Introduction to Deep Learning
Module 4

Ganapathy Krishnamurthi

Organization

- Introduction
- CNN Operations
- CNN Training
- Illustrative Example (“Hello World”) - MNIST digit classification
- Image Recognition-SoTA model(s)

Introduction
The ImageNet Challenge

- The ImageNet Large Scale Visual Recognition Challenge- object category classification and detection of hundreds of object categories and millions of images - running since 2010

- The benchmark dataset was created to test algorithms for such vision tasks

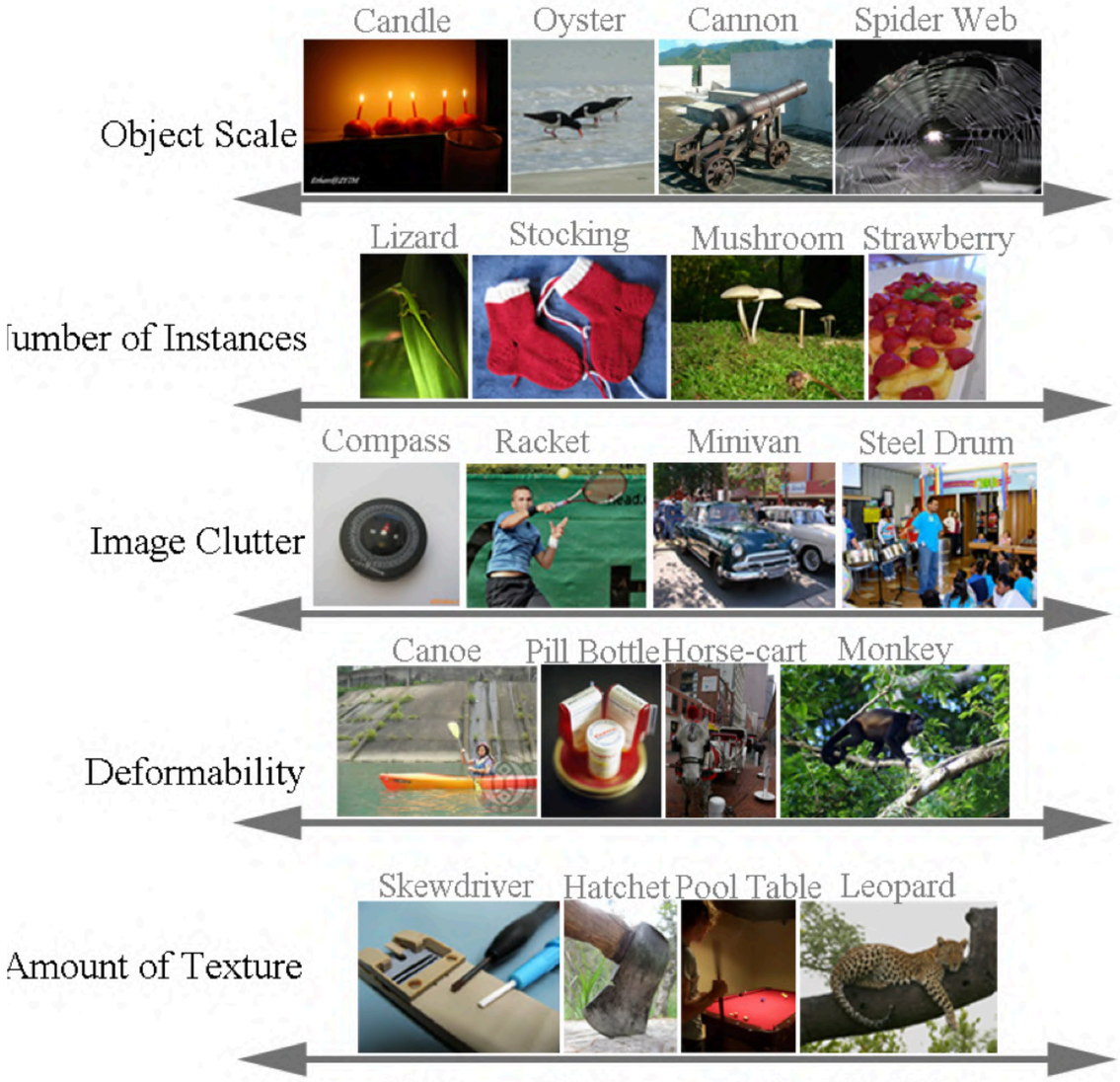
- Public dataset consists of manually annotated training images. A set of test images with annotations withheld is also available

https://arxiv.org/pdf/1409.0575.pdf

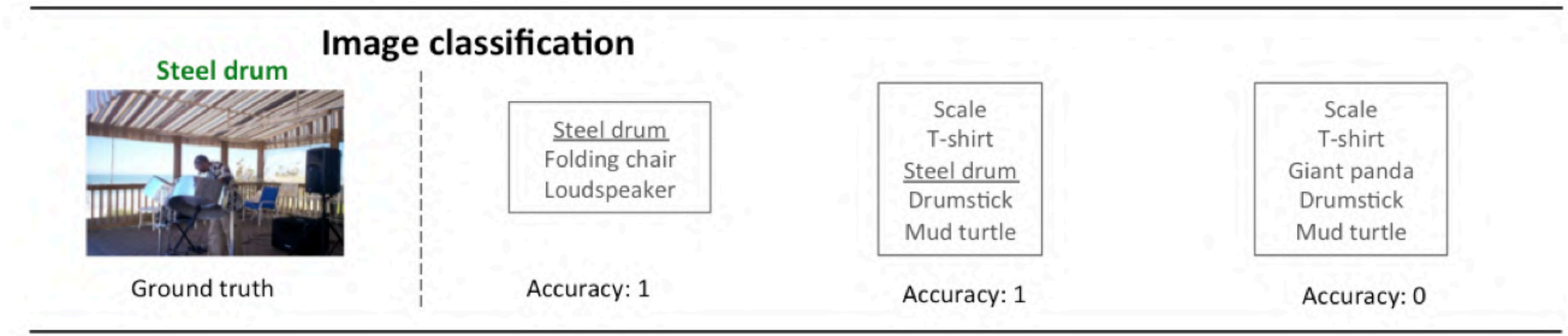
ImageNet Challenge

- Image classification/recognition Millions of images, 1000 categories labeled
- Single Object localisation task introduced in 2011- draw a tight bounding box around the object of interest i.e label
- Object detection - localizes multiple object categories in the same image
-

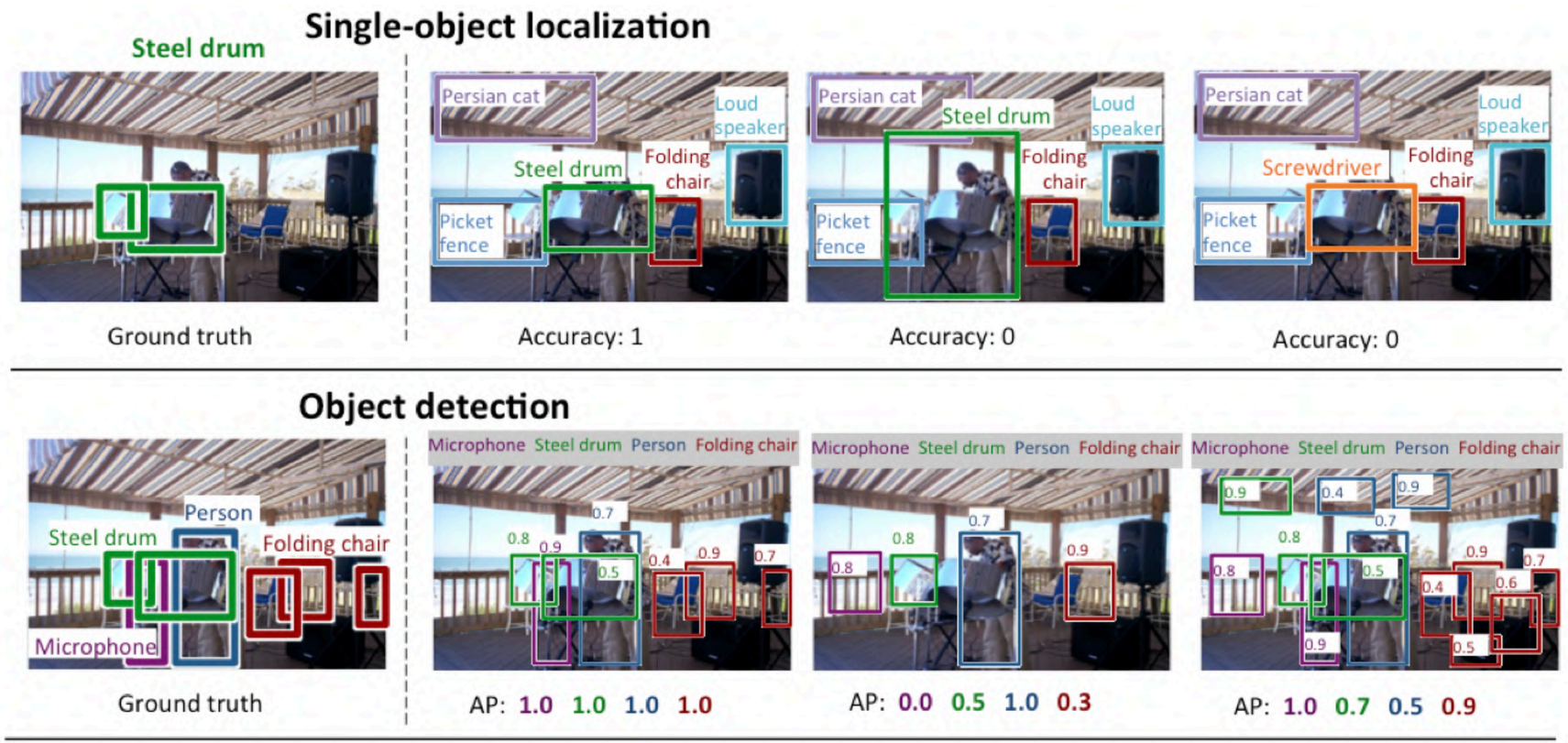
ImageNet challenge



ImageNet Challenge



ImageNet Challenge



ImageNet Challenge

Codename	CLS	Insitutions	Contributors and references
Hminmax	54.4	Massachusetts Institute of Technology	Jim Mutch, Sharat Chikkerur, Hristo Paskov, Ruslan Salakhutdinov, Stan Bileschi, Hueihan Jhuang
IBM	70.1	IBM research†, Georgia Tech‡	Lexing Xie†, Hua Ouyang‡, Apostol Natsev†
ISIL	44.6	Intelligent Systems and Informatics Lab., The University of Tokyo	Tatsuya Harada, Hideki Nakayama, Yoshitaka Ushiku, Yuya Yamashita, Jun Imura, Yasuo Kuniyoshi
ITNLP	78.7	Harbin Institute of Technology	Deyuan Zhang, Wenfeng Xuan, Xiaolong Wang, Bingquan Liu, Chengjie Sun
LIG	60.7	Laboratoire d'Informatique de Grenoble	Georges Quénot
NEC	**28.2**	NEC Labs America†, University of Illinois at Urbana-Champaign‡, Rutgers∓	Yuanqing Lin†, Fengjun Lv†, Shenghuo Zhu†, Ming Yang†, Timothee Cour†, Kai Yu†, LiangLiang Cao‡, Zhen Li‡, Min-Hsuan Tsai‡, Xi Zhou‡, Thomas Huang‡, Tong Zhang∓ (Lin et al., 2011)
NII	74.2	National Institute of Informatics, Tokyo,Japan†, Hefei Normal Univ. Heifei, China‡	Cai-Zhi Zhu†, Xiao Zhou‡, Shiníchi Satoh†
NTU	58.3	CeMNet, SCE, NTU, Singapore	Zhengxiang Wang, Liang-Tien Chia
Regularities	75.1	SRI International	Omid Madani, Brian Burns
UCI	46.6	University of California Irvine	Hamed Pirsiavash, Deva Ramanan, Charless Fowlkes
XRCE	33.6	Xerox Research Centre Europe	Jorge Sanchez, Florent Perronnin, Thomas Mensink (Perronnin et al., 2010)

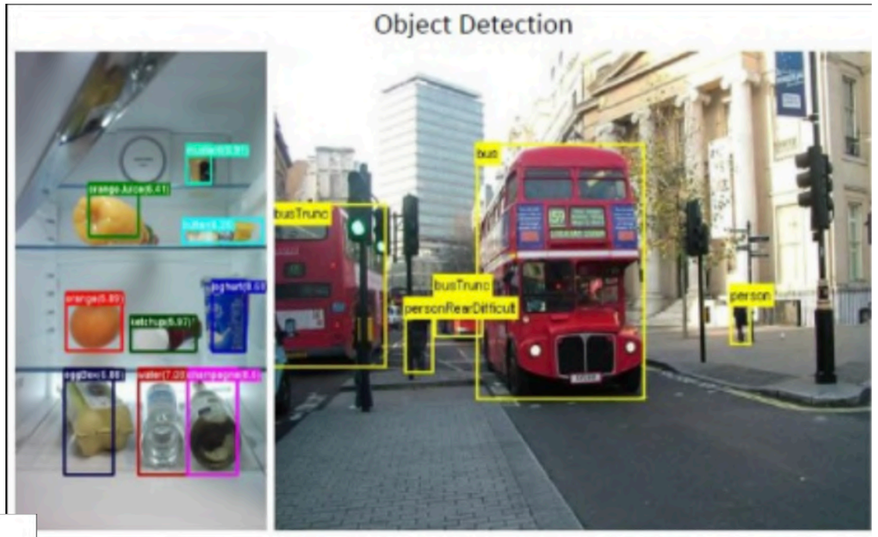
ILSVRC 2011

Codename	CLS	LOC	Institutions	Contributors and references
ISI	36.0	-	Intelligent Systems and Informatics lab, University of Tokyo	Tatsuya Harada, Asako Kanezaki, Yoshitaka Ushiku, Yuya Yamashita, Sho Inaba, Hiroshi Muraoka, Yasuo Kuniyoshi
NII	50.5	-	National Institute of Informatics, Japan	Duy-Dinh Le, Shiníchi Satoh
UvA	31.0	**42.5**	University of Amsterdam†, University of Trento‡	Koen E. A. van de Sande†, Jasper R. R. Uijlings‡, Arnold W. M. Smeulders†, Theo Gevers†, Nicu Sebe‡, Cees Snoek† (van de Sande et al., 2011b)
XRCE	**25.8**	56.5	Xerox Research Centre Europe†, CIII‡	Florent Perronnin†, Jorge Sanchez†‡ (Sanchez and Perronnin, 2011)

ILSVRC 2012

Codename	CLS	LOC	Institutions	Contributors and references
ISI	26.2	53.6	University of Tokyo†, JST PRESTO‡	Naoyuki Gunji†, Takayuki Higuchi†, Koki Yasumoto†, Hiroshi Muraoka†, Yoshitaka Ushiku†, Tatsuya Harada†‡, Yasuo Kuniyoshi† (Harada and Kuniyoshi, 2012)
LEAR	34.5	-	LEAR INRIA Grenoble†, TVPA Xerox Research Centre Europe‡	Thomas Mensink†‡, Jakob Verbeek†, Florent Perronnin‡, Gabriela Csurka‡ (Mensink et al., 2012)
VGG	27.0	50.0	University of Oxford	Karen Simonyan, Yusuf Aytar, Andrea Vedaldi, Andrew Zisserman (Arandjelovic and Zisserman, 2012; Sanchez et al., 2012)
SuperVision	**16.4**	**34.2**	University of Toronto	Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton (Krizhevsky et al., 2012)
UvA	29.6	-	University of Amsterdam	Koen E. A. van de Sande, Amir Habibian, Cees G. M. Snoek (Sanchez and Perronnin, 2011; Scheirer et al., 2012)
XRCE	27.1	-	Xerox Research Centre Europe†, LEAR INRIA ‡	Florent Perronnin†, Zeynep Akata†‡, Zaid Harchaoui‡, Cordelia Schmid‡ (Perronnin et al., 2012)

CNN-Applications



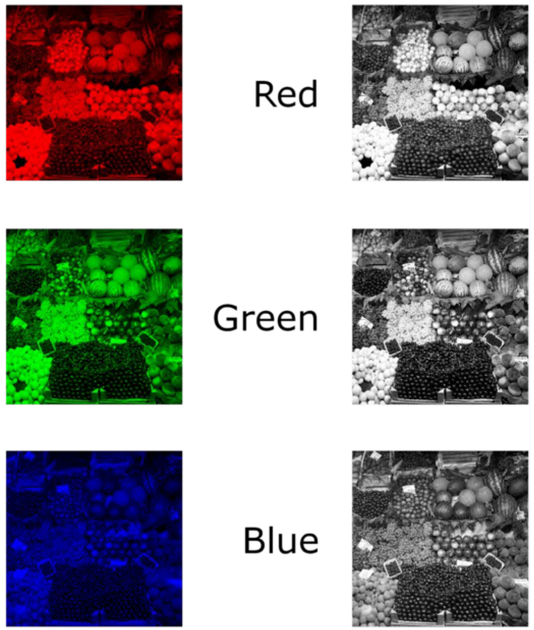
https://github.com/Microsoft/CNTK/wiki/Object-Detection-using-Fast-R-CNN

Introduction

- Analysing images using fully connected ANNs can be computationally expensive
- CNNs are a type of feed forward network that is used for analysing structured data, specifically grid data in 1D, 2D or 3D

Introduction
Image Parametrization

- RGB images



Grayscale Image

© Nevit Dilmen [CC BY-SA 3.0 (https://creativecommons.org/licenses/by-sa/3.0) or GFDL (http://www.gnu.org/copyleft/fdl.html)], via Wikimedia Commons

CNNs- From Fully connected to convolutional Layers

- A typically fully connected feed forward neural network- All Neurons in previous layer are connected to a Neuron in the current layer

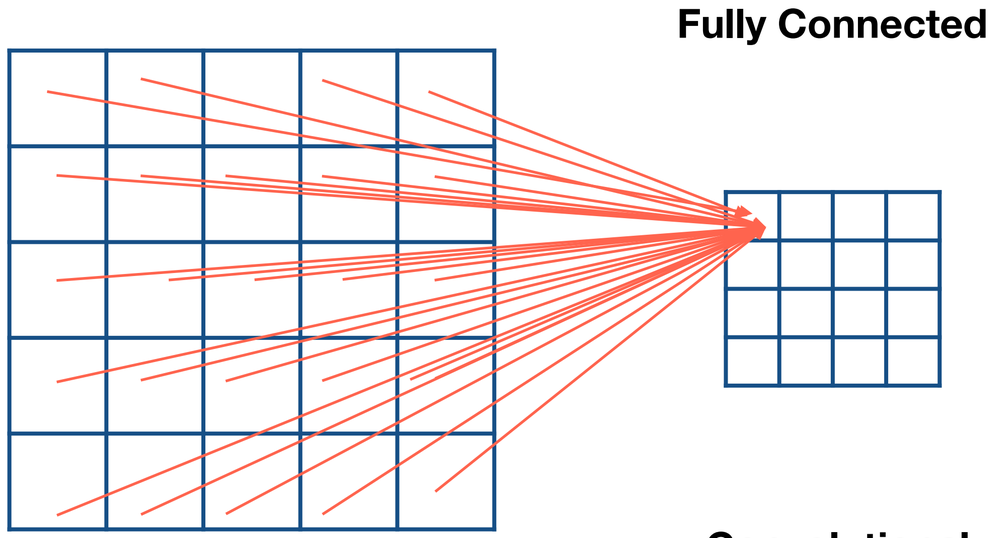
- In a CNN, only a subset of neurons in the previous layer connect to a Neuron in the current layer- Sparse Connections

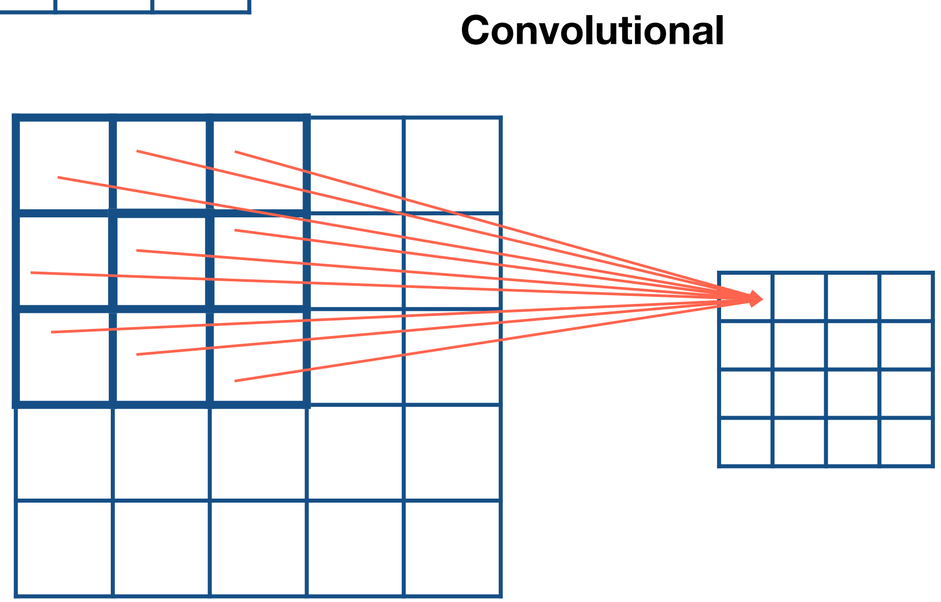
- The same set of weights connect Neuron(s) in the present layer to Neurons in the previous layer- Weight Sharing

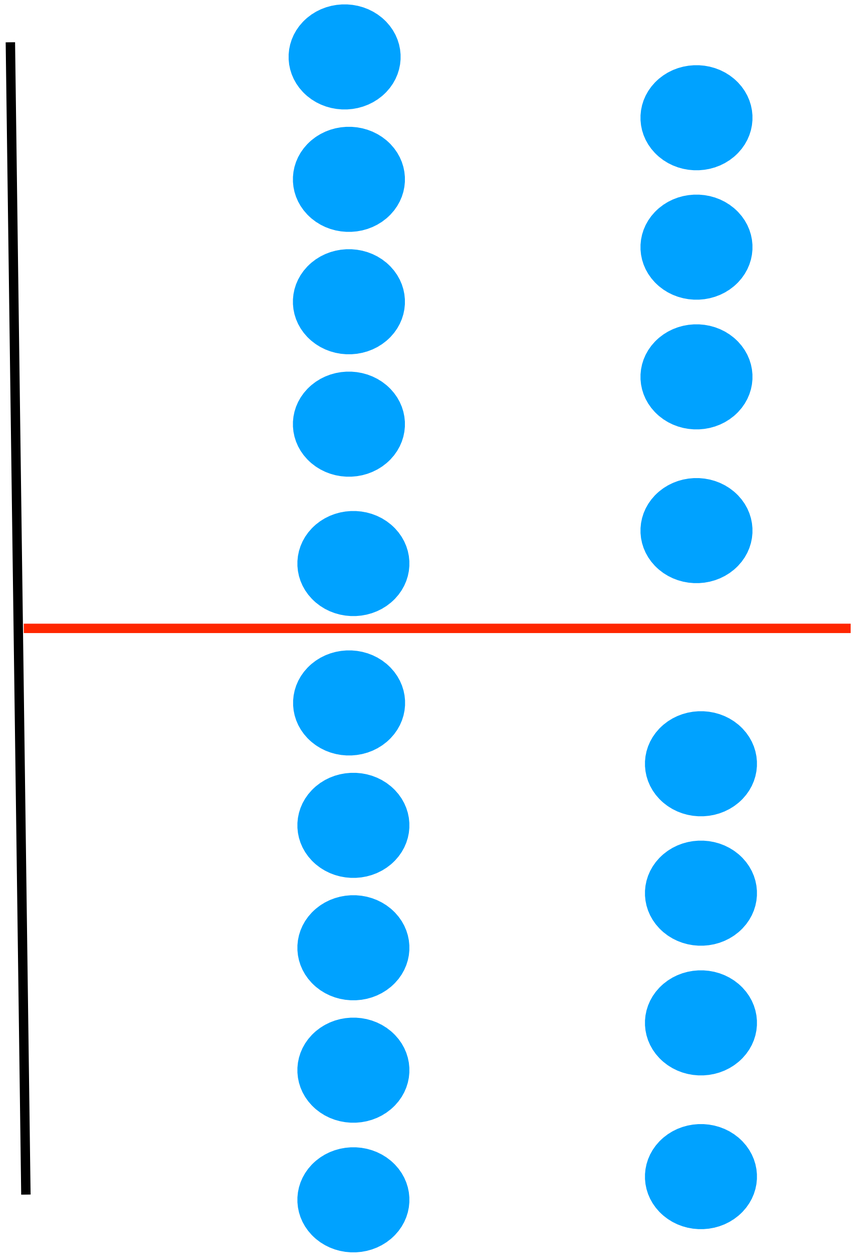
- The weights are typically visualised as filter kernels- a small matrix (2x2,3x3, 5x5 etc) that can be applied to Neurons in the previous layer to generate Neurons in the current layer.

- Hierarchical Learning

Fully Connected to convolutional





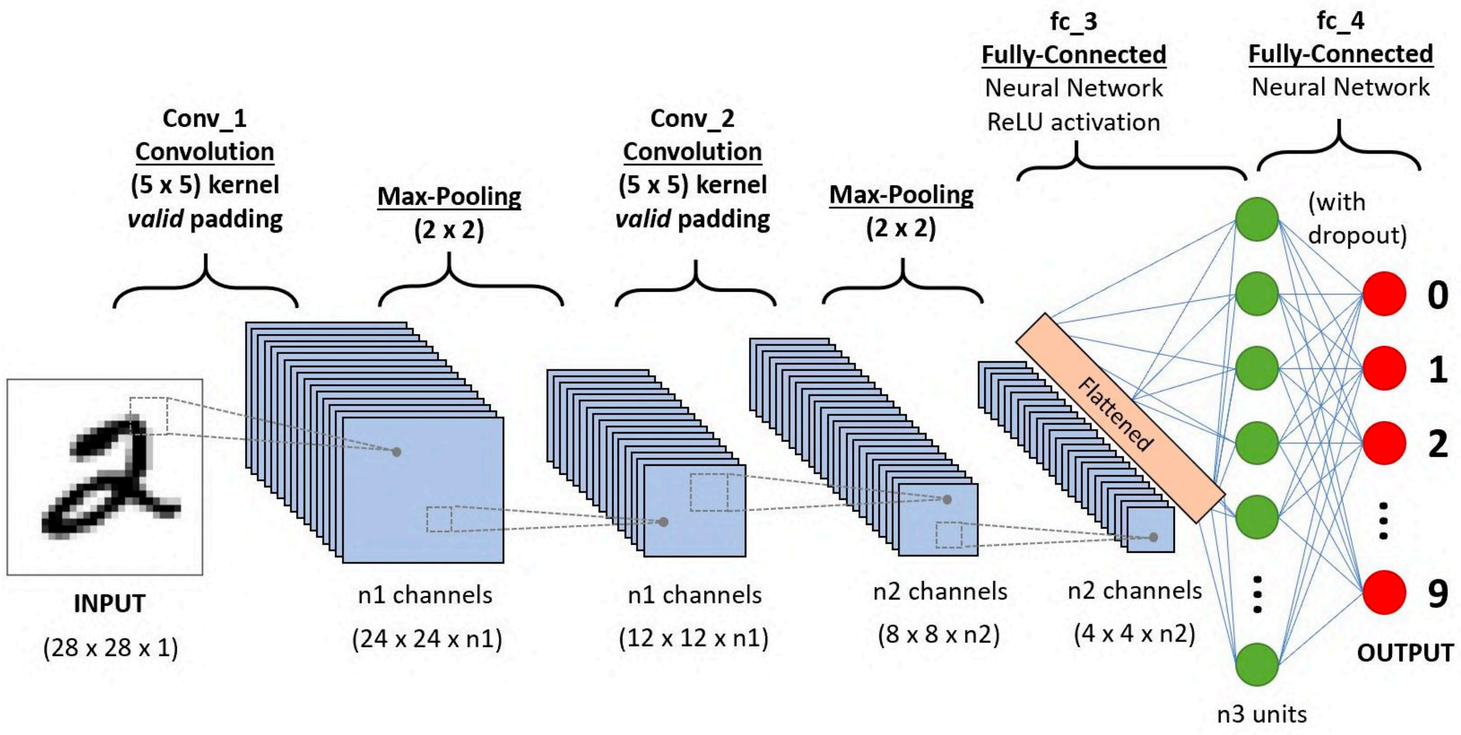


Fully Connected To Convolution

- ANNs take a vector of inputs and produce as output another hidden layer vector fully connected to the input
- For small image sizes the number of weights/parameters to be estimated are not large – but consider a 224x224x3 image - RGB images.
- •A single neuron in the output layer will have 224x224x3 weights coming into it.
- •A ‘Volume’ image input like RGB images will lead to an explosion in the number of weights – Requires more memory, computations and data

Convolutional Neural Networks

- Networks designed to handle gridded data especially images- lots of applications in computer vision
- Locality and translational invariance
- Weight sharing, sparse connections and invariance to translation are unique aspects of these networks.



https://editor.analyticsvidhya.com/uploads/90650dnn2.jpeg

CNN- Convolution

Convolution

1	0	2	2	1
0	2	1	1	3
7	0	1	2	1
5	1	3	2	2
2	3	6	1	5

Image [5X5]

*

Convolution

1	1	0
2	1	1
1	0	2

3x3 Kernel

=

Convolution

1 1	0 1	2 0	2	1
0 2	2 1	1 1	1	3
7 1	0 0	1 2	2	1
5	1	3	2	2
2	3	6	1	5

Image [5X5]

*

Convolution

1	1	0
2	1	1
1	0	2

3x3 Kernel

=

13		

Convolution

1	0 1	2 1	2 0	1
0	2 2	1 1	1 1	3
7	0 1	1 0	2 2	1
5	1	3	2	2
2	3	6	1	5

Image [5X5]

*

Convolution

1	1	0
2	1	1
1	0	2

3x3 Kernel

=

13	12	

Convolution

1	0	2 1	2 1	1 0
0	2	1 2	1 1	3 1
7	0	1 1	2 0	1 2
5	1	3	2	2
2	3	6	1	5

Image [5X5]

*

Convolution

1	1	0
2	1	1
1	0	2

3x3 Kernel

=

13	12	13

Convolution

1	0	2	2	1
0 1	2 1	1 0	1	3
7 2	0 1	1 1	2	1
5 1	1 0	3 2	2	2
2	3	6	1	5

Image [5X5]

*

Convolution

1	1	0
2	1	1
1	0	2

3x3 Kernel

=

13	12	13
28		

Convolution

1	0	2	2	1
0 1	2 1	1 0	1	3
7 2	0 1	1 1	2	1
5 1	1 0	3 2	2	2
2	3	6	1	5

Image [5X5]

*

Convolution

1	1	0
2	1	1
1	0	2

3x3 Kernel

=

13	12	13
28		

Convolution

1	0	2	2	1
0	2	1 1	1 1	3 0
7	0	1 2	2 1	1 1
5	1	3 1	2 0	2 2
2	3	6	1	5

Image [5X5]

*

Convolution

1	1	0
2	1	1
1	0	2

3x3 Kernel

=

13	12	13
28	11	14

Convolution

1	0	2	2	1
0	2	1	1	3
7 1	0 1	1 0	2	1
5 2	1 1	3 1	2	2
2 1	3 0	6 2	1	5

Image [5X5]

*

Convolution

1	1	0
2	1	1
1	0	2

3x3 Kernel

=

13	12	13
28	11	14
35		

Convolution

1	0	2	2	1
0	2	1	1	3
7	0 1	1 1	2 0	1
5	1 2	3 1	2 1	2
2	3 1	6 0	1 2	5

Image [5X5]

*

Convolution

1	1	0
2	1	1
1	0	2

3x3 Kernel

=

13	12	13
28	11	14
35	13	

Convolution

1	0	2	2	1
0	2	1	1	3
7	0	1 1	2 1	1 0
5	1	3 2	2 1	2 1
2	3	6 1	1 0	5 2

Image [5X5]

*

Convolution

1	1	0
2	1	1
1	0	2

3x3 Kernel

=

13	12	13
28	11	14
35	13	29

Strided Convolutions

Strided Convolution – Stride = 1

1	0	2	2	1
0	2	1	1	3
7	0	1	2	1
5	1	3	2	2
2	3	6	1	5

Image [5X5]

*

Convolution

1	1	0
2	1	1
1	0	2

3x3 Kernel

=

Strided Convolution

1 1	0 1	2 0	2	1
0 2	2 1	1 1	1	3
7 1	0 0	1 2	2	1
5	1	3	2	2
2	3	6	1	5

Image [5X5]

*

Convolution

1	1	0
2	1	1
1	0	2

3x3 Kernel

=

13		

Strided Convolution

1	0	2 1	2 1	1 0
0	2	1 2	1 1	3 1
7	0	1 1	2 0	1 2
5	1	3	2	2
2	3	6	1	5

Image [5X5]

*

Convolution

1	1	0
2	1	1
1	0	2

3x3 Kernel

=

13		13

Strided Convolution

1	0	2	2	1
0	2	1	1	3
7 1	0 1	1 0	2	1
5 2	1 1	3 1	2	2
2 1	3 0	6 2	1	5

Image [5X5]

*

Convolution

1	1	0
2	1	1
1	0	2

3x3 Kernel

=

13		13
35		

Strided Convolution

1	0	2	2	1
0	2	1	1	3
7	0	1 1	2 1	1 0
5	1	3 2	2 1	2 1
2	3	6 1	1 0	5 2

Image [5X5]

*

Convolution

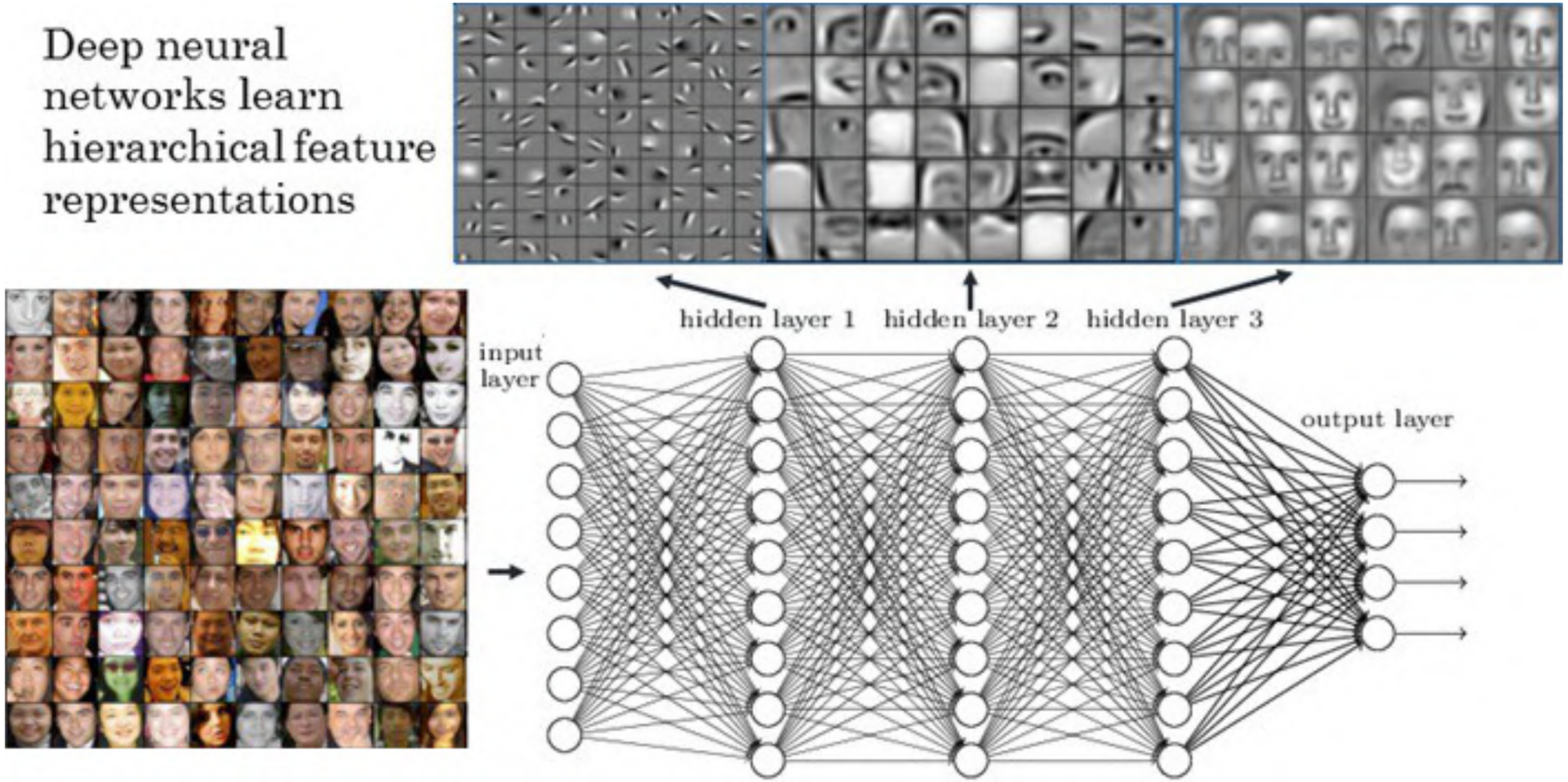
1	1	0
2	1	1
1	0	2

3x3 Kernel

=

13		13
35		29

Hierarchical Learning



https://medium.com/@fenjiro/face-id-deep-learning-for-face-recognition-324b50d916d1

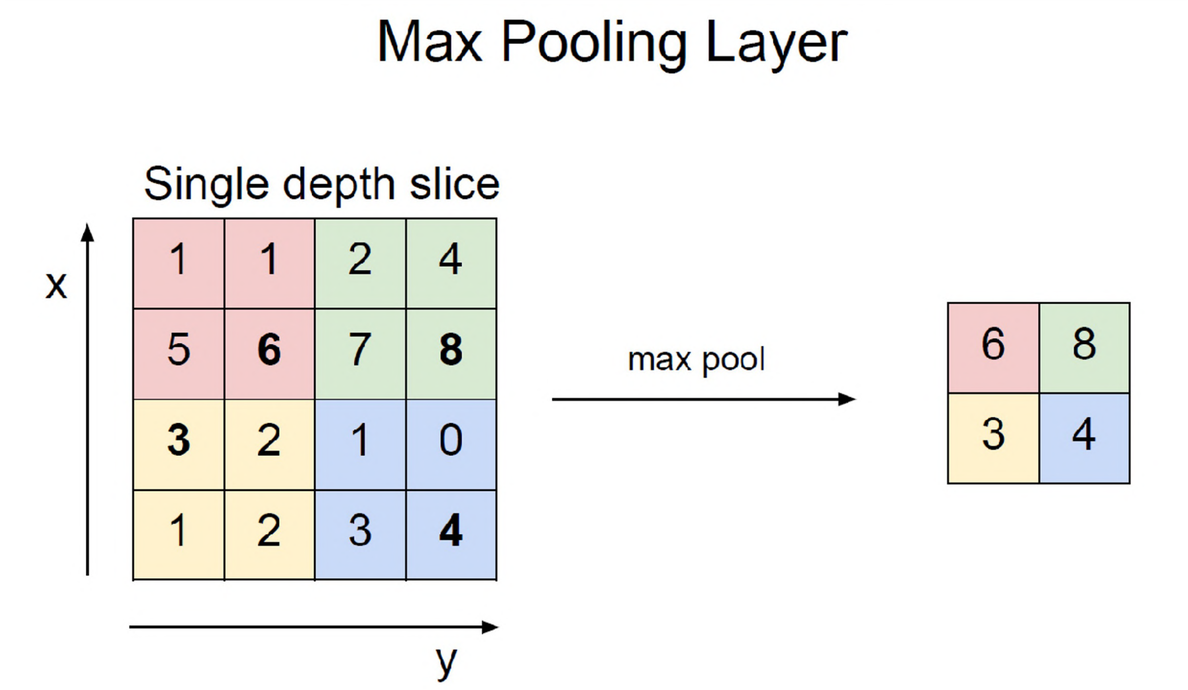
Understanding CNNs

- Two ways to understand CNNs
  - CNNs autonomously learns filters that are typically either hand crafted or emerge as output of some optimisation criterion
  - Advantage is that certain filters hard to formulate mathematically are learnt- problem dependent

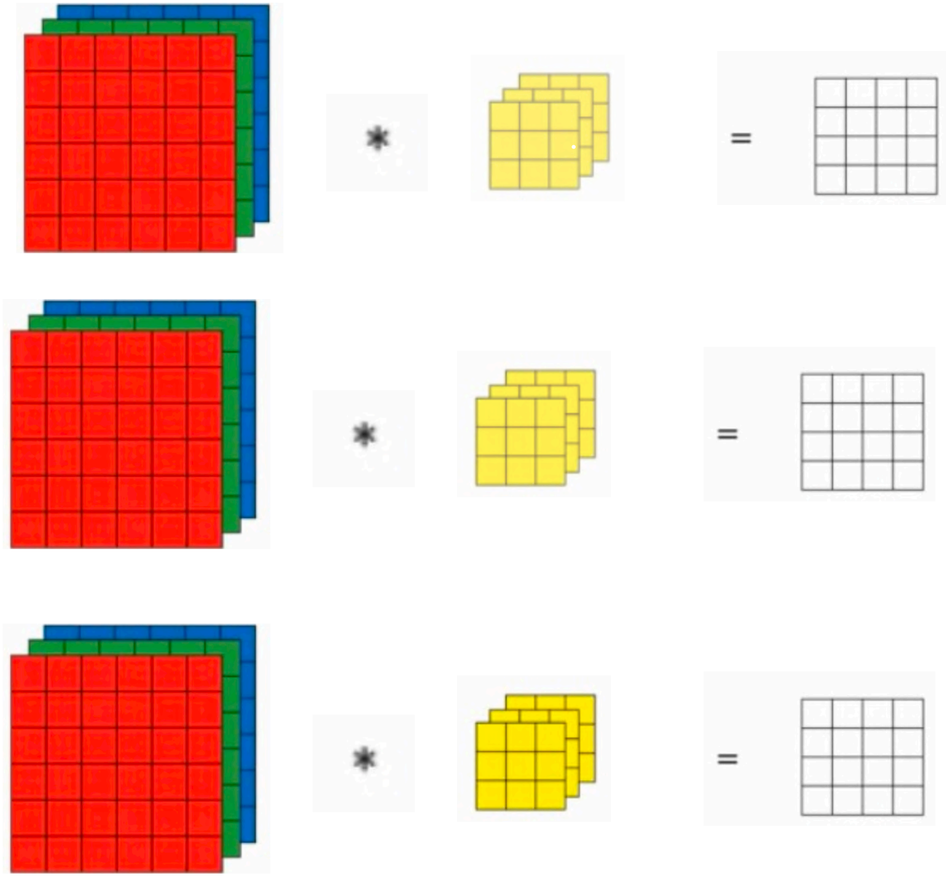
Understanding CNNs

- Projection in to lower dimensional space
  - Consider NxN images as a point in NxN dimensional space
  - Understanding and extracting features in this space becomes harder as N increases
  - CNNs project these images into smaller dimensions and eventually in to a vector (for instance 1024 hidden units in a linear layer)
  - The projection is accomplished using a matrix multiplication. A related example is Fourier transform where you project the signal into sinusoidal basis functions
  - In this case, the basis functions ie the weight matrices are learnt during training

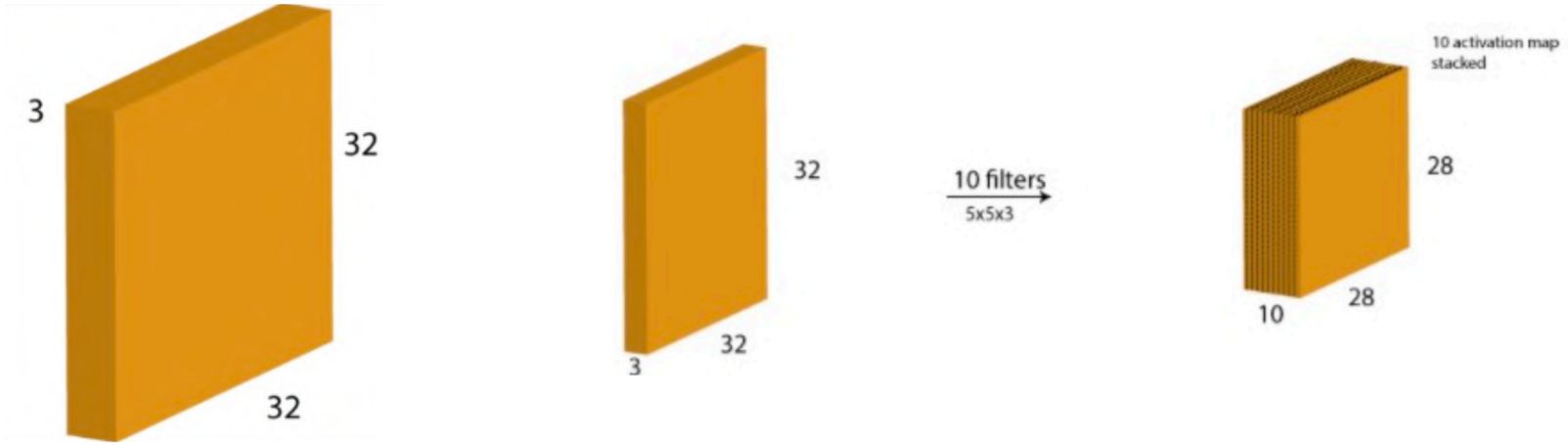
CNN Operations- Max Pooling



CNN Operations - Convolution Filters

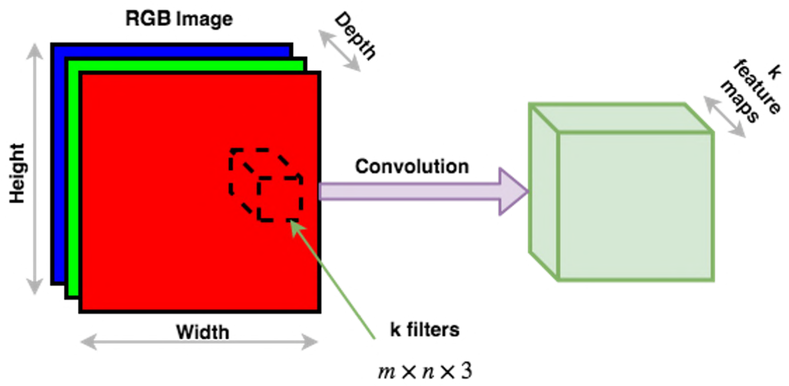


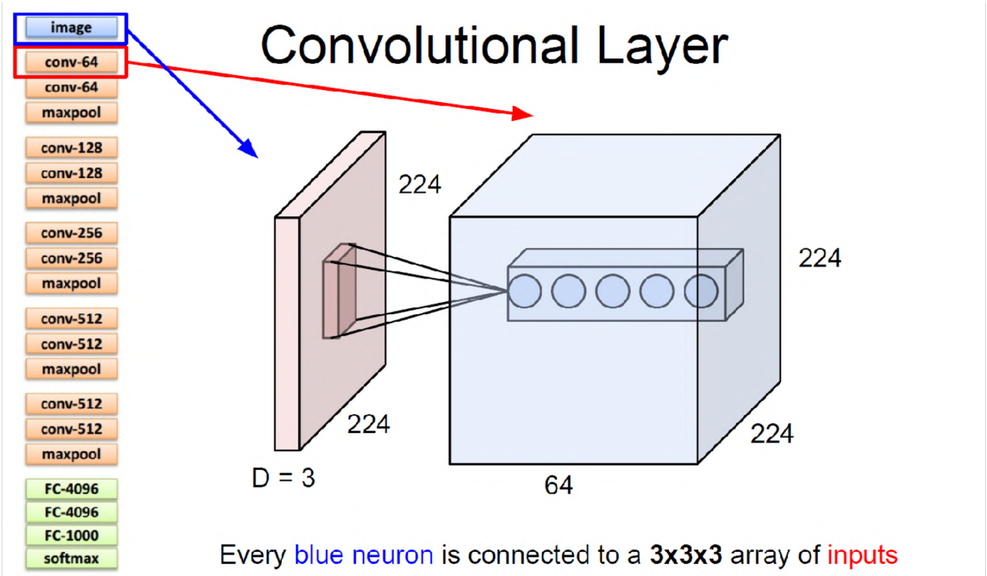
CNN Operations - Convolution

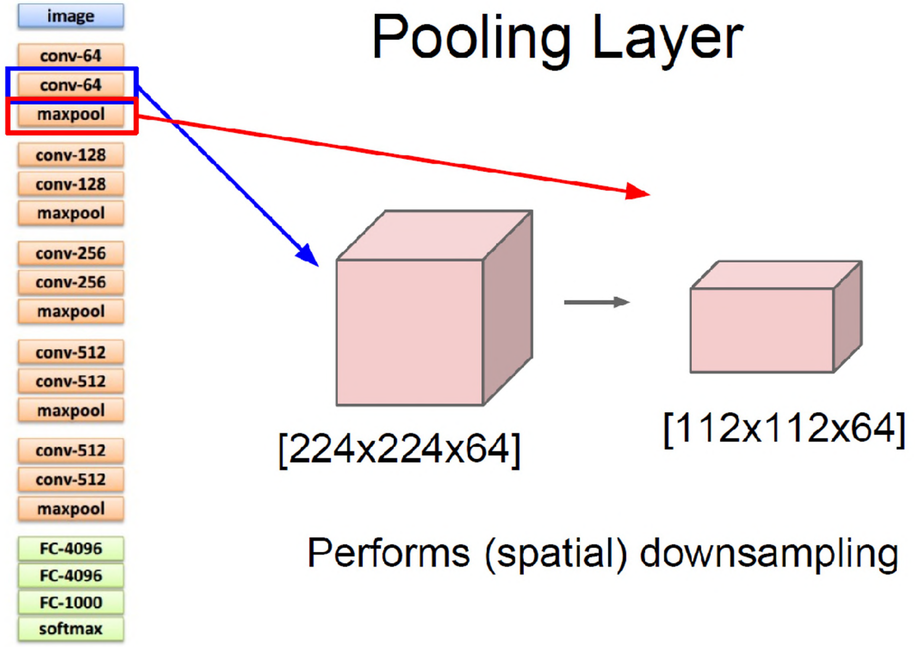


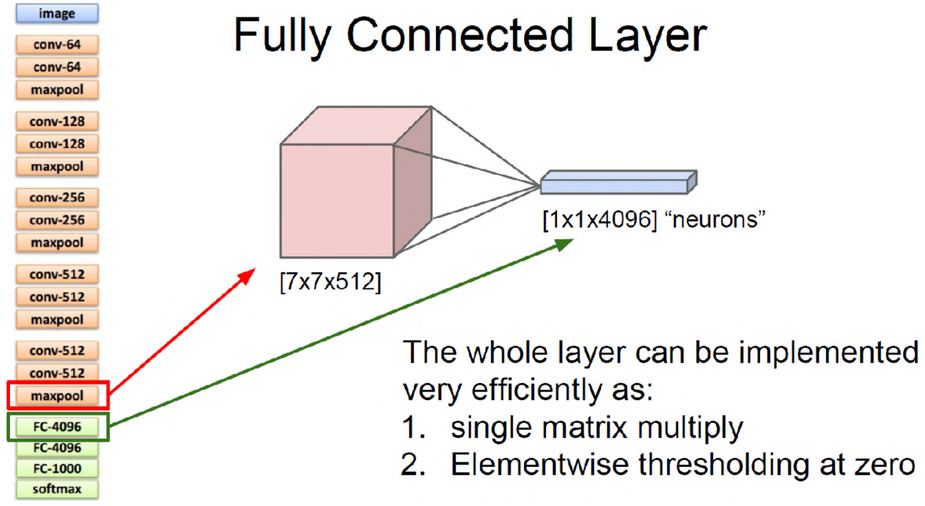
CNN Operations - Convolutions

- Takes a 3D volume of numbers
- Outputs a 3D volume of numbers




Convolutional Layer
image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
224
224
224
224
D = 3
64
Every blue neuron is connected to a 3x3x3 array of inputs


Pooling Layer
image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
[224x224x64]
[112x112x64]
Performs (spatial) downsampling


Fully Connected Layer
image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
[7x7x512]
[1x1x4096] "neurons"
The whole layer can be implemented very efficiently as:
1. single matrix multiply
2. Elementwise thresholding at zero

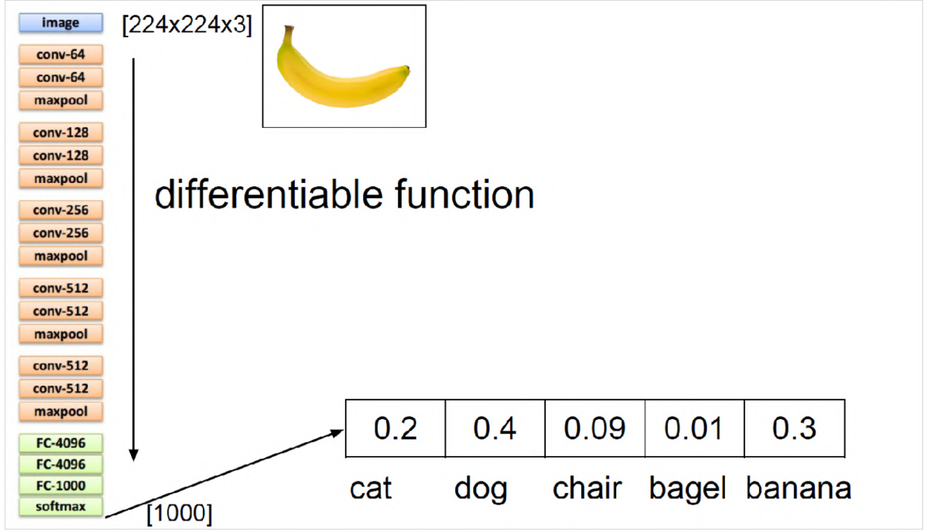


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
[224x224x3]
differentiable function
[1000]
0.2 0.4 0.09 0.01 0.3
cat dog chair bagel banana

Types of Convolutions

- Valid Convolutions
- Same Convolutions
- Strided Convolutions

A guide to convolution arithmetic for deep learning

Vincent Dumoulin[1★] and Francesco Visin[2★†]

[★]MILA, Université de Montréal
[†]AIRLab, Politecnico di Milano

January 12, 2018

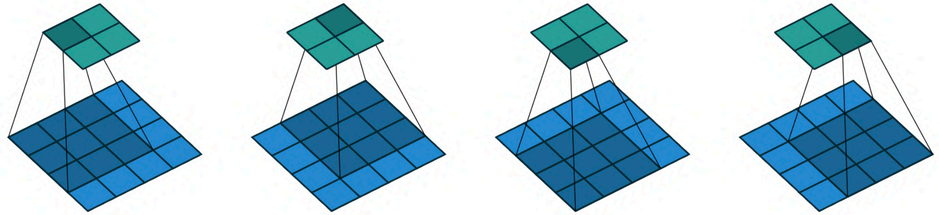


Figure 2.1: (No padding, unit strides) Convolving a 3×3 kernel over a 4×4 input using unit strides (i.e., $i = 4$, $k = 3$, $s = 1$ and $p = 0$).

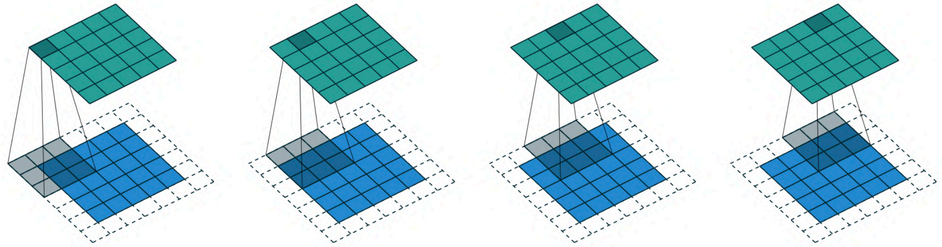


Figure 2.3: (Half padding, unit strides) Convolving a 3×3 kernel over a 5×5 input using half padding and unit strides (i.e., $i = 5$, $k = 3$, $s = 1$ and $p = 1$).

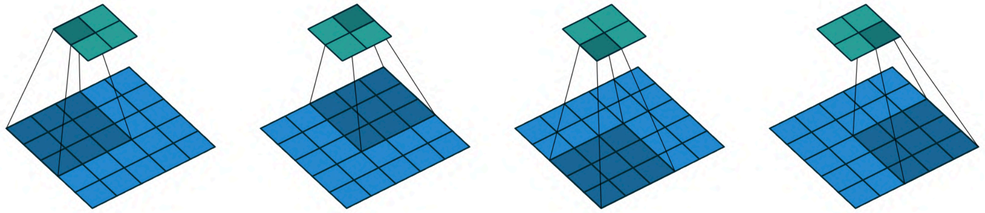


Figure 2.5: (No zero padding, arbitrary strides) Convolving a 3×3 kernel over a 5×5 input using 2×2 strides (i.e., $i = 5$, $k = 3$, $s = 2$ and $p = 0$).

Type of Convolutions

- Dilated Convolutions
- Depthwise Convolutions
- Depthwise Separable Convolutions

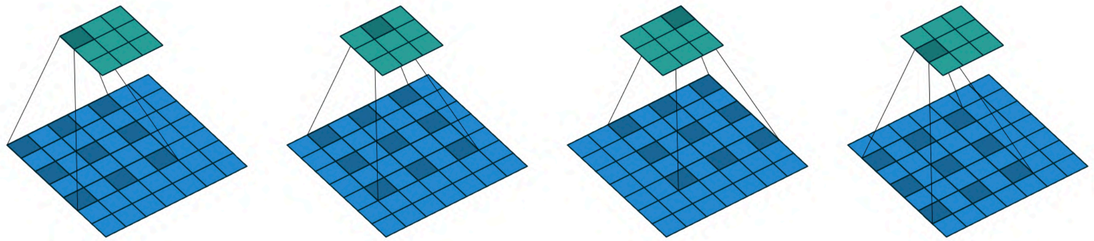
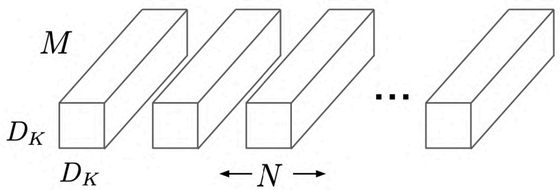
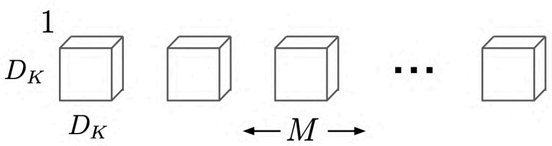


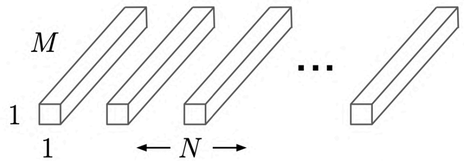
Figure 5.1: (Dilated convolution) Convolving a 3×3 kernel over a 7×7 input with a dilation factor of 2 (i.e., $i = 7$, $k = 3$, $d = 2$, $s = 1$ and $p = 0$).



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



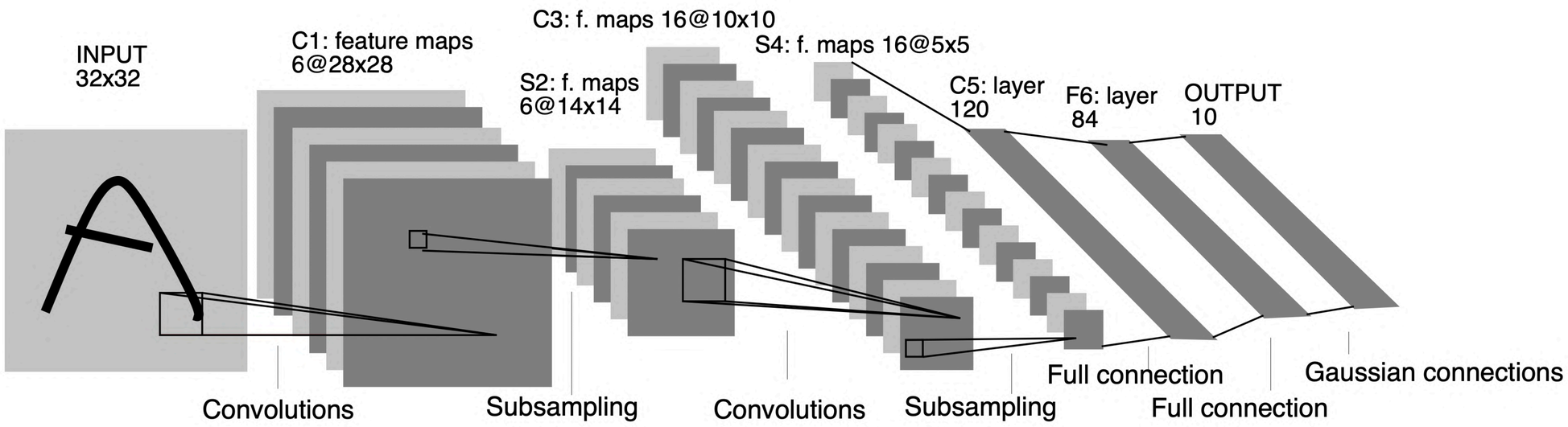
(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications

Andrew G. Howard Menglong Zhu Bo Chen Dmitry Kalenichenko
Weijun Wang Tobias Weyand Marco Andreetto Hartwig Adam

Google Inc.

MNIST Digit Classification
LeNet

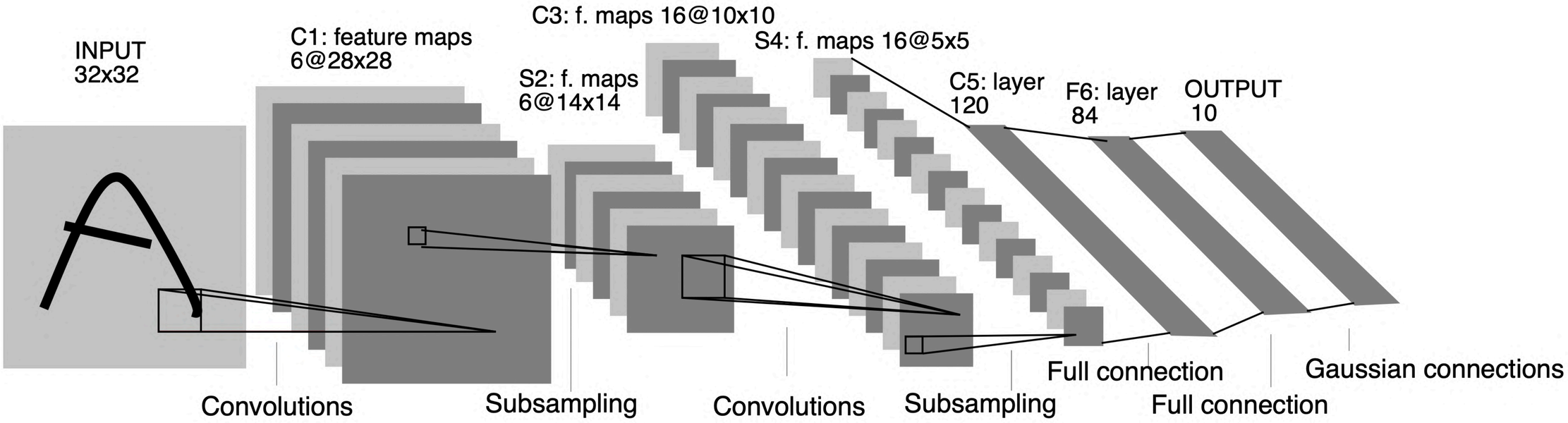


C1 is convolutional layer with 6 feature maps output by 6 Filter Kernels
Each filter kernel is of size 5x5
32x32 -> output size of 28x28

Gradient-Based Learning Applied to Document Recognition

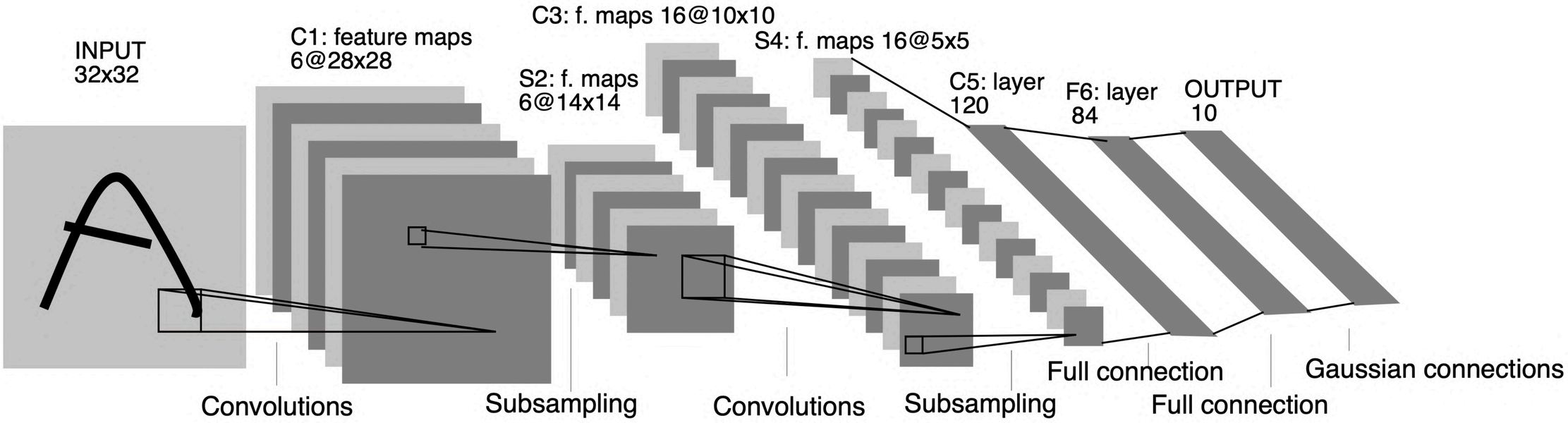
YANN LECUN, MEMBER, IEEE, LÉON BOTTOU, YOSHUA BENGIO, AND PATRICK HAFFNER

LeNet



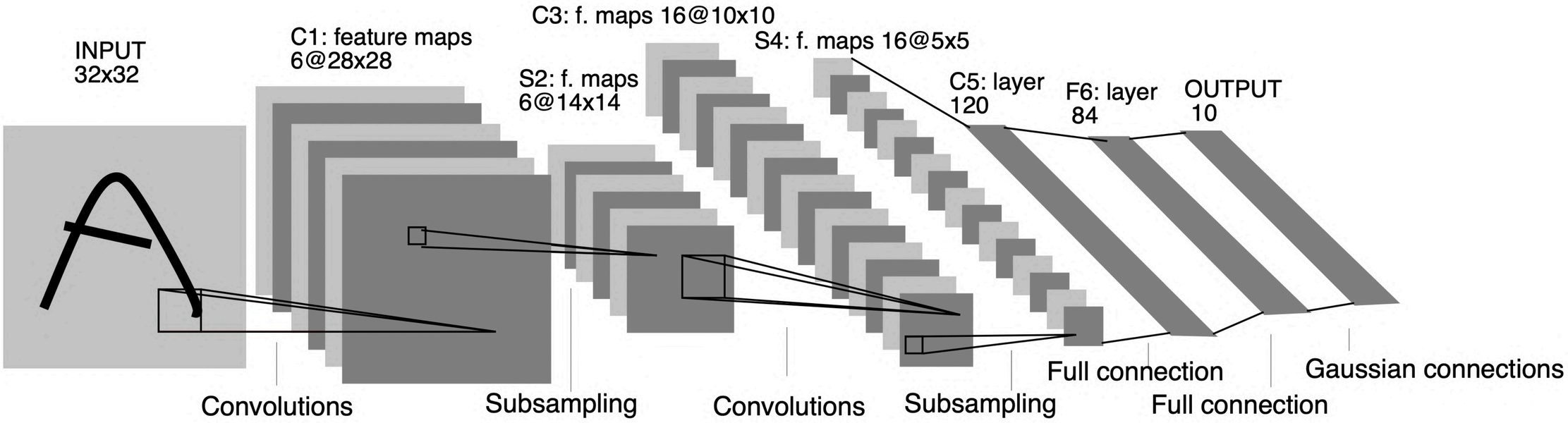
S2 is subsampling layer with
6 feature maps
2x2 max pooling - different
in the original paper which
as trainable parameters

LeNet



Layer C3 is a convolutional layer with 16 feature maps
Each filter kernel is 5x5
Input 14x14-> output 10x10

LeNet

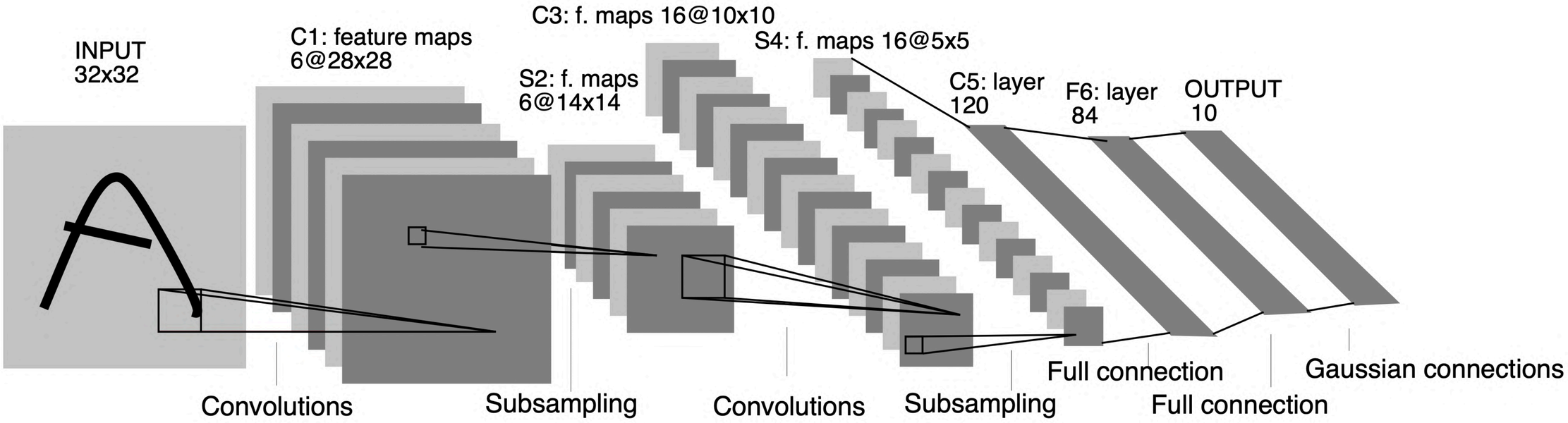


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

TABLE I

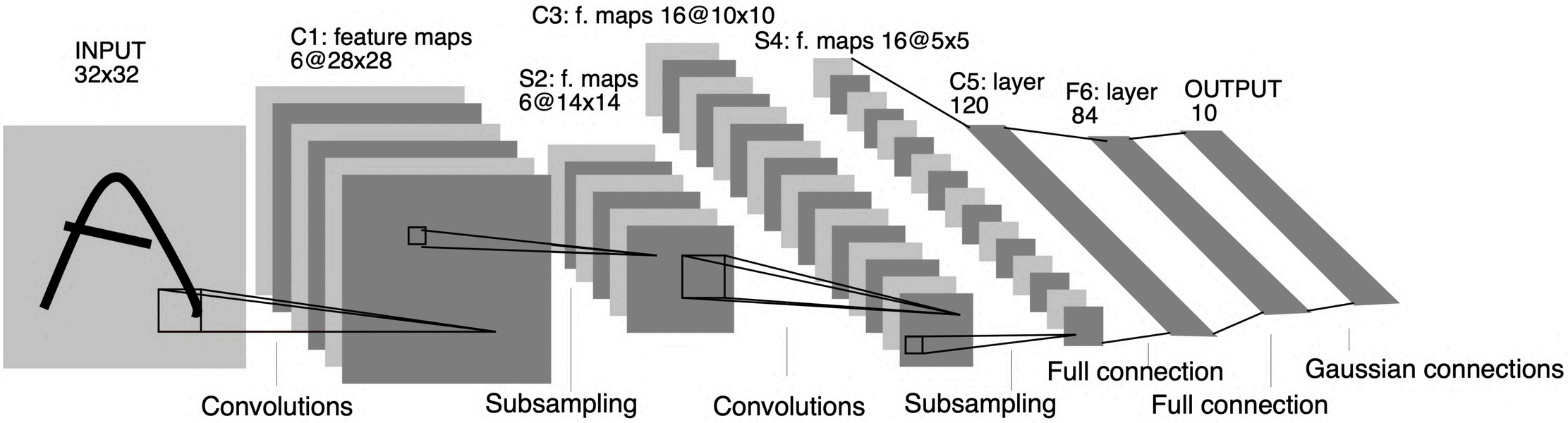
EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.

LeNet



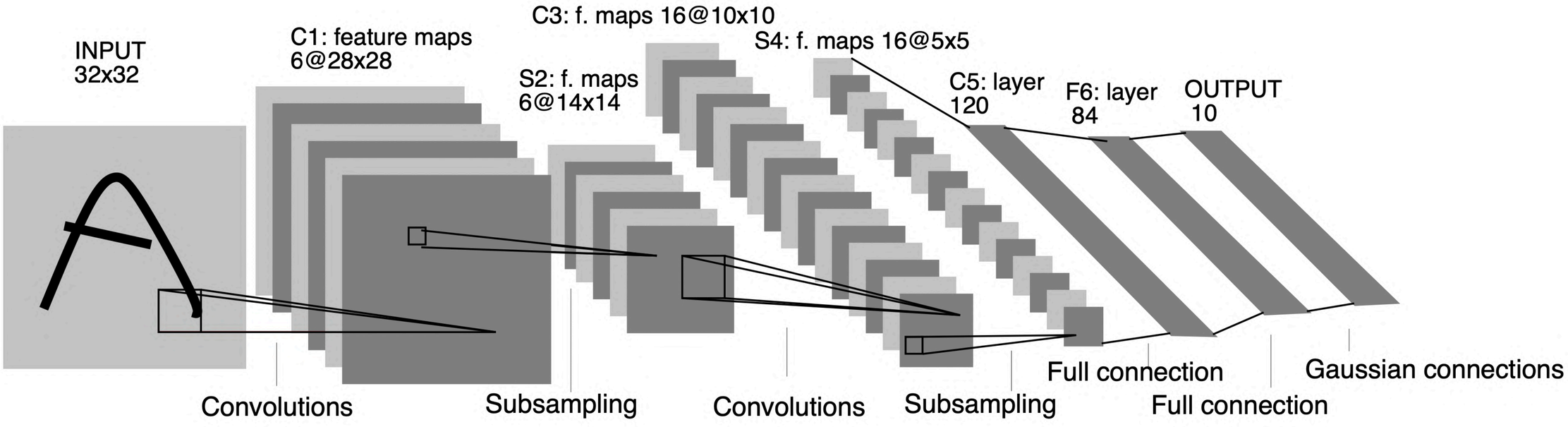
Layer S4 is a sub-sampling layer with 16 feature maps of size 5x5. Every feature map is obtained using 2x2 pooling from C3.

LeNet



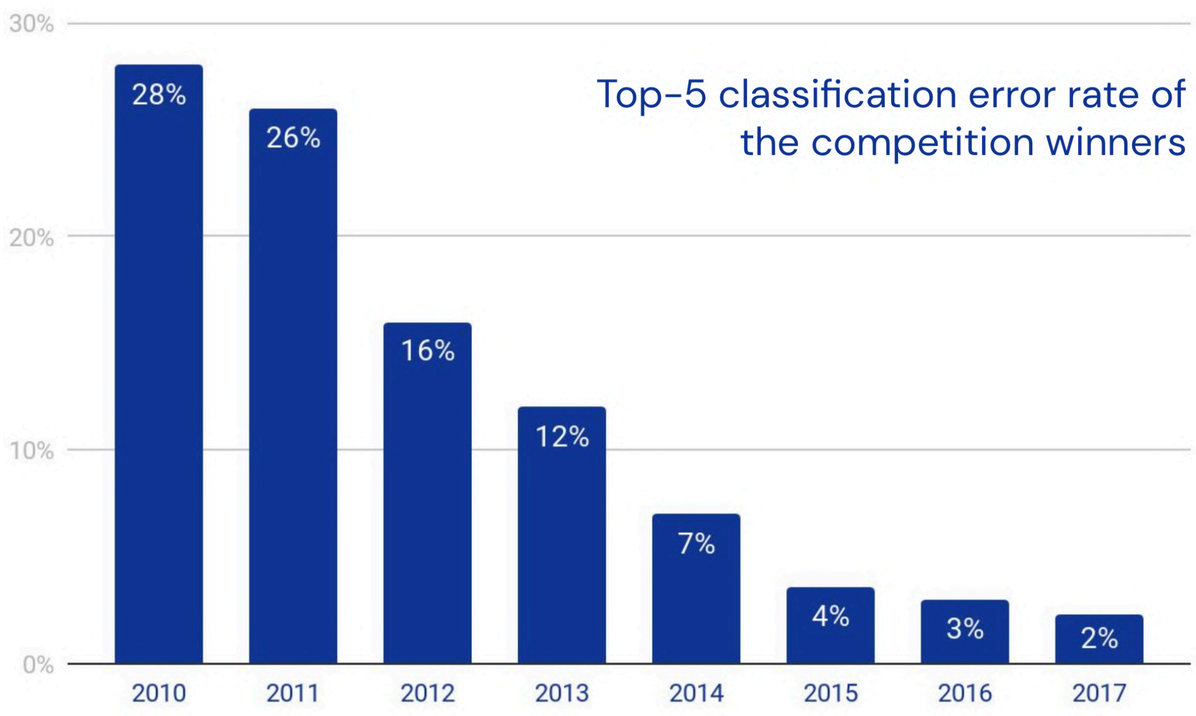
Layer C5 is a convolutional layer with 120 feature maps
Each filter kernel is of size 5x5x16
Output is of size 1x1

LeNet



Layer F6 contains 84 units
And is fully connected to C5

ImageNet Challenge



1. Training set of 1.4 Million Images
2. 1000 classes
3. Image recognition challenge
4. Top 5 prediction error rates

1.2010 and 2011 best results are computer vision techniques-Conventional
2. Alexnet was 16%
3. VGGNet and Inception got 7%
4. Resnet dropped to 4%

CNN Architectures

- AlexNet
- VGG
- ResNet
- DenseNet
- MobileNet
- ShuffleNet
- NAS

LeNet Architecture

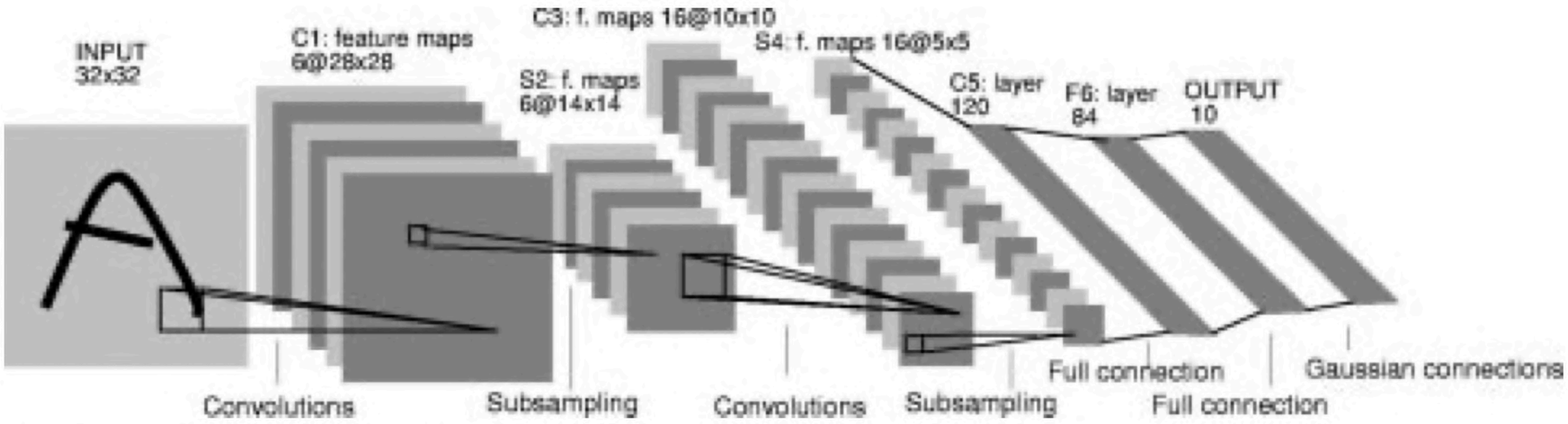


Fig. 2. Architecture of LeNet-5, a convolutional NN, here used for digits recognition. Each plane is a feature map, i.e., a set of units whose weights are constrained to be identical.

Gradient-Based Learning Applied to Document Recognition

YANN LECUN, MEMBER, IEEE, LÉON BOTTOU, YOSHUA BENGIO, AND PATRICK HAFFNER

AlexNet

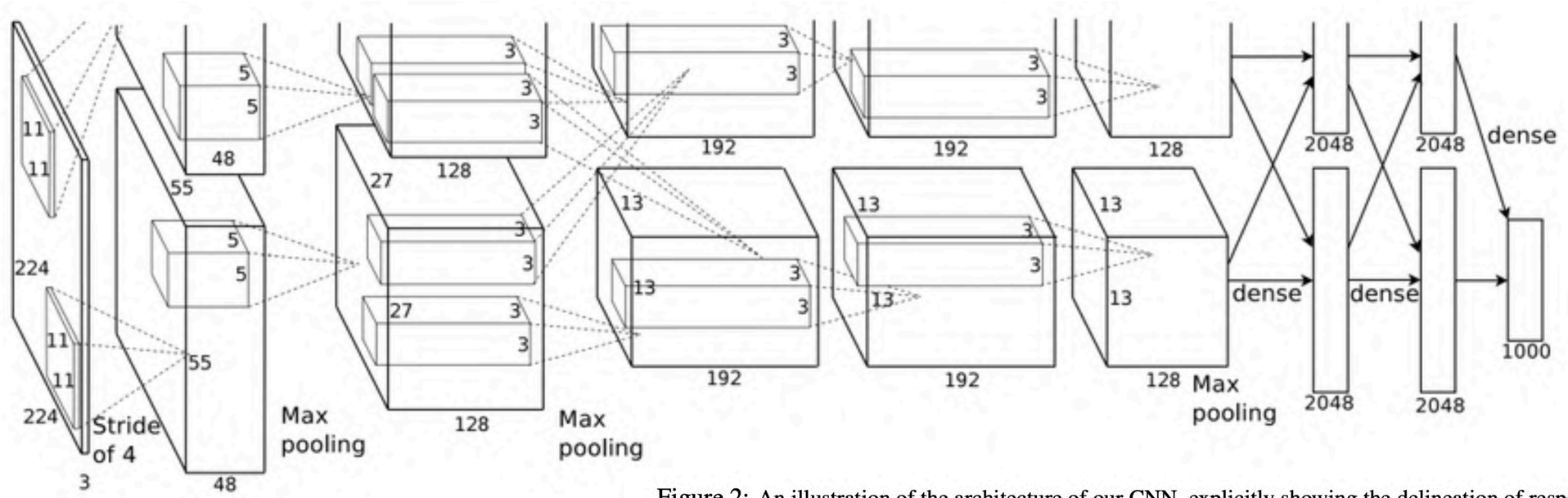


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

ImageNet Classification with Deep Convolutional Neural Networks

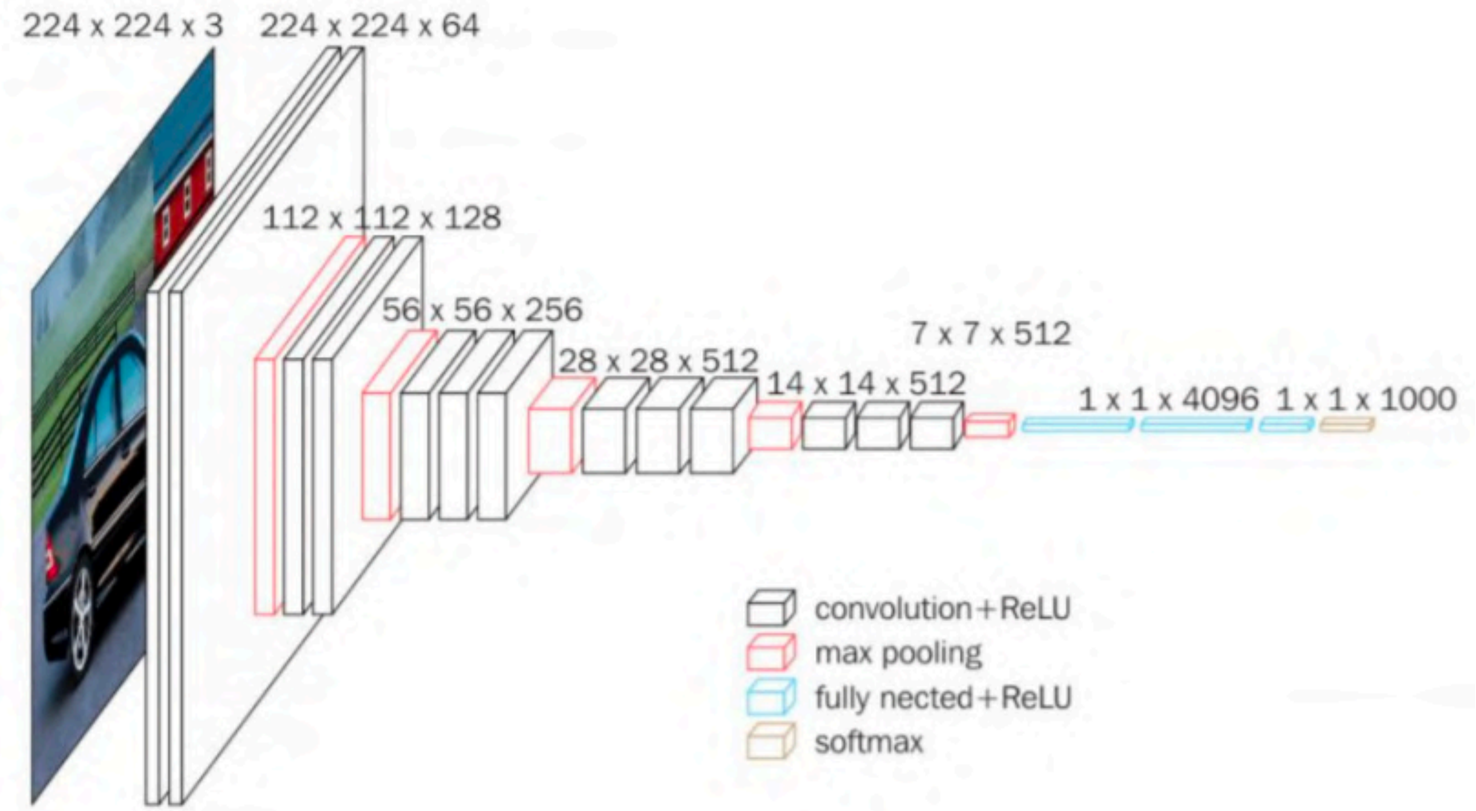
Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

neurons in a kernel map). The second convolutional layer takes as input the (response-normalized and pooled) output of the first convolutional layer and filters it with 256 kernels of size $5 \times 5 \times 48$. The third, fourth, and fifth convolutional layers are connected to one another without any intervening pooling or normalization layers. The third convolutional layer has 384 kernels of size $3 \times 3 \times 256$ connected to the (normalized, pooled) outputs of the second convolutional layer. The fourth convolutional layer has 384 kernels of size $3 \times 3 \times 192$, and the fifth convolutional layer has 256 kernels of size $3 \times 3 \times 192$. The fully-connected layers have 4096 neurons each.

VGGNet

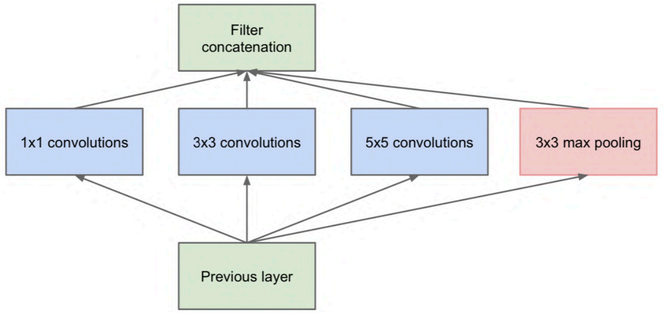


VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

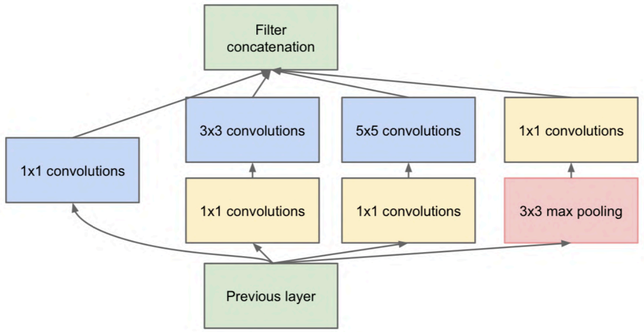
Karen Simonyan* **& Andrew Zisserman+**
Visual Geometry Group, Department of Engineering Science, University of Oxford
{karen,az}@robots.ox.ac.uk

11,13 ,16 and 19 layers. The best result is for 16 layers. It actually slightly worsens for 19 layers

GoogleNet



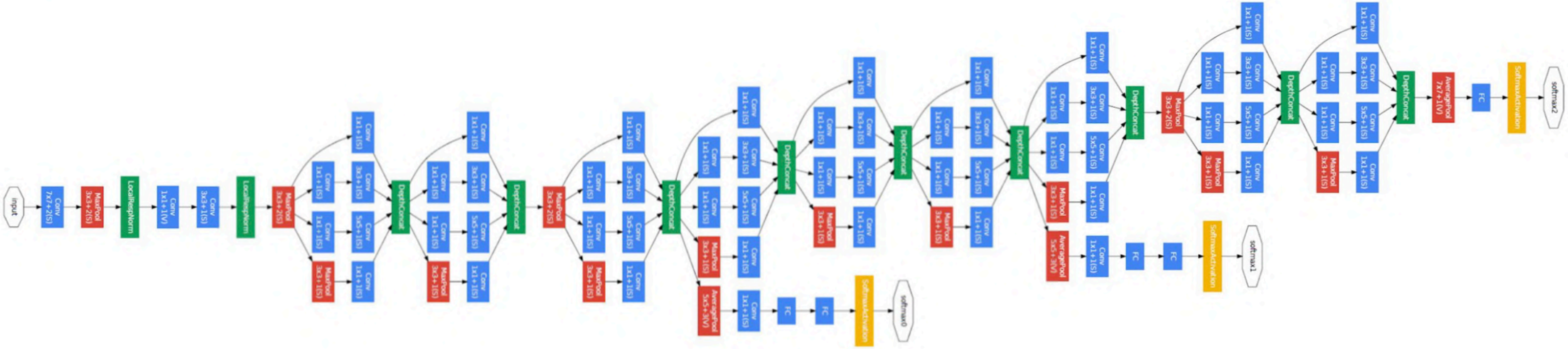
(a) Inception module, naïve version



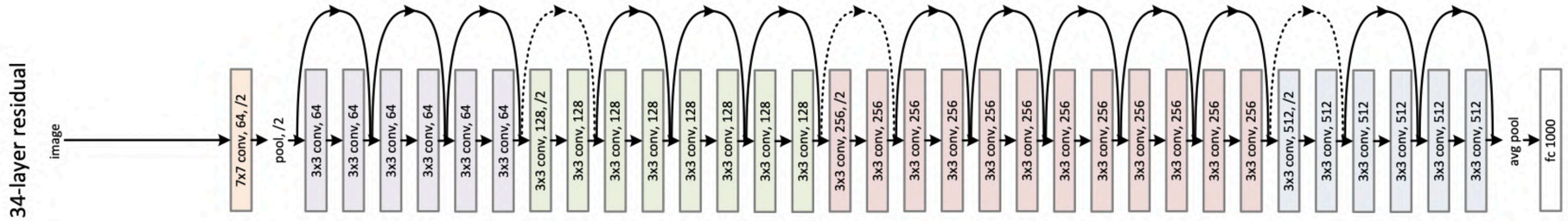
(b) Inception module with dimension reductions

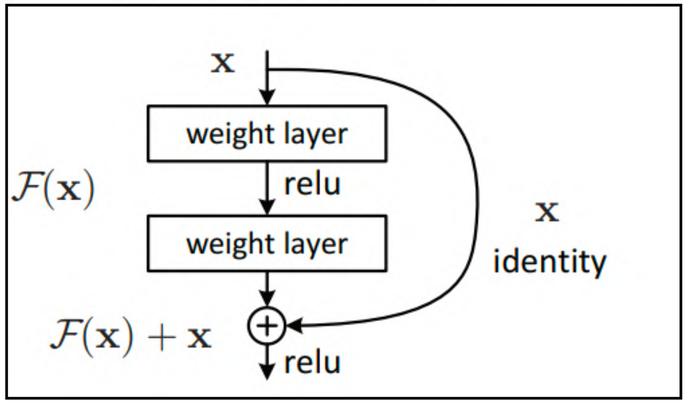
Going deeper with convolutions

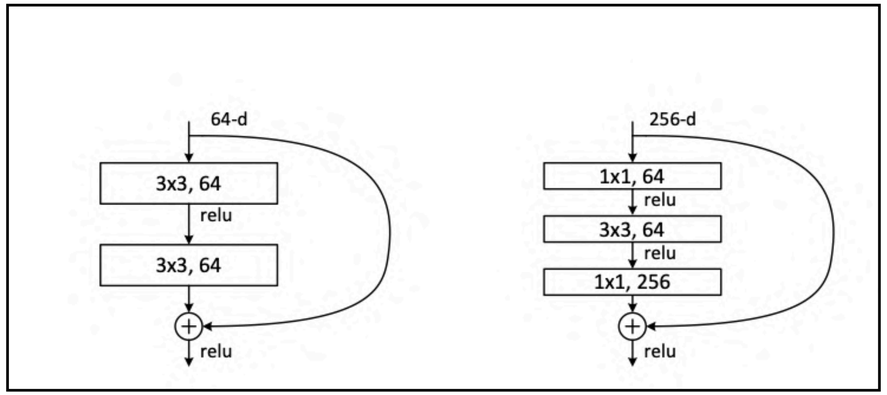
Christian Szegedy Google Inc.
Wei Liu University of North Carolina, Chapel Hill
Yangqing Jia Google Inc.
Pierre Sermanet Google Inc.
Scott Reed University of Michigan
Dragomir Anguelov Google Inc.
Dumitru Erhan Google Inc.
Vincent Vanhoucke Google Inc.
Andrew Rabinovich Google Inc.



ResNet (ResNet -34 architecture)







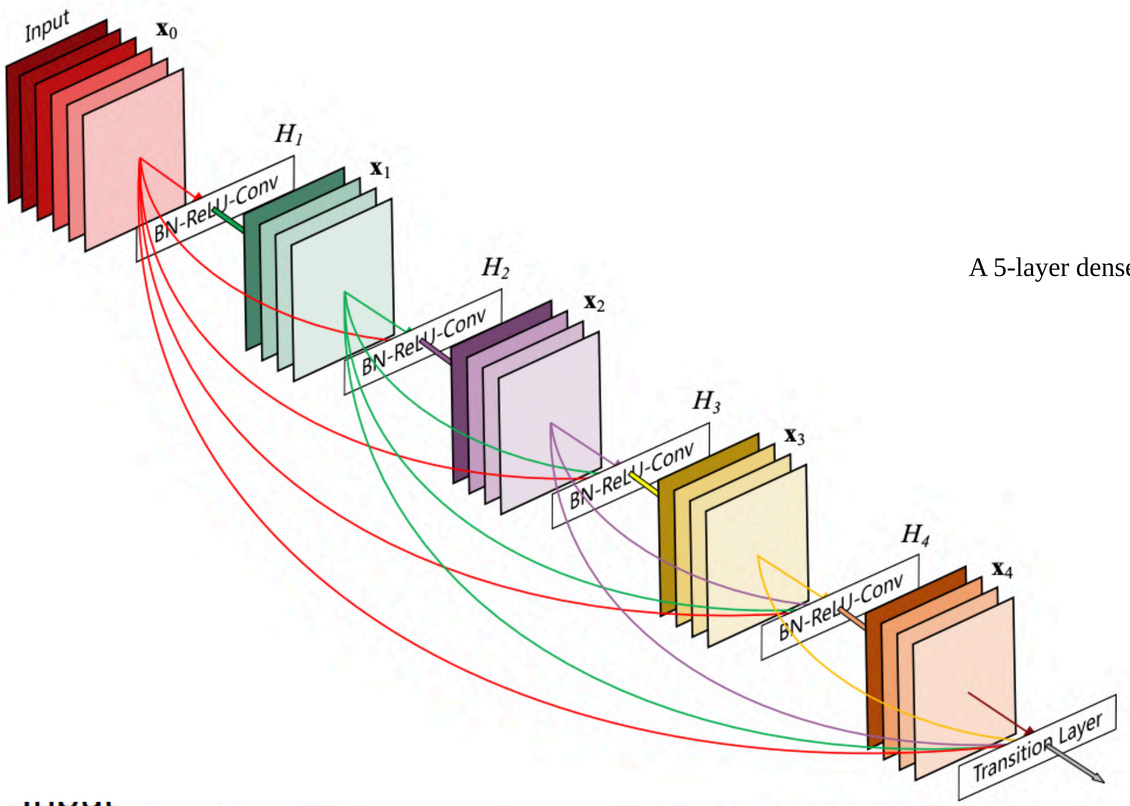
Skip (Shortcut) connection

Deep Residual Learning for Image Recognition

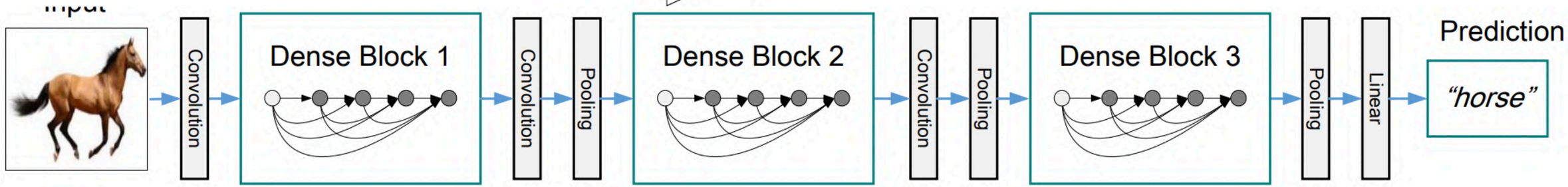
Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Built to address the vanishing gradient problem in deep networks

DenseNet



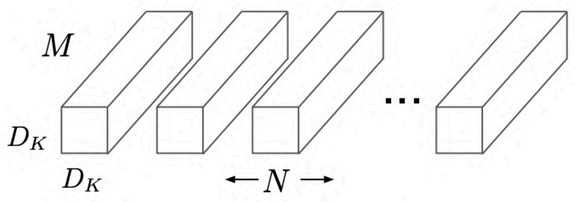
A 5-layer dense block with a growth rate of k = 4. Each layer takes all preceding feature-maps as input



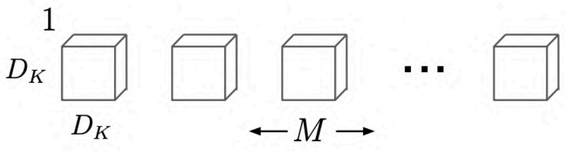
A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling

Figures from , DOI: 10.1109/CVPR.2017.243

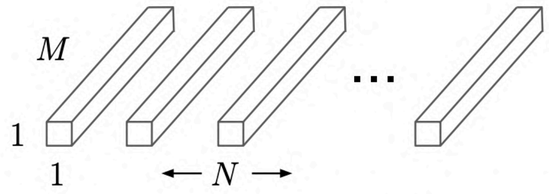
MobileNet



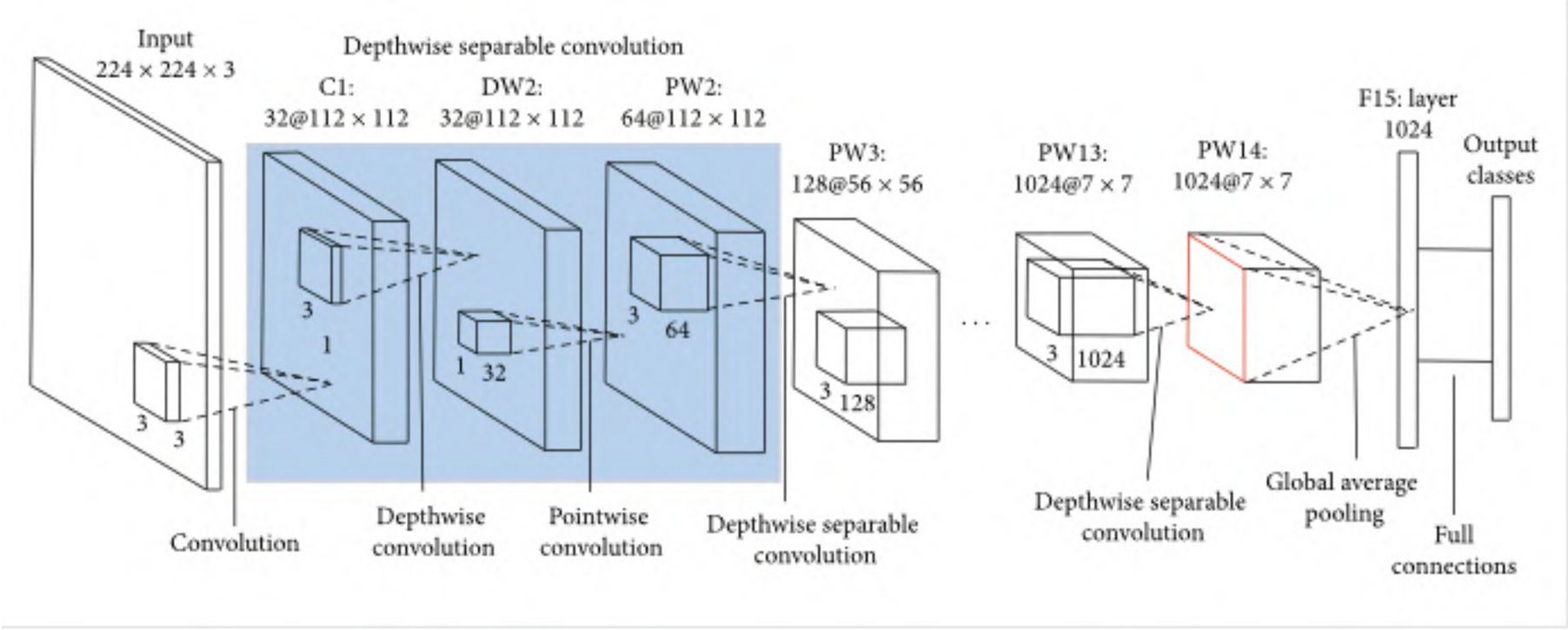
(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



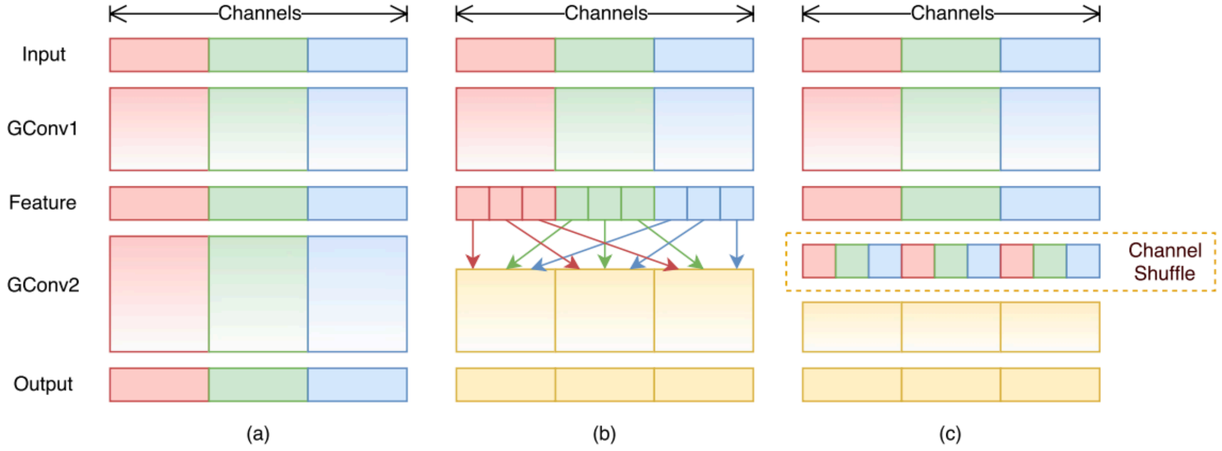
(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution



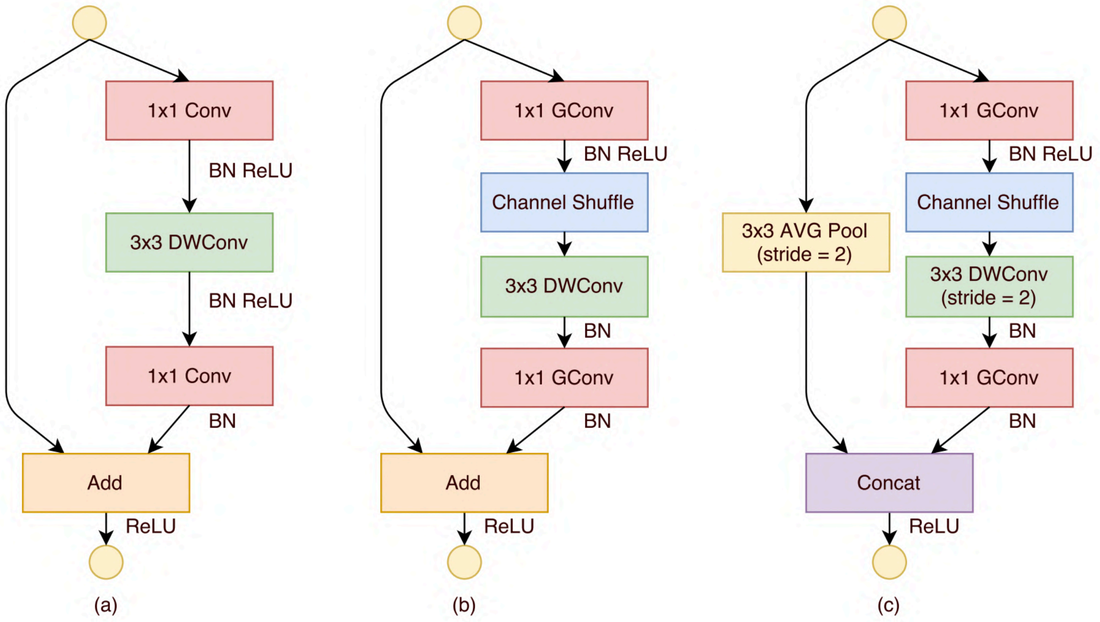
The standard convolutional filters in (a) are replaced by two layers: depthwise convolution in (b) and pointwise convolution in (c) to build a depthwise separable filter.

https://doi.org/10.48550/arXiv.1704.04861

- ShuffleNet



Channel shuffle with two stacked group convolutions. GConv stands for group convolution. a) two stacked convolution layers with the same number of groups. Each output channel only relates to the input channels within the group. No cross talk; b) input and output channels are fully related when GConv2 takes data from different groups after GConv1; c) an equivalent implementation to b) using channel shuffle



ShuffleNet Units. a) bottleneck unit [9] with depthwise convolution (DWConv) b) ShuffleNet unit with pointwise group convolution (GConv) and channel shuffle; c) ShuffleNet unit with stride = 2

https://doi.org/10.48550/arXiv.1707.01083

Layer	Output size	KSize	Stride	Repeat	Output channels (g groups)				
					$g = 1$	$g = 2$	$g = 3$	$g = 4$	$g = 8$
Image	224×224				3	3	3	3	3
Conv1	112×112	3×3	2	1	24	24	24	24	24
MaxPool	56×56	3×3	2						
Stage2[1]	28×28		2	1	144	200	240	272	384
	28×28		1	3	144	200	240	272	384
Stage3	14×14		2	1	288	400	480	544	768
	14×14		1	7	288	400	480	544	768
Stage4	7×7		2	1	576	800	960	1088	1536
	7×7		1	3	576	800	960	1088	1536
GlobalPool	1×1	7×7							
FC					1000	1000	1000	1000	1000
Complexity[2]					143M	140M	137M	133M	137M

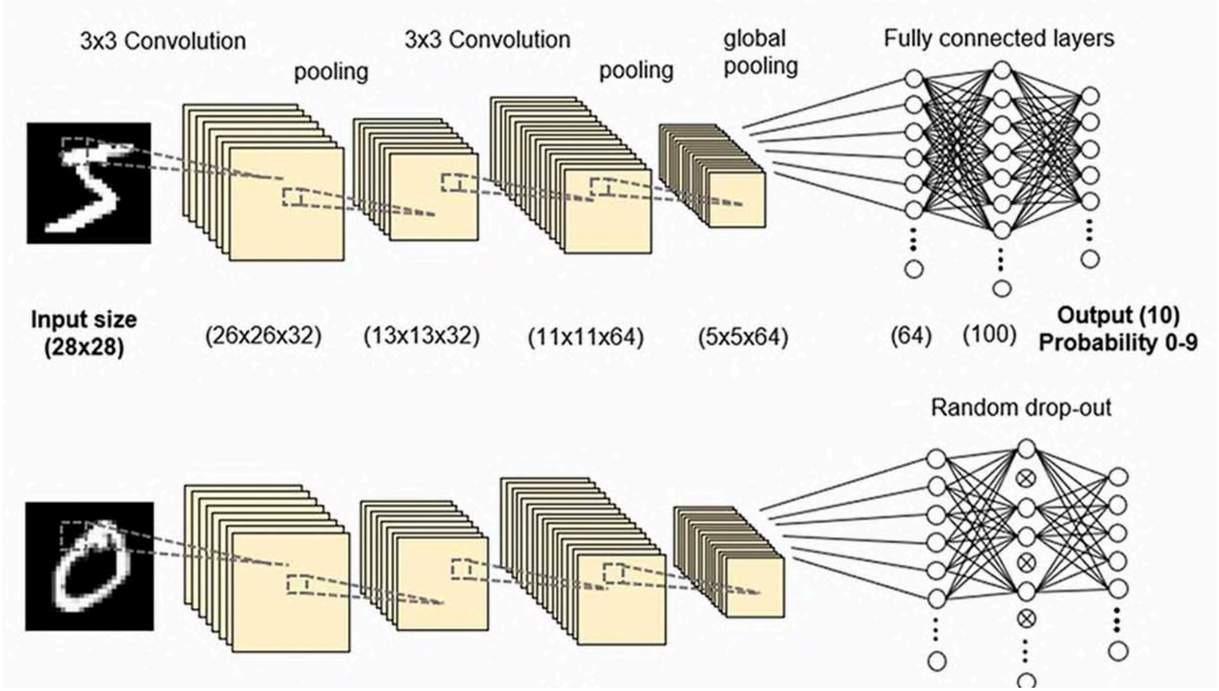
ShuffleNet Architecture

https://doi.org/10.48550/arXiv.1707.01083

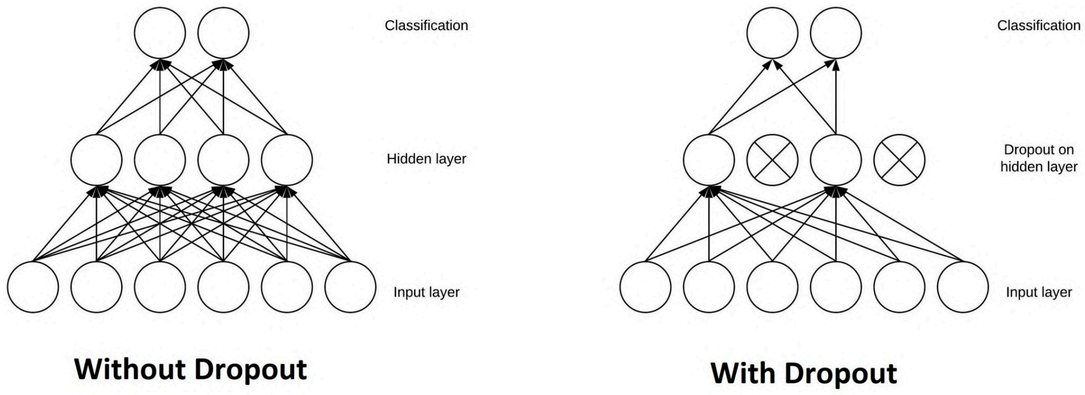
CNN Training

- Dropouts
- Normalisation Layers
- Hyperparameter optimisation
- Weight Initialization
- Data Augmentation

CNN Training-Dropouts



https://www.frontiersin.org/files/Articles/571894/fnagi-12-00273-HTML-r1/image_m/fnagi-12-00273-g001.jpg



https://www.baeldung.com/wp-content/uploads/sites/4/2020/05/2-1-2048x745-1.jpg

CNN Training- Addressing Covariate Shift

- Batch Norm
- Layer Norm
- Instance Norm
- Group Norm

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

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

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

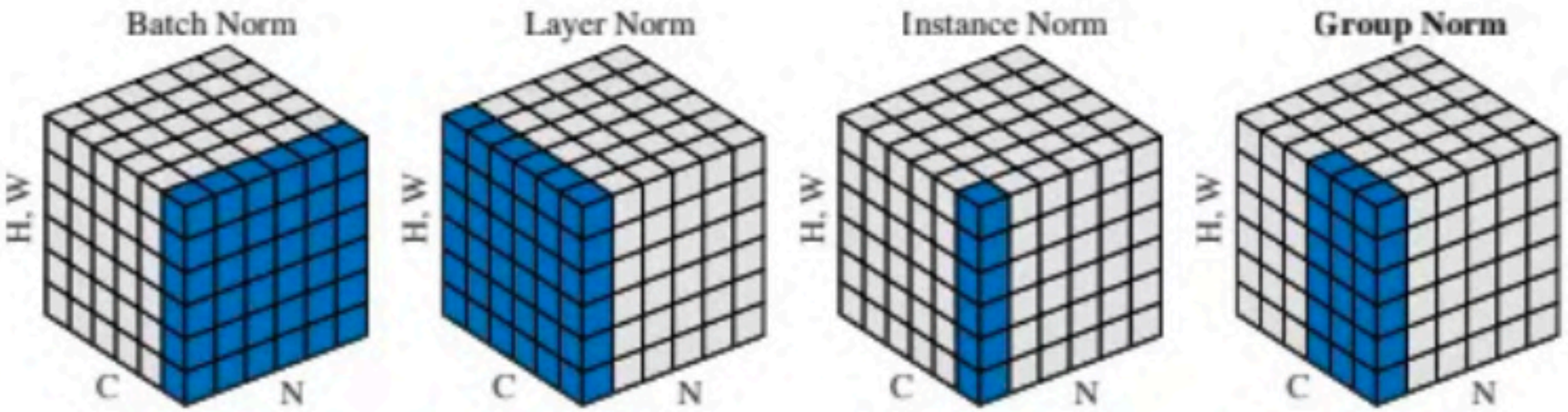
$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

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

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

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

arXiv:1502.03167v3



A visual comparison of various normalization methods

PowerNorm: Rethinking Batch Normalization in Transformers

Sheng Shen [*1] Zhewei Yao [*1] Amir Gholami [1] Michael W. Mahoney [1] Kurt Keutzer [1]

There has also been work on alternative normalization techniques, and in particular Layer Normalization (LN), proposed by (Ba et al., 2016). LN normalizes across the channel/feature dimension as shown in Figure 1. This could be extended to Group Norm (GN) (Wu & He, 2018), where the normalization is performed across a partition of the features/channels with different pre-defined groups. Instance Normalization (IN) (Ulyanov et al., 2016) is another technique, where per-channel statistics are computed for each sample.

$$\mu_i = \frac{1}{m}\sum_{k\in\mathcal{S}_i} x_k, \quad \sigma_i = \sqrt{\frac{1}{m}\sum_{k\in\mathcal{S}_i}(x_k-\mu_i)^2+\epsilon},$$

S_i is defined below.

Group Norm

$$\mathcal{S}_i = \{k \mid k_N = i_N, \lfloor\frac{k_C}{C/G}\rfloor = \lfloor\frac{i_C}{C/G}\rfloor\}.$$

x is the feature computed by a layer, and i is an index. In the case of 2D images, **i** = (**i**N , **i**C , **i**H, **i**W) is a 4D vector indexing the features in (N, C, H, W) order, where N is the batch axis, **C** is the channel axis, and **H** and **W** are the spatial height and width axes. **G** is the number of groups, which is a pre-defined hyper-parameter. C/G is the number of channels per group. ⌊.⌋ is the floor operation, and "**⌊kC/(C/G)⌋ = ⌊iC/(C/G)⌋**" means that the indexes **i** and **k** are in the same group of channels, assuming each group of channels are stored in a sequential order along the C axis. GN computes μ and σ along the (H, W) axes and along a group of C/G channels

$$\hat{x}_i = \frac{1}{\sigma_i}(x_i - \mu_i).$$

$$y_i = \gamma\hat{x}_i + \beta,$$

https://arxiv.org/abs/1803.08494

CNN Training- Hyper parameter Optimisation

- Grid Search
- Random Search
- Hand Tuning

Random Search for Hyper-Parameter Optimization

James Bergstra JAMES.BERGSTRA@UMONTREAL.CA
Yoshua Bengio YOSHUA.BENGIO@UMONTREAL.CA
Département d'Informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC, H3C 3J7, Canada

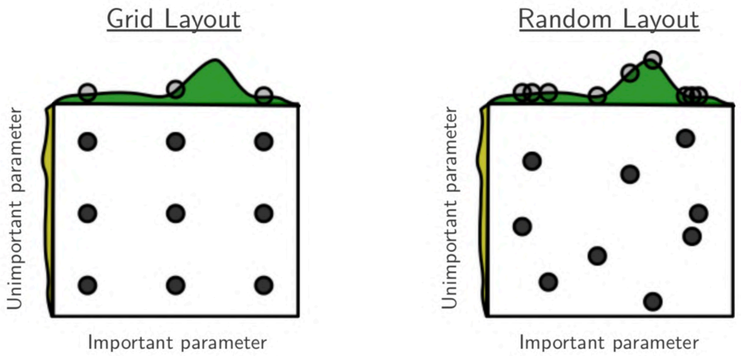
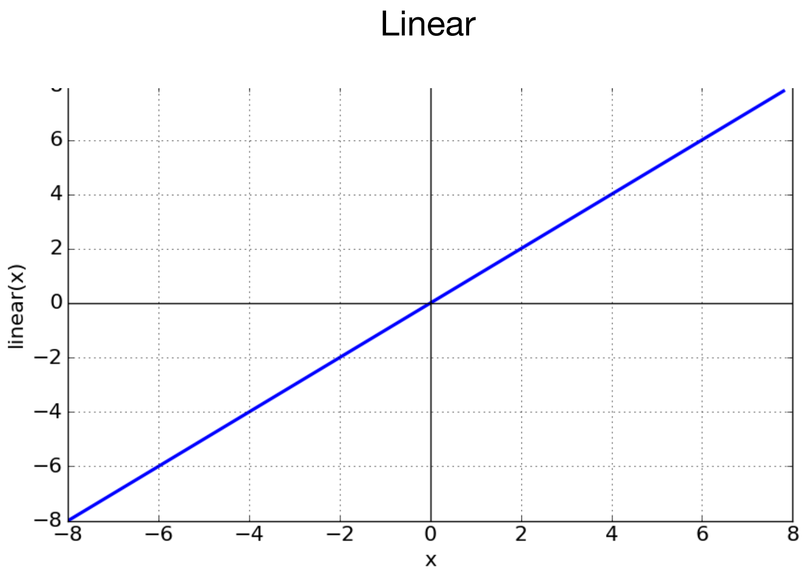
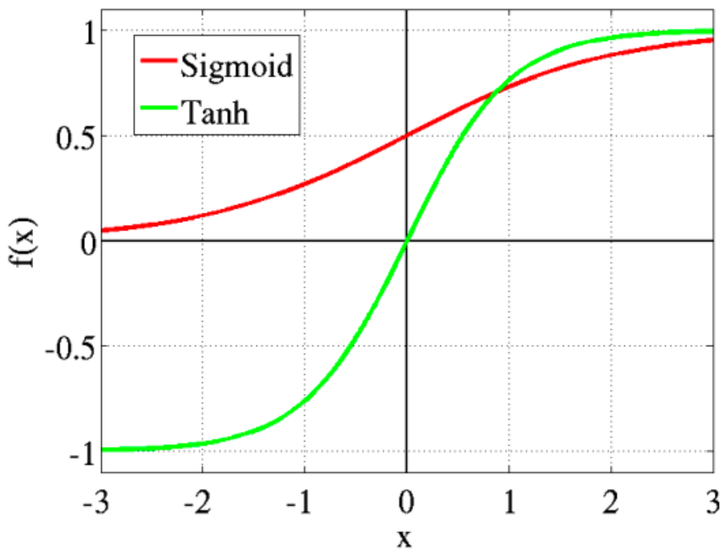


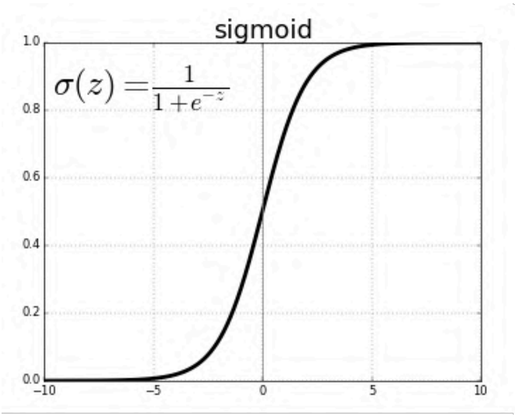
Figure 1: Grid and random search of nine trials for optimizing a function $f(x,y) = g(x) + h(y) \approx g(x)$ with low effective dimensionality. Above each square $g(x)$ is shown in green, and left of each square $h(y)$ is shown in yellow. With grid search, nine trials only test $g(x)$ in three distinct places. With random search, all nine trials explore distinct values of g. This failure of grid search is the rule rather than the exception in high dimensional hyper-parameter optimization.

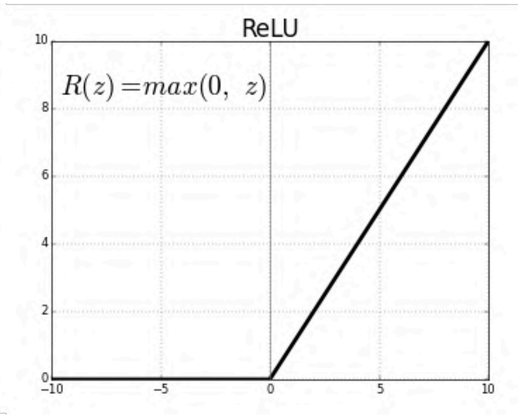
CNN Training-Non-Linearities & Loss functions

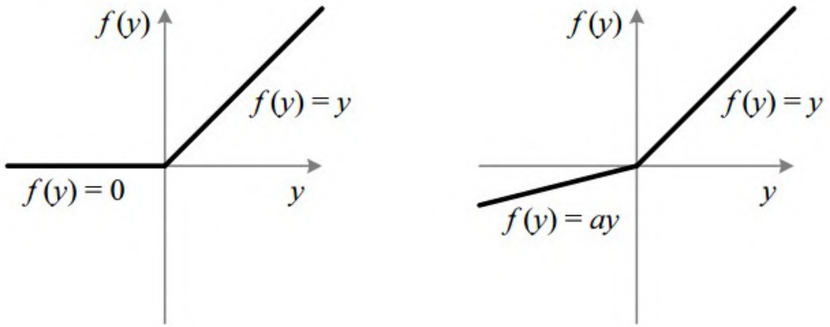
- Non-Linearities
- Loss Functions



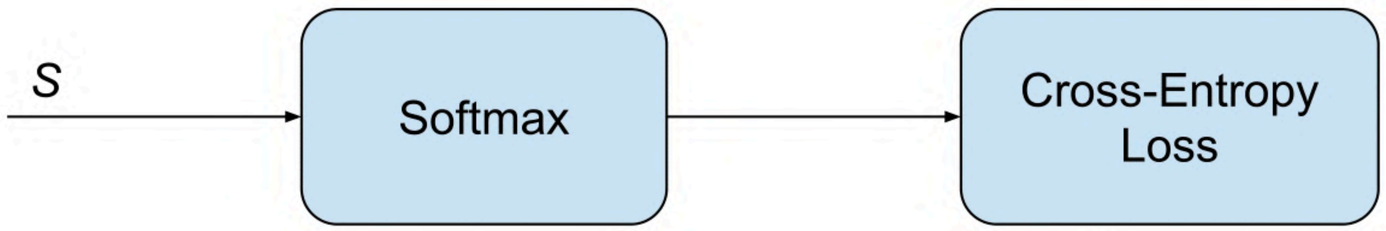








ReLU v/s Leaky ReLU



$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \quad CE = -\sum_i^C t_i log(f(s)_i)$$

https://gombru.github.io/assets/cross_entropy_loss/softmax_CE_pipeline.png

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

https://miro.medium.com/max/808/1*-e1QGatrODWpJkEwqP4Jyg.png

Visualising CNNs - Saliency Maps

More formally, let $S_c(I)$ be the score of the class c, computed by the classification layer of the ConvNet for an image I. We would like to find an L_2-regularised image, such that the score S_c is high:

$$\arg\max_I S_c(I) - \lambda \|I\|_2^2, \tag{1}$$

where λ is the regularisation parameter. A locally-optimal I can be found by the back-propagation method. The procedure is related to the ConvNet training procedure, where the back-propagation is used to optimise the layer weights. The difference is that in our case the optimisation is performed with respect to the input image, while the weights are fixed to those found during the training stage. We initialised the optimisation with the zero image (in our case, the ConvNet was trained on the zero-centred image data), and then added the training set mean image to the result. The class model visualisations for several classes are shown in Fig. 1.

Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps

Karen Simonyan **Andrea Vedaldi** **Andrew Zisserman**
Visual Geometry Group, University of Oxford
{karen,vedaldi,az}@robots.ox.ac.uk

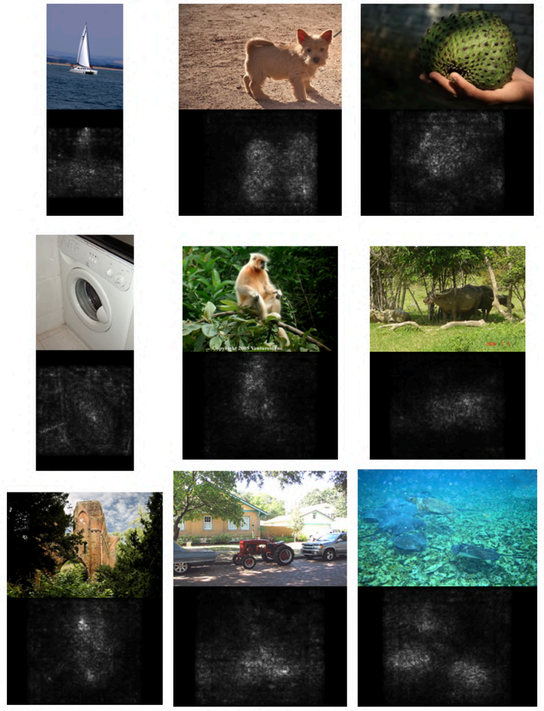


Figure 2: **Image-specific class saliency maps for the top-1 predicted class in ILSVRC-2013 test images.** The maps were extracted using a single back-propagation pass through a classification ConvNet. No additional annotation (except for the image labels) was used in training.

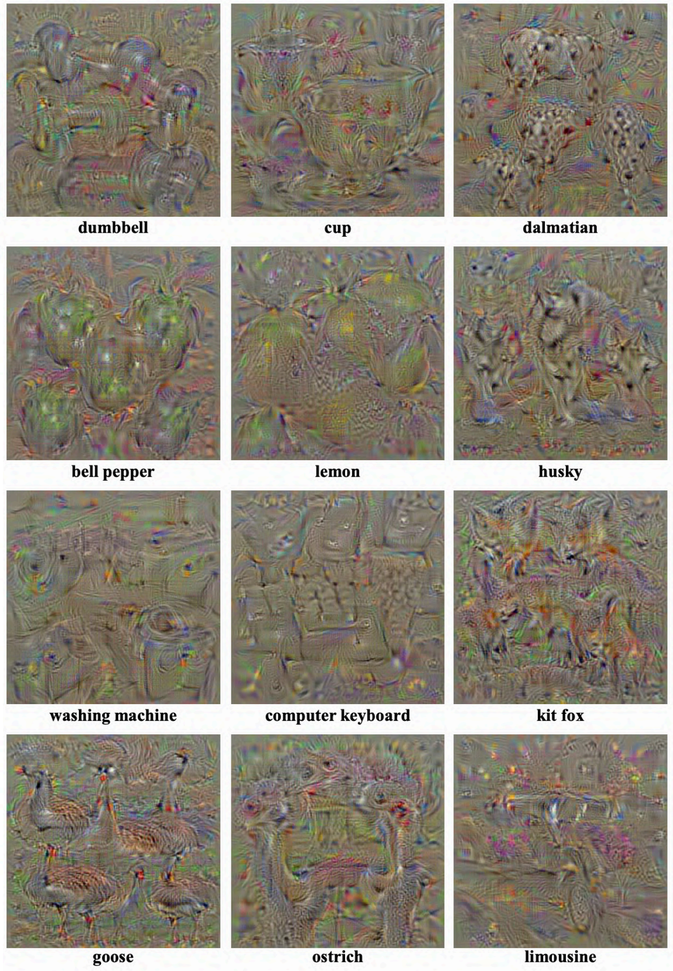
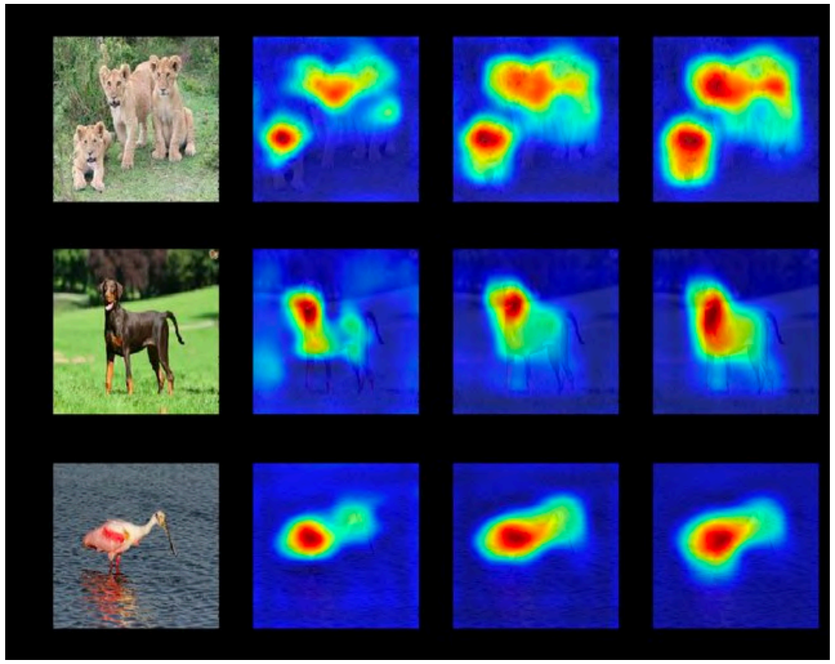


Figure 1: **Numerically computed images, illustrating the class appearance models, learnt by a ConvNet, trained on ILSVRC-2013.** Note how different aspects of class appearance are captured in a single image. Better viewed in colour.

Visualising CNNs
Gradient weighted Class Activated Maps

$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}}$$

$$L_{\text{Grad-CAM}}^c = ReLU\underbrace{\left(\sum_k \alpha_k^c A^k\right)}_{\text{linear combination}}$$



Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization

Ramprasaath R. Selvaraju · Michael Cogswell · Abhishek Das · Ramakrishna Vedantam · Devi Parikh · Dhruv Batra

CAM] FCN
SEMANTIC]
OBJECT DETECTION