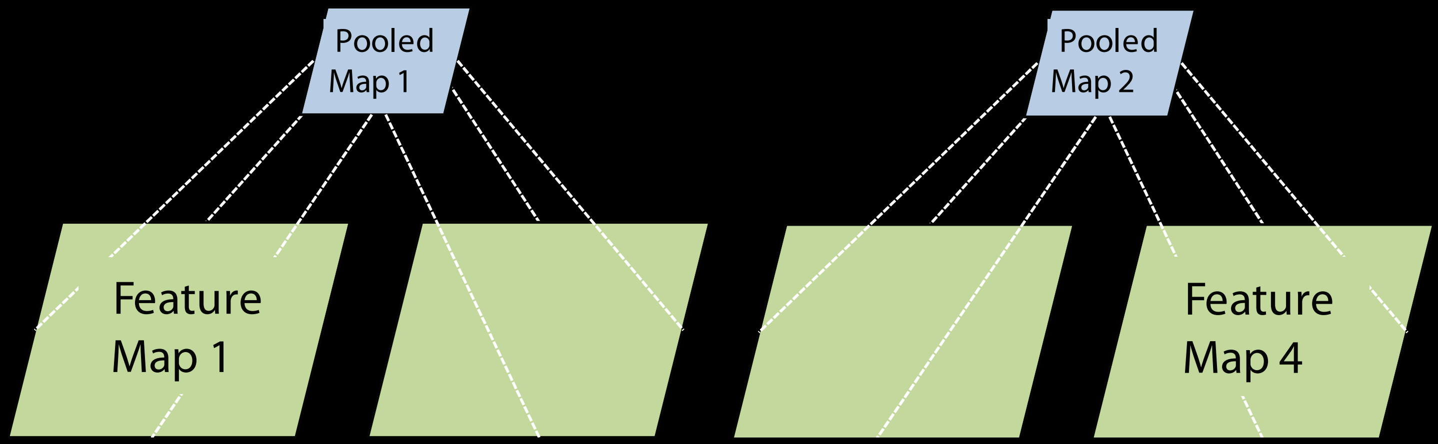




Slide: R. Fergus

Pooling

- Pooling across feature groups
  - Additional form of inter-feature competition
  - MaxOut Networks [Goodfellow et al. ICML 2013]

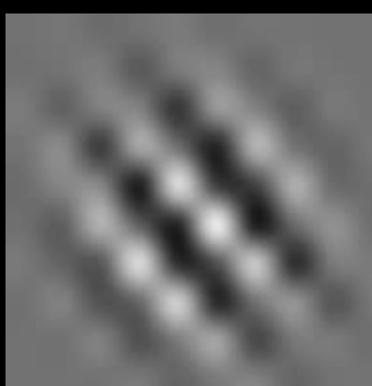


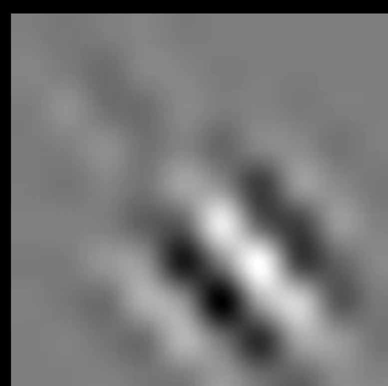
Slide: R. Fergus

Role of Pooling

- Spatial pooling
  - Invariance to small transformations
  - Larger receptive fields (see more of input)

Visualization technique from [Le et al. NIPS'10]:





Zeiler, Fergus [arXiv 2013]

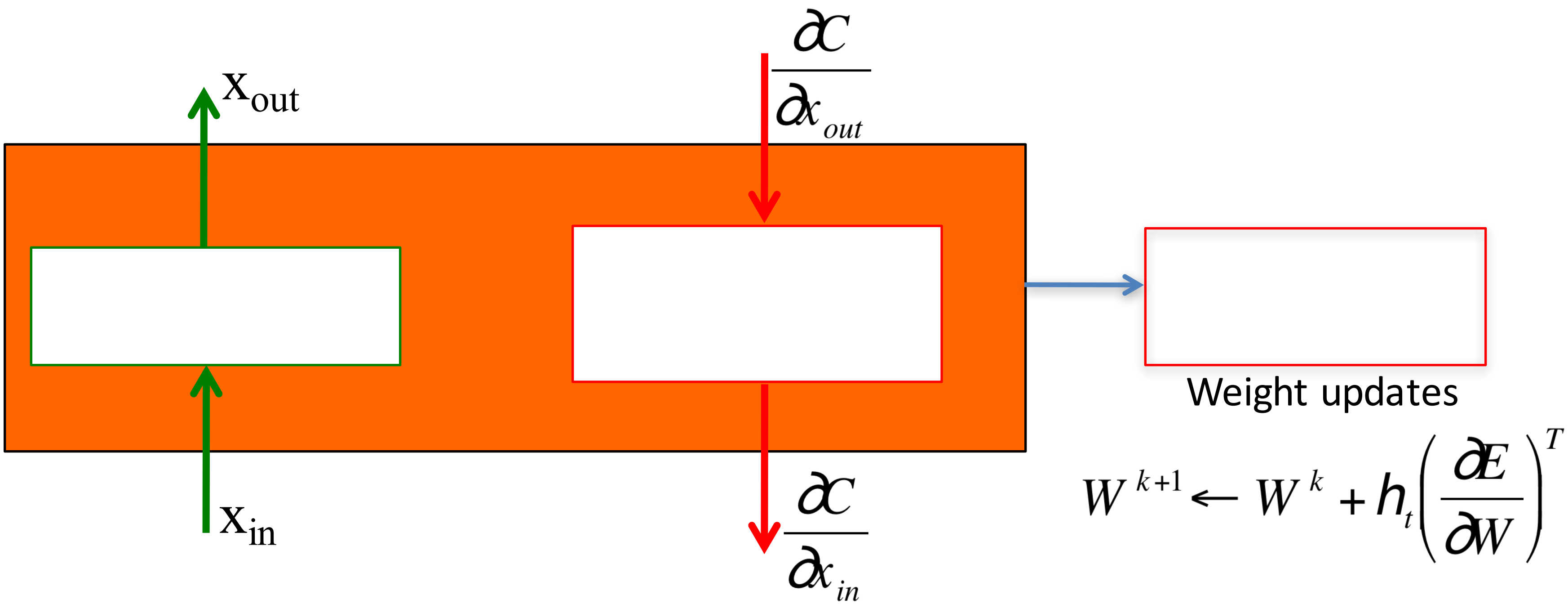
Videos from: http://ai.stanford.edu/~quocle/TCNNweb

Slide: R. Fergus

Pset: max pooling Module

(grad course, optional for undergrads)

Assume the input x_{in} and output x_{out} are 1D signals of different lengths.

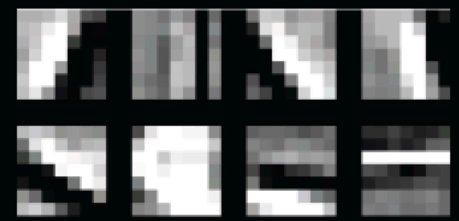


Derive the equations that go inside each box.
Discuss how you handle the boundaries.

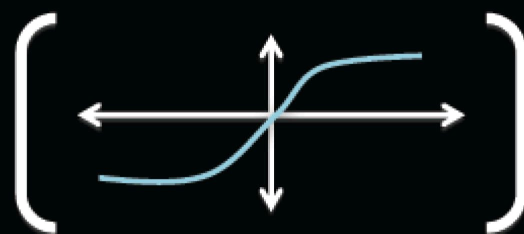
Components of Each Layer

Pixels / Features

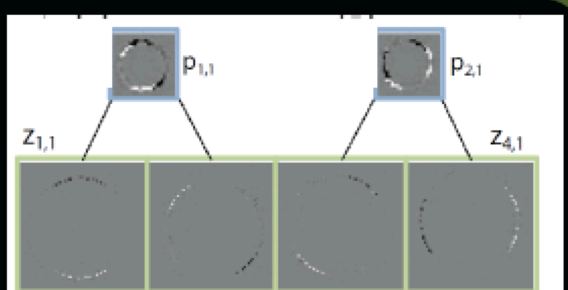
Filter with learned dictionary



Non-linearity



Spatial local max pooling



[Optional] Normalization across data/features

Output Features

Slide: R. Fergus

Normalization

- Contrast normalization
  - See Divisive Normalization in Neuroscience

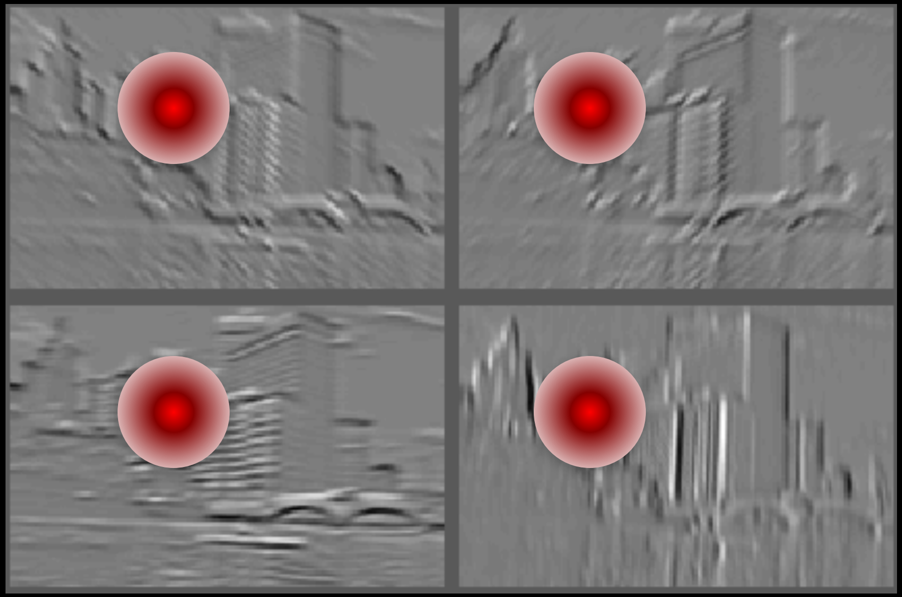
Input

Filters

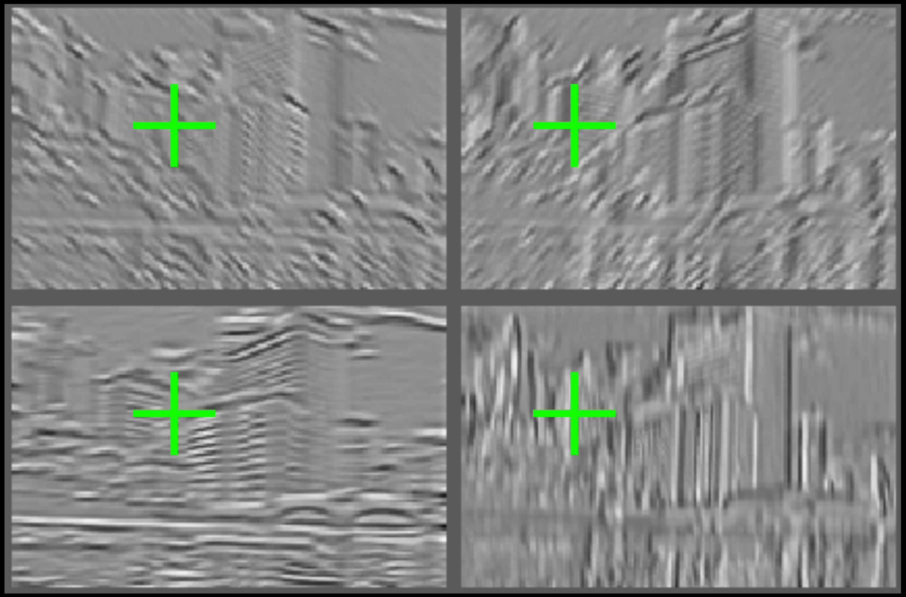
Slide: R. Fergus

Normalization

- Contrast normalization (across feature maps)
  - Local mean = 0, local std. = 1, "Local" → 7x7 Gaussian
  - Equalizes the features maps



Feature Maps



Feature Maps
After Contrast Normalization

Slide: R. Fergus

Role of Normalization

- Introduces local competition between features
  - "Explaining away" in graphical models
  - Just like top-down models
  - But more local mechanism
- Also helps to scale activations at each layer better for learning
  - Makes energy surface more isotropic
  - So each gradient step makes more progress
- Empirically, seems to help a bit (1-2%) on ImageNet
- Recent models do not use normalization

Normalization across Data

- Batch Normalization

[Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, Sergey Ioffe, Christian Szegedy, arXiv:1502.03167]

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

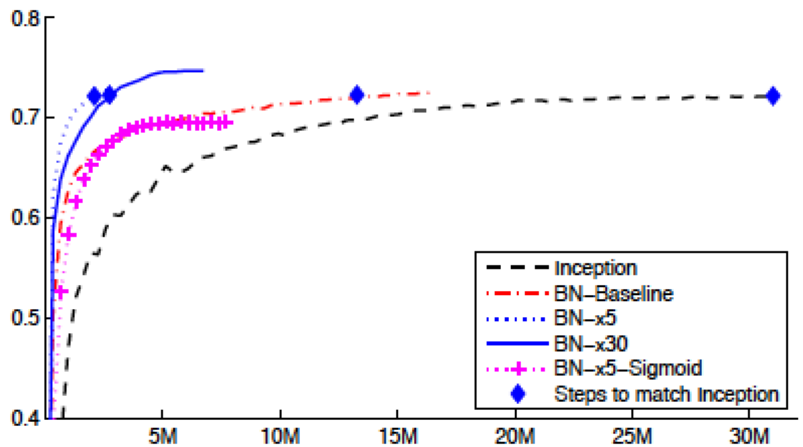
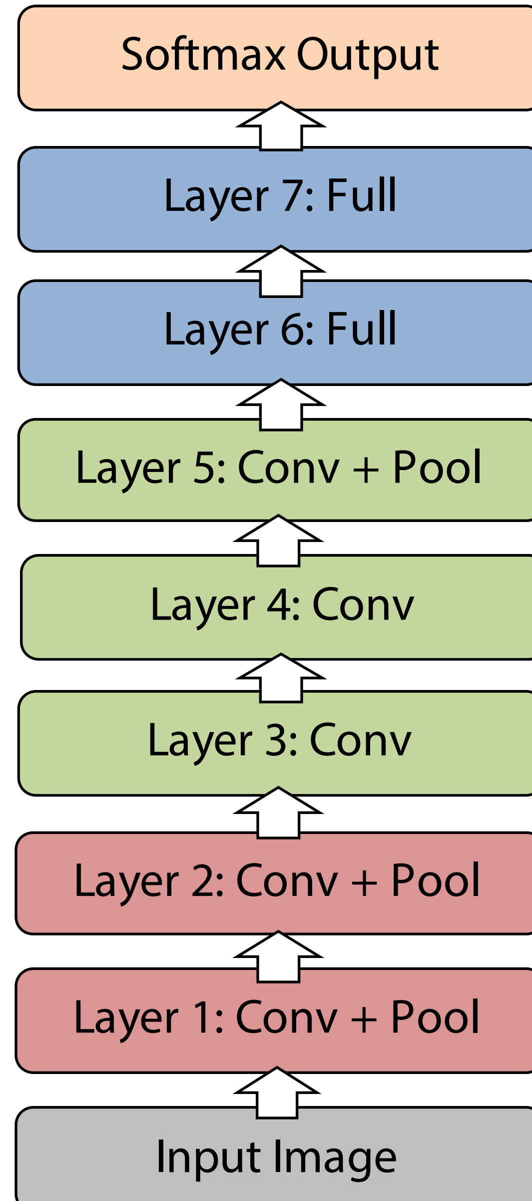


Figure 2: *Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.*

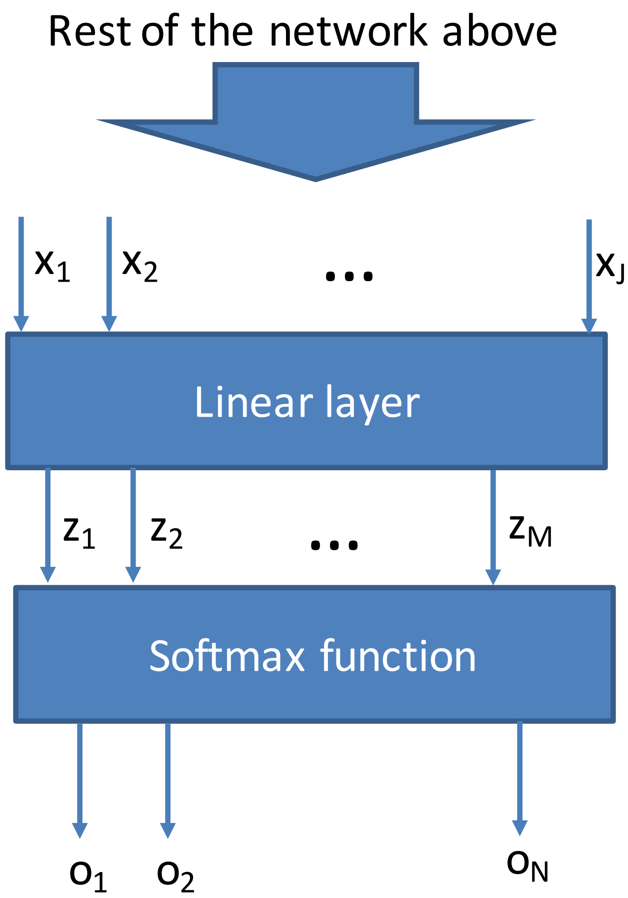
Slide: R. Fergus

Architecture of Krizhevsky et al.

- 8 layers total



Softmax



If we have N classes

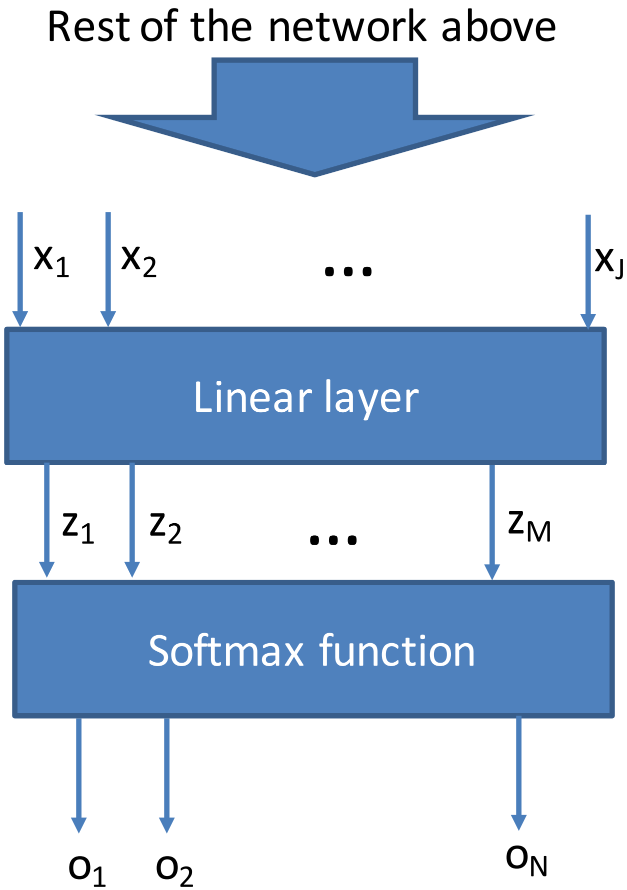
$$z_m = \sum_{j=1}^{J} w_{m,j} \, x_j$$

$$o_n = \frac{exp\,(z_m)}{\sum_{m=1}^{M} exp\,(z_m)}$$

Note that:

$$\sum_{n=1}^{N} o_n = 1$$

Cross-entropy loss



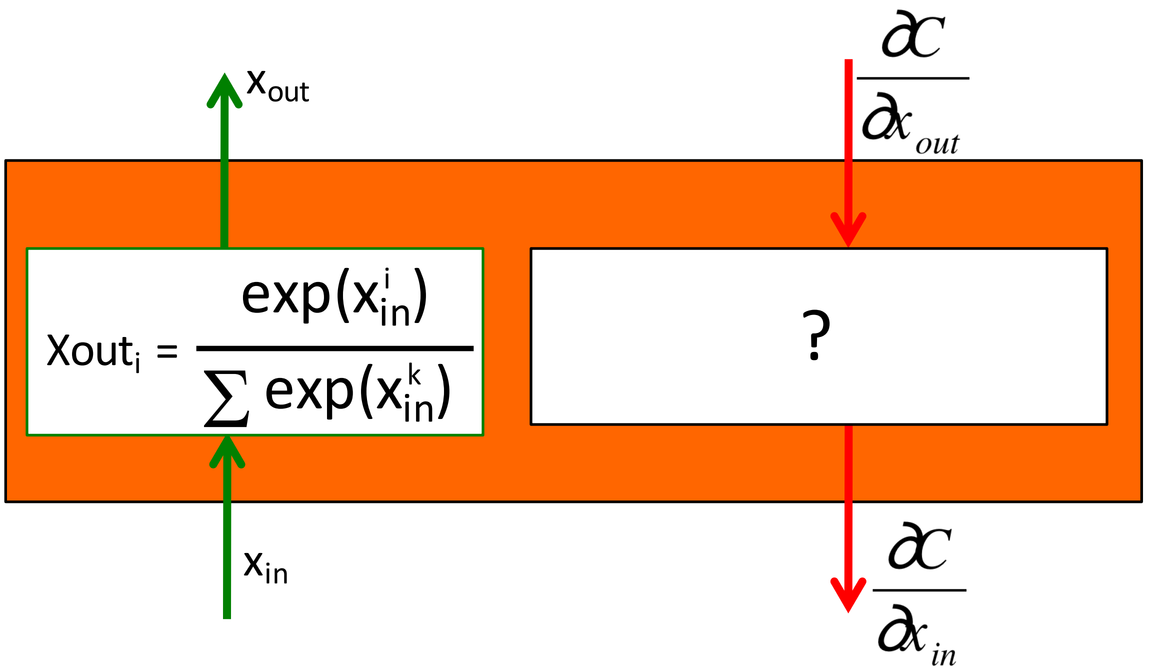
If we have N classes

Ground truth label for a training example:
$y = [y_1, y_2, y_3, \ldots y_N] = [0, 0, 1, 0, 0, \ldots 0]$

$$C = -\sum_n y_n \log(o_n)$$

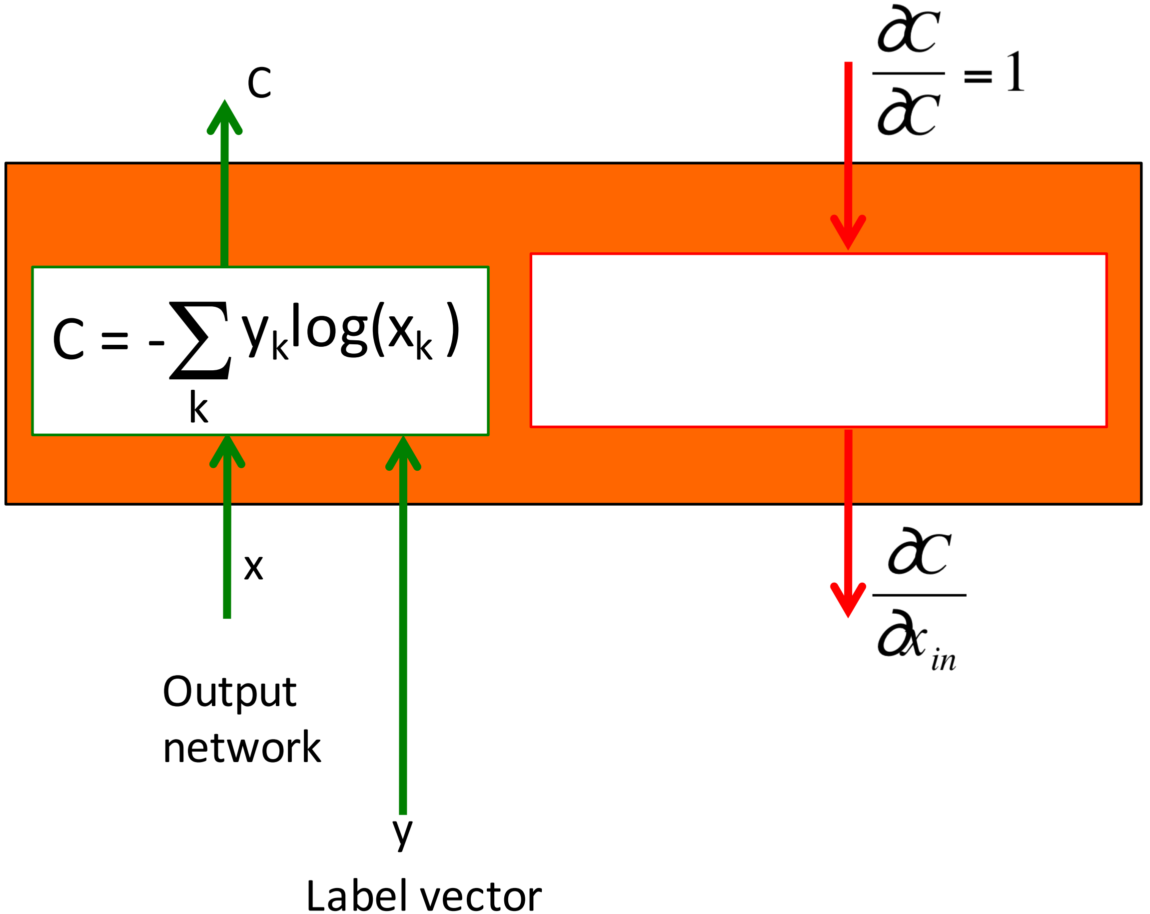
E = sum over training examples

Softmax layer



x_{out} The length of the output is the number of classes

Cross-entropy cost module



$$C = -\sum_k y_k \log(x_k)$$ Sum is over classes.

Architecture

- Big issue: how to select
  - Depth
  - Width
  - Parameter count

- Manual tuning of features has turn into manual tuning of Architectures

How we choose the architecture?

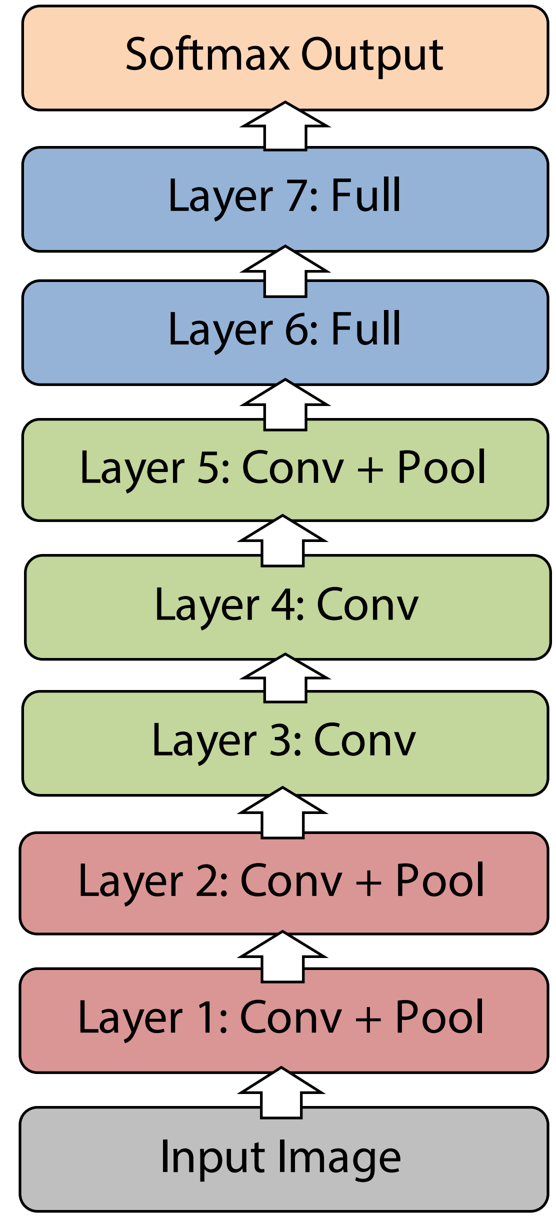
- Many hyper-parameters:
- – # layers, # feature maps
- Cross-validation
- Grid search (need lots of GPUs)
- Smarter strategies:
  – Random [Bergstra & Bengio JMLR 2012]
  – Gaussian processes [Hinton]

How important is Depth

- "Deep" in Deep Learning
- Ablation study
- Tap off features

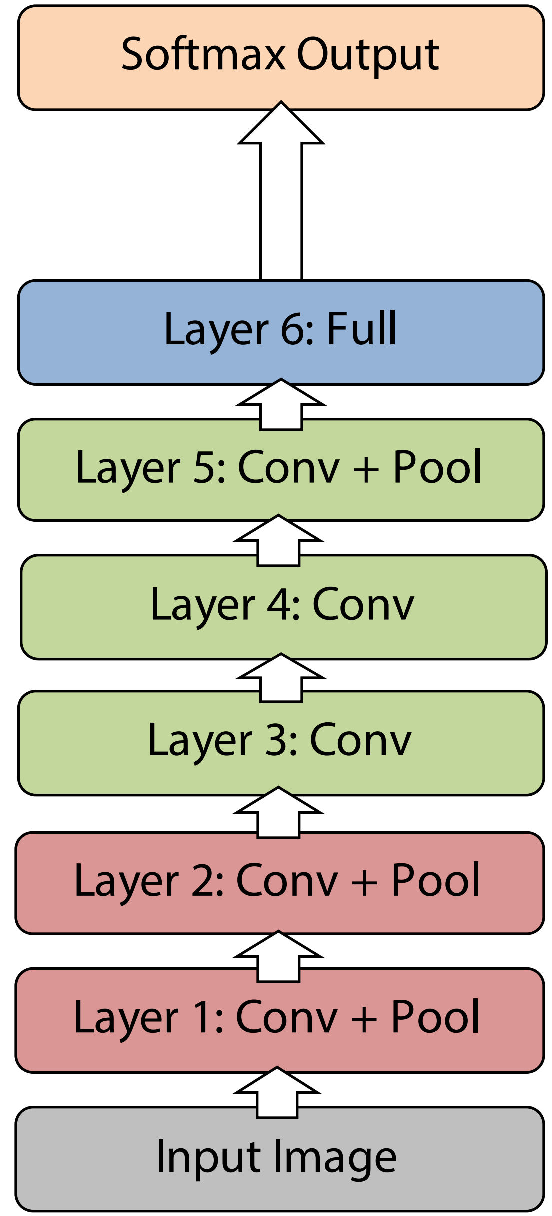
Architecture of Krizhevsky et al.

- 8 layers total
- Trained on Imagenet dataset [Deng et al. CVPR'09]
- 18.2% top-5 error
- Our reimplementation:
  18.1% top-5 error



Architecture of Krizhevsky et al.

- Remove top fully connected layer
  - Layer 7
- Drop 16 million parameters
- Only 1.1% drop in performance!

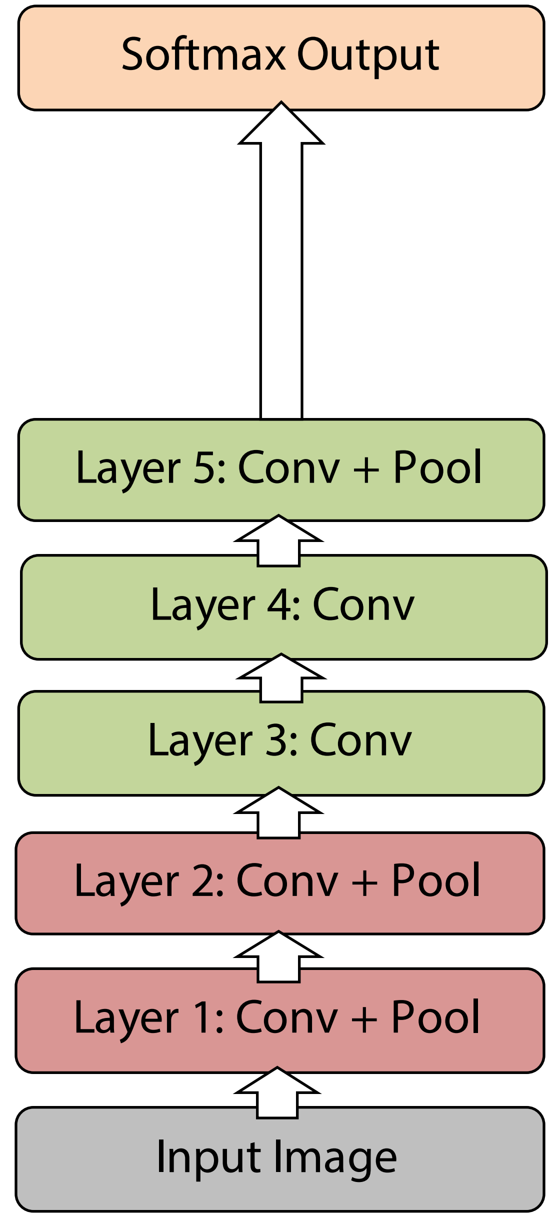


Architecture of Krizhevsky et al.

- Remove both fully connected layers
  - Layer 6 & 7

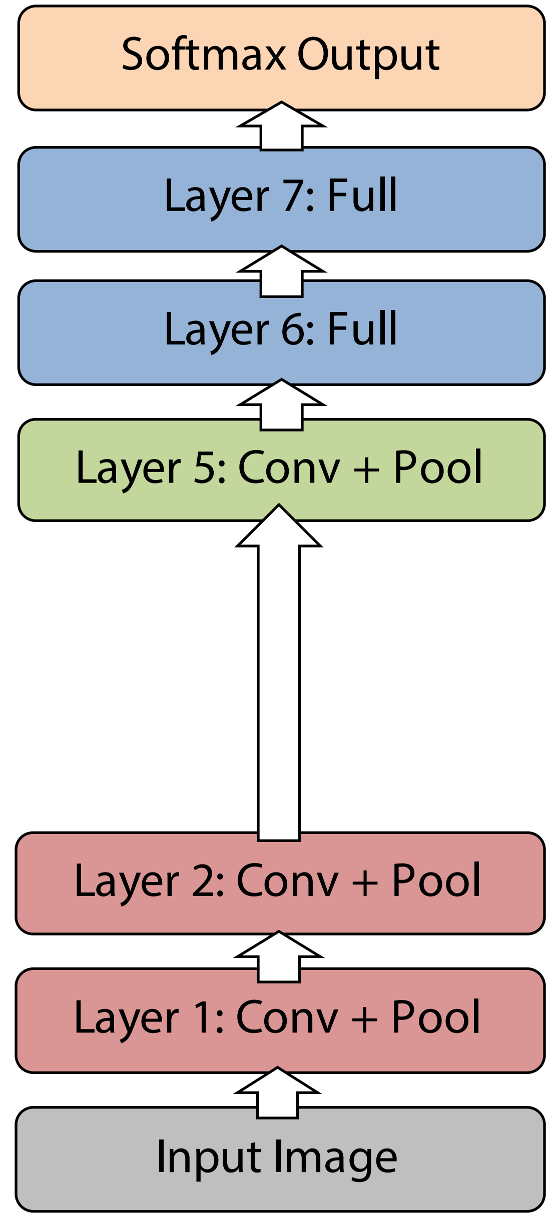
- Drop ~50 million parameters

- 5.7% drop in performance



Architecture of Krizhevsky et al.

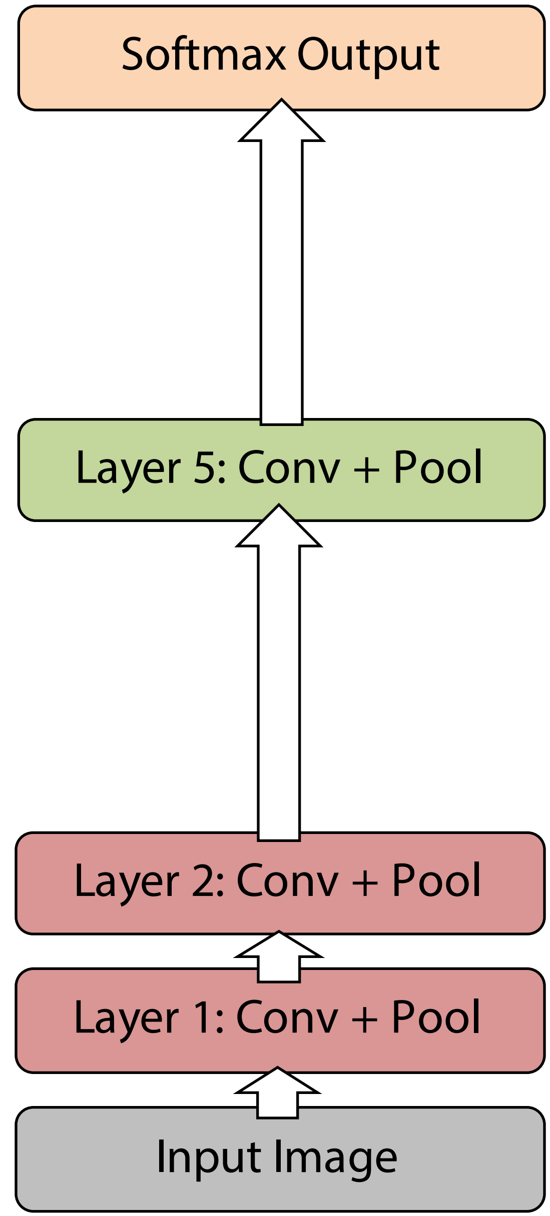
- Now try removing upper feature extractor layers:
  - Layers 3 & 4
- Drop ~1 million parameters
- 3.0% drop in performance



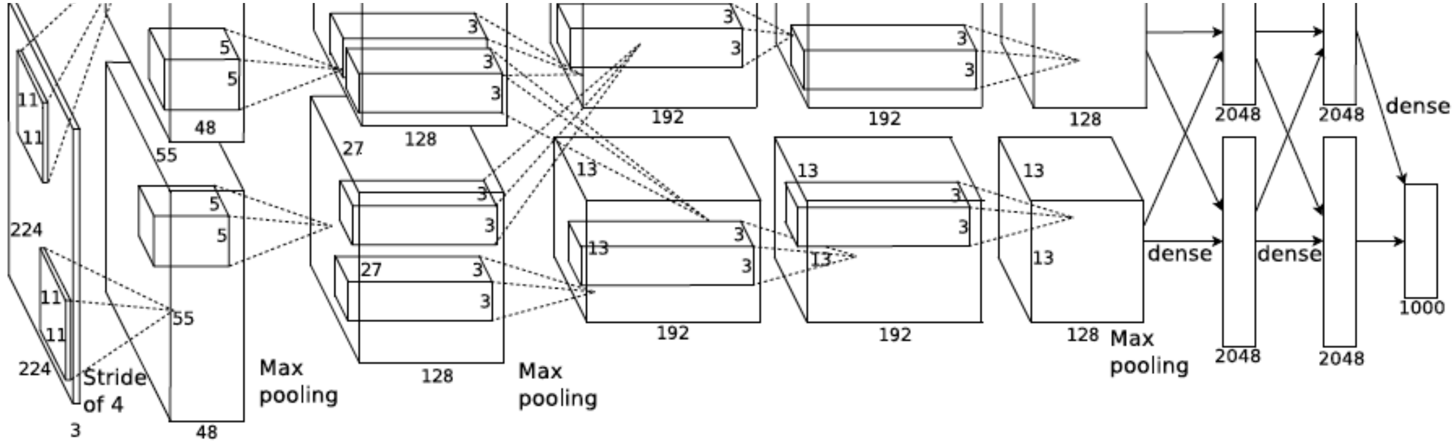
Architecture of Krizhevsky et al.

- Now try removing upper feature extractor layers & fully connected:
  - Layers 3, 4, 6 ,7
- Now only 4 layers
- 33.5% drop in performance

→Depth of network is key

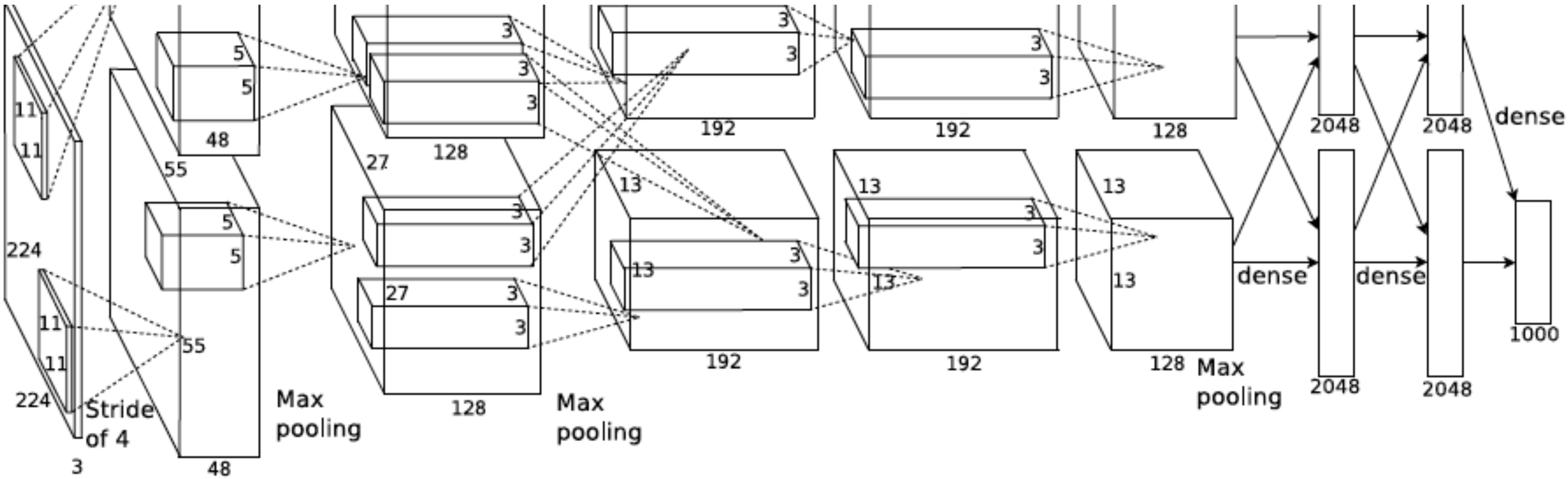


Krizhevsky et al. [NIPS2012]



AlexNet architecture:

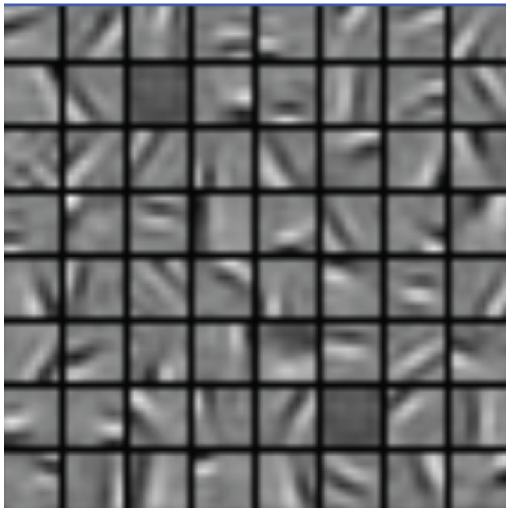
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)



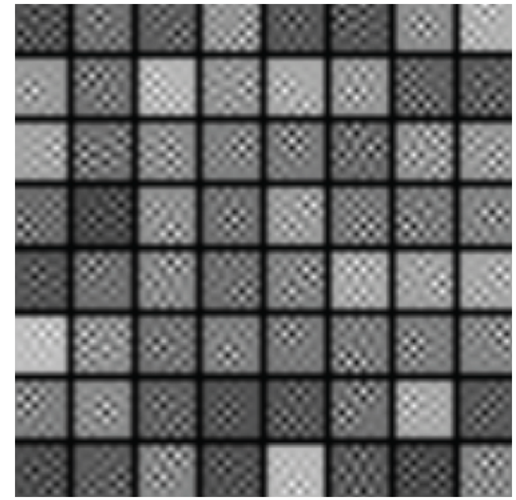
What filters are learned?

What filters are learned?

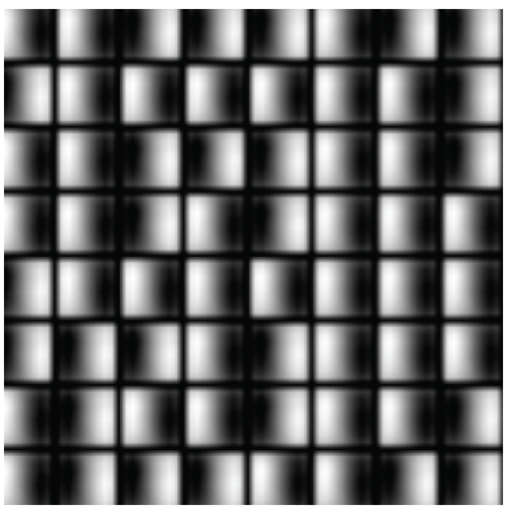
A



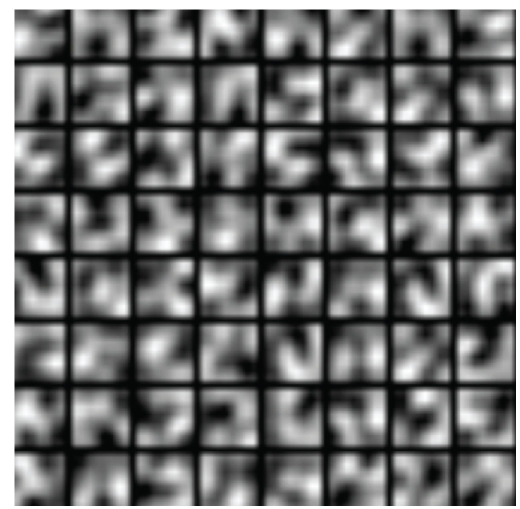
B



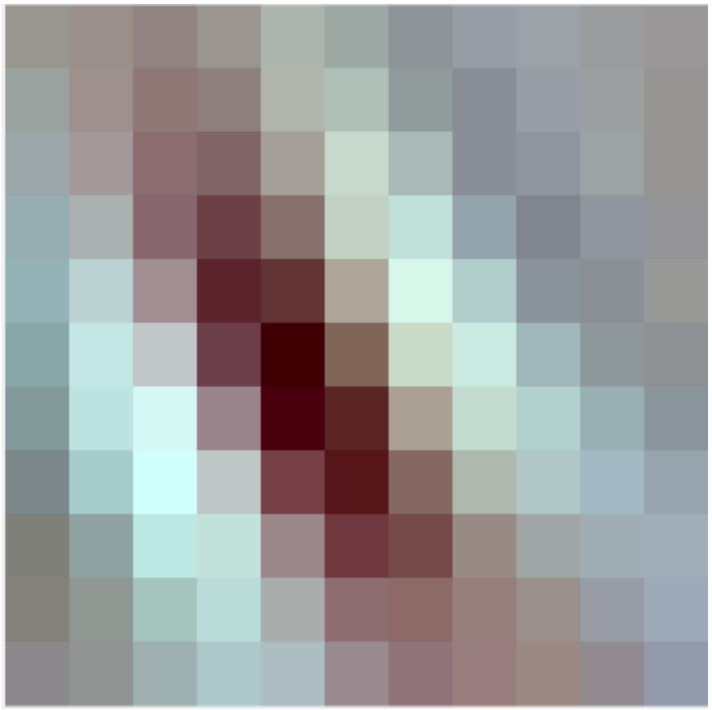
C



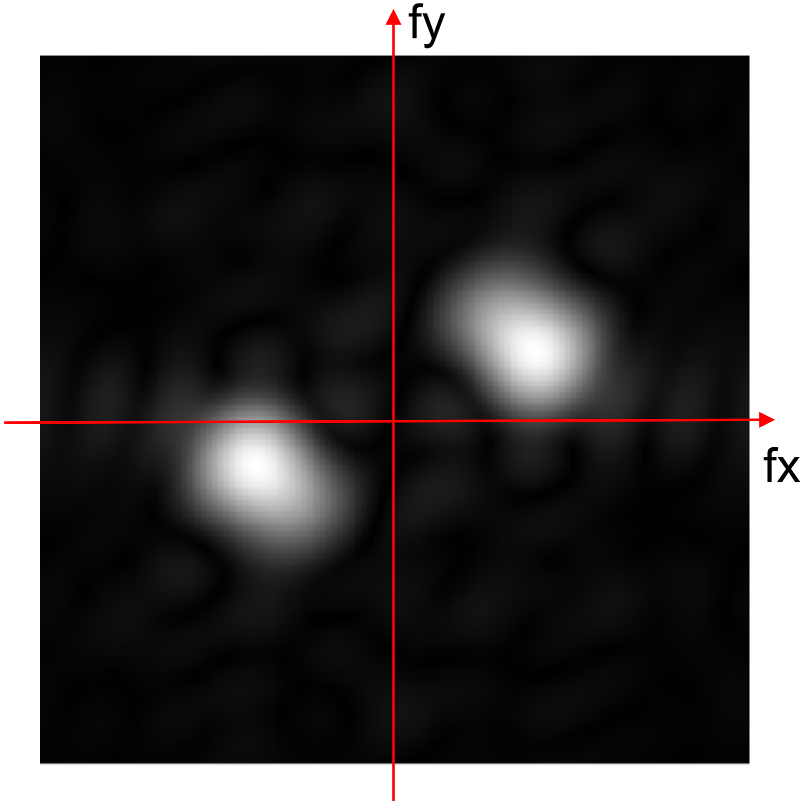
D



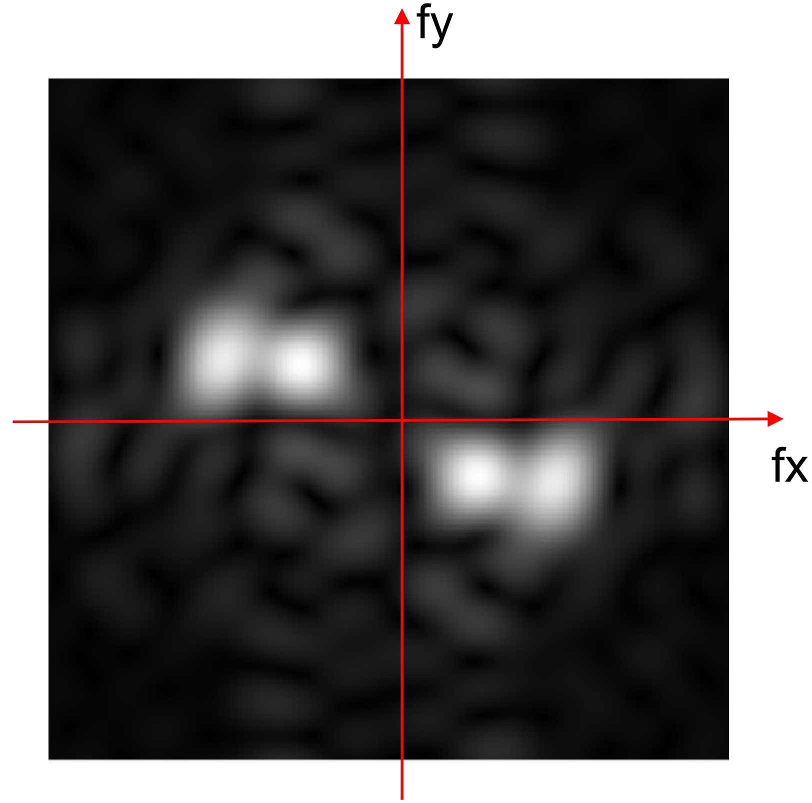
Get to know your units



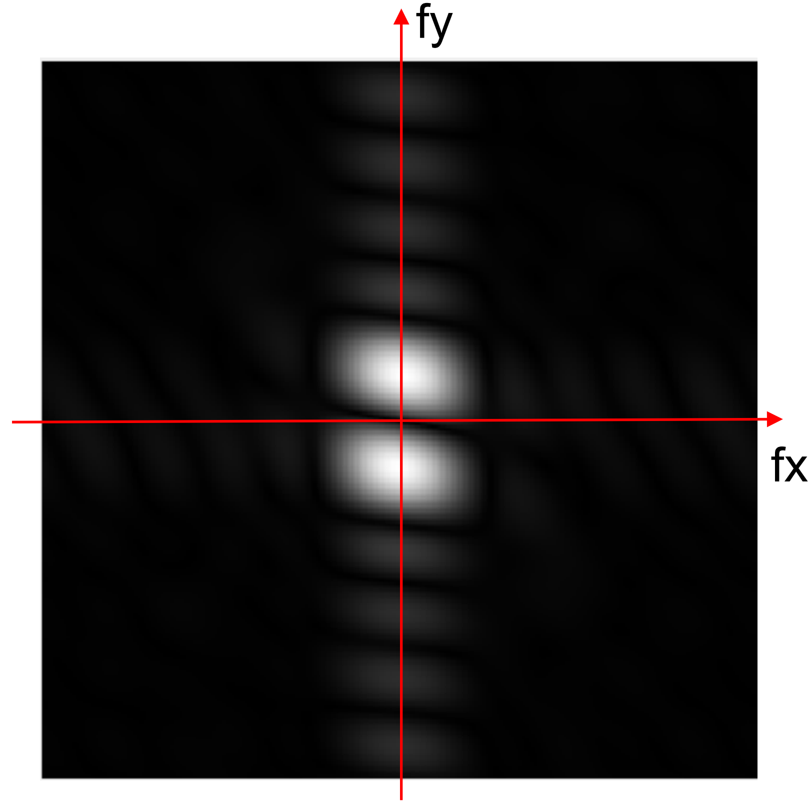
11x11 convolution kernel
(3 color channels)



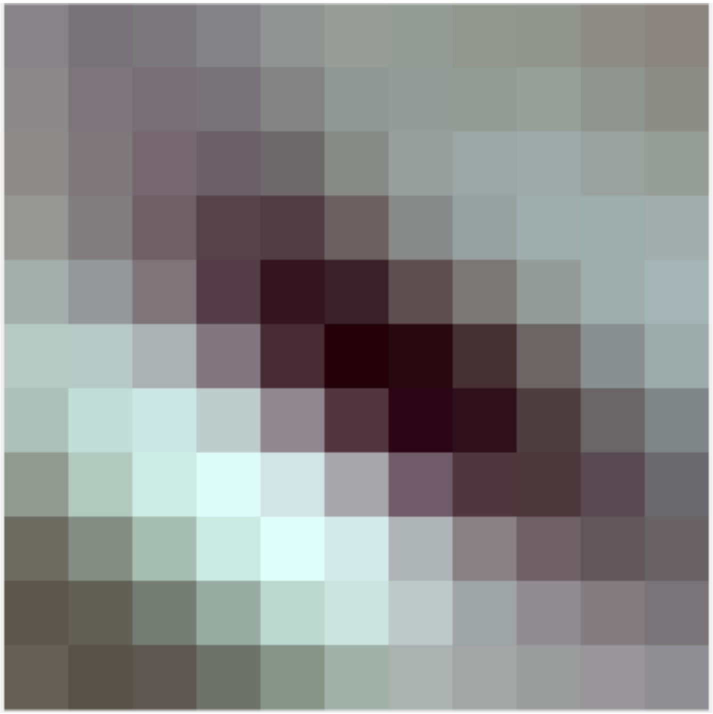
Get to know your units

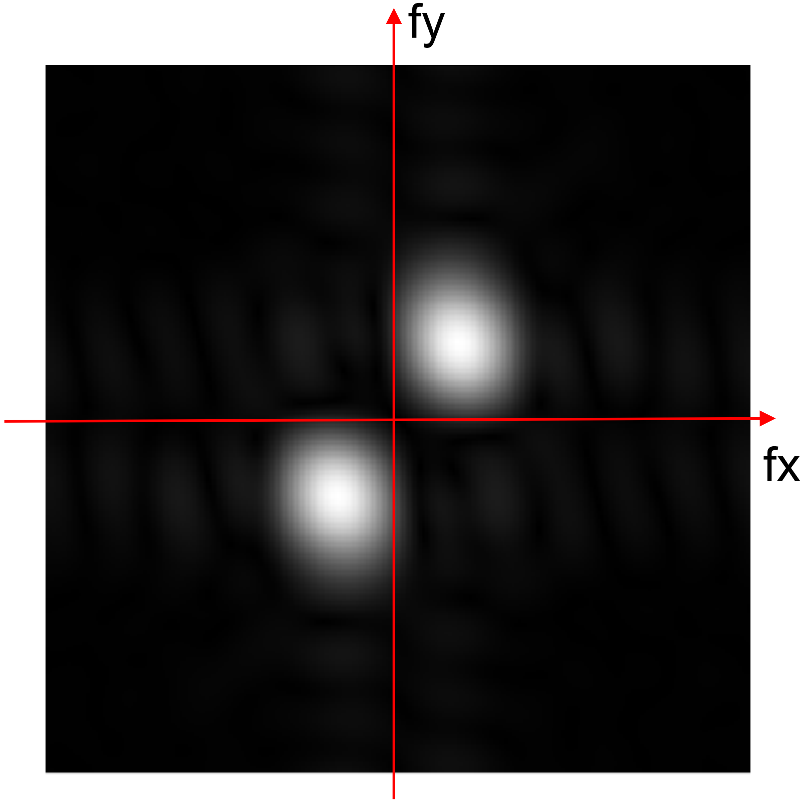


Get to know your units

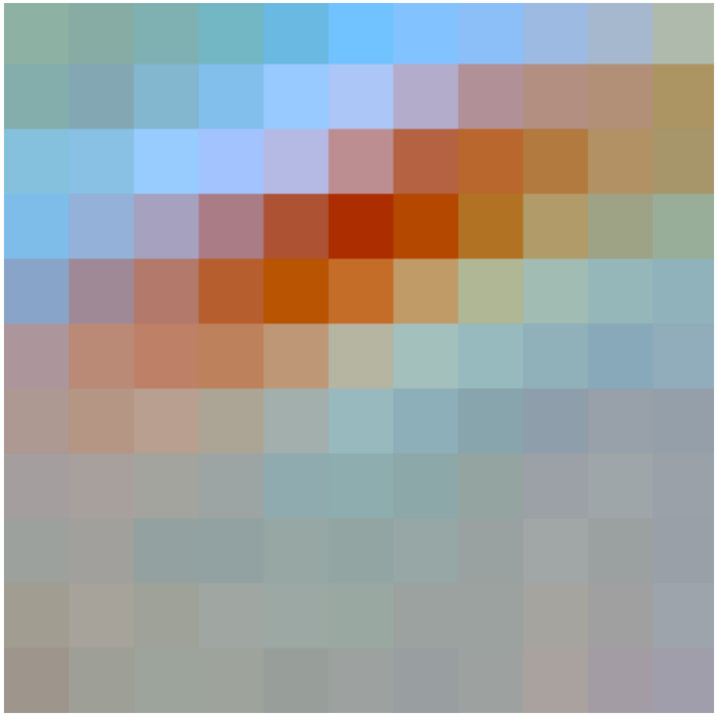


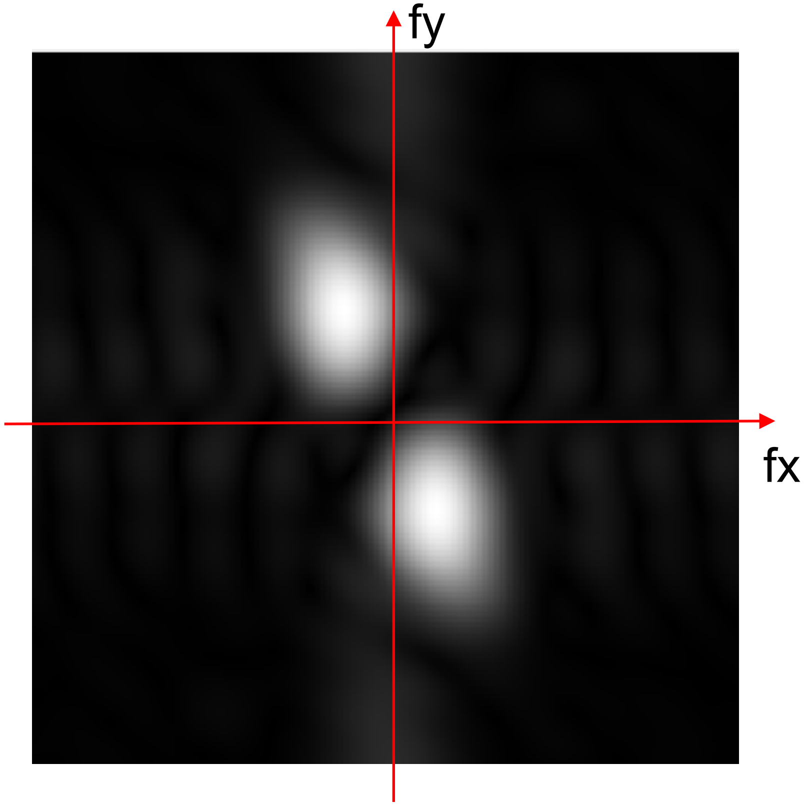
Get to know your units



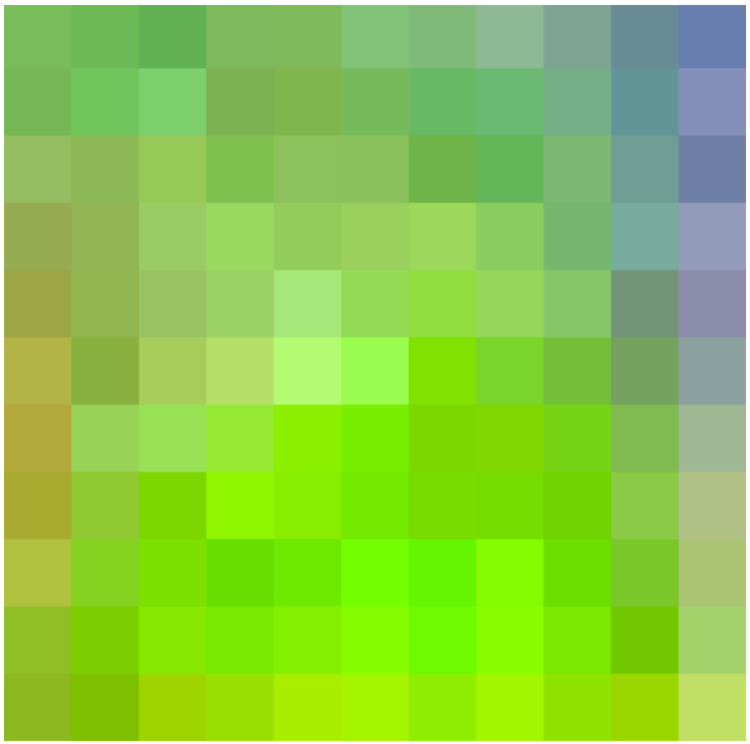


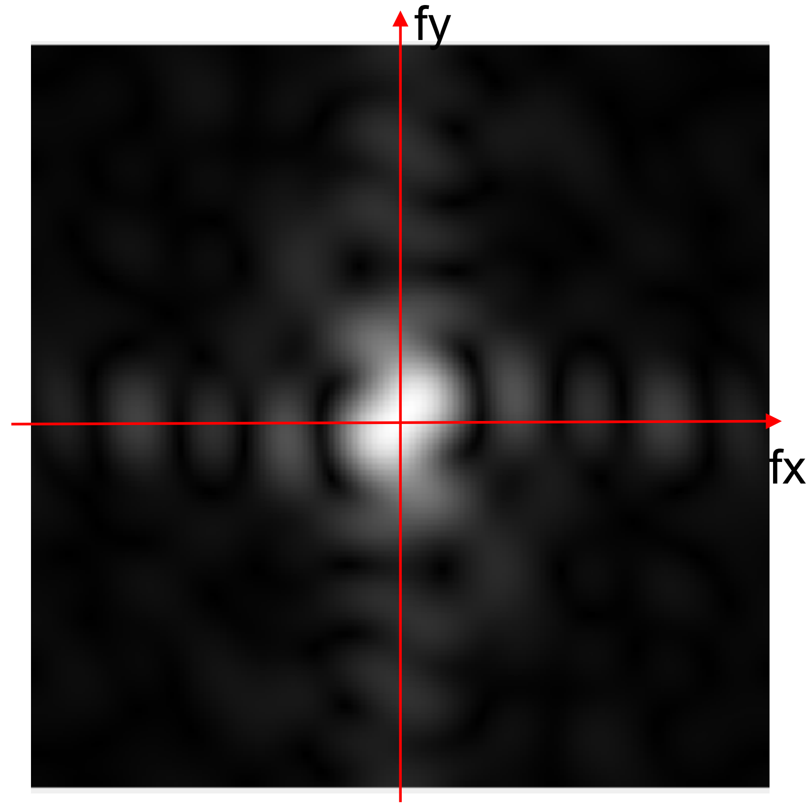
Get to know your units



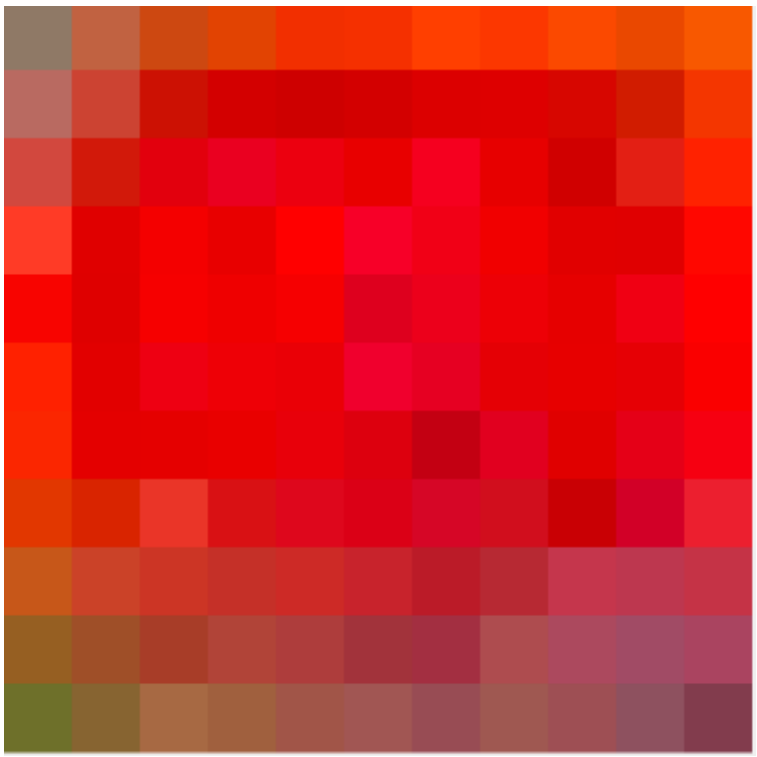


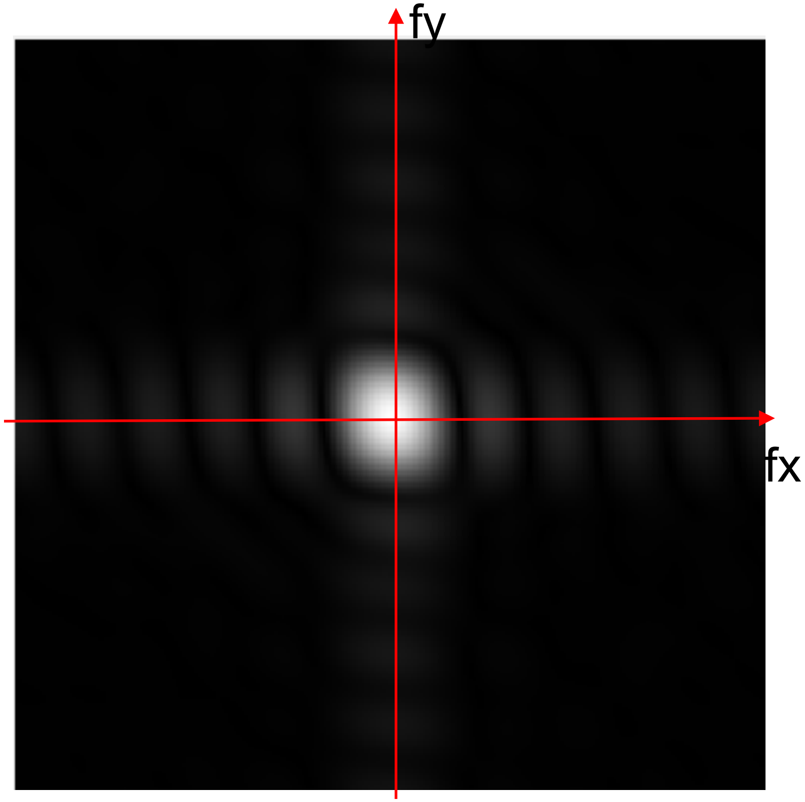
Get to know your units



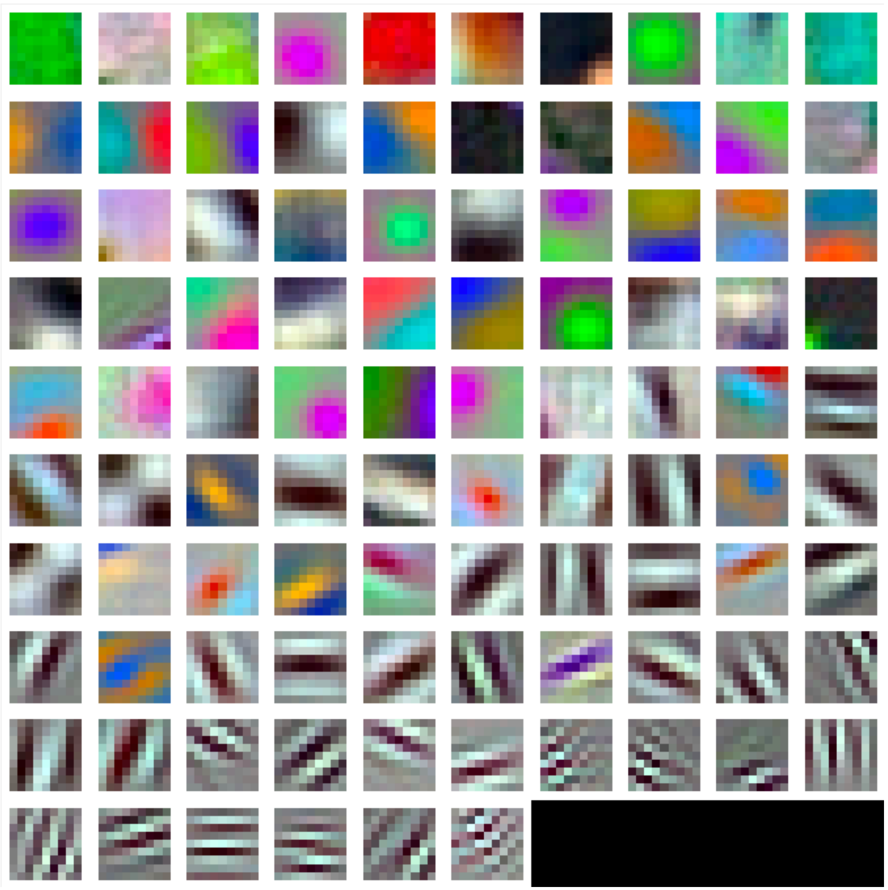


Get to know your units

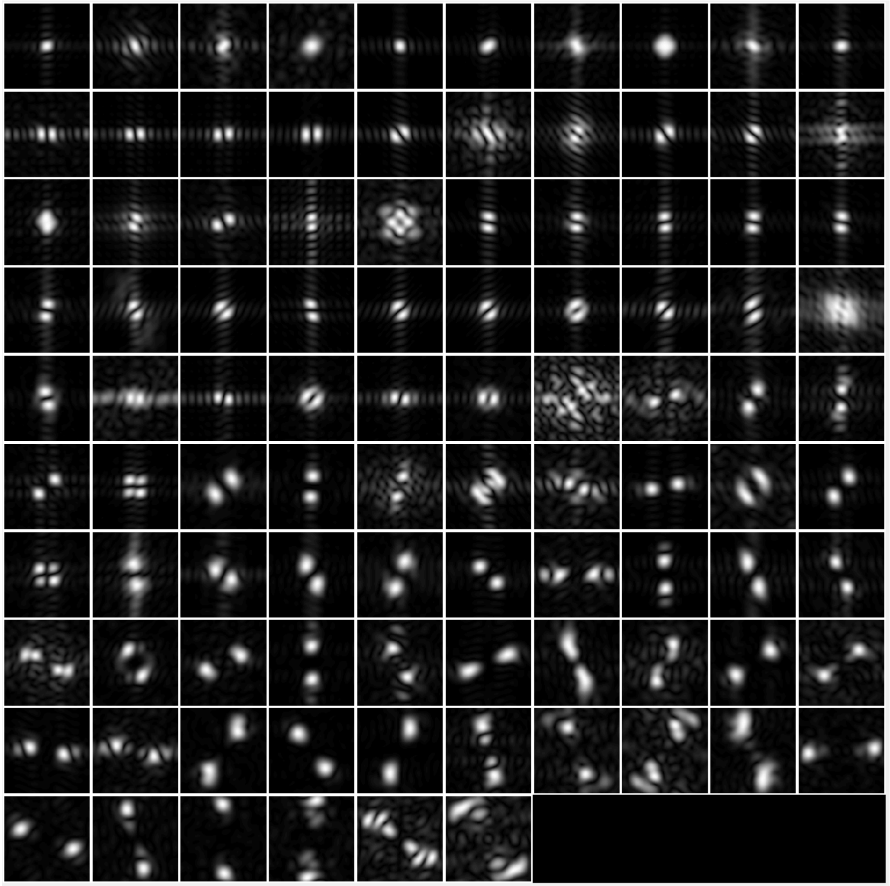




Get to know your units

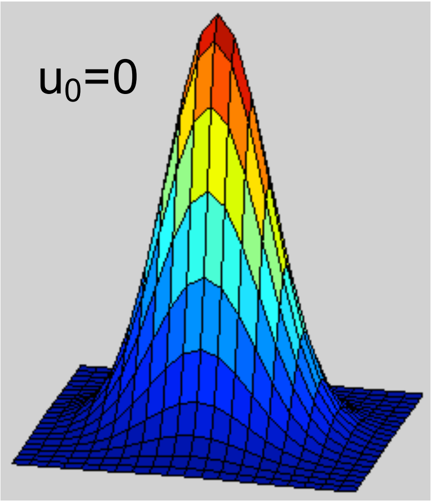


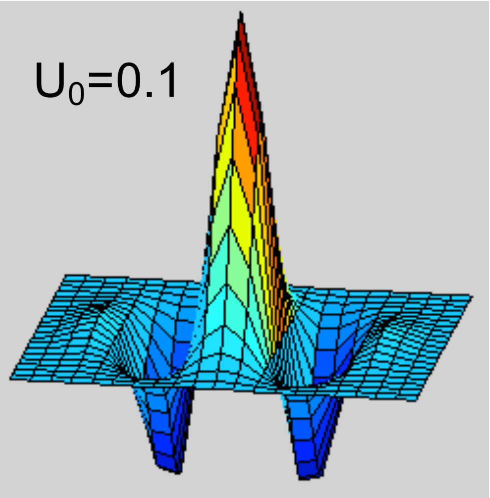
96 Units in conv1

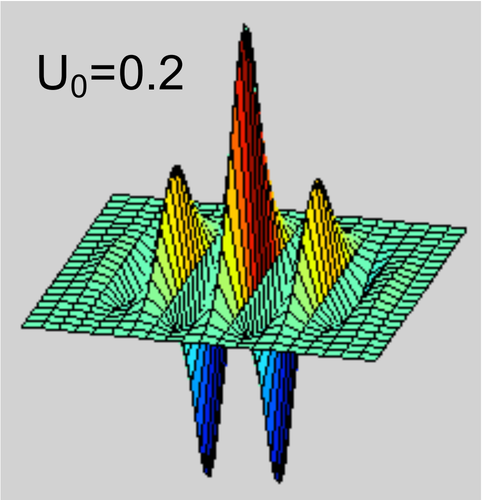


Gabor wavelets

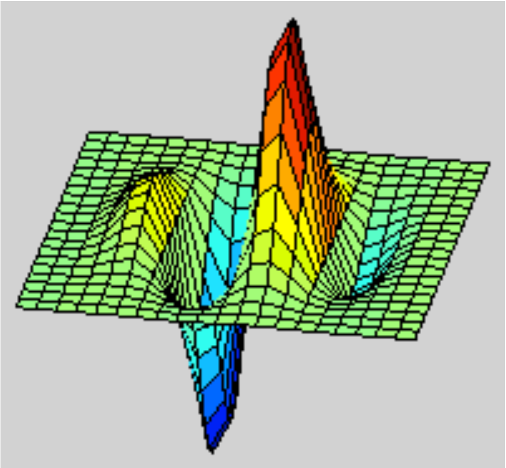
$$\psi_c(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \cos(2\pi u_0 x)$$

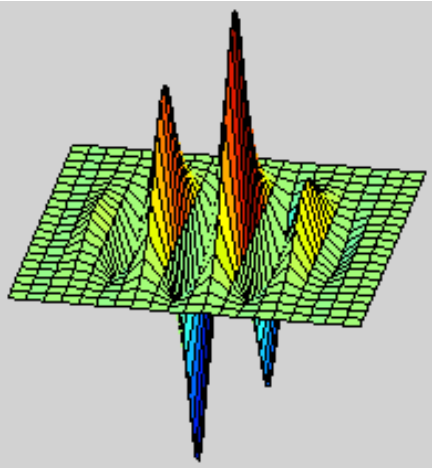




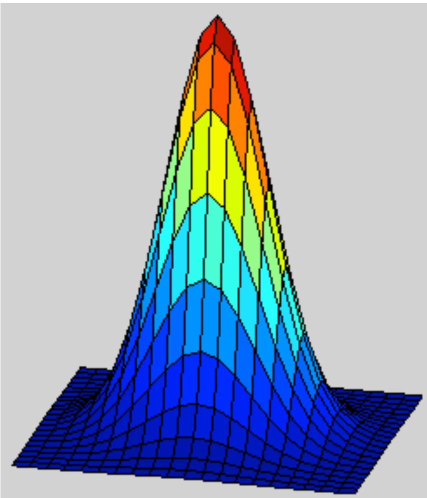


$$\psi_s(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \sin(2\pi u_0 x)$$

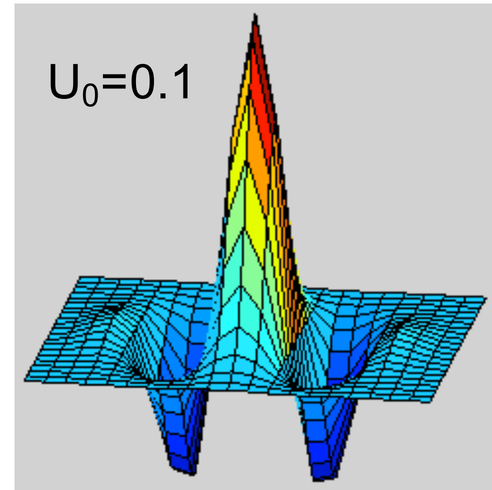


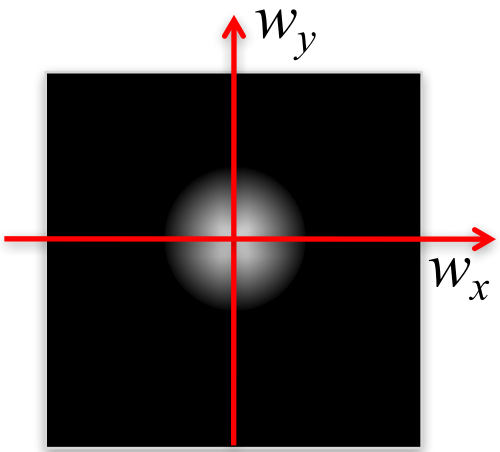


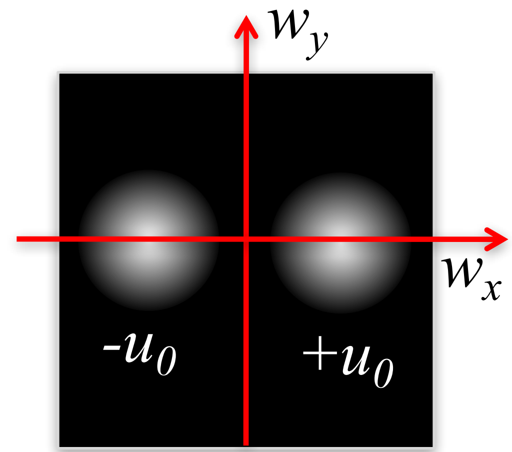
Fourier transform of a Gabor wavelet

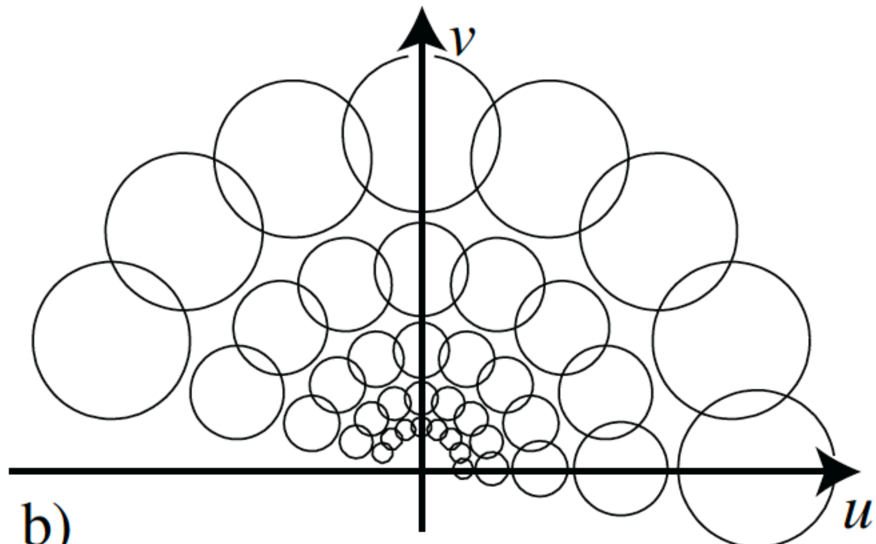


$$\psi_c(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \cos(2\pi u_0 x)$$

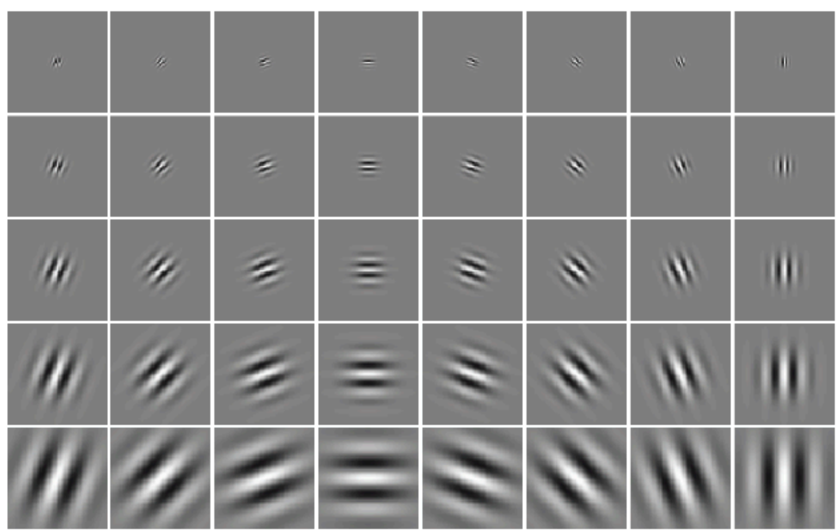








b)



Comparing Human and Machine Perception

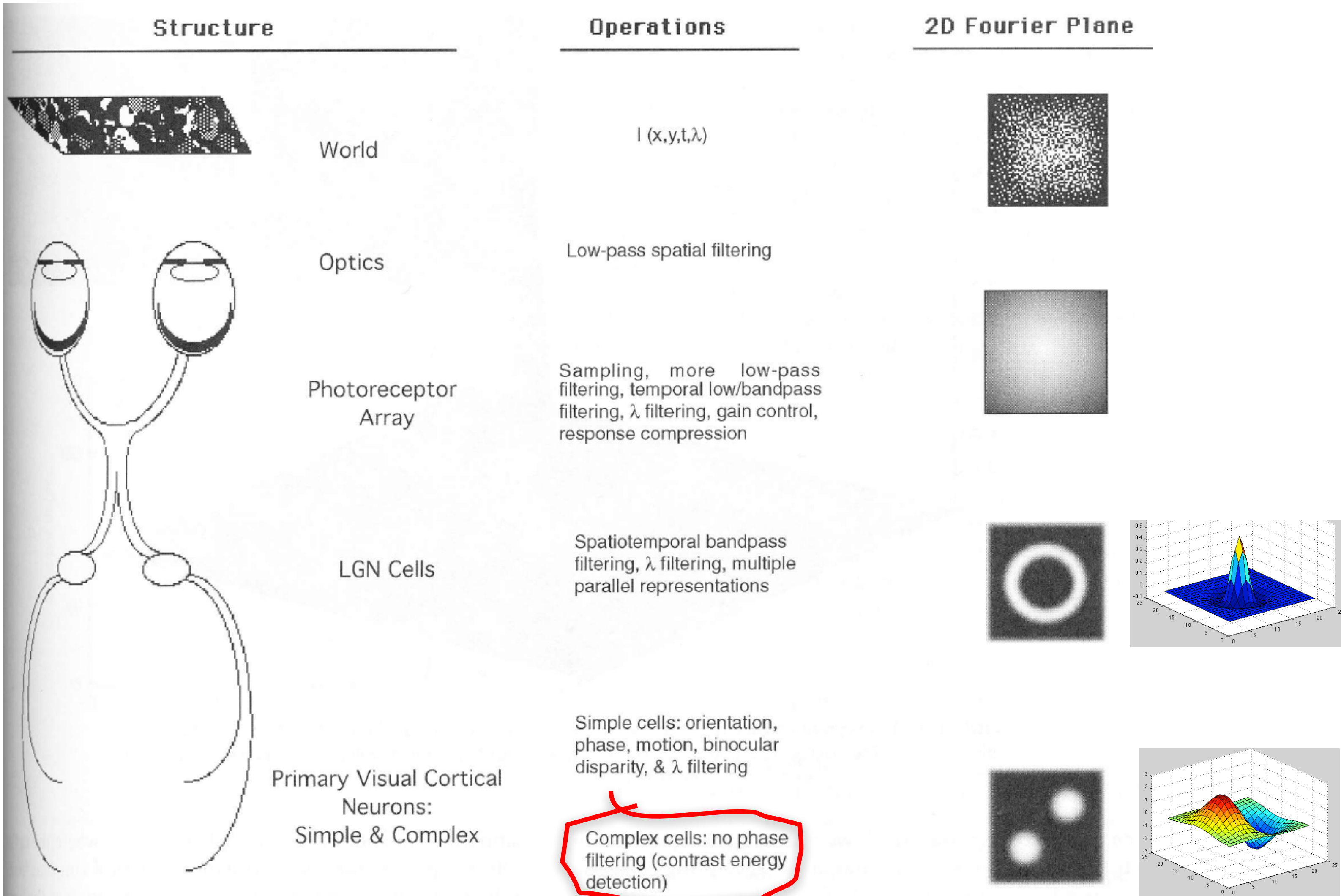


FIGURE 1 Schematic overview of the processing done by the early visual system. On the left, are some of the major structures to be discussed; in the middle, are some of the major operations done at the associated structure; in the right, are the 2-D Fourier representations of the world, retinal image, and sensitivities typical of a ganglion and cortical cell.

John Daugman, 1988

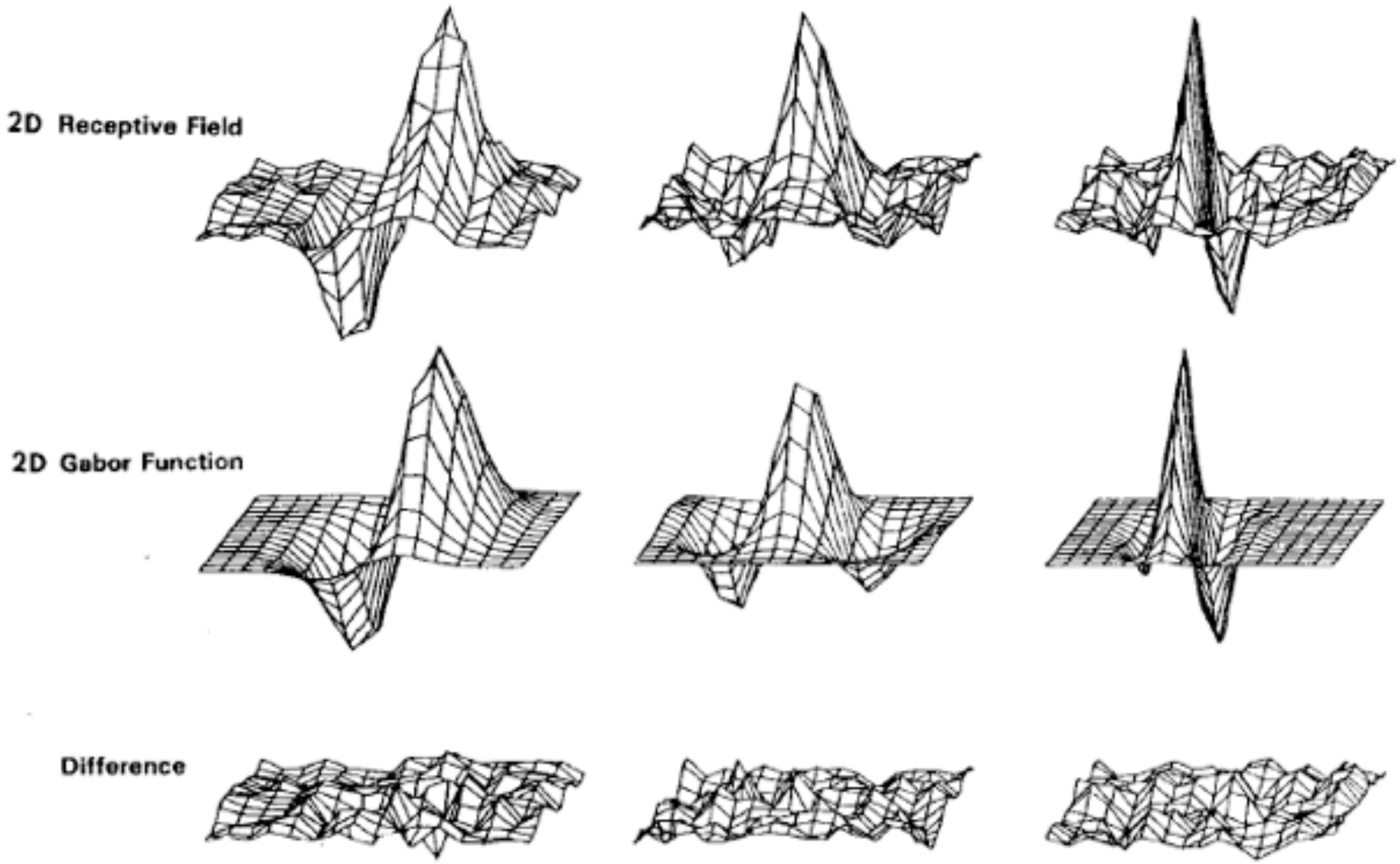
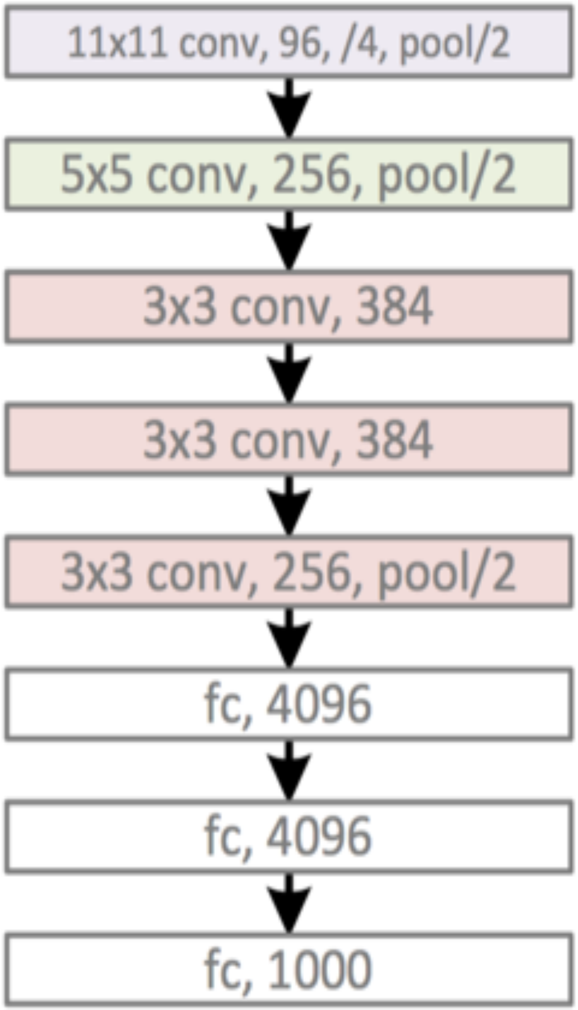


Fig. 5. Top row: illustrations of empirical 2-D receptive field profiles measured by J. P. Jones and L. A. Palmer (personal communication) in simple cells of the cat visual cortex. Middle row: best-fitting 2-D Gabor elementary function for each neuron, described by (10). Bottom row: residual error of the fit, indistinguishable from random error in the Chi-squared sense for 97 percent of the cells studied.

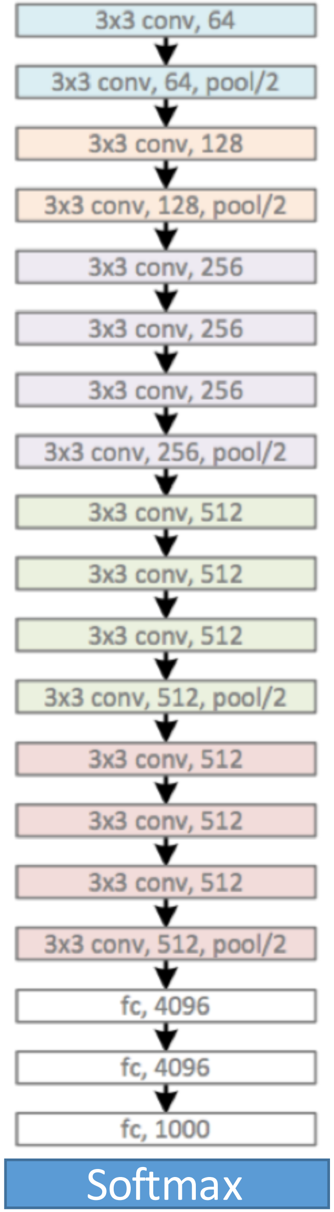
52

2012: AlexNet
5 conv. layers



Error: 15.3%

2014: VGG
16 conv. layers

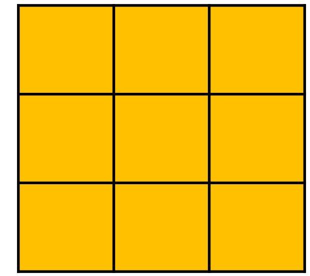


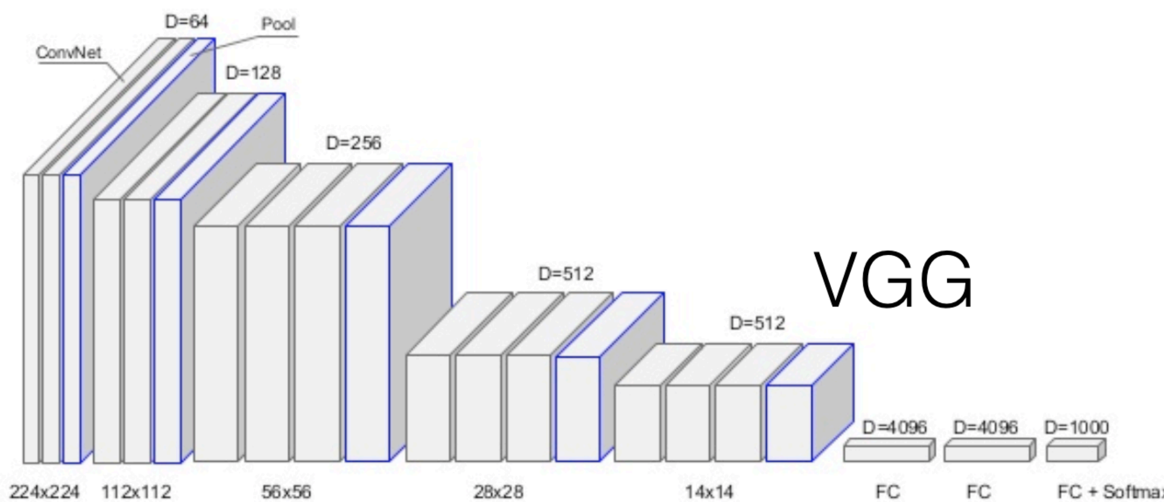
Error: 8.5%

VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

https://arxiv.org/pdf/1409.1556.pdf

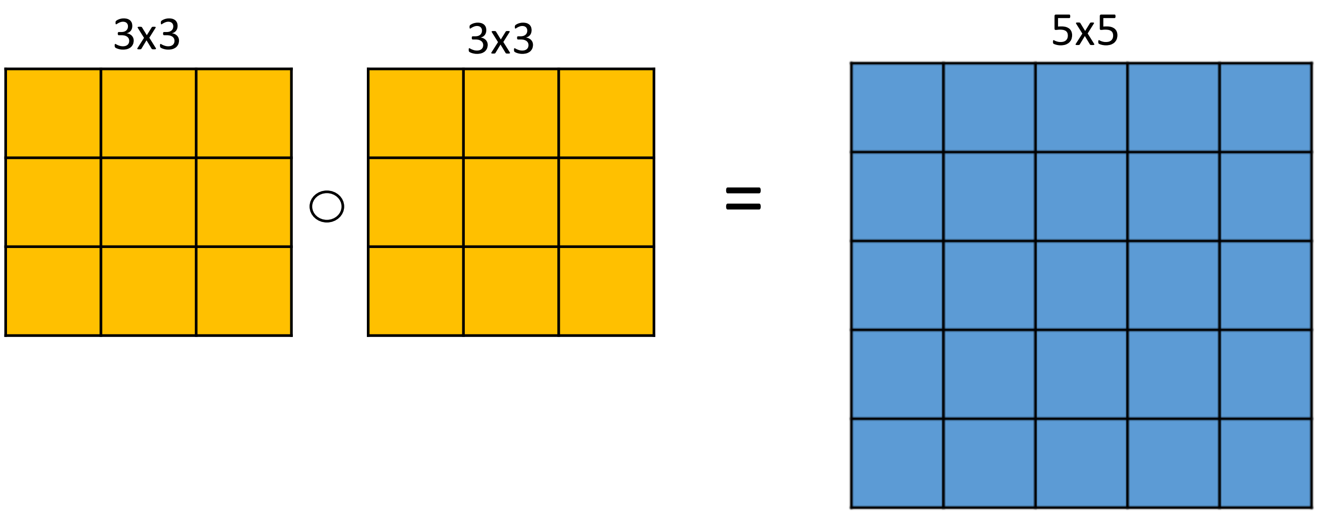
Small convolutional kernels: 3x3
ReLu non-linearities
>100 million parameters.



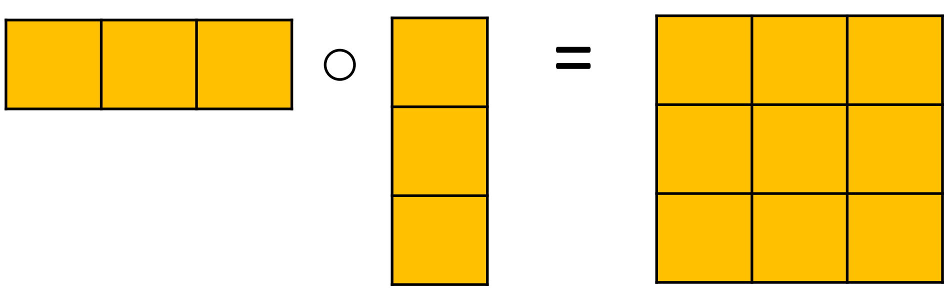


VGG

Chaining convolutions

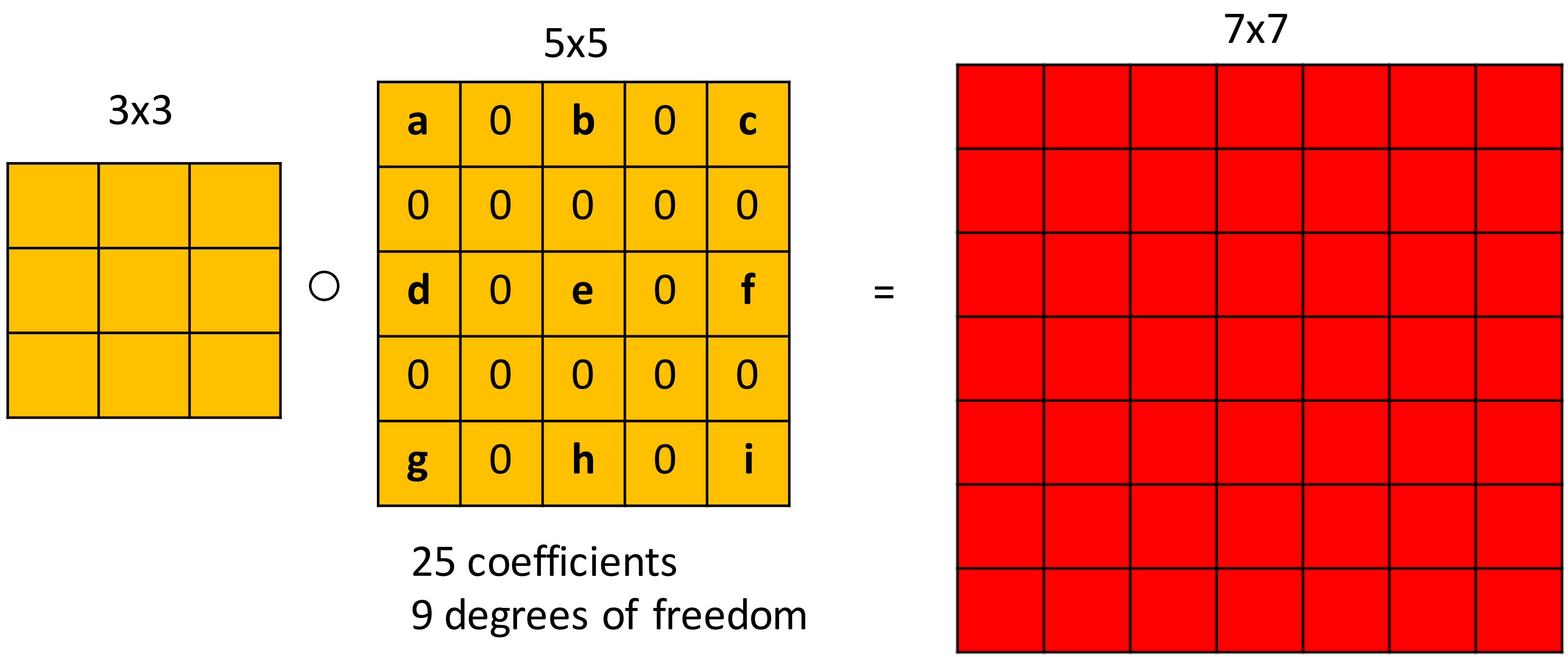


25 coefficients, but only
18 degrees of freedom



9 coefficients, but only
6 degrees of freedom.
Only separable filters… would this be enough?

Dilated convolutions



25 coefficients
9 degrees of freedom

49 coefficients
18 degrees of freedom

What is lost?

https://arxiv.org/pdf/1511.07122.pdf

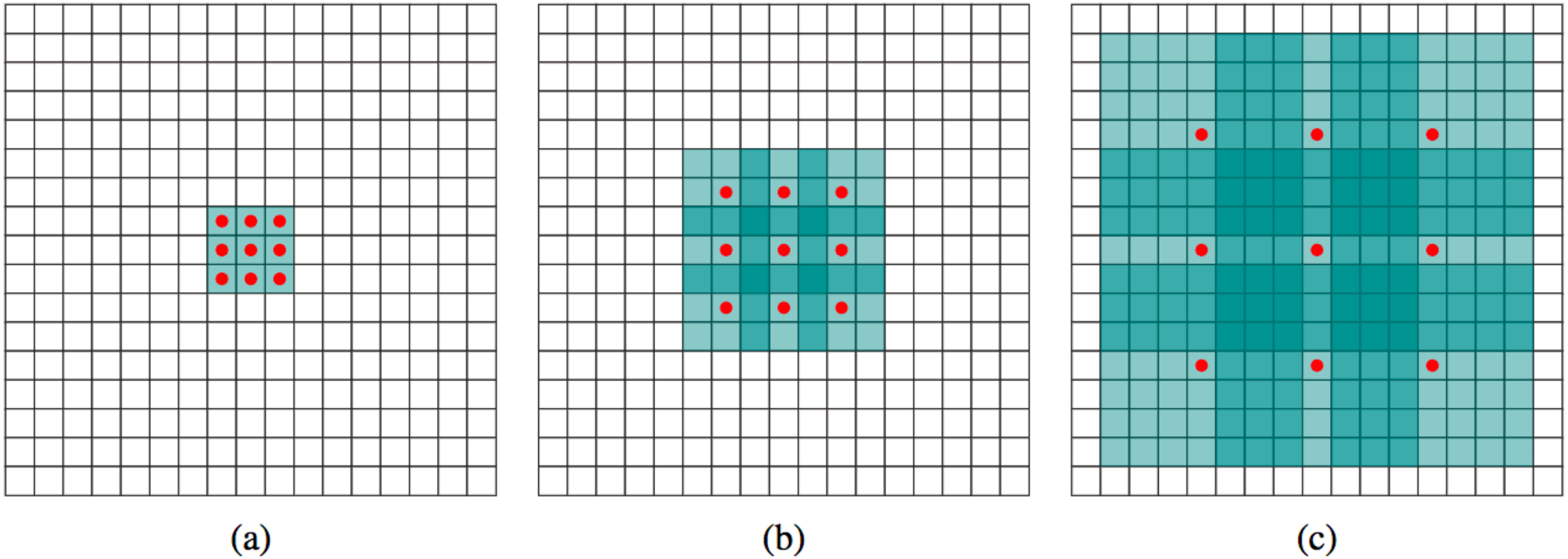


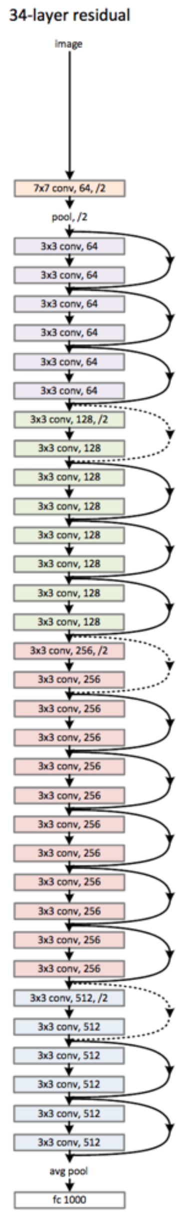
Figure 1: Systematic dilation supports exponential expansion of the receptive field without loss of resolution or coverage. (a) F_1 is produced from F_0 by a 1-dilated convolution; each element in F_1 has a receptive field of 3×3. (b) F_2 is produced from F_1 by a 2-dilated convolution; each element in F_2 has a receptive field of 7×7. (c) F_3 is produced from F_2 by a 4-dilated convolution; each element in F_3 has a receptive field of 15×15. The number of parameters associated with each layer is identical. The receptive field grows exponentially while the number of parameters grows linearly.

https://arxiv.org/pdf/1511.07122.pdf

2016: ResNet
>100 conv. layers

Deep Residual Learning for Image Recognition

https://arxiv.org/pdf/1512.03385.pdf



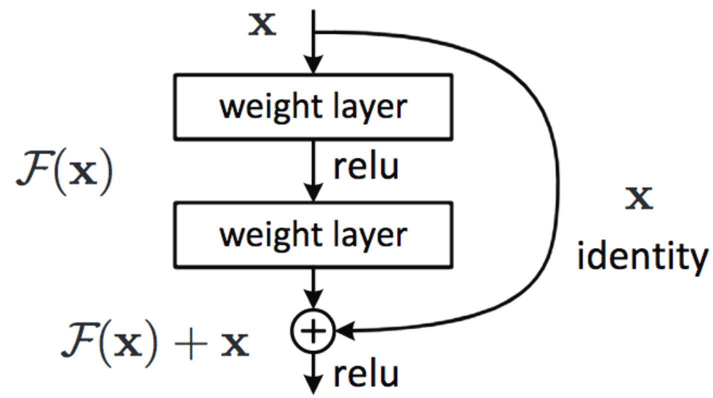
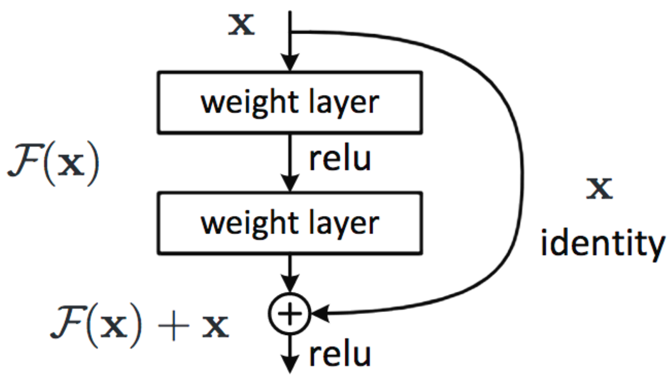


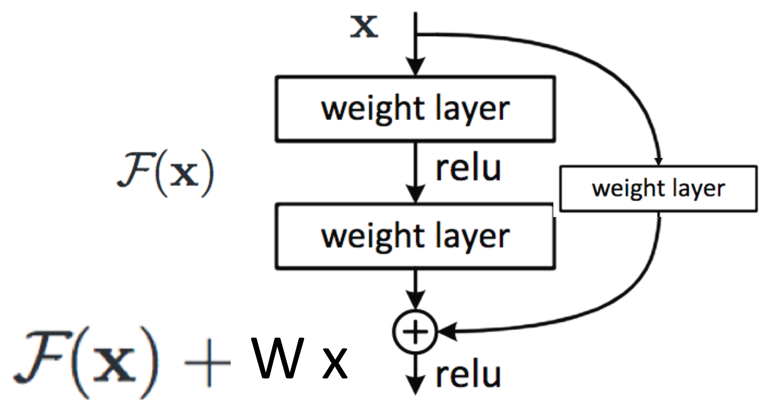
Figure 2. Residual learning: a building block.

Error: 4.4%

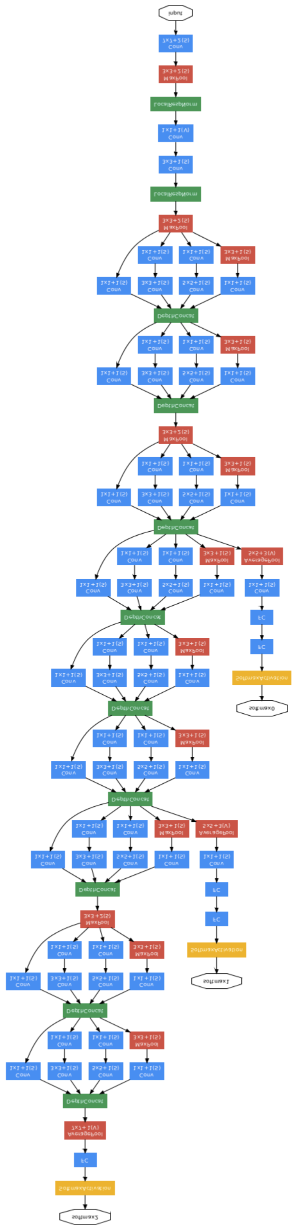
If output has same size as input:



If output has a different size:



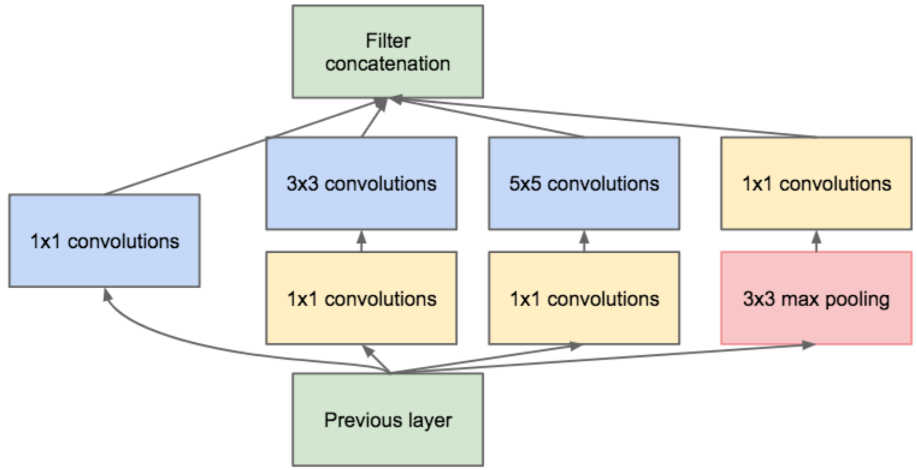
2015: GoogLeNet
22 conv. layers



Error: 7.8%

Inception GoogLeNet

https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43022.pdf



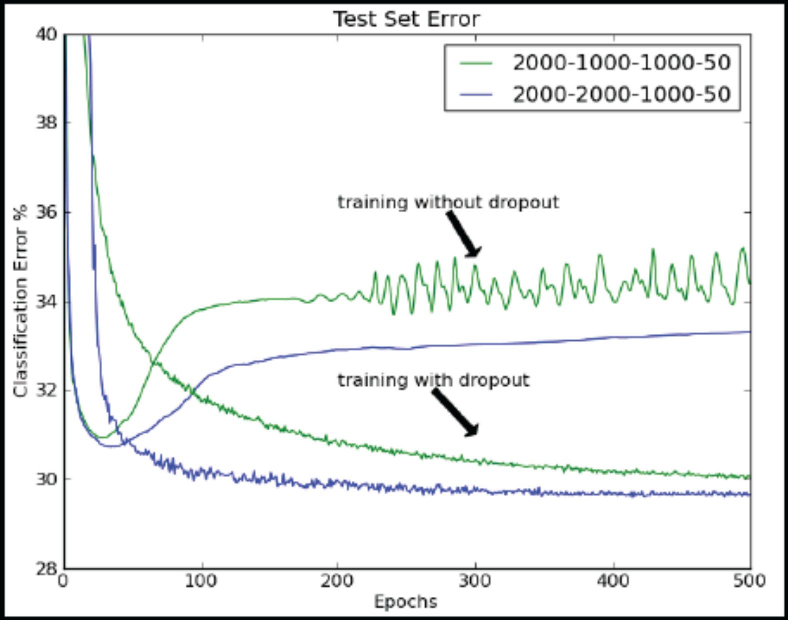
(b) Inception module with dimensionality reduction

Figure 2: Inception module

DropOut

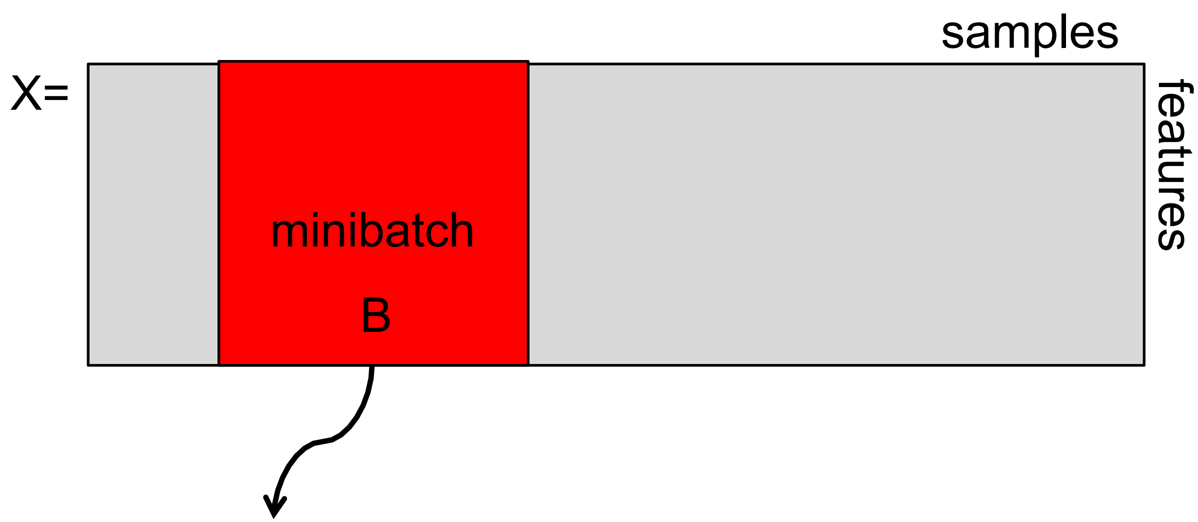
- G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov, *Improving neural networks by preventing co-adaptation of feature detectors*, arXiv:1207.0580 2012

- Fully connected layers only
- Randomly set activations in layer to zero
- Gives ensemble of models
- Similar to bagging [Breiman'94], but differs in that parameters are shared.



Slide Rob Fergus

Batch normalization



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

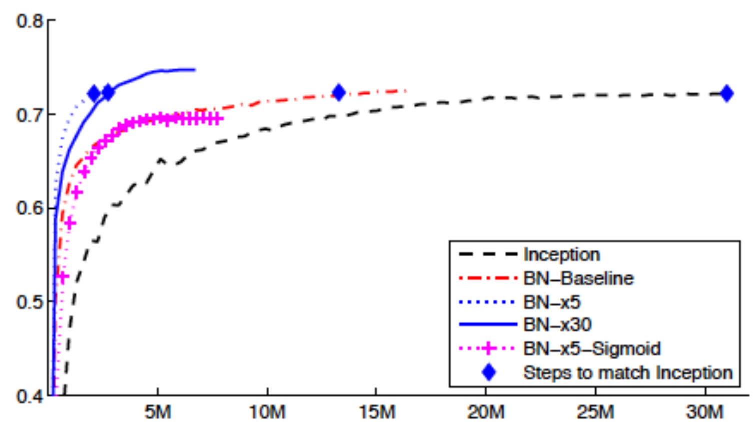


Figure 2: *Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.*

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, Sergey Ioffe, Christian Szegedy, arXiv:1502.03167

Batch normalization

- Training: take into account the normalization in backdrop

  Derivative wrt x_i depends on the partial derivative of the mean and stddev

  Must also update γ and β

- Test time: use the global mean stddev at test time

  Removes the stochasticity of the mean and stddev

  Requires a final phase where, from the first to the last hidden layer

  1. propagate all training data to that layer
  2. compute and store the global mean and stddev of each unit

Fooling Convnets

- Search for images that are misclassified by the network
- Intriguing properties of neural networks, Christian Szegedy et al. arXiv 1312.6199, 2013
- Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images, Anh Nguyen, Jason Yosinski, Jeff Clune, arXiv 1412.1897.
- Problem common to any discriminative method

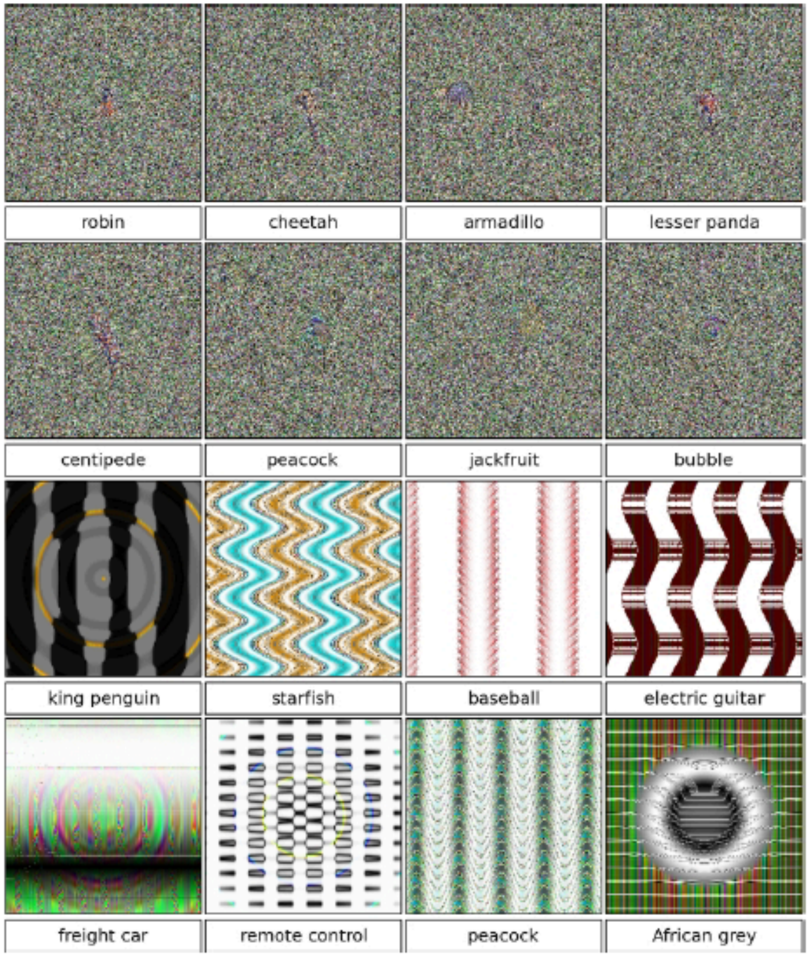
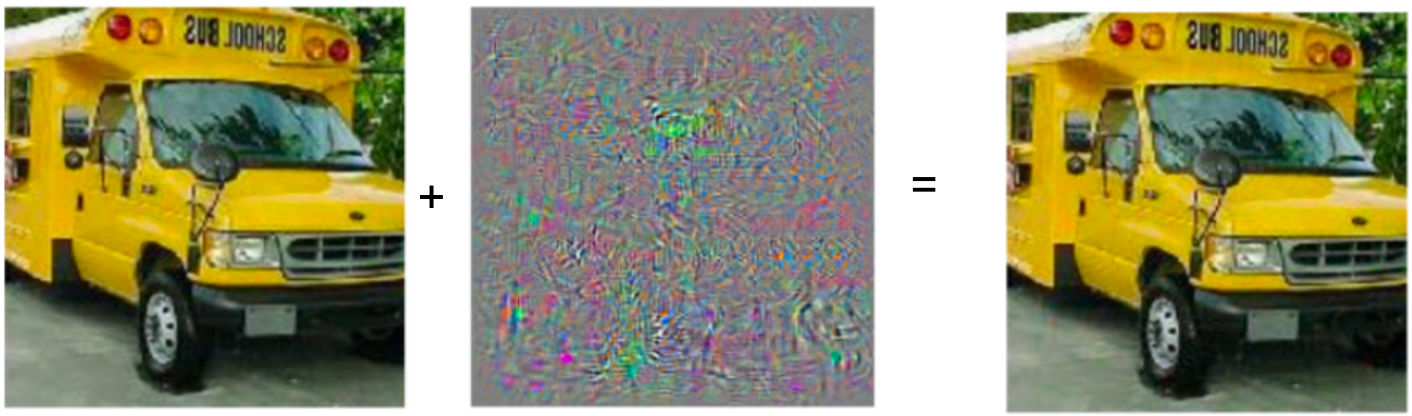


Figure 1. Evolved images that are unrecognizable to humans, but that state-of-the-art DNNs trained on ImageNet believe with $\geq$ 99.6% certainty to be a familiar object. This result highlights differences between how DNNs and humans recognize objects.

Slide Rob Fergus

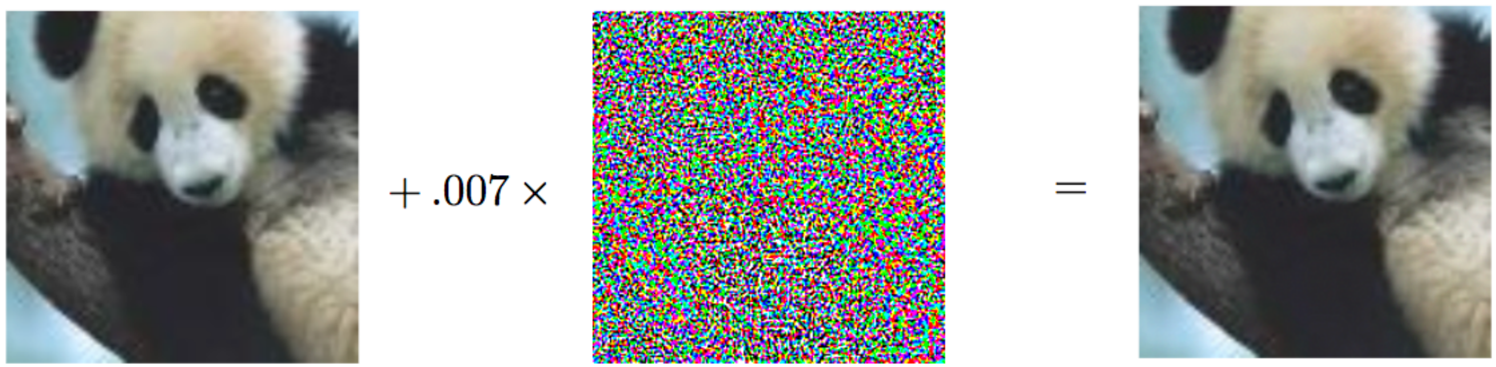
Intriguing properties of neural networks

https://arxiv.org/pdf/1312.6199.pdf



Bus

Ostrich



$\boldsymbol{x}$
"panda"
57.7% confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence

$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

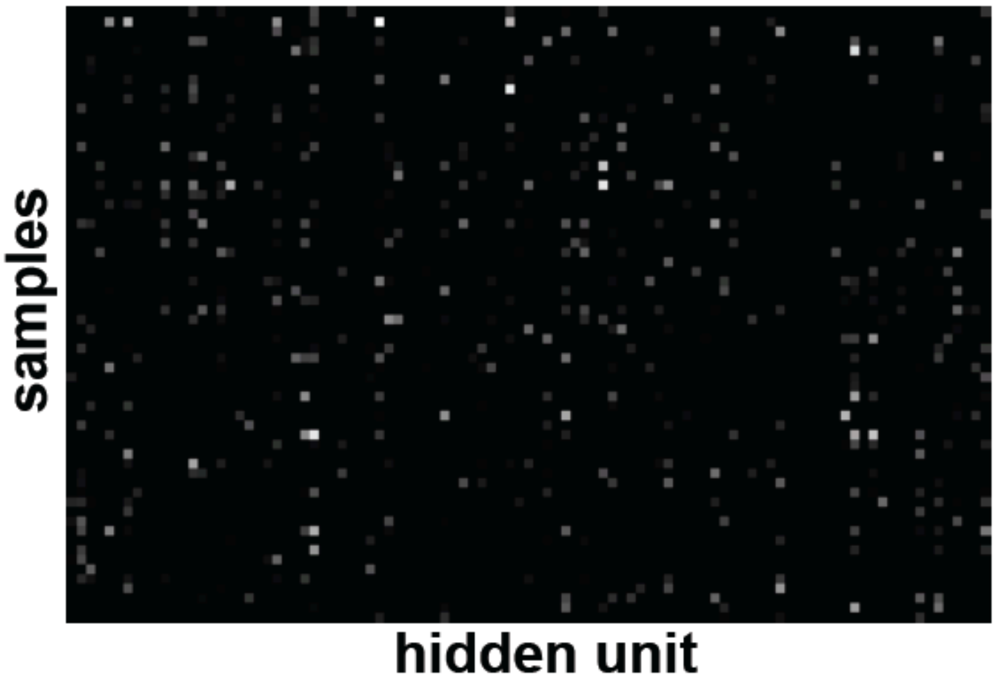
https://en.wikipedia.org/wiki/Gibbon

EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES

https://arxiv.org/pdf/1412.6572.pdf

OTHER THINGS GOOD TO KNOW

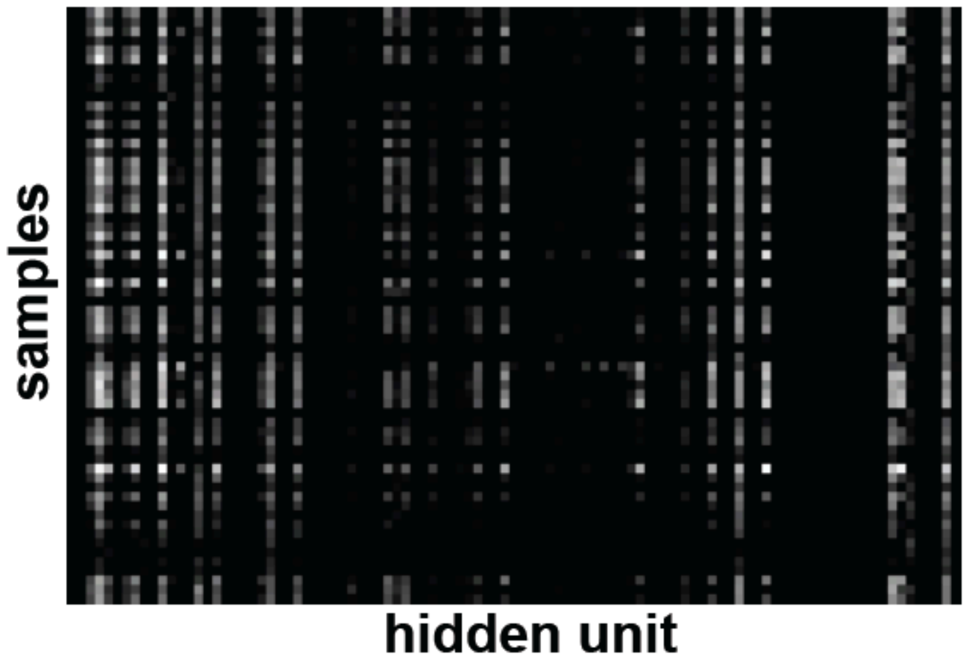
- Check gradients numerically by finite differences
- Visualize features (feature maps need to be uncorrelated) and have high variance.



Good training: hidden units are sparse across samples and across features.

OTHER THINGS GOOD TO KNOW

- Check gradients numerically by finite differences
- Visualize features (feature maps need to be uncorrelated) and have high variance.

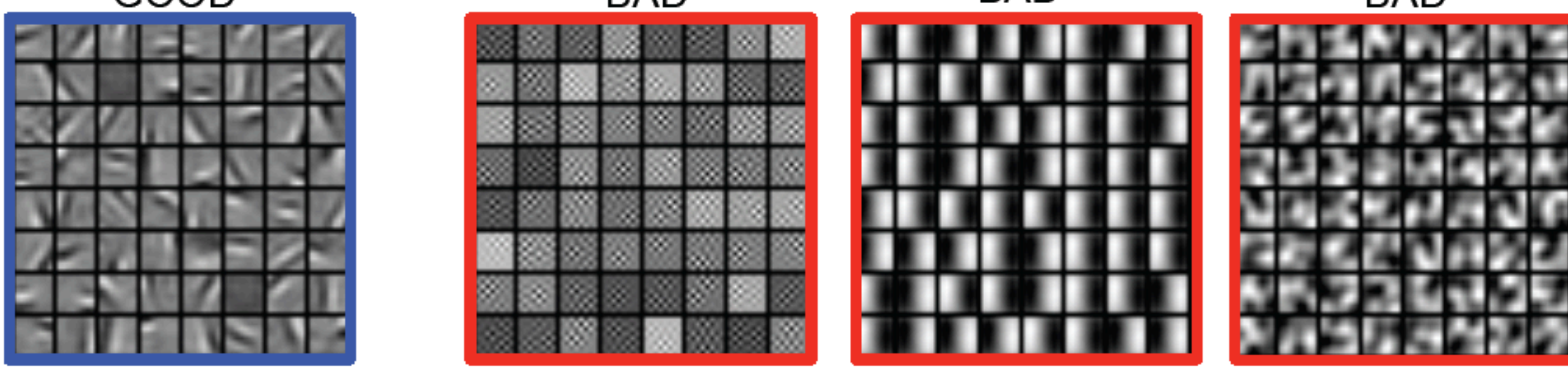


Bad training: many hidden units ignore the input and/or exhibit strong correlations.

Ranzato

OTHER THINGS GOOD TO KNOW

- Check gradients numerically by finite differences
- Visualize features (feature maps need to be uncorrelated) and have high variance.
- Visualize parameters



GOOD | BAD (too noisy) | BAD (too correlated) | BAD (lack structure)

Good training: learned filters exhibit structure and are uncorrelated.

Ranzato

OTHER THINGS GOOD TO KNOW

- Check gradients numerically by finite differences
- Visualize features (feature maps need to be uncorrelated) and have high variance.
- Visualize parameters
- Measure error on both training and validation set.
- Test on a small subset of the data and check the error $\rightarrow 0$.