Deep Learning 2

Jian Li

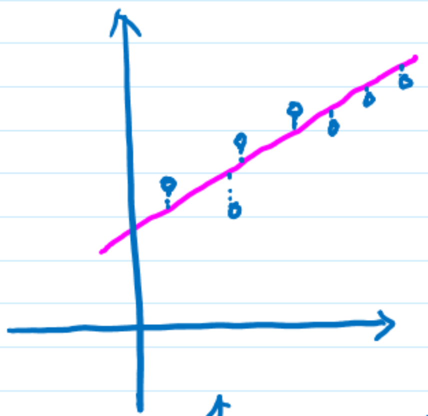
IIIS, Tsinghua

Some Linear Algebra, PCA, Eigenface

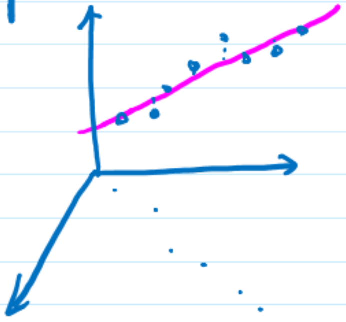
Least Square

Least square Problem (LS)

$$\inf_{x \in \mathbb{R}^n} \|Ax - b\|_2^2 \qquad A \in \mathbb{R}^{m \times n} \qquad \begin{array}{c c} & n \\ m & \begin{bmatrix} A \end{bmatrix} \end{array} \begin{bmatrix} x \end{bmatrix} - \begin{bmatrix} b \end{bmatrix} m$$



$$f(x) = (Ax-b)^T(Ax-b) = x^TA^TAx - 2b^TAx + b^Tb$$



Let $\nabla f = 2A^TAx - 2A^Tb = 0$

$$A^TAx = A^Tb$$

if $\text{rank}(A) = n$, A^TA is invertible , so $x = \underline{(A^TA)^{-1}A^T}b$
(full col rank)

if not full col rank, we need to solve the problem in the row subspace
can do it via SVD :

Moore-Penrose Psendoinverse
if $\text{rank}(A) = n$

SVD

Moore-Penrose inverse

SVD: $A = U\Sigma V^T$ $\quad \operatorname{rank}(A) = r$

$$\begin{array}{c} \\ m \end{array}\begin{array}{c} n \\ \left[\; A \;\right] \end{array} = \begin{array}{c} \\ m \end{array}\begin{array}{c} r \\ \left[\; U \;\right] \end{array} \begin{array}{c} r \\ \left[\begin{matrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{matrix}\right] \end{array} \begin{array}{c} \\ r \end{array}\begin{array}{c} n \\ \left[\; V^T \;\right] \end{array}$$

$\leftarrow$ orthonormal rows $\quad V^T V = I_{r \times r}$

a basis of row(A)

singular values

orthonormal col $\quad U^T U = I_{r \times r}$

a basis of col(A)

$A^{inv} = V \Sigma U^T$

$$\begin{array}{c} n \end{array}\begin{array}{c} \left[\; A^{inv} \;\right] \\ m \end{array} = \begin{array}{c} \\ n \end{array}\begin{array}{c} r \\ \left[\; V \;\right] \end{array} \begin{array}{c} \\ \left[\begin{matrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{matrix}\right] \end{array} \begin{array}{c} \\ r \end{array}\begin{array}{c} m \\ \left[\; U^T \;\right] \end{array}$$

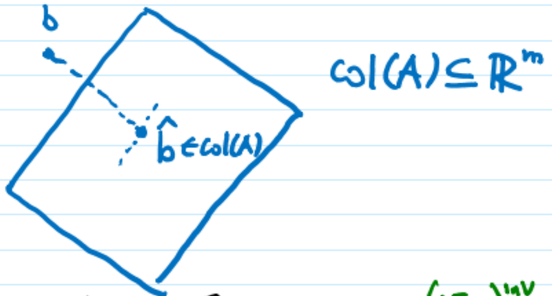
The solution to the LS is still $A^{inv} b$

Geometric View

Now let us derive it geometrically

Consider the subspace spanned by col of A : col(A)

So $Ax \in \mathrm{col}(A)$, $\|Ax-b\|_2$ is the dist between Ax and b



Def: Projection operator (onto col(A))
(orthogonal)

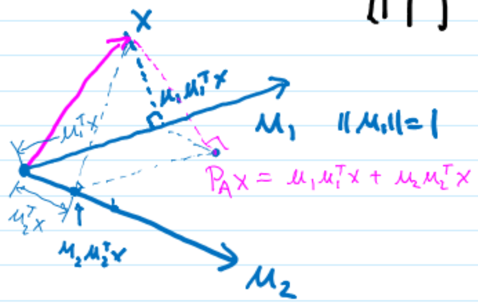
(Projection: $PP = P$

orthogonal proj: $P = P^T$ (for real matrix))

$$P_A = UU^T \left(= A(A^TA)^{inv}A^T = U\Sigma V^T (V\Sigma^{-2}V^T) V\Sigma U^T. \text{ here we use } (A^TA)^{inv} = A^{inv}(A^T)^{inv}\right)$$

Geometric intuition: Consider $P_A x = UU^T x = \begin{bmatrix} | & | & \\ u_1 & u_2 & \cdots \\ | & | & \end{bmatrix} \begin{bmatrix} - & u_1 & - \\ - & u_2 & - \end{bmatrix} \begin{bmatrix} x \end{bmatrix}$

$$\begin{bmatrix} U^T \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} u_1^T x \\ u_2^T x \end{bmatrix}$$



Geometric View

Orthognality:

$x - P_A x = (I - P_A)x$ should be orthognal to col(A) :

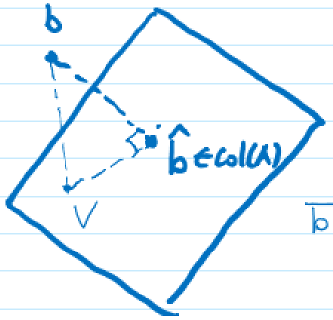
for every u_i : $u_i^T(I-P_A)x = (u_i^T - u_i^T U U^T)x = (u_i^T - (0,0,\cdots,\overset{i\text{th}}{1},\cdots 0)\,U^T)\,x = 0$

why orthognal Proj is the minimizer ?

for any vector $V \in \mathrm{col}(A)$,

$$\|V - b\|_2^2 = \|P_A b + (v - P_A b) - b\|_2^2 = \|P_A b - b\|_2^2 + \|V - P_A b\|_2^2$$

orthognal $\hookrightarrow \in \mathrm{col}(A)$



$\mathrm{col}(A) \subseteq \mathbb{R}^m$

$\overline{bV}^2 = \overline{b\hat{b}}^2 + |\hat{b}V|^2$

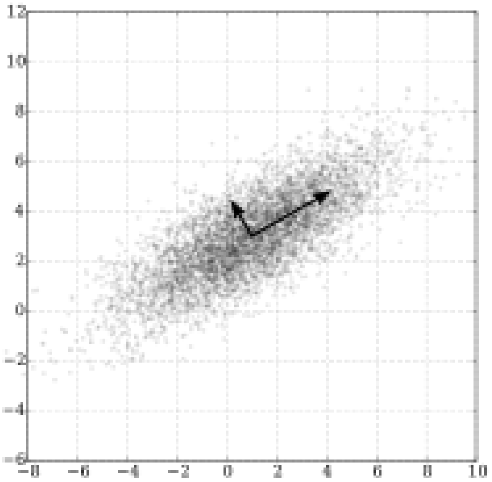
(Pythagorean THM)

Principle Component Analysis

- First principle component: the direction that maximizes the variance (which is the first eigen-vector of the covariance matrix $X^T X$)

$$\mathbf{w}_{(1)} = \arg\max \left\{ \frac{\mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w}}{\mathbf{w}^T \mathbf{w}} \right\}$$

- 2nd principle component: the direction orthogonal to 1st PC and maximizes the variance
- Dimension reduction: project to the first few PC

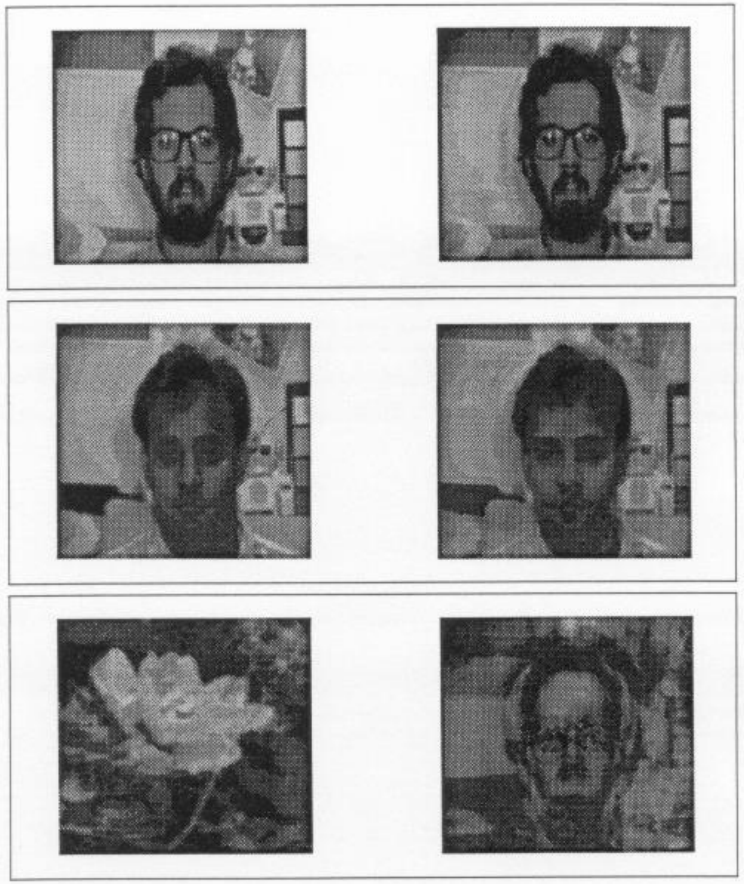


Eigen-face [Turk, Pentland '91]

- Treat each face as a vector
- Eigen face: just principle components

1. Detect whether a figure is a face (see the distance from it to the subspace spanned by the first few PC

Figure 4. Three images and their projections onto the face space defined by the eigenfaces of Figure 2. The relative measures of distance from face space are (a) 29.8, (b) 58.5, (c) 5217.4. Images (a) and (b) are in the original training set.



Eigen-face

1. Detect and locate a face in a figure (like CNN)
2. Tracking movement of a face
3. Reconstruct occluded image (ask student)
    - Dictionary learning

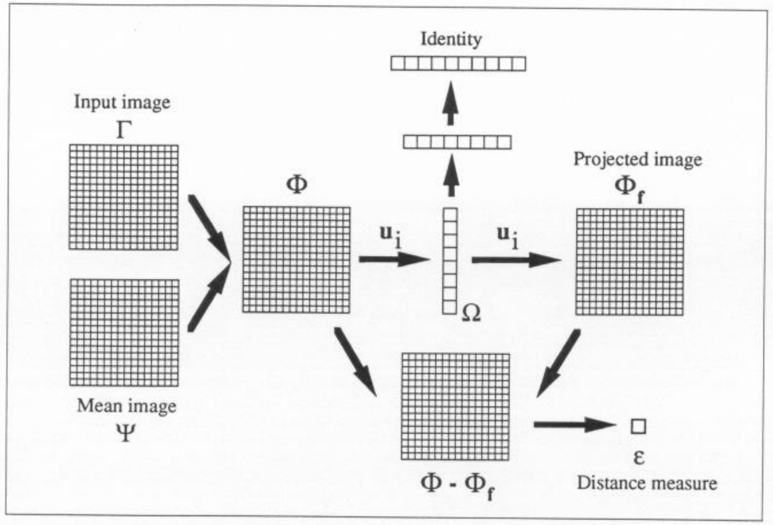


Figure 12. Collection of networks to implement computation of the pattern vector, projection into face space, distance from face space measure, and identification.

Figure 13. **(a)** Partially occluded face image and **(b)** its reconstruction using the eigenfaces.

Convolutional Neural Network

Convolution

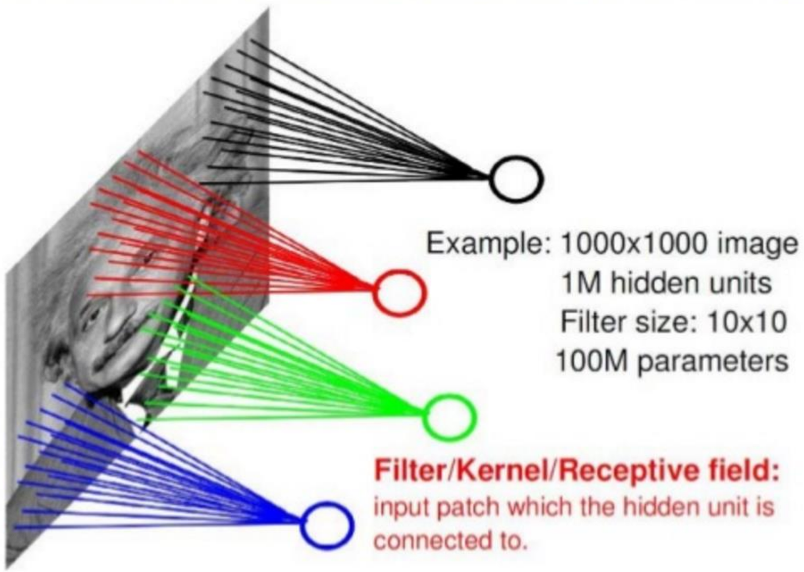
- 1d convolution (continuous):

$$s(t) = \int x(a)w(t-a)da \qquad s(t) = (x * w)(t)$$

- 1d convolution (discrete):

$$s[t] = (x * w)(t) = \sum_{a=-\infty}^{\infty} x[a]w[t-a]$$

Convolution

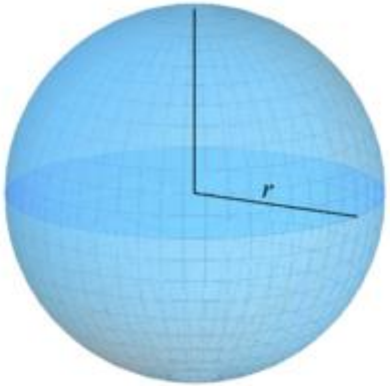


For a 2-D image **H** and a 2-D kernel **F**,

- Convolution Operator: $G = H \star F$

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

Random Vectors in High Dimension



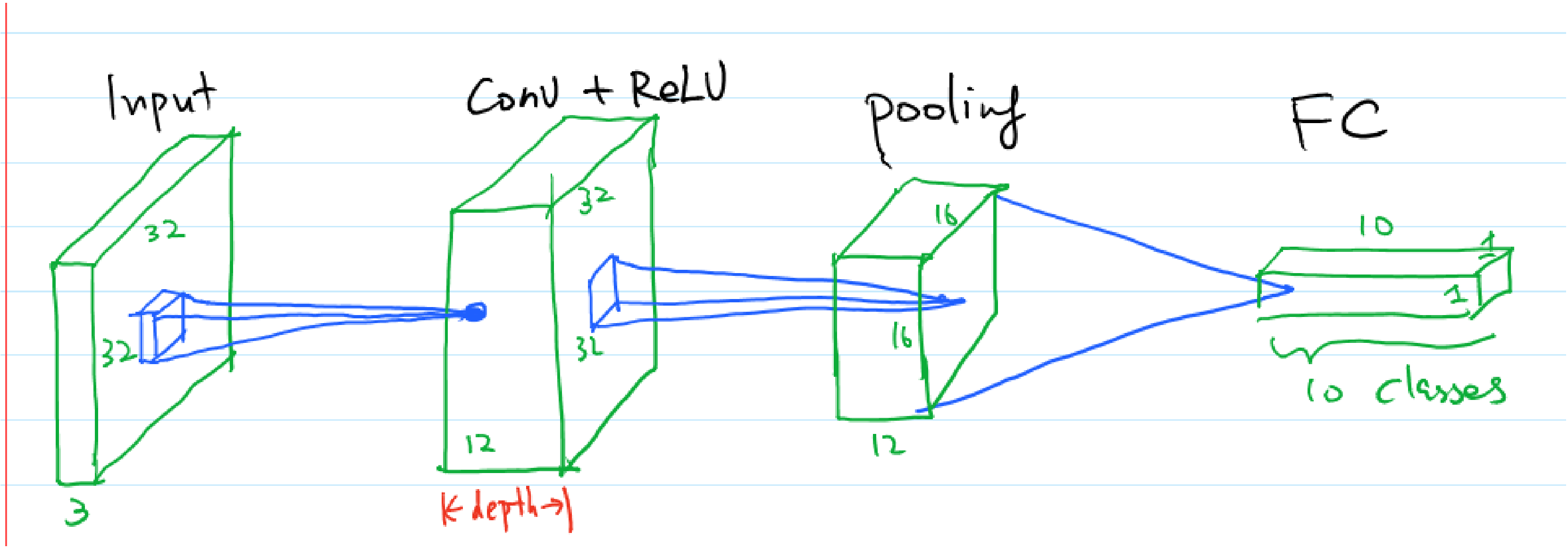
- Pick two i.i.d. n-dimensional Gaussian N(0,I) X, Y
- As n becomes large, X and Y are nearly orthogonal (i.e., $< X, Y >\approx 0$)

- Pick two points X, Y uniformly randomly from n-dimensional unit sphere
- As n becomes large, X and Y are nearly orthogonal (i.e., $< X, Y >\approx 0$)

- For two points X, Y, if $< X, Y >$ is far away from 0, they must be highly correlated.

High dimension phenomena – not true in low dimensions

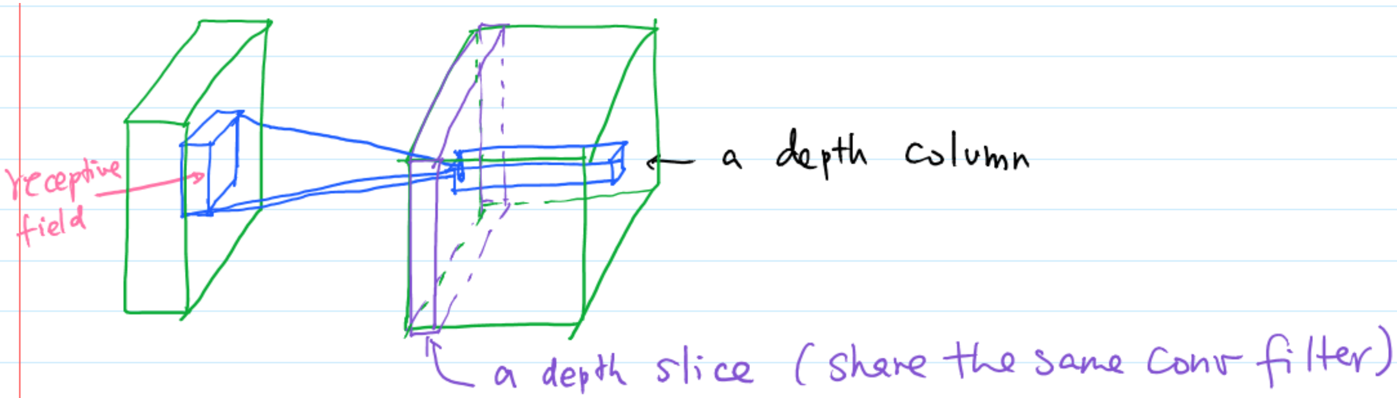
Basic architecture



- *Example Architecture: Overview*. We will go into more details below, but a simple ConvNet for CIFAR-10 classification could have the architecture [INPUT - CONV - RELU - POOL - FC]. In more detail:
- INPUT [32x32x3] will hold the raw pixel values of the image, in this case an image of width 32, height 32, and with three color channels R,G,B.
- CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and the region they are connected to in the input volume. This may result in volume such as [32x32x12].
- RELU layer will apply an elementwise activation function, such as the max(0,x)max(0,x) thresholding at zero. This leaves the size of the volume unchanged ([32x32x12]).
- POOL layer will perform a downsampling operation along the spatial dimensions (width, height), resulting in volume such as [16x16x12].
- FC (i.e. fully-connected) layer will compute the class scores, resulting in volume of size [1x1x10], where each of the 10 numbers correspond to a class score, such as among the 10 categories of CIFAR-10. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

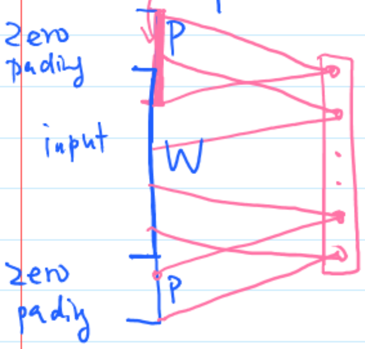
http://cs231n.github.io/convolutional-networks/

Convolution Layer



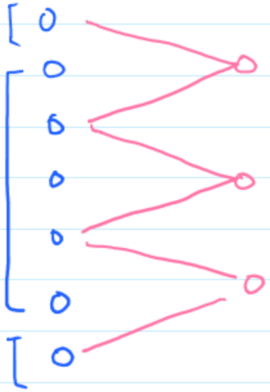
zero padding the boundary of input

F : size of receptive field.



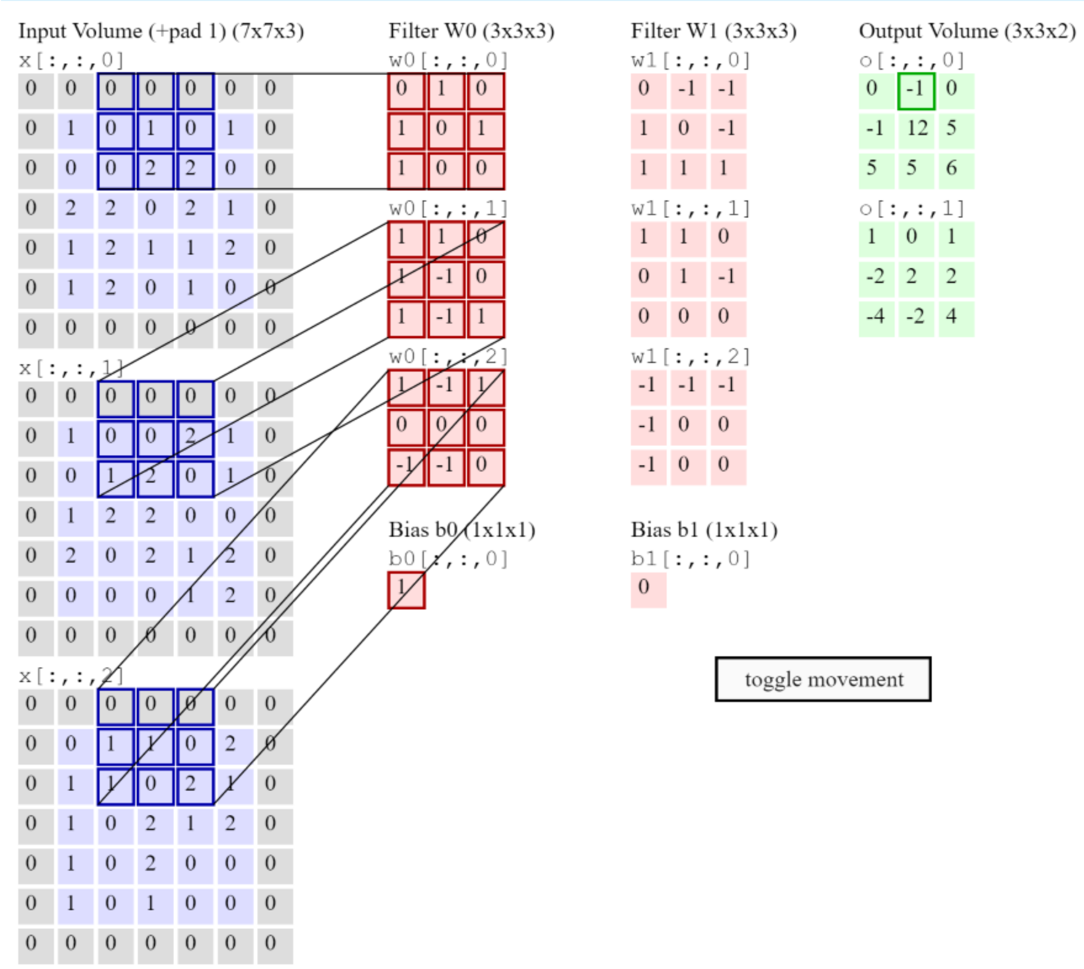
S: stride

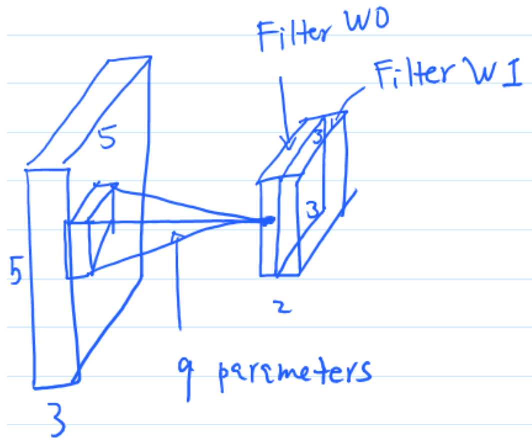
$$\#\text{Conv layer neurons} = \frac{W+2P-F}{S} + 1$$



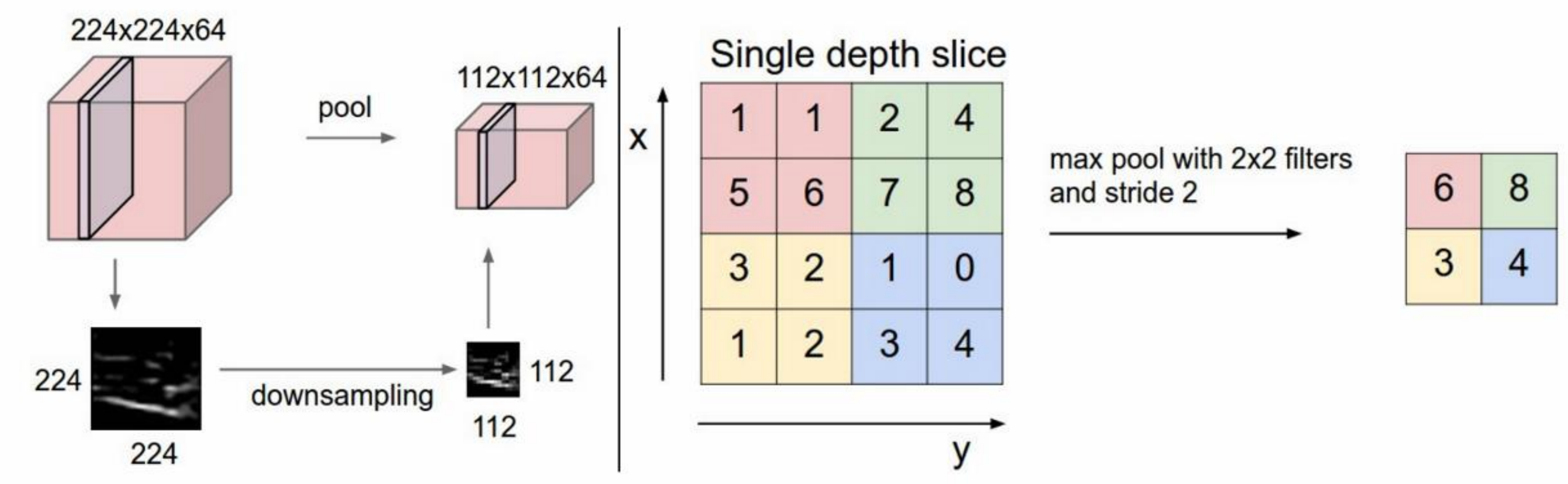
$W=5,\ P=1,\ F=3,\ S=2$

Convolution Layer



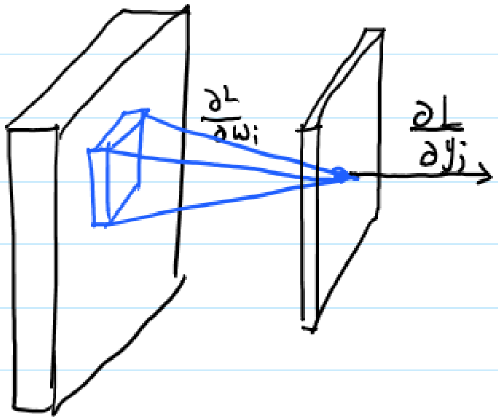


Pooling Layer



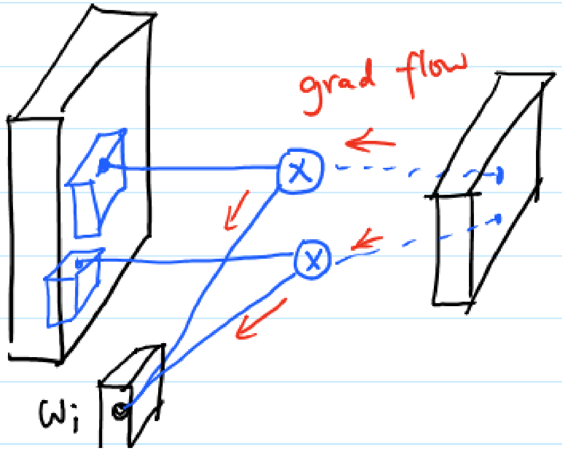
- fractional pooling: randomized 1×1, 1×2, 2×1, 2×2 pooling
- all convolutional Net

BP in CNN



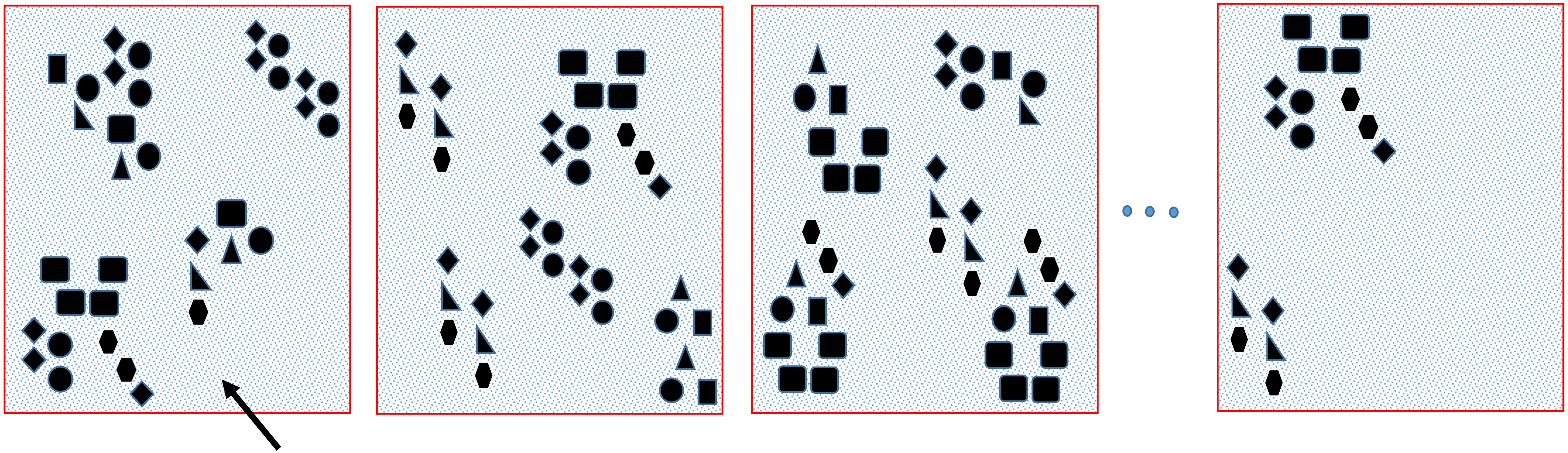
$$\frac{\partial L}{\partial w_i} = \sum_j \frac{\partial L}{\partial y_j} \cdot \frac{\partial y_j}{\partial w_i}$$

can be viewed as

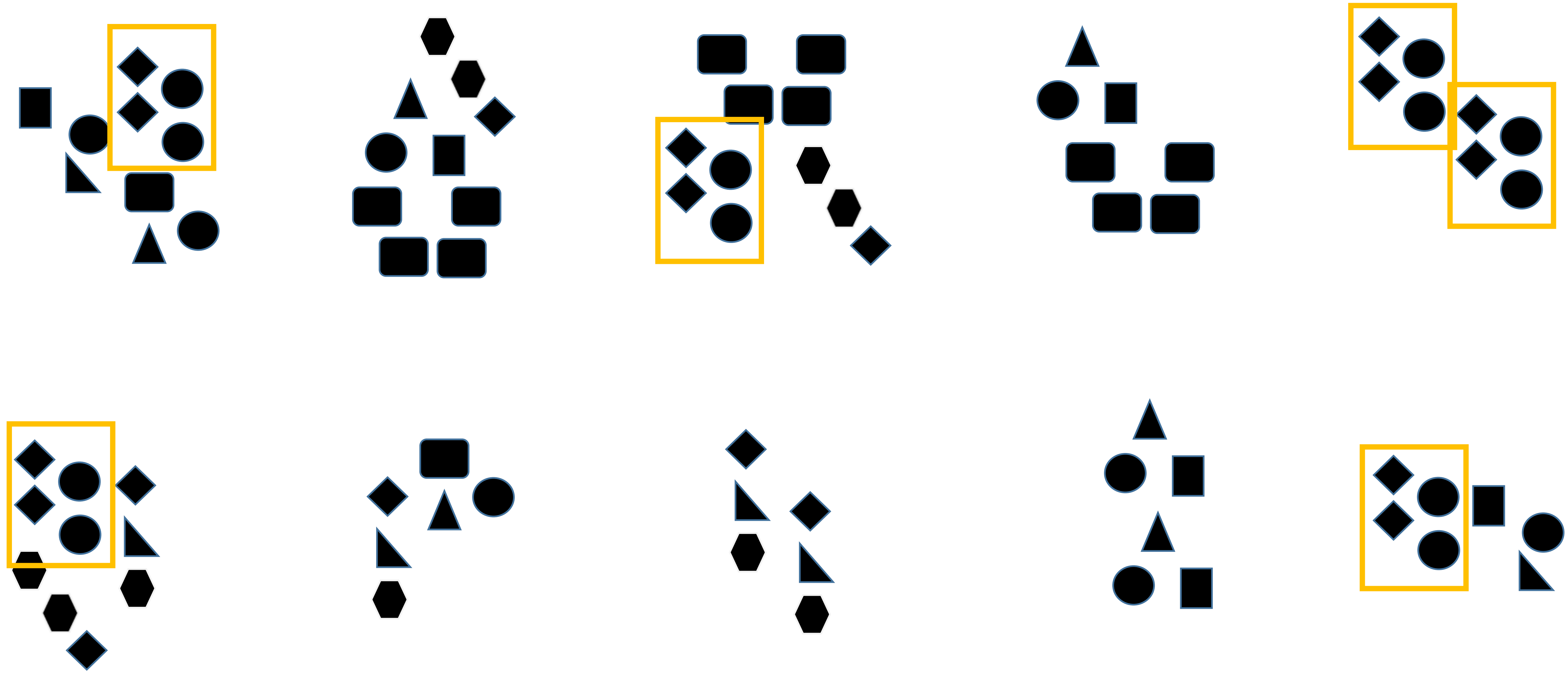


A Hierarchy of Features

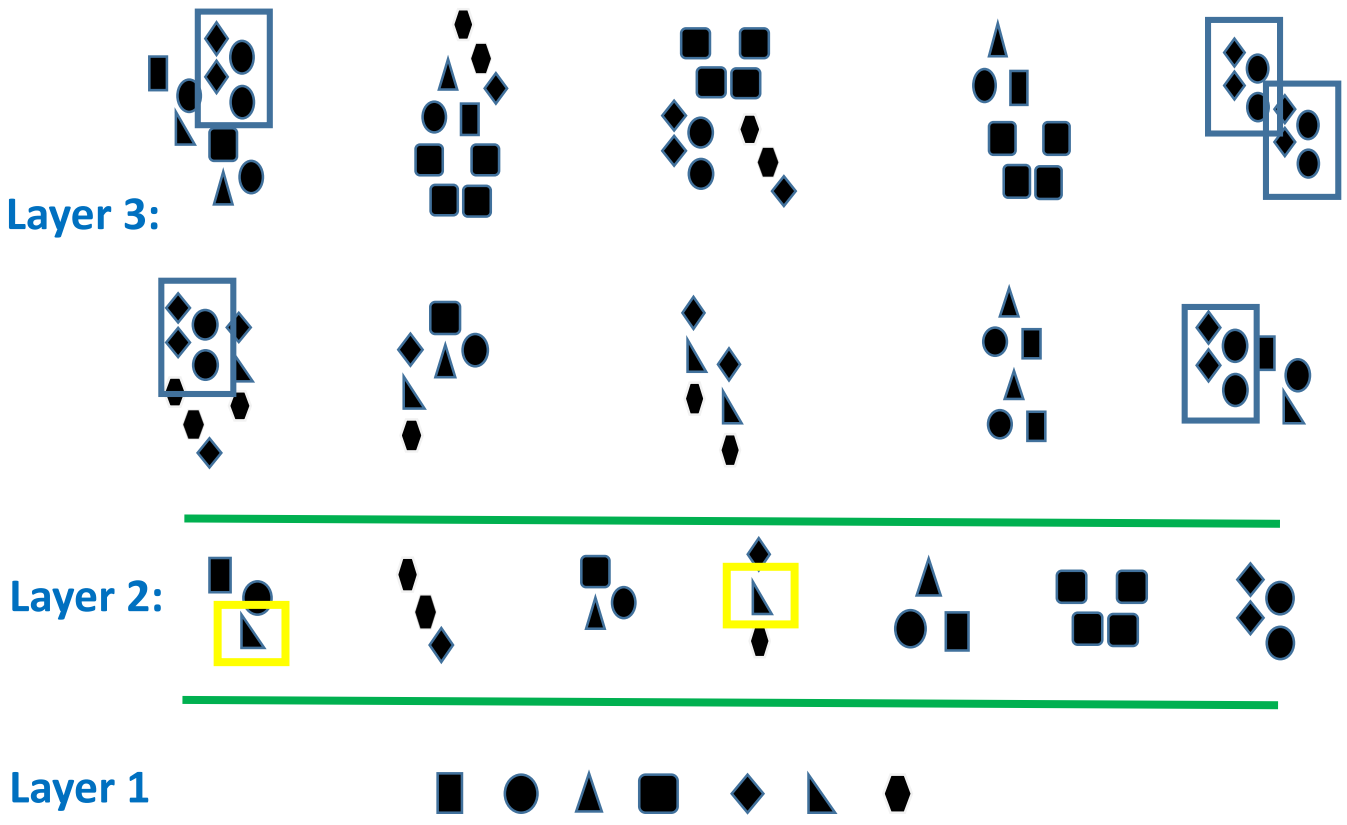
- Toy training images



A Hierarchy of Features

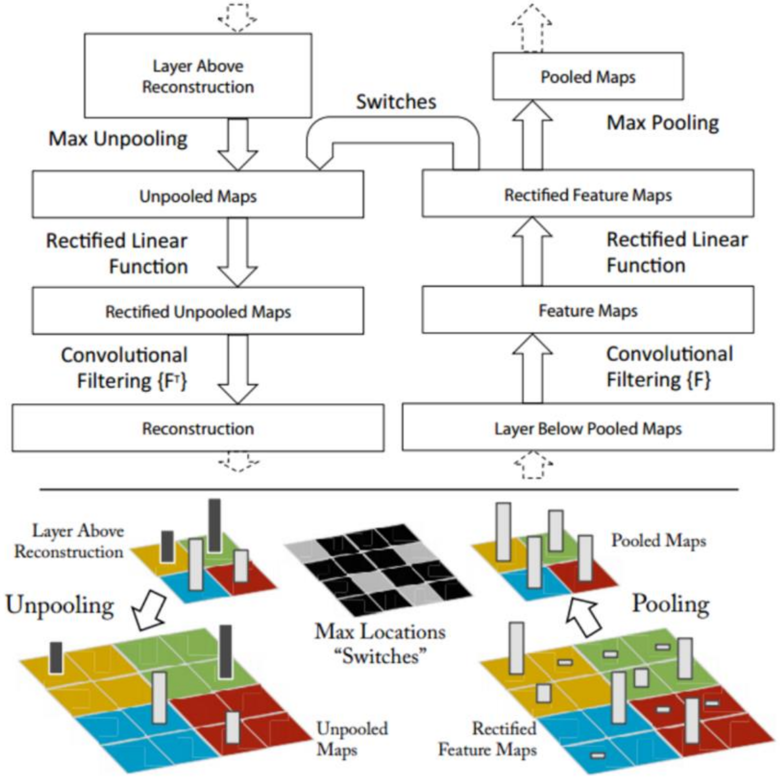


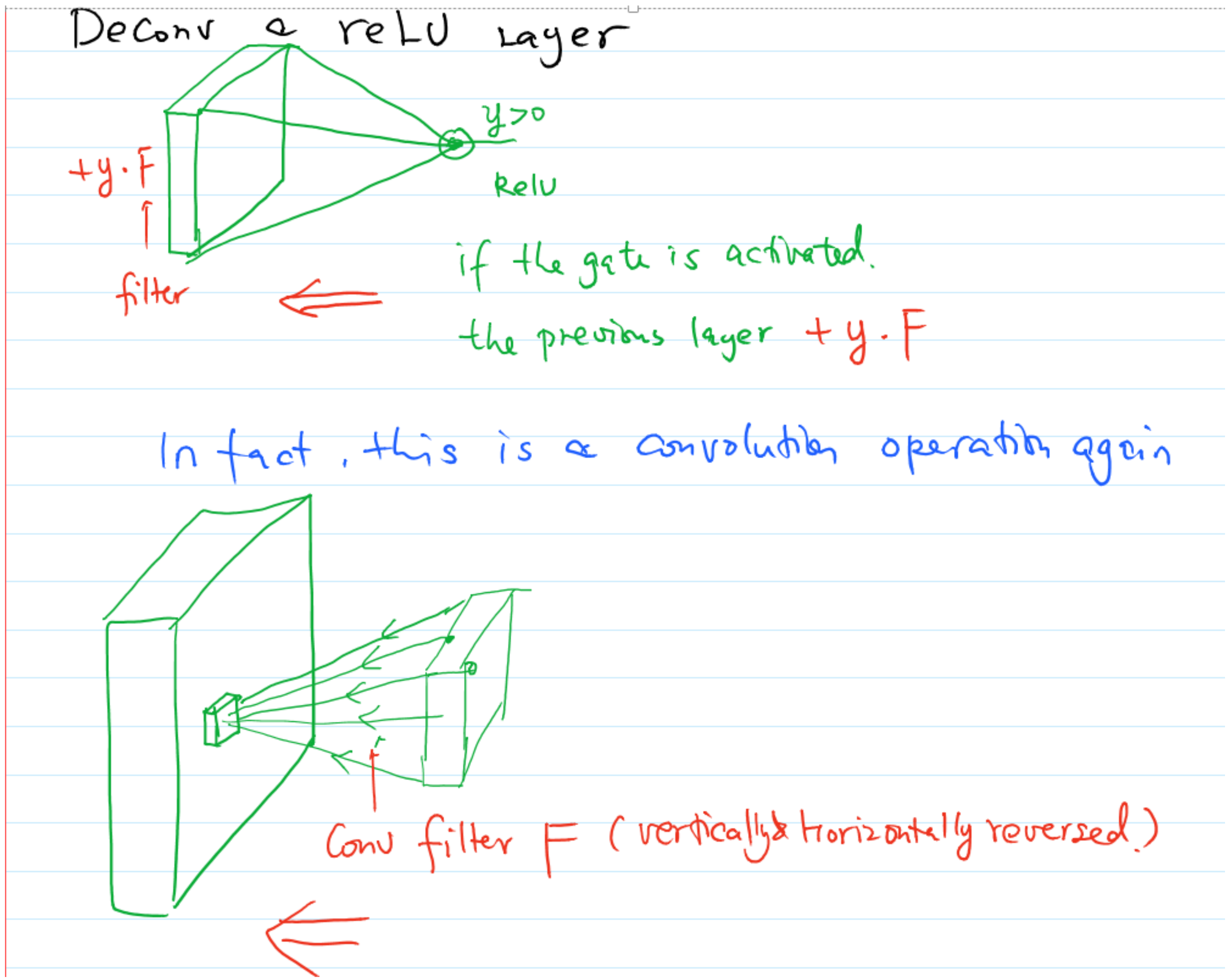
A Hierarchy of Features



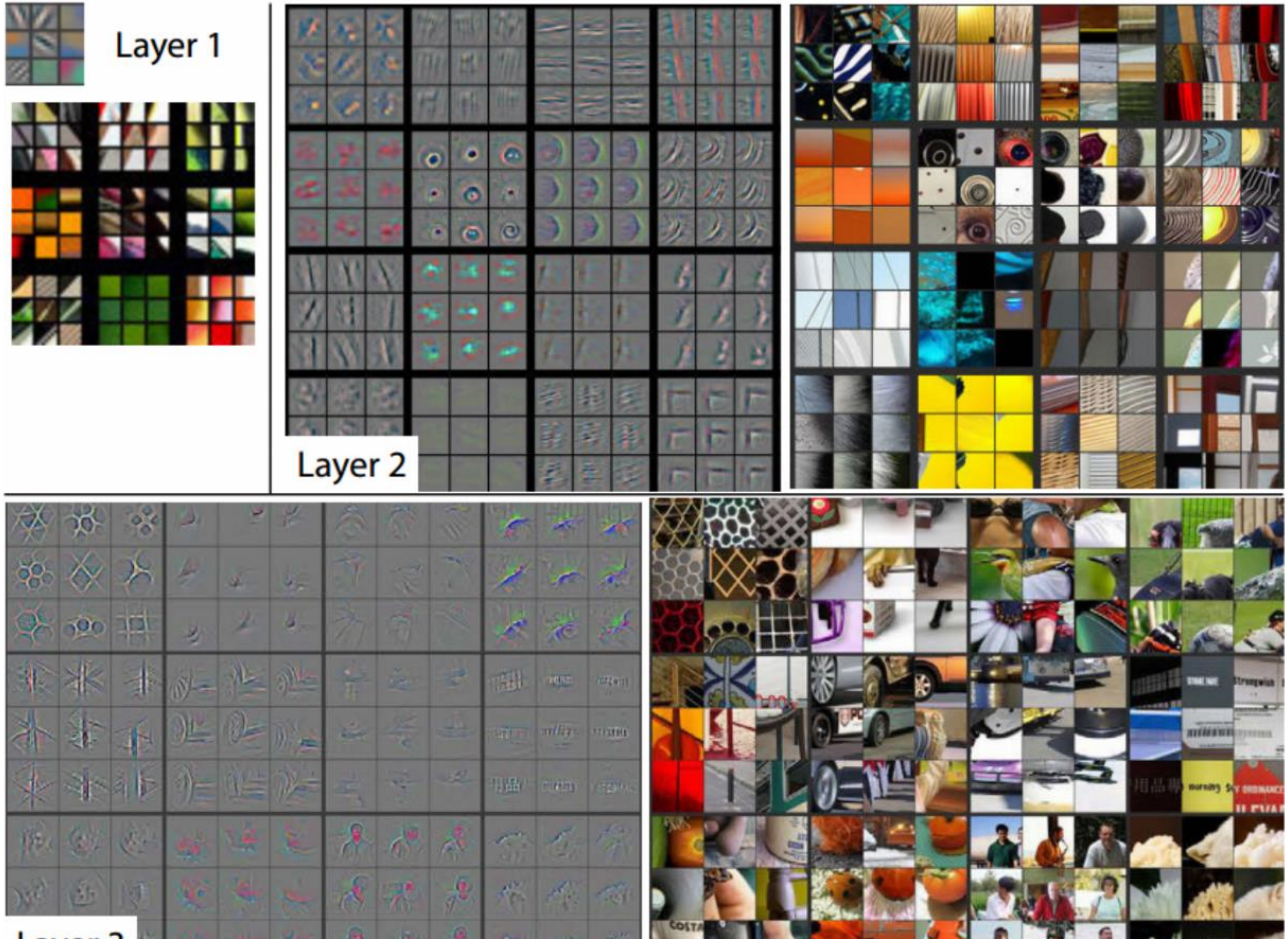
Visualizing CNN

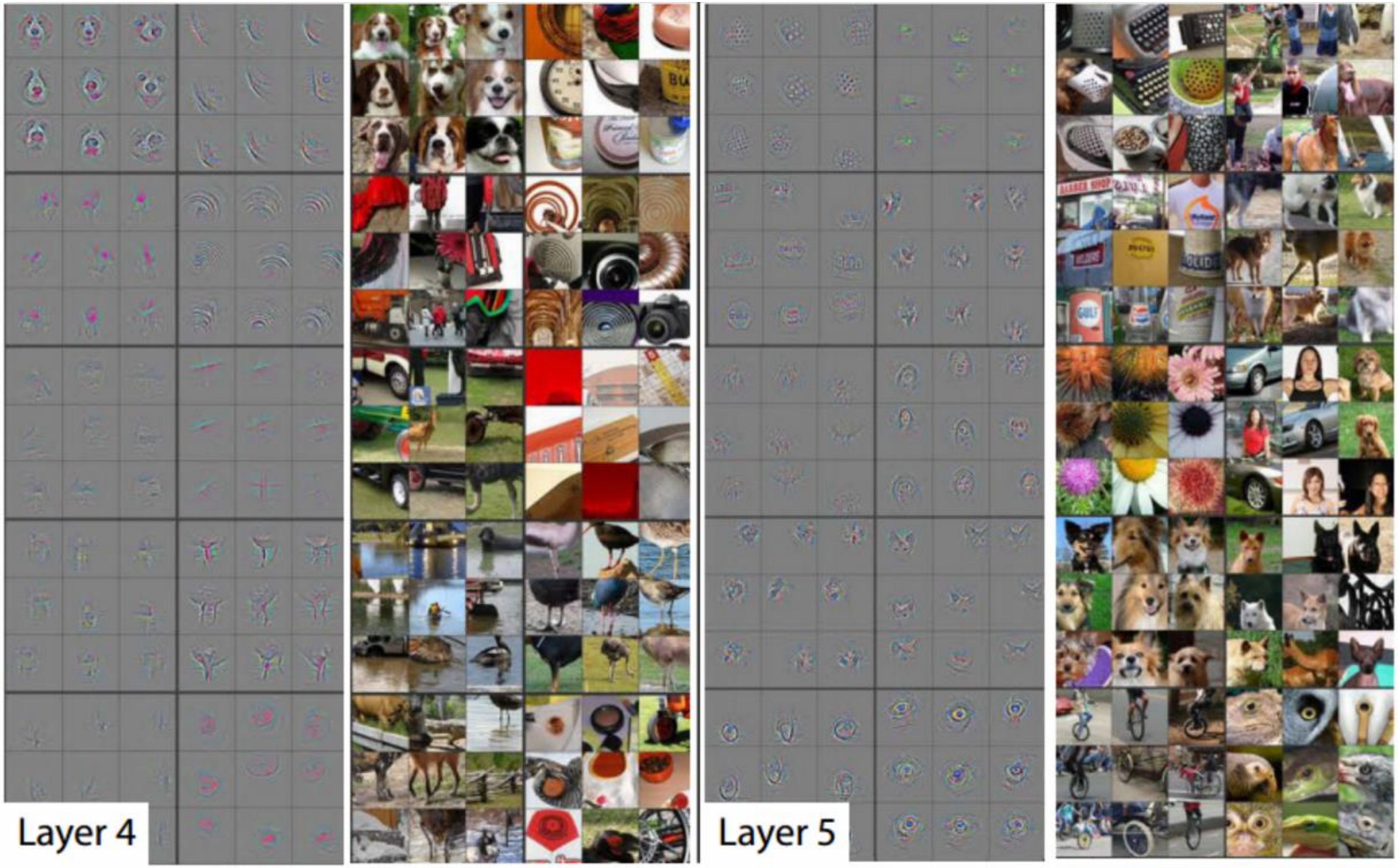
Deconv Net and Visualizing CNN [Matthew D. Zeiler and Rob Fergus]





Try to figure this by yourself!


Layer 1
Layer 2

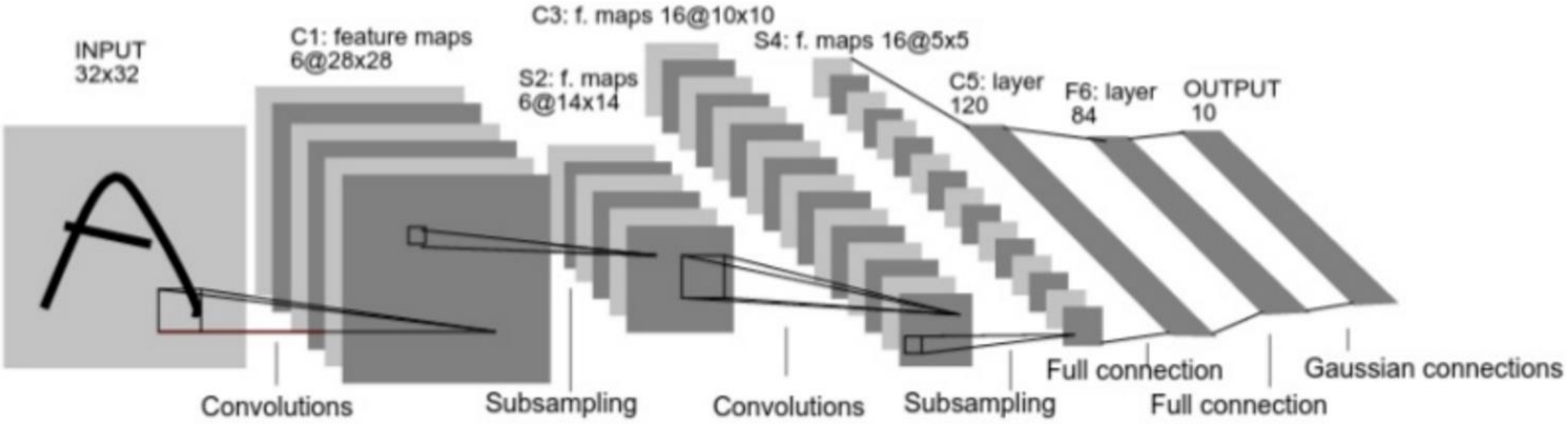

Layer 4
Layer 5

T-SNE [van der Maaten, Hinton]

- t-distributed stochastic neighbor embedding
  - A nonlinear dimension reduction
- Think the CNN code of an image as its feature vector (highly nonlinear features)
- Two images are closer if their CNN codes are closer in the feature space

Some popular CNN architectures

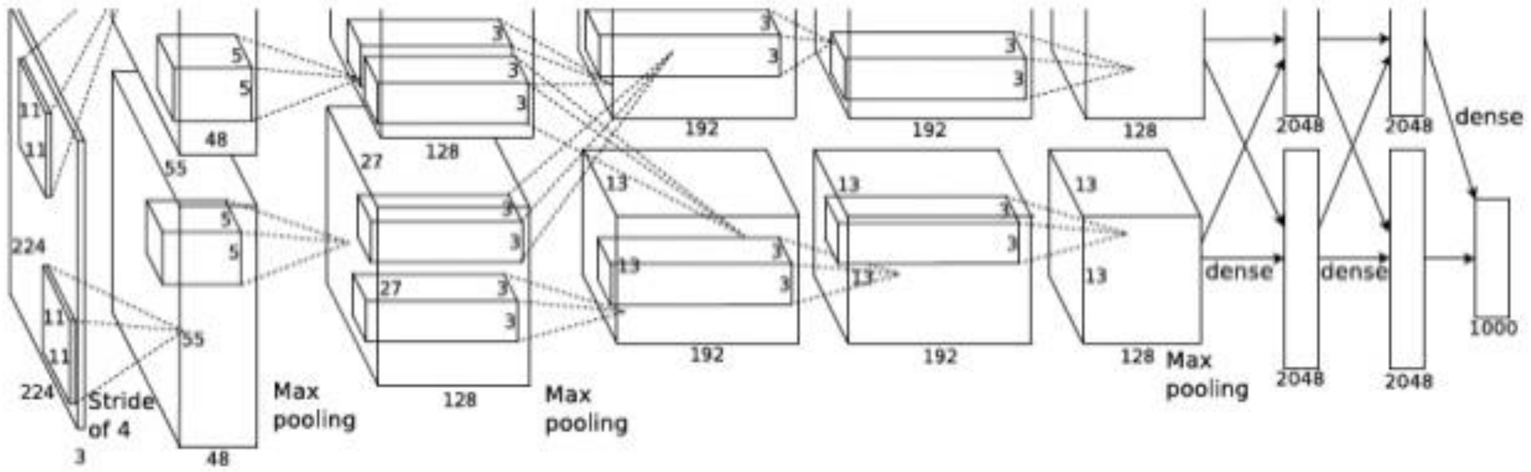
LeNet (Lecun-98)



Lenet-5 (Lecun-98), Convolutional Neural Network for digits recognition

Alexnet

- Similar framework to LeCun'98 but:
  - Bigger model (7 hidden layers, 650,000 units, 60,000,000 params)
  - More data (10^6 vs. 10^3 images)
  - GPU implementation (50x speedup over CPU)
    - Trained on two GPUs for a week



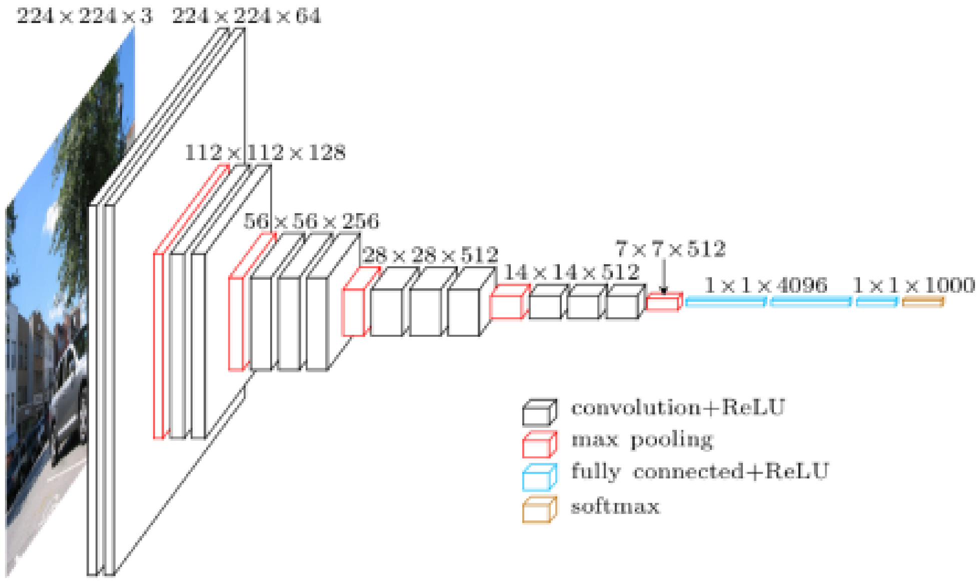
A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

- https://github.com/BVLC/caffe/tree/master/models/bvlc_alexnet

VGG Net [Simonyan, Zisserman]

<table>
<tr><th rowspan="2">Model</th><th colspan="2">top-5 classification error on ILSVRC-2012 (%)</th></tr>
<tr><th>validation set</th><th>test set</th></tr>
<tr><td>16-layer</td><td>7.5%</td><td>7.4%</td></tr>
<tr><td>19-layer</td><td>7.5%</td><td>7.3%</td></tr>
<tr><td>model fusion</td><td>7.1%</td><td>7.0%</td></tr>
</table>

Top-5 error in ImageNet (1000 classes)



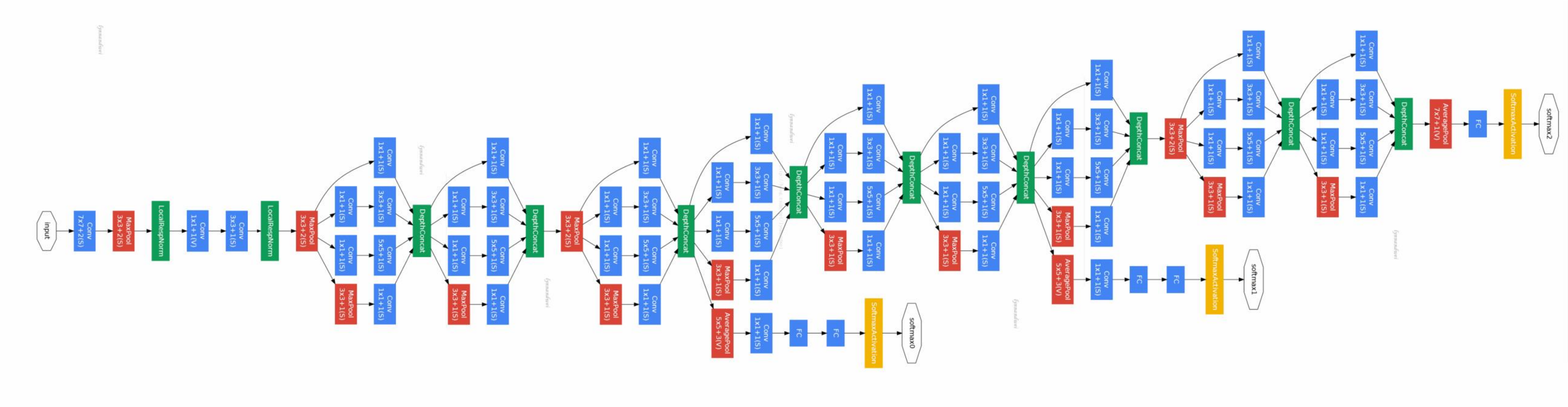
VGG16

Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

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					

- Implemented in Caffe
- You can download the weight from http://www.robots.ox.ac.uk/~vgg/research/very_deep/
- In Tensorflow: https://www.cs.toronto.edu/~frossard/post/vgg16/

GoogleNet [Szegedy et al.]

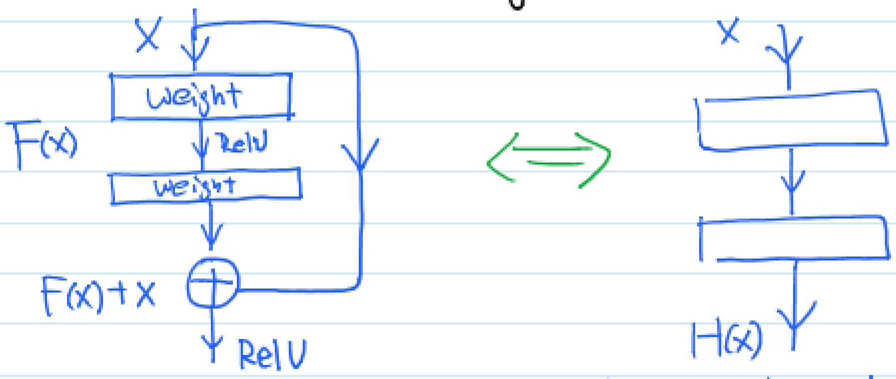


- https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet

ResNet [He et al.]

Stack many plain layers may even increase the training error.

ResNet

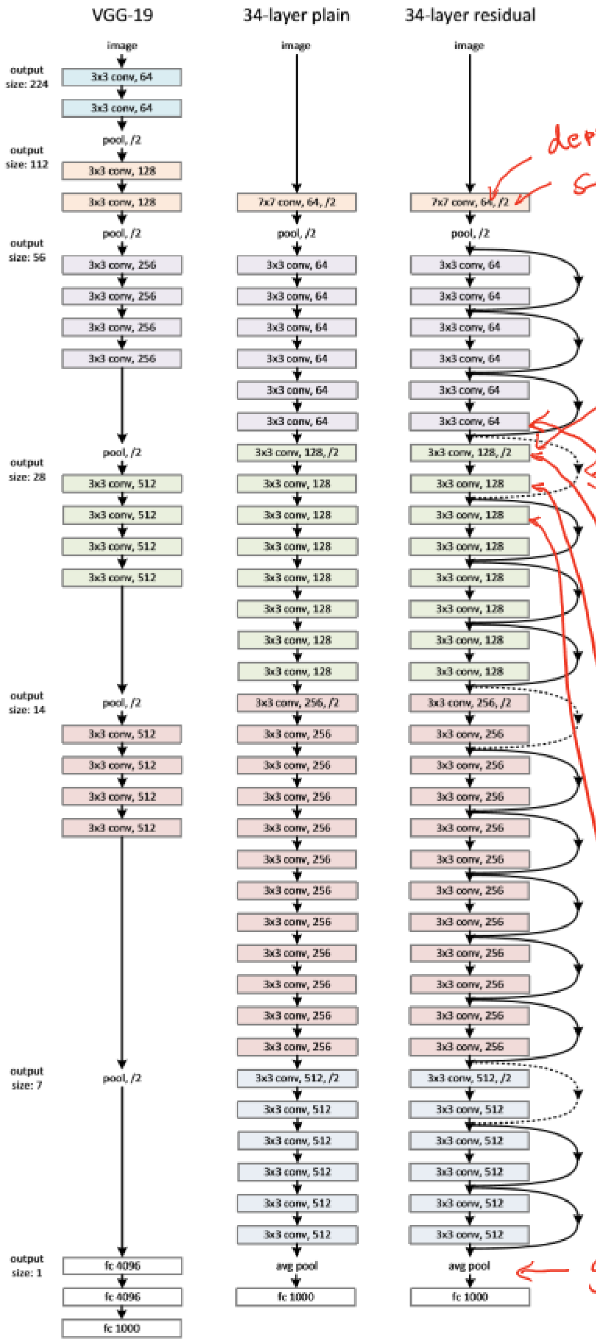


(we hypothesis $H(x)$ is closer to x)

More generally $y = F(x, \{W_i\}) + x$

F can be a general function. e.g. $F = W_2 \sigma(W_1 x)$ in above

ResNet

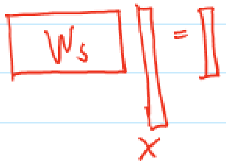


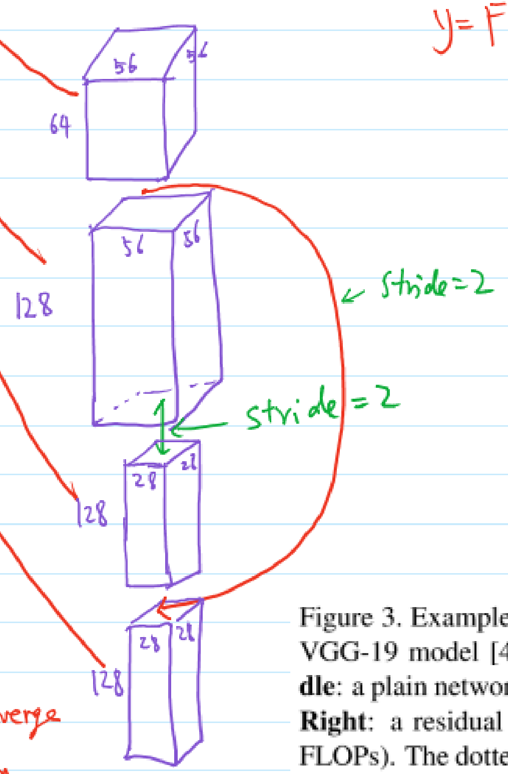
depth
stride.

stride=2 . output size halves

(A) The shortcut still performs identity mapping, with extra zero entries padded for increasing dimensions. This option introduces no extra parameter; (B) The projection shortcut in Eqn.(2) is used to match dimensions (done by 1×1 convolutions). For both options, when the shortcuts go across feature maps of two sizes, they are performed with a stride of 2.

$y = F(x, \{W_i\}) + W_s x$





stride=2

stride=2

global averge pooling

Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

ResNet

- https://github.com/KaimingHe/deep-residual-networks

- A later improved model has 1000 layers

Fractal Net [Larsson et al.]

- The network is defined recursively

$$f_1(z) = \mathrm{conv}(z)$$

$$f_{C+1}(z) = [(f_C \circ f_C)(z)] \oplus [\mathrm{conv}(z)]$$

$\circ$ denotes composition and $\oplus$ a join operation

- Instead of adding shortcut, FracNet provides a combination of short and long paths
  - neural information processing pathway

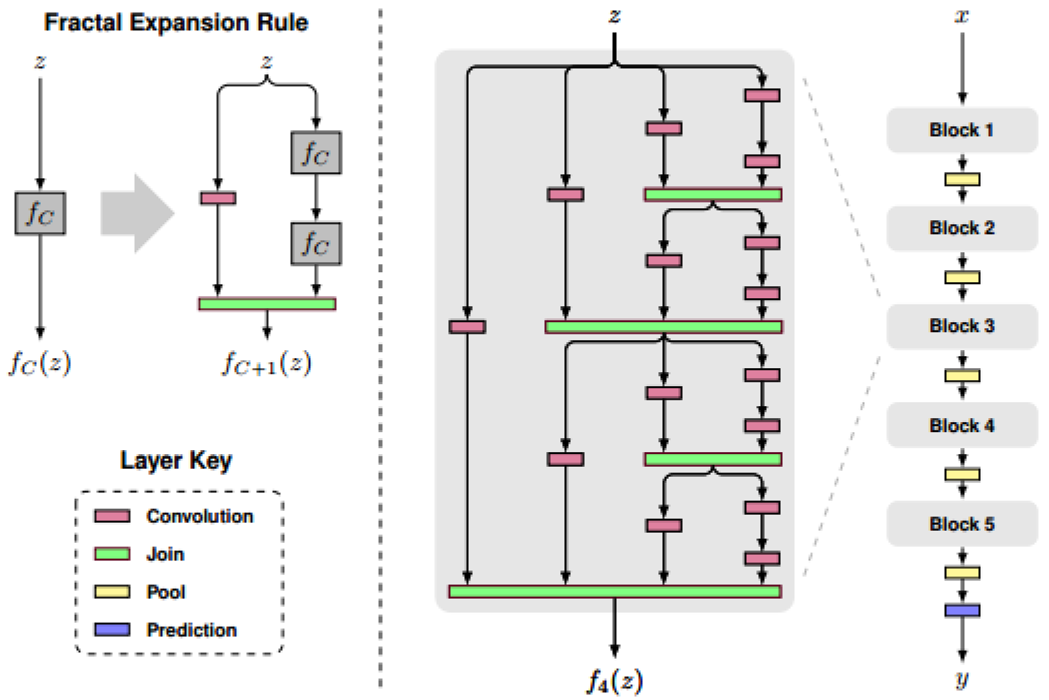


Figure 1: **Fractal architecture.** *Left:* A simple expansion rule generates a fractal architecture with C intertwined columns. The base case, $f_1(z)$, has a single layer of the chosen type (*e.g.* convolutional) between input and output. Join layers compute element-wise mean. *Right:* Deep convolutional networks periodically reduce spatial resolution via pooling. A fractal version uses f_C as a building block between pooling layers. Stacking B such blocks yields a network whose total depth, measured in terms of convolution layers, is $B \cdot 2^{C-1}$. This example has depth 40 ($B = 5$, $C = 4$).

Fractal Net

- Drop-path: a generalization of dropout

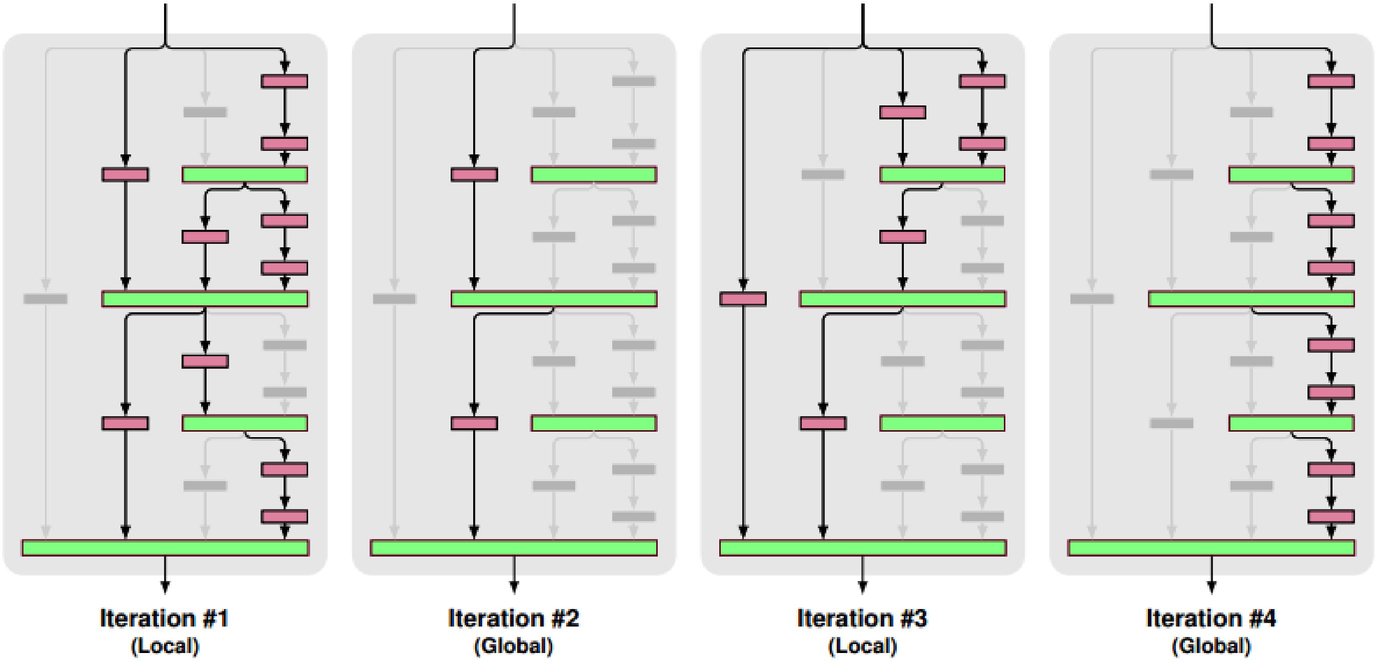


Figure 2: **Drop-path.** A fractal network block functions with some connections between layers disabled, provided some path from input to output is still available. Drop-path guarantees at least one such path, while sampling a subnetwork with many other paths disabled. During training, presenting a different active subnetwork to each mini-batch prevents co-adaptation of parallel paths. A global sampling strategy returns a single column as a subnetwork. Alternating it with local sampling encourages the development of individual columns as performant stand-alone subnetworks.

Performance

Method	C100	C100+	C100++	C10	C10+	C10++	SVHN
Network in Network [21]	35.68	-	-	10.41	8.81	-	2.35
Generalized Pooling [17]	32.37	-	-	7.62	6.05	-	1.69
Recurrent CNN [19]	31.75	-	-	8.69	7.09	-	1.77
Competitive Multi-scale [20]	27.56	-	-	6.87	-	-	1.76
FitNet [27]	-	35.04	-	-	8.39	-	2.42
Deeply Supervised [18]	-	34.57	-	9.69	7.97	-	1.92
All-CNN [30]	-	33.71	-	9.08	7.25	4.41	-
Highway Network [31]	-	32.39	-	-	7.72	-	-
ELU [2]	-	24.28	-	-	6.55	-	-
Scalable BO [29]	-	-	27.04	-	-	6.37	1.77
Fractional Max-Pooling [5]	-	-	26.32	-	-	3.47	-
FitResNet (LSUV) [23]	-	27.66	-	-	5.84	-	-
ResNet [8]	-	-	-	-	6.61	-	-
ResNet (reported by [11])	44.76	27.22	-	13.63	6.41	-	2.01
ResNet: Stochastic Depth [11]	37.80	24.58	-	11.66	5.23	-	1.75
ResNet: Identity Mapping [9]	-	22.68	-	-	4.69	-	-
ResNet in ResNet [33]	-	22.90	-	-	5.01	-	-
FractalNet	35.34	23.30	22.85	10.18	5.22	5.11	2.01
FractalNet+dropout/drop-path	28.20	23.73	23.36	7.33	4.60	4.59	1.87
↳ Deepest column alone	29.05	24.32	23.60	7.27	4.68	4.63	1.89

Table 1: **CIFAR-100/CIFAR-10/SVHN.** We compare test error (%) with other leading methods, trained with either no data augmentation, translation/mirroring (+), or more substantial augmentation (++). Our main point of comparison is ResNet. We closely match its state-of-the-art results using data augmentation, and outperform it by large margins without data augmentation. Training with drop-path, we can extract from FractalNet simple single-column networks that are highly competitive.

Stochastic Depth [Huang et al.]

- Very deep residual network: very hard and very slow to train
- Idea: randomly drop a subset of layers (treating them as Identity) (for each mini-batch)
- Allow one to go beyond 1200 layers

p_l:survival probabilities

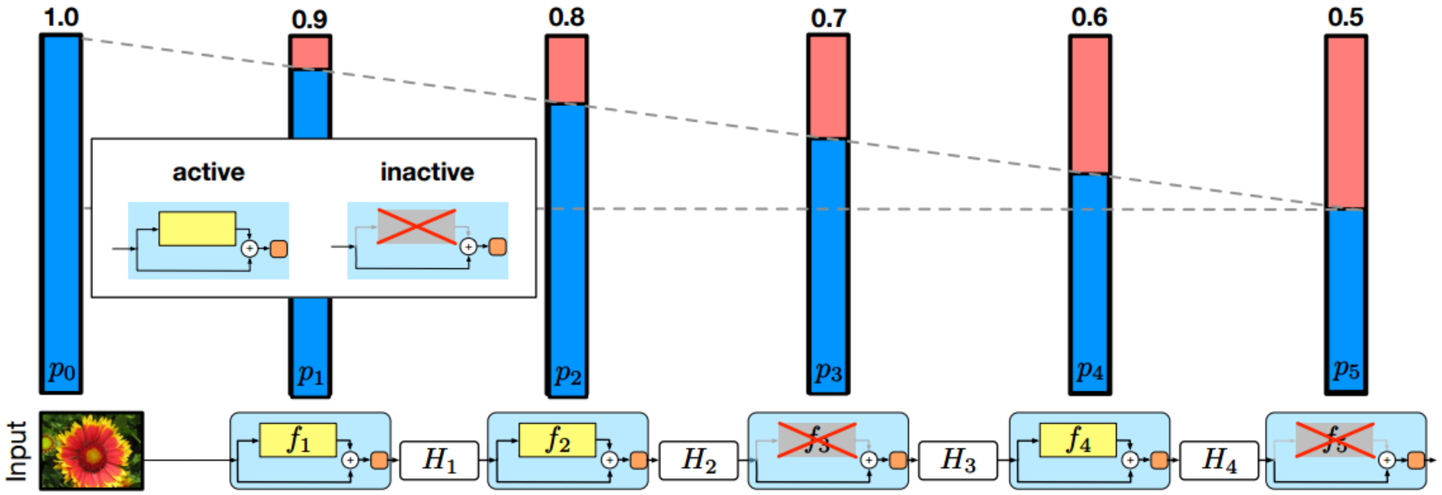


Fig. 2. The linear decay of p_ℓ illustrated on a ResNet with stochastic depth for $p_0 = 1$ and $p_L = 0.5$. Conceptually, we treat the input to the first ResBlock as H_0, which is always active.

Stochastic Depth

- https://github.com/yueatsprograms/Stochastic_Depth

Table 1. Test error (%) of ResNets trained with stochastic depth compared to other most competitive methods previously published (whenever available). A "+" in the name denotes standard data augmentation. ResNet with constant depth refers to our reproduction of the experiments by He et al.

	CIFAR10+	CIFAR100+	SVHN	ImageNet
Maxout [21]	9.38	-	2.47	-
DropConnect [20]	9.32	-	1.94	-
Net in Net [24]	8.81	-	2.35	-
Deeply Supervised [13]	7.97	-	1.92	33.70
Frac. Pool [25]	-	27.62	-	-
All-CNN [6]	7.25	-	-	41.20
Learning Activation [26]	7.51	30.83	-	-
R-CNN [27]	7.09	-	1.77	-
Scalable BO [28]	6.37	27.40	1.77	-
Highway Network [29]	7.60	32.24	-	-
Gen. Pool [30]	6.05	-	1.69	28.02
ResNet with constant depth	6.41	27.76	1.80	21.78
ResNet with stochastic depth	5.25	24.98	1.75	21.98

Table 2. Training time comparison on benchmark datasets.

	CIFAR10+	CIFAR100+	SVHN
Constant Depth	20h 42m	20h 51m	33h 43m
Stochastic Depth	15h 7m	15h 20m	25h 33m

Applications

Image Reconstruction [Mahendran, Vedaldi 2014]

Find an image such that:
- Its code is similar to a given code
- It "looks natural" (image prior regularization)

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x} \in \mathbb{R}^{H \times W \times C}} \ell(\Phi(\mathbf{x}), \Phi_0) + \lambda \mathcal{R}(\mathbf{x})$$

$$\ell(\Phi(\mathbf{x}), \Phi_0) = \|\Phi(\mathbf{x}) - \Phi_0\|^2$$

$R(x)$: regularizer to encourge "natural image"

$$R(x) = \|x\|_\alpha^\alpha \quad (\text{eg. } \alpha = 6)$$

$$R_{TV}(x) = \sum_{ij} \left((x_{i+1,j} - x_{ij})^2 + (x_{i,j+1} - x_{ij})^2 \right)^{\beta/2}$$

Image Reconstruction

Understanding Deep Image Representations by Inverting Them [Mahendran and Vedaldi, 2014]

original image

reconstructions from the 1000 log probabilities for ImageNet (ILSVRC) classes

Image Reconstruction

Reconstructions from the representation after last last pooling layer
(immediately before the first Fully Connected layer)

Image Reconstruction

Reconstructions from intermediate layers

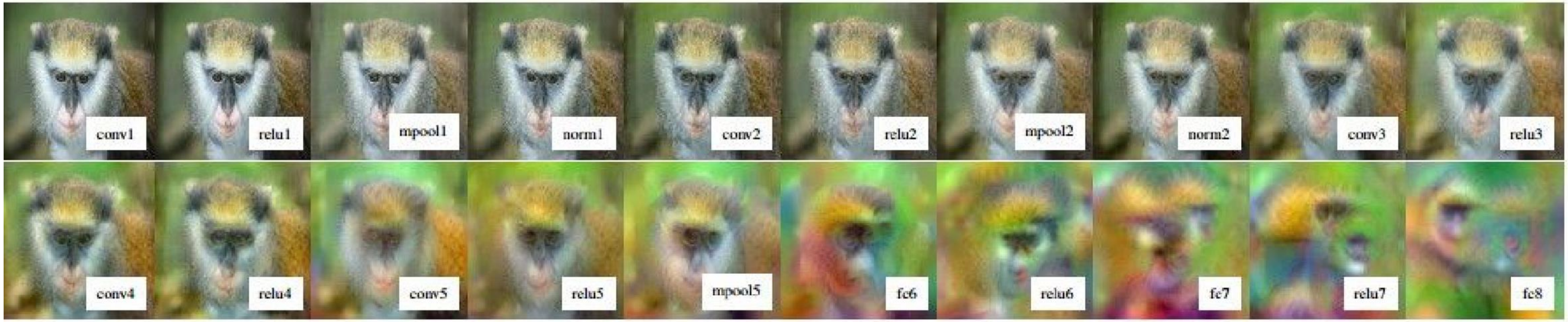


Image Reconstruction

- https://github.com/aravindhm/deep-goggle

Deep Dream

inception_4c/output

Deep Dream

IDEA: if a neuron is activated, activate it further!

we don't have a loss function

DeepDream: set dx = x :)

a layer in googlenet

```
def objective_L2(dst):
    dst.diff[:] = dst.data

def make_step(net, step_size=1.5, end='inception_4c/output',
              jitter=32, clip=True, objective=objective_L2):
    '''Basic gradient ascent step.'''

    src = net.blobs['data'] # input image is stored in Net's 'data' blob
    dst = net.blobs[end]

    ox, oy = np.random.randint(-jitter, jitter+1, 2)
    src.data[0] = np.roll(np.roll(src.data[0], ox, -1), oy, -2) # apply jitter shift

    net.forward(end=end)
    objective(dst)  # specify the optimization objective
    net.backward(start=end)
    g = src.diff[0]
    # apply normalized ascent step to the input image
    src.data[:] += step_size/np.abs(g).mean() * g

    src.data[0] = np.roll(np.roll(src.data[0], -ox, -1), -oy, -2) # unshift image

    if clip:
        bias = net.transformer.mean['data']
        src.data[:] = np.clip(src.data, -bias, 255-bias)
```

← a forward computation (from 'data' to 'end'), we get activation values at 'end'

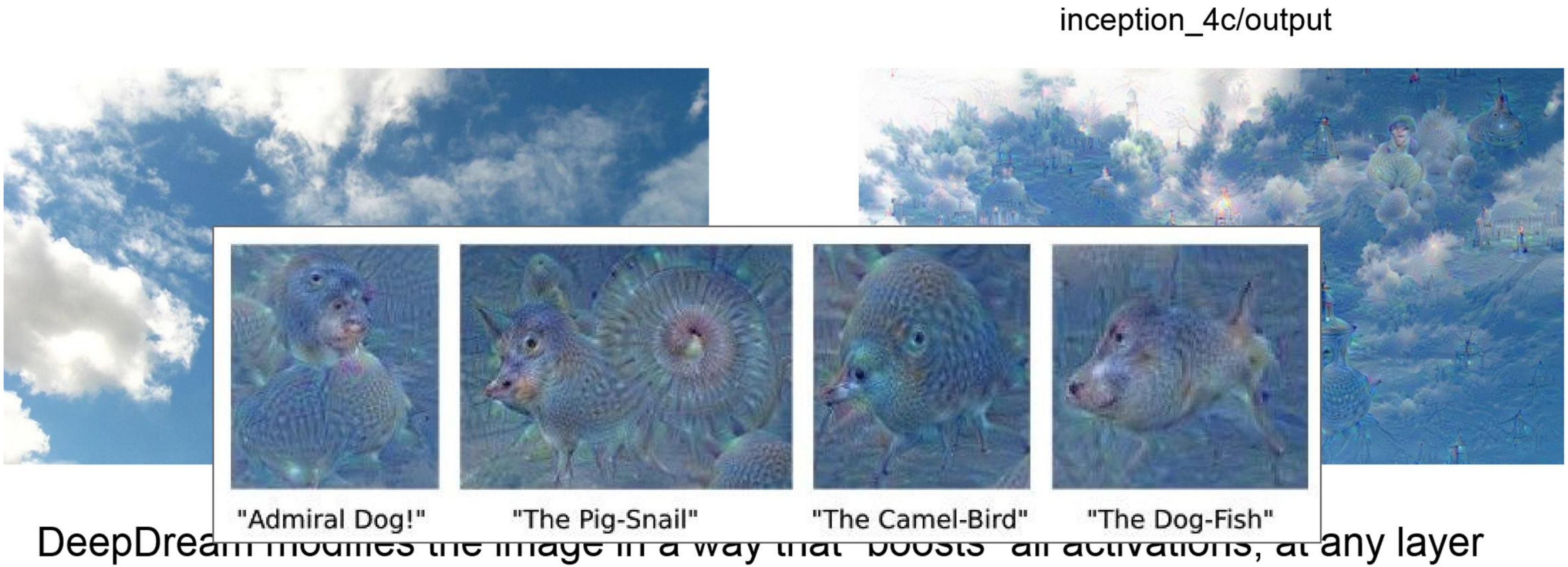
← backward computation, starting from 'end'

jitter regularizer

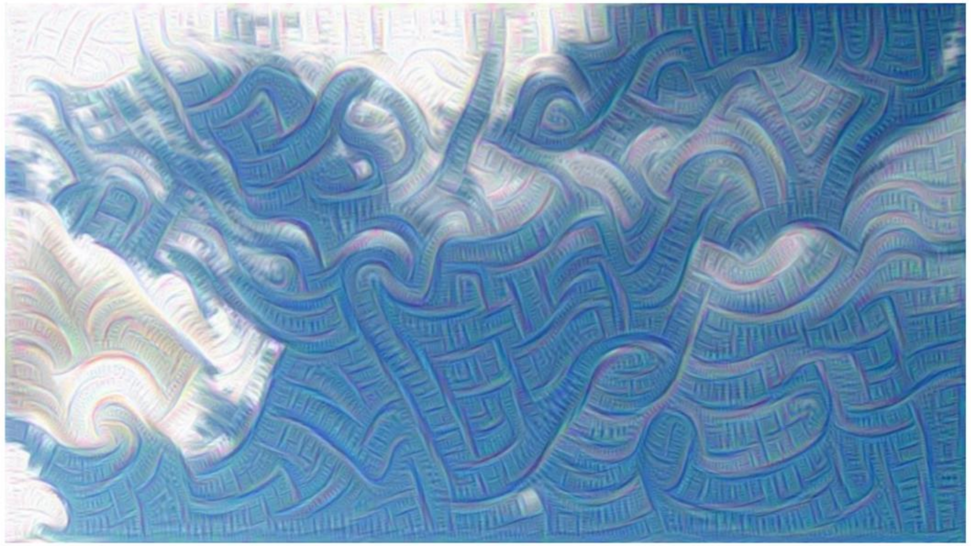
only marginally useful

"image update"

Deep Dream


inception_4c/output
"Admiral Dog!"
"The Pig-Snail"
"The Camel-Bird"
"The Dog-Fish"

Deep Dream



DeepDream modifies the image in a way that “boosts” all activations, at any layer

Deep Dream

Inceptionism!

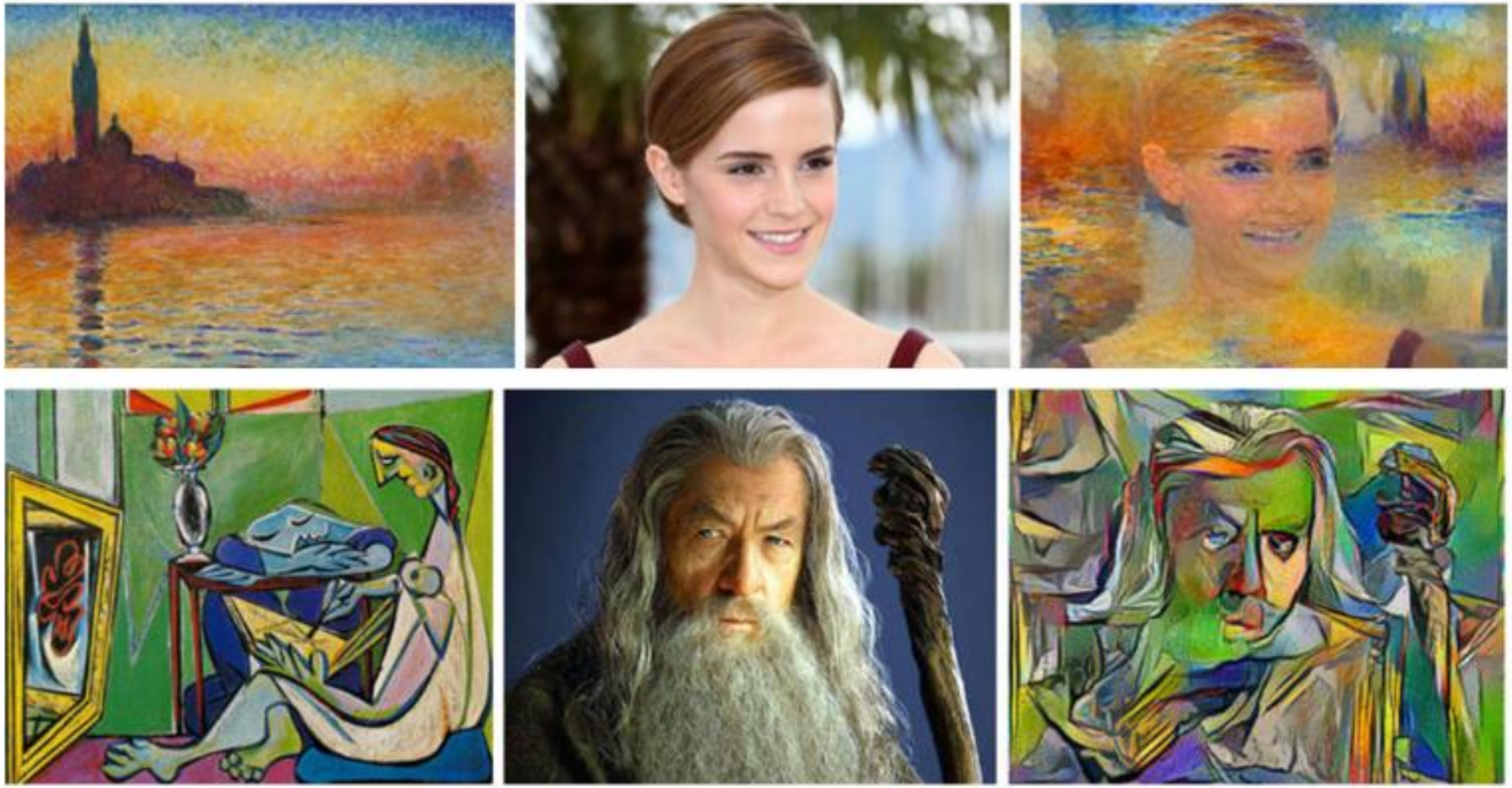


Deep Dream

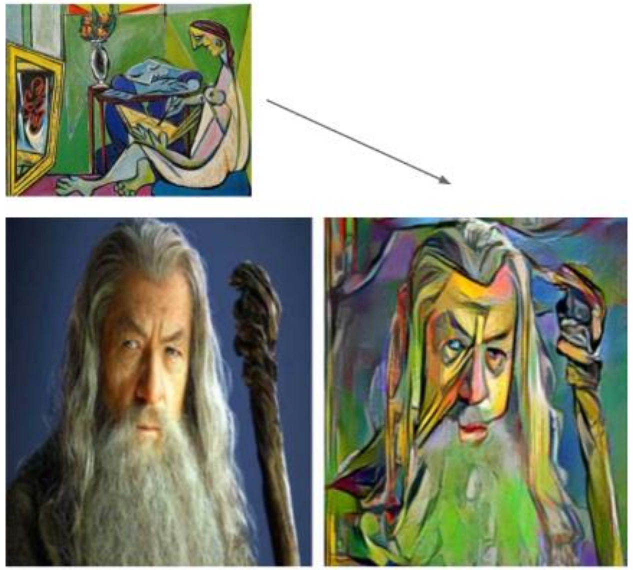
- https://github.com/google/deepdream

- http://www.pyimagesearch.com/2015/07/06/bat-country-an-extendible-lightweight-python-package-for-deep-dreaming-with-caffe-and-convolutional-neural-networks/#show_and_tell

Neuralstyle [Gatys et al. 2015]



Neuralstyle

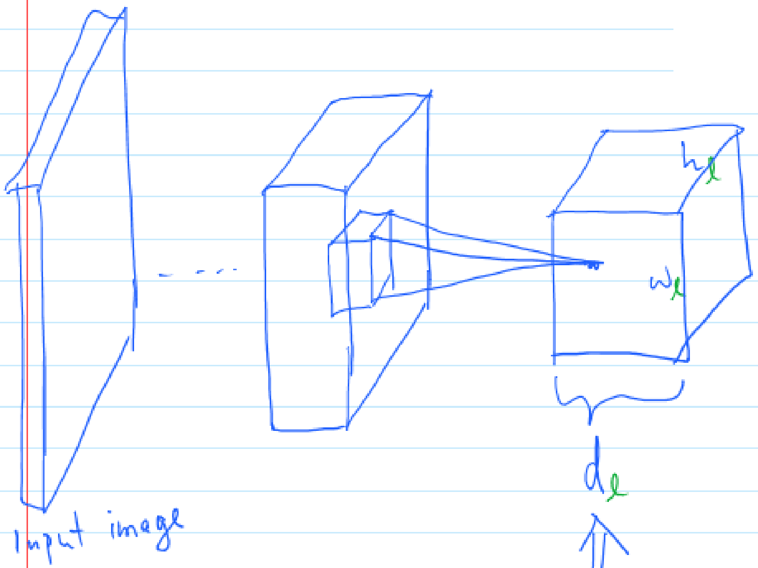


① try to match the content from the original figure.

② try to match the $\boxed{\text{style}}$ from the art work

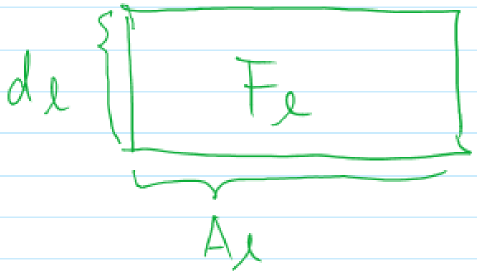
Correlation of filter response

Neuralstyle

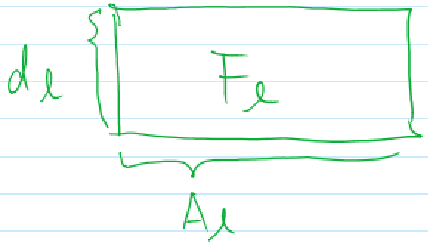


$A_\ell = w_\ell \times h_\ell$

Layer. ℓ all response can be stored in tensor $\mathbb{R}^{d_\ell \times w_\ell \times h_\ell}$. flatten it into $F_\ell \in \mathbb{R}^{d_\ell \times A_\ell}$



Neuralstyle



(find an image)

matching the content.

$$\text{loss: } \mathcal{L}_{content}(\vec{p}, \vec{x}, \ell) = \frac{1}{2}\sum_{i,j}\left(F^{\ell}_{ij} - P^{\ell}_{ij}\right)^2$$

$\vec{p}$: given input image

$\vec{x}$: we want to generate x

ℓ: layer ℓ

F^{ℓ}_{ij}: feature representation of $\vec{x}$ in layer ℓ

P^{ℓ}_{ij}: feature representation of $\vec{p}$ in layer ℓ

How to get x ?

initially $x \leftarrow$ white noise.

iterate GD

the network is fixed. but x is variable

so $\nabla_x \mathcal{L}$ is well-defined

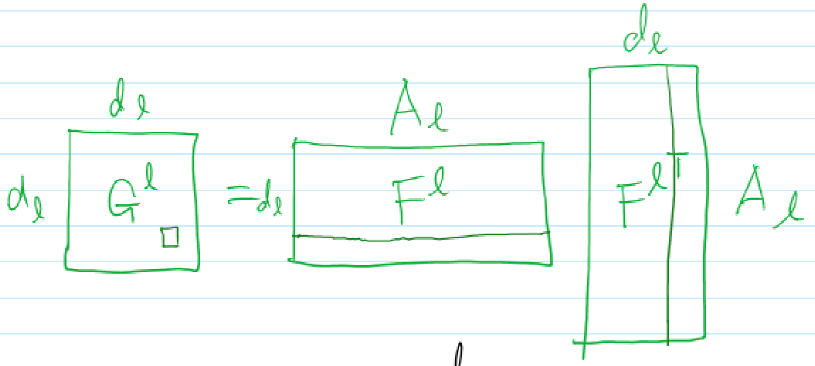
$$x_t \leftarrow x_{t-1} - \lambda_t \nabla_x \mathcal{L}$$

Neuralstyle

Matching the style.

Feature correlation

$G^l \in \mathbb{R}^{d_l \times d_l}$



Loss function: $\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l \left(\frac{1}{4 d_l^2 A_l^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2 \right)$

art work

we want to generate x

weight for layers

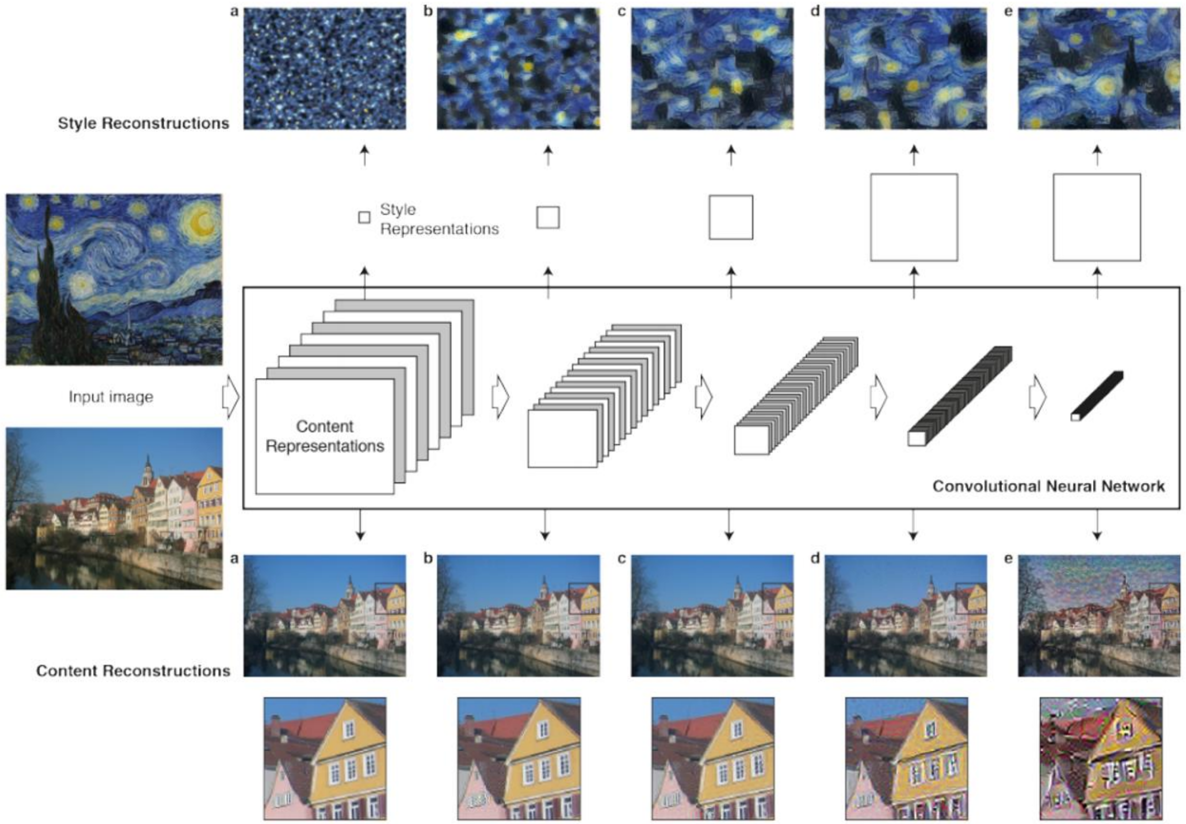
feature corr for $\vec{x}$

feature correlation for the art work

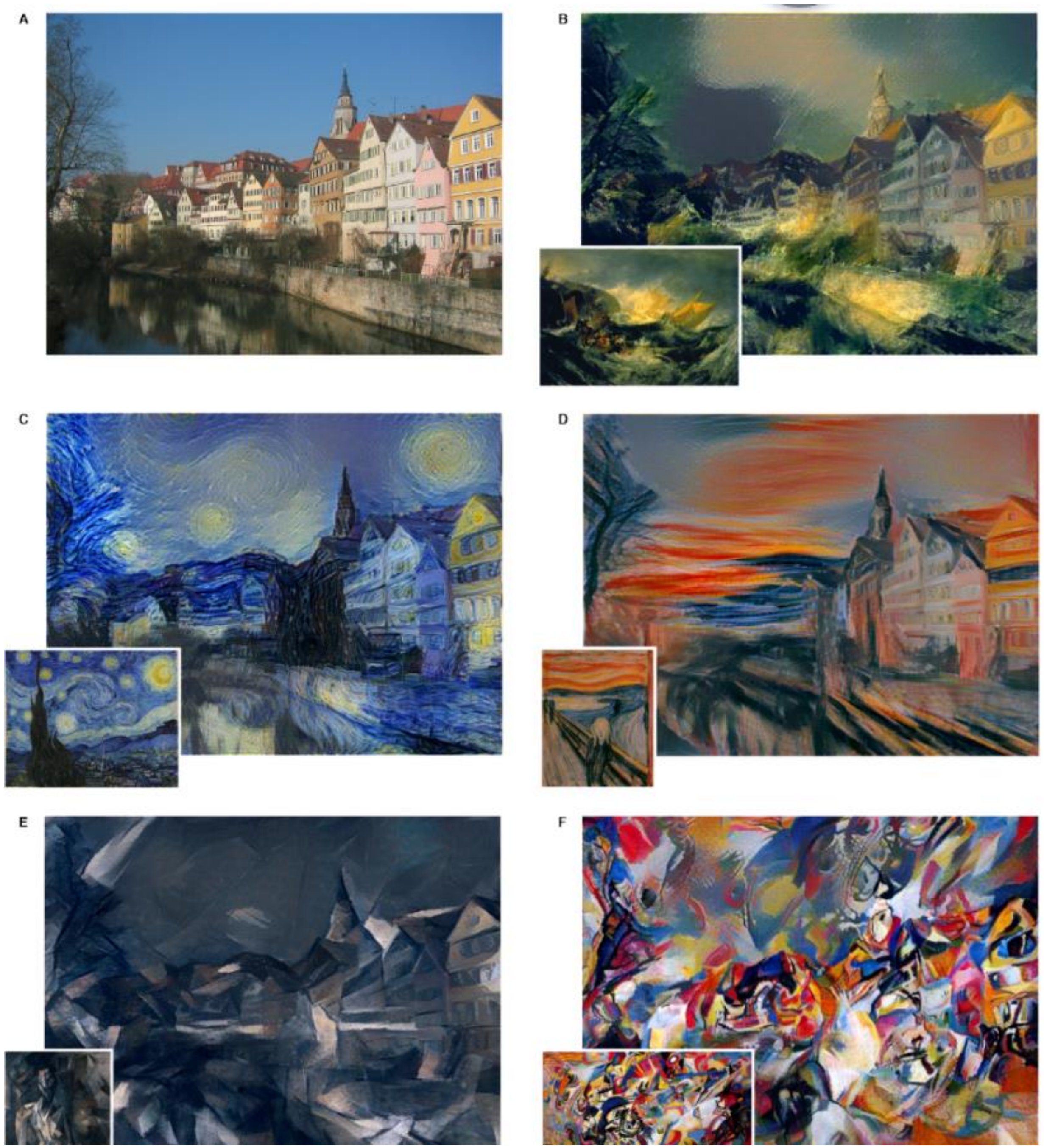
training is the same (start from white noise)

Overall loss: $\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$

Neuralstyle



Neuralstyle

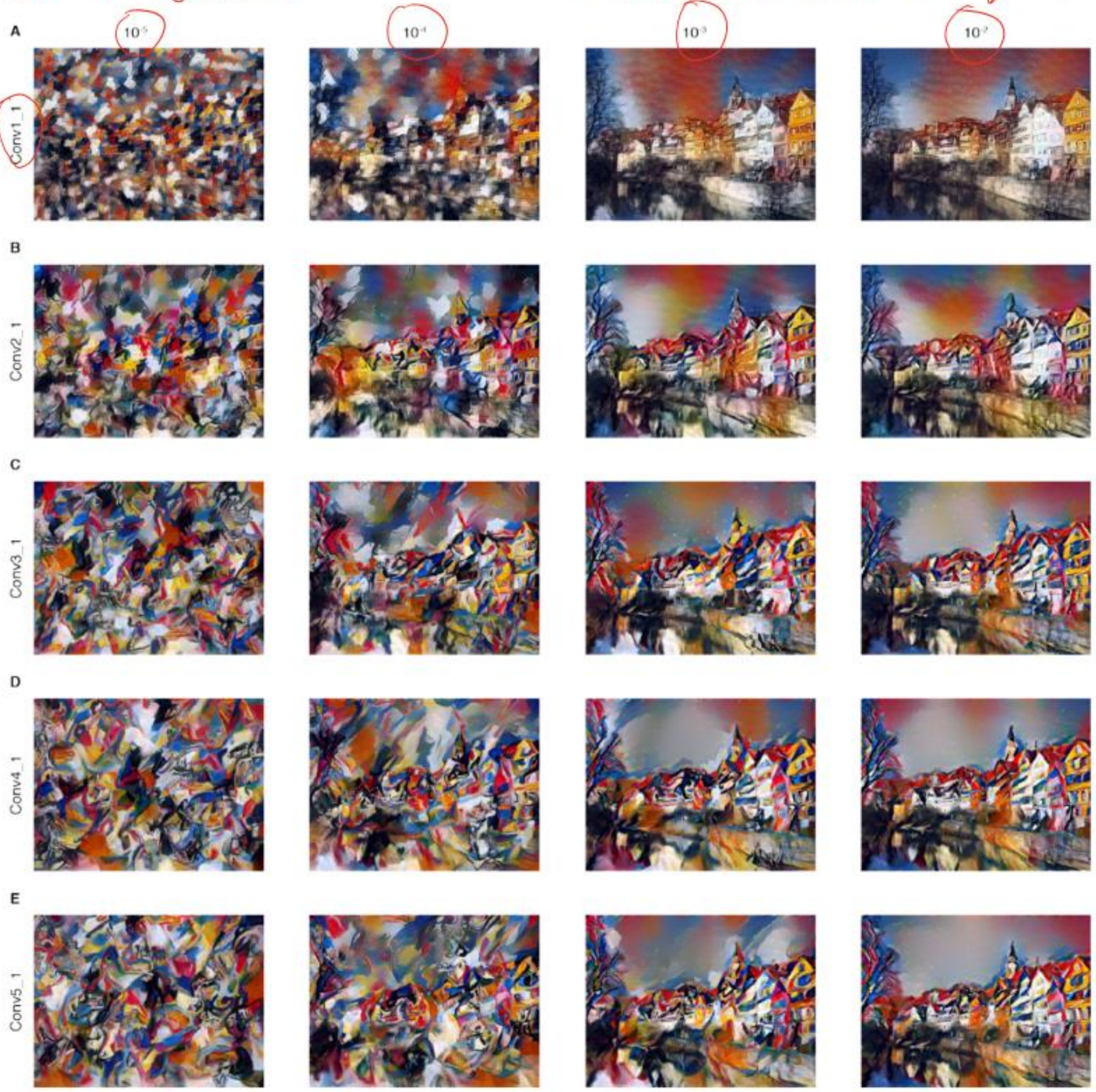


Neuralstyle

more focus on style ← → more focus on content

different α/β value

which style layer to match



↑ style more local

depending on size of receptive field

style more globle ↓

Neuralstyle

- In tensorflow:
  - https://github.com/anishathalye/neural-style

- Mxnet
  - https://github.com/dmlc/mxnet/tree/master/example/neural-style

- Some slides borrowed from Gaurav Mittal’s slides, Lawrence Carin’s slides, cs231n at Stanford