NATIONAL RESEARCH UNIVERSITY
HIGHER SCHOOL OF ECONOMICS

Laboratory of Algorithms and Technologies for Network Analysis (LATNA)

EFFICIENT IMAGE RECOGNITION WITH CONVOLUTIONAL NEURAL NETWORKS

Andrey V. Savchenko
Dr. of Sci., Prof.
Email: avsavchenko@hse.ru
URL: www.hse.ru/en/staff/avsavchenko

Summer School on Operational Research and Applications - 2018

- **Development of Preference Prediction Engine using Visual Data**
  - **Deep understanding of user characteristics by analyzing user images and videos in a mobile device.**
  - **Categorizing user's characteristics (taxonomy, classification, demographics, hobbies, occupation, lifestyle, etc.) → Generate user profile**

Depending on the user profile, recommends [Product / Shop / Content] to suitable users

1. **Introduction to image processing**
2. **Convolutional Neural Networks (CNN)**
3. **Performance optimization in image recognition**

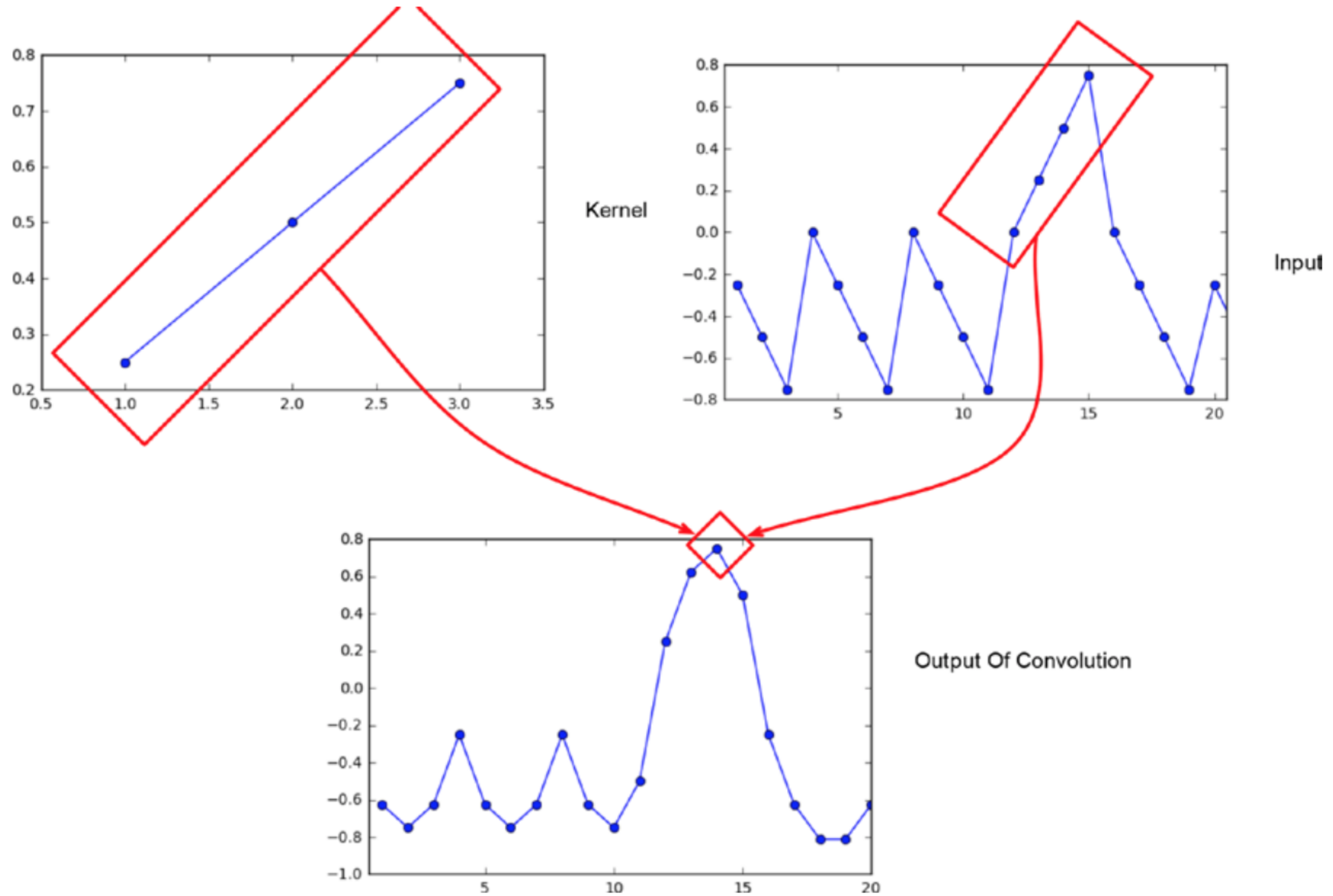
Introduction to image processing

Convolution

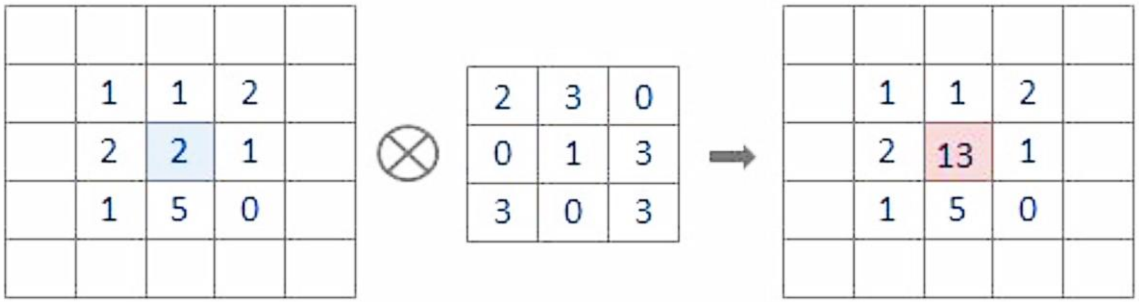
$$s(t) = \sum_a I(t-a) \cdot K(a)$$

Cross-correlation

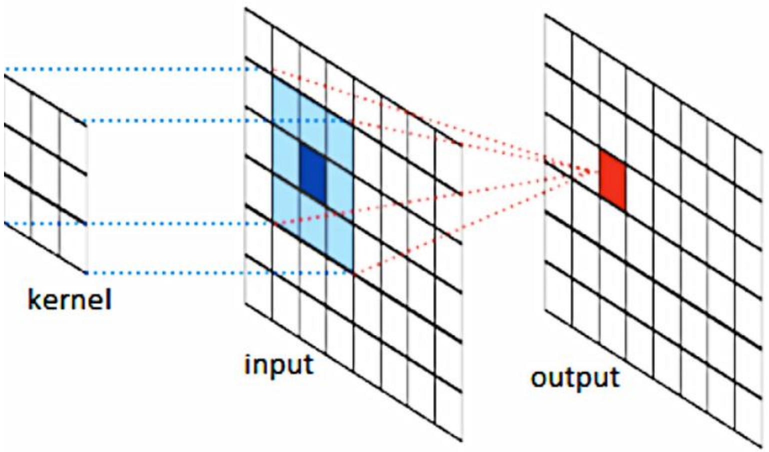
$$s(t) = \sum_a I(t+a) \cdot K(a)$$

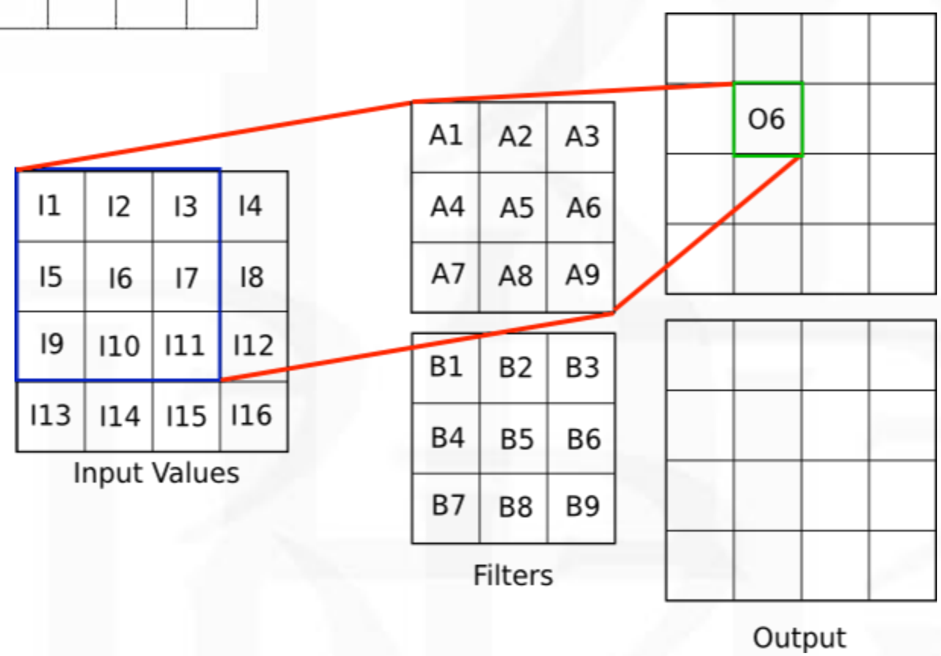


Ketkar "Deep Learning with Python"



$$x_{ij} = \sum_{m=-M}^{M} \sum_{n=-N}^{N} p_{i-m,j-n} f_{m,n}$$



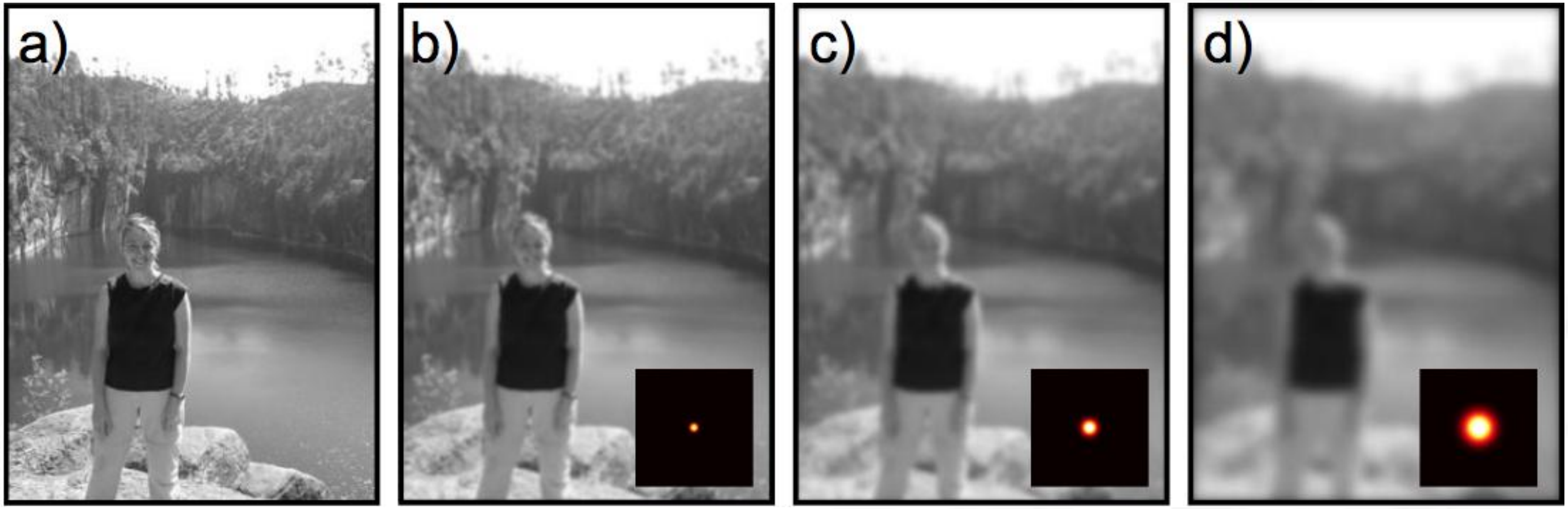


http://intellabs.github.io/RiverTrail/tutorial/

OpenCV: filter2D()

DLI-Teaching-Kit

$$f(m,n) = \frac{1}{2\pi\sigma^2}\exp\left[-\frac{m^2+n^2}{2\sigma^2}\right]$$



Ponce «Computer vision: models, learning and inference»

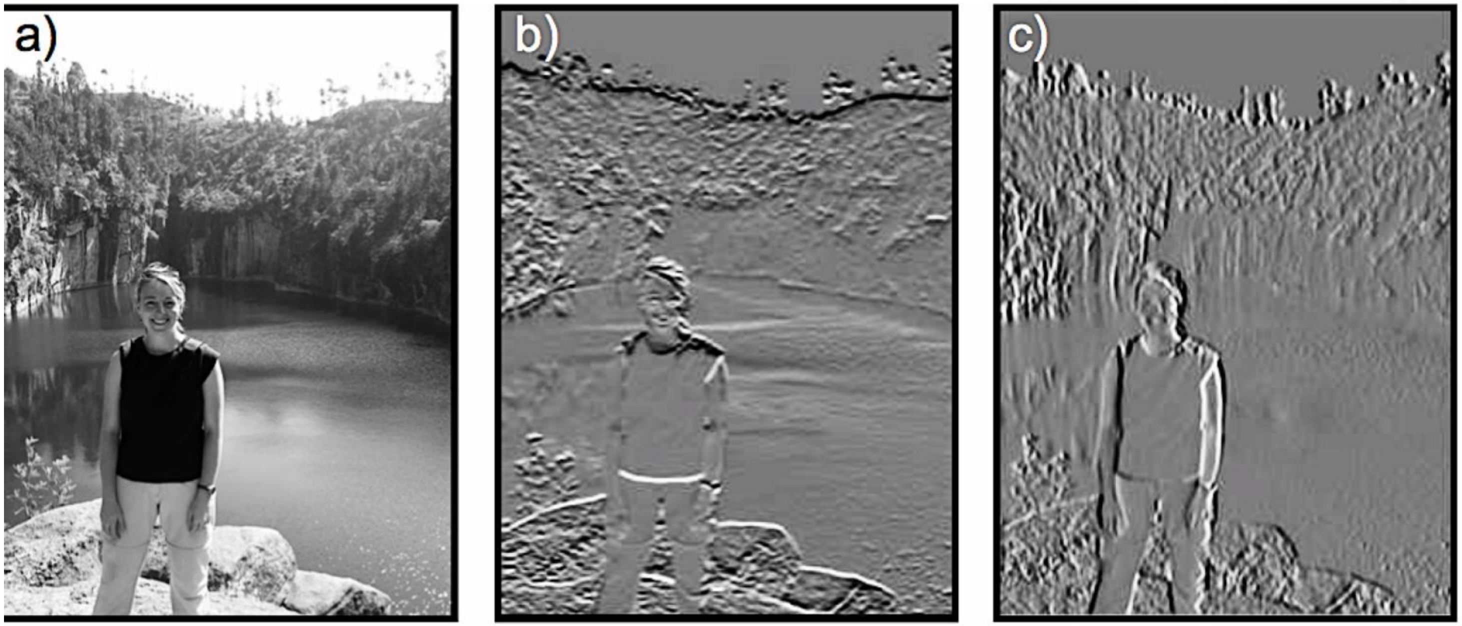
OpenCV: GaussianBlur()

Prewitt filter

$$\mathbf{F}_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}, \qquad \mathbf{F}_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Sobel filter

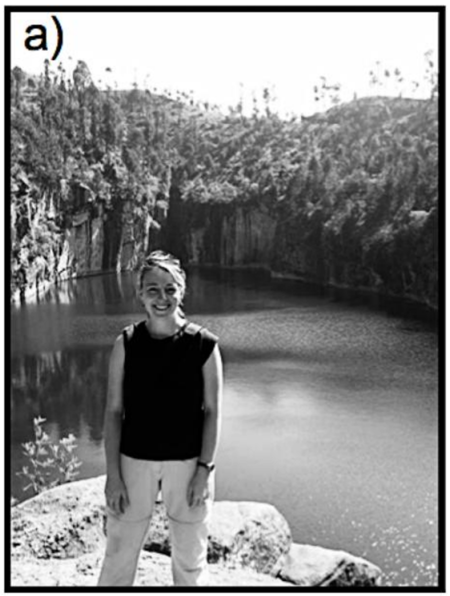
$$\mathbf{F}_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \qquad \mathbf{F}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$



Original image — Prewitt (vertical) — Prewitt (horizontal)

Ponce «Computer vision: models, learning and inference»

Edge detection (2). Second derivatives. Laplacian filter

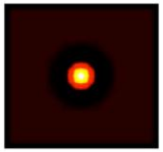


Original image

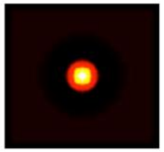
Laplacian

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Laplacian of Gaussian



Difference of Gaussians

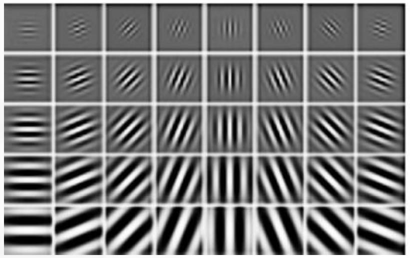


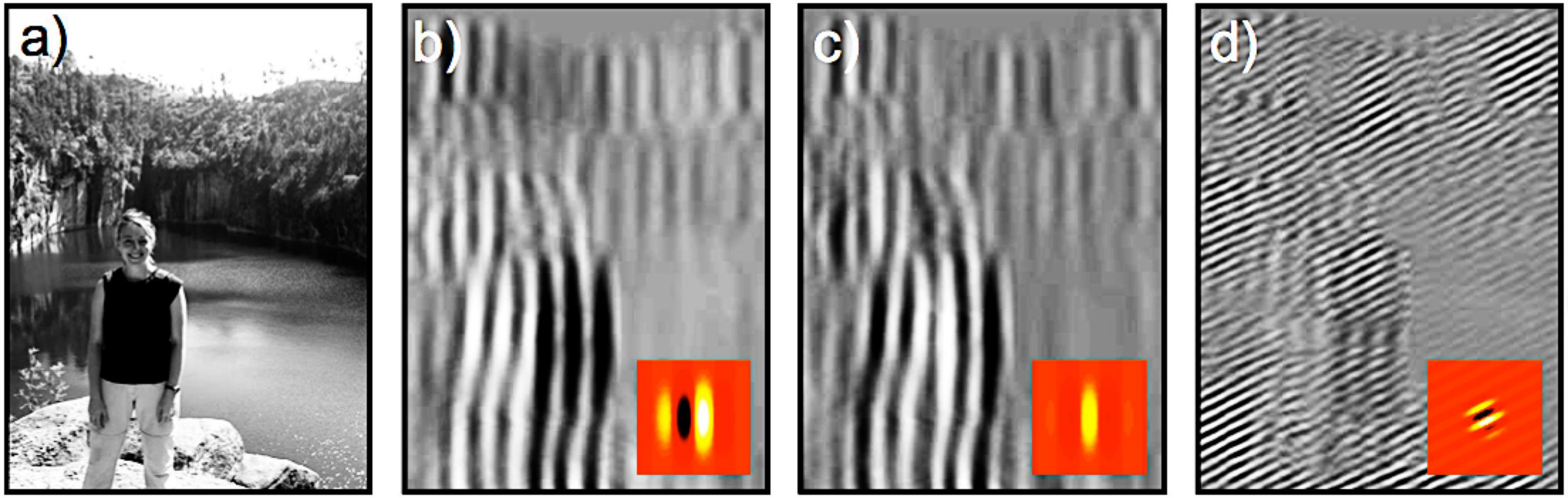
$$LoG(x;\sigma) = (\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2})G(x;\sigma)$$

OpenCV:
Laplacian(), filter2D

Ponce «Computer vision: models, learning and inference»

$$f_{mn} = \frac{1}{2\pi\sigma^2}\exp\left[-\frac{m^2+n^2}{2\sigma^2}\right]\sin\left[\frac{2\pi(\cos[\omega]m+\sin[\omega]n)}{\lambda}+\phi\right]$$



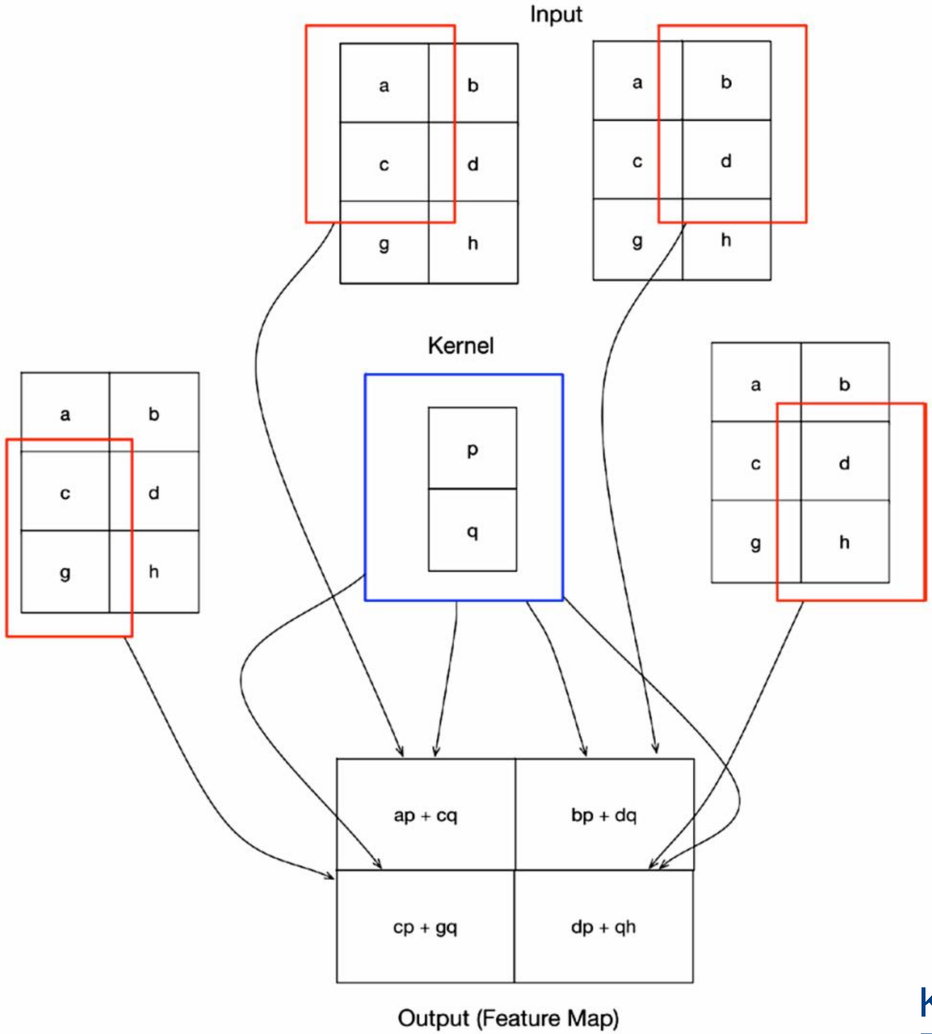


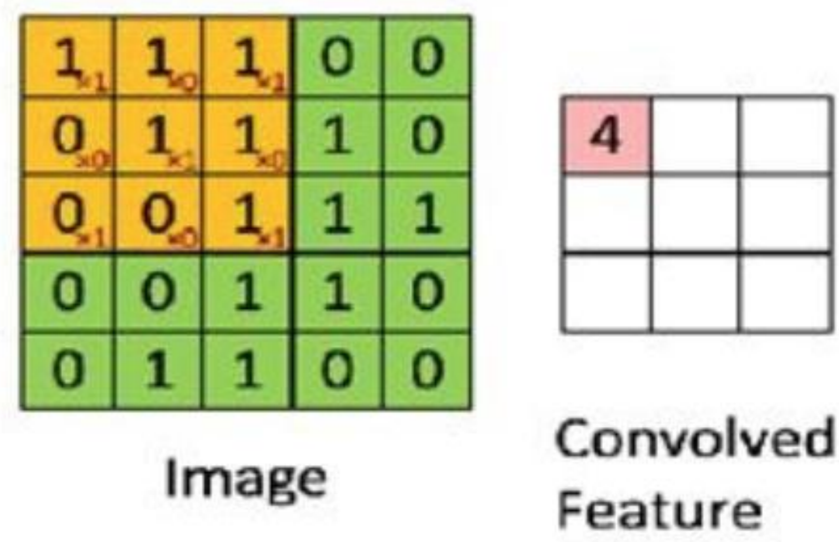
OpenCV: getGaborFilter()

Ponce «Computer vision: models, learning and inference»

Convolutional Neural Networks (CNN)

Convolution layer





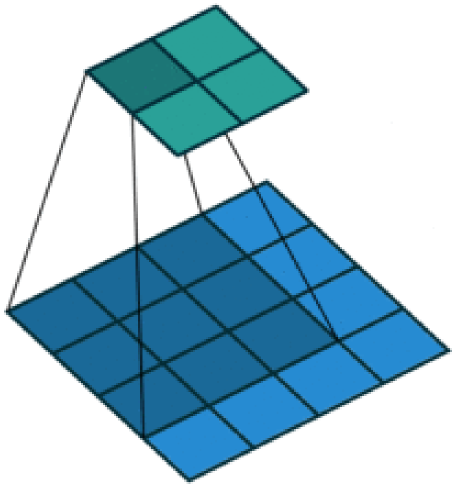
Filter weights are learnable

Ketkar "Deep Learning with Python"
Zaccone et al, Deep learning with TensorFlow

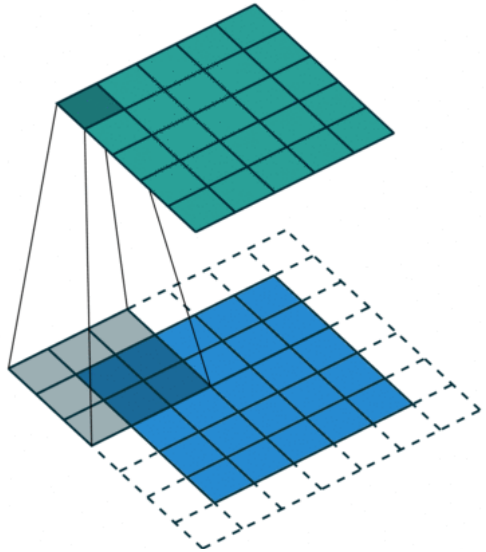
Filter weights are learnable

1. **Kernel size K**
2. **Number of filters (feature map), depth D**
3. **Stride S**
4. **Zero padding Z**

Z=0



Z=1



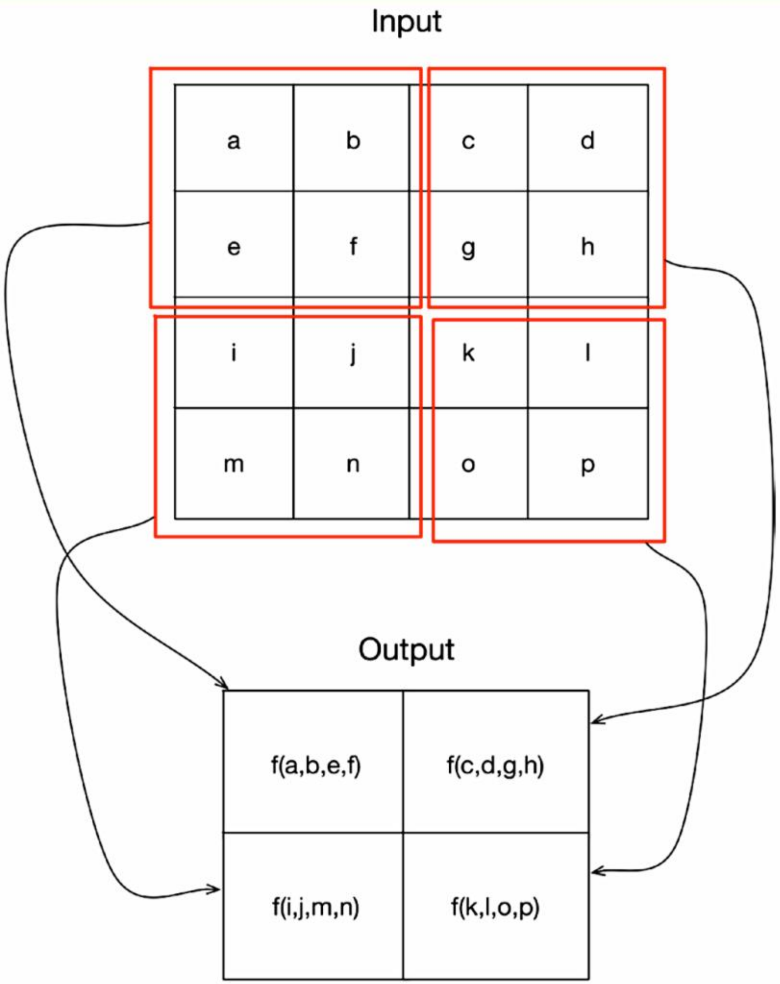
Output width/height

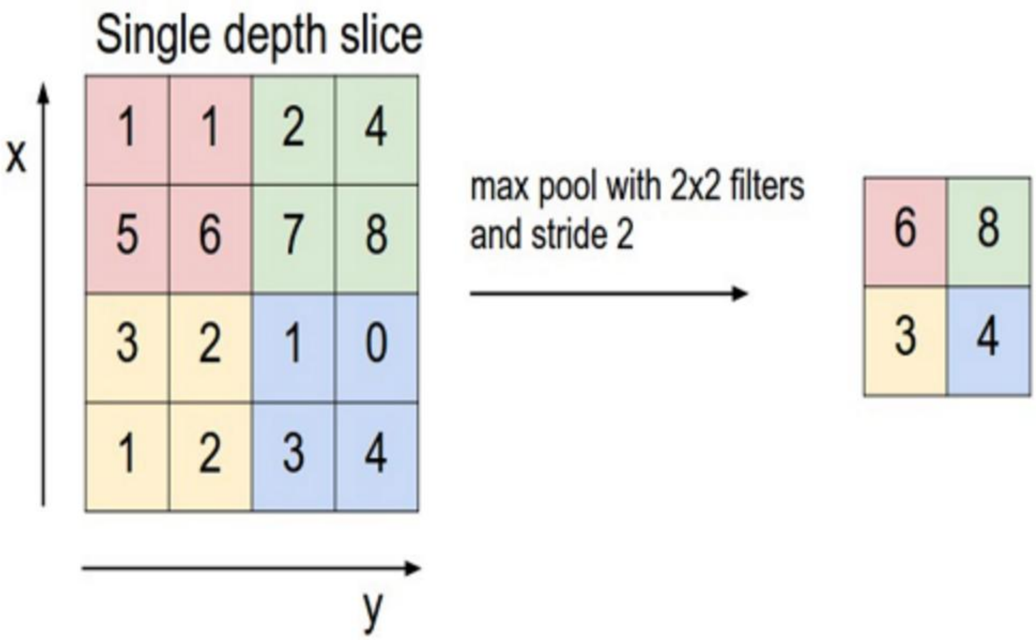
$$O = \frac{(W - K + 2P)}{S}$$

$$O_m(i,j) = a\left(\sum_{d=1}^{D}\sum_{u=-2k-1}^{2k+1}\sum_{v=-2k-1}^{2k+1} F^{(1)}_{m_d}(u,v) I_d(i-u, j-v)\right) \quad m = 1, \cdots, n$$

https://gist.github.com/akiross/754c7b87a2af8603da78b46cdaaa5598

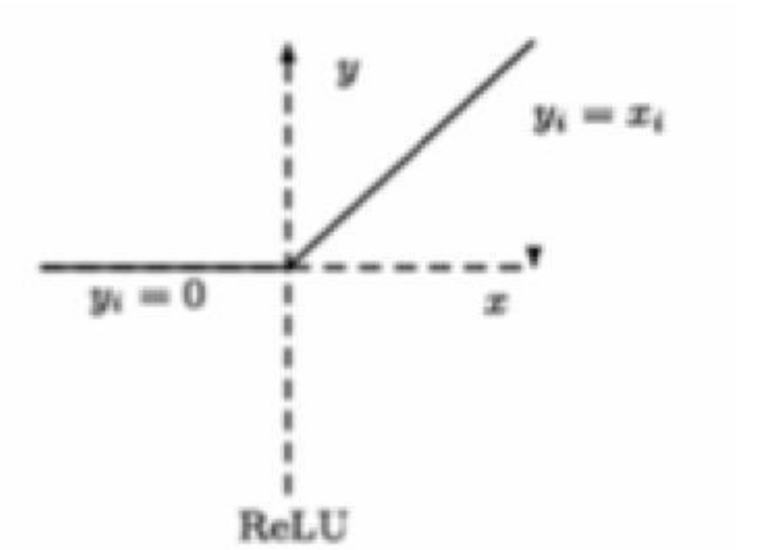
Pooling layer (max, avg,…)



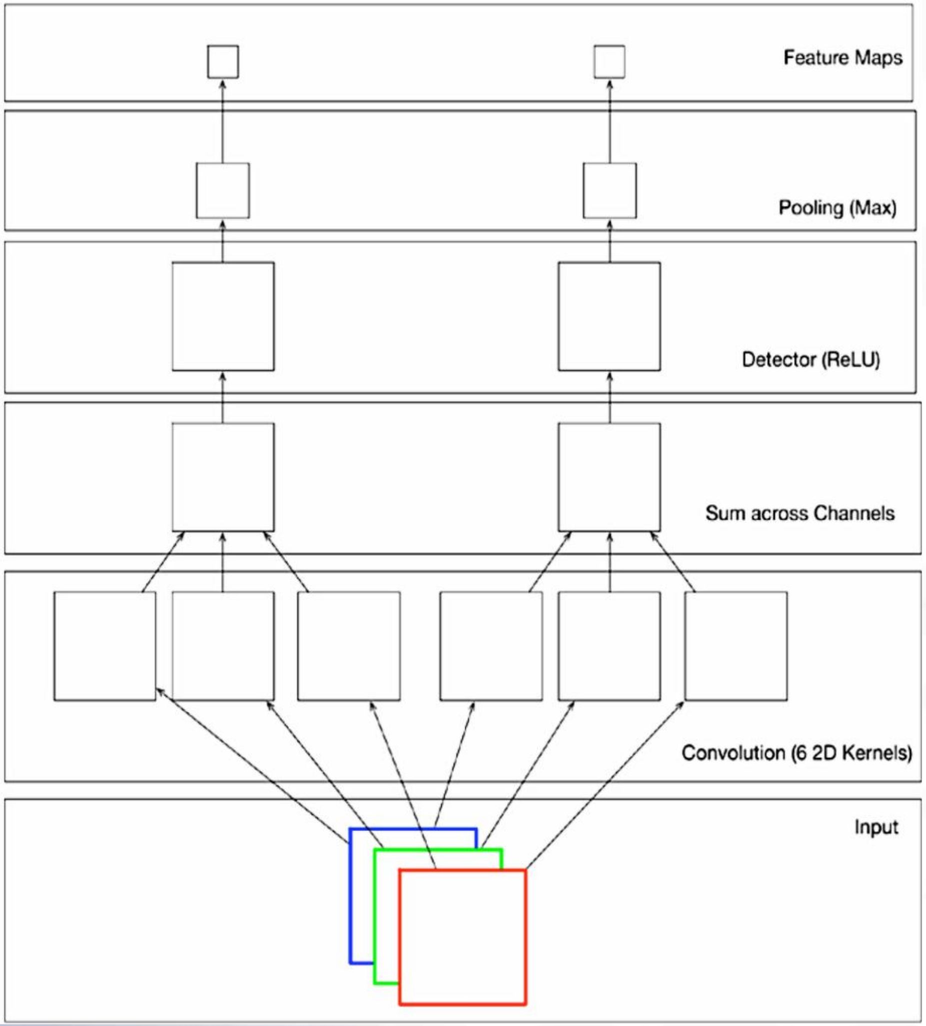


Ketkar "Deep Learning with Python"
https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/

Detector – ReLU (Rectified Linear Unit) nonlinearity

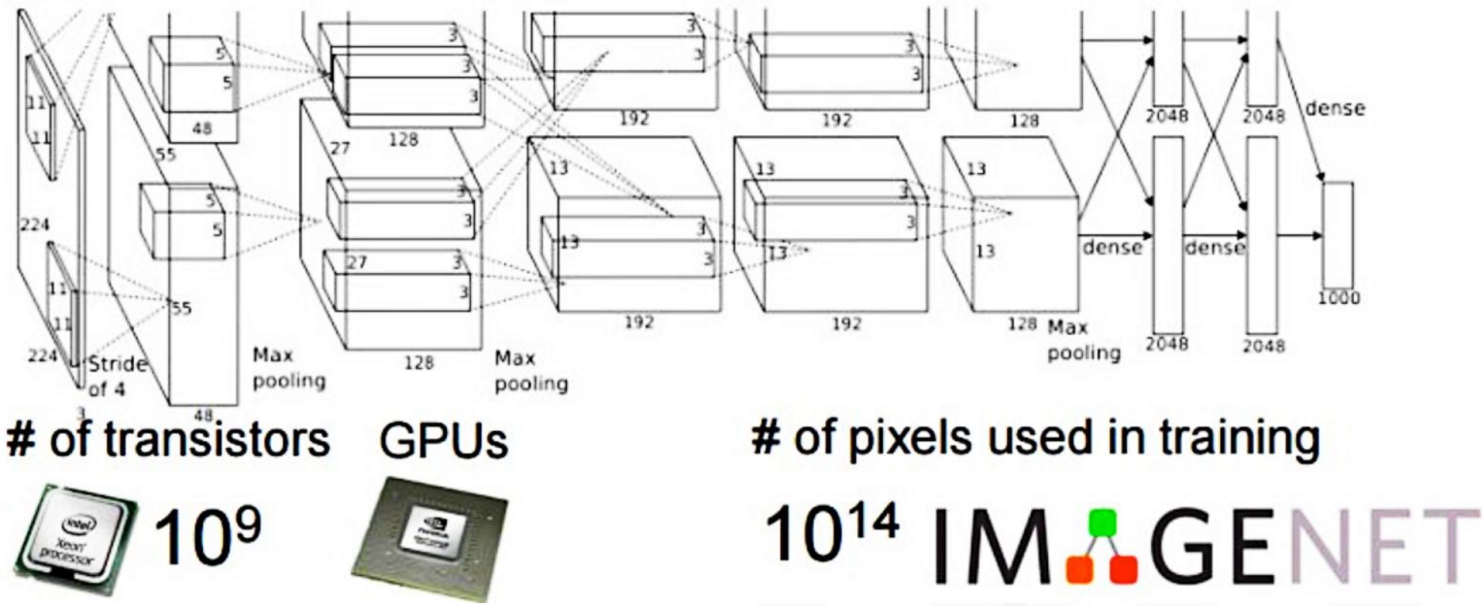


Convolution-detector-pooling block



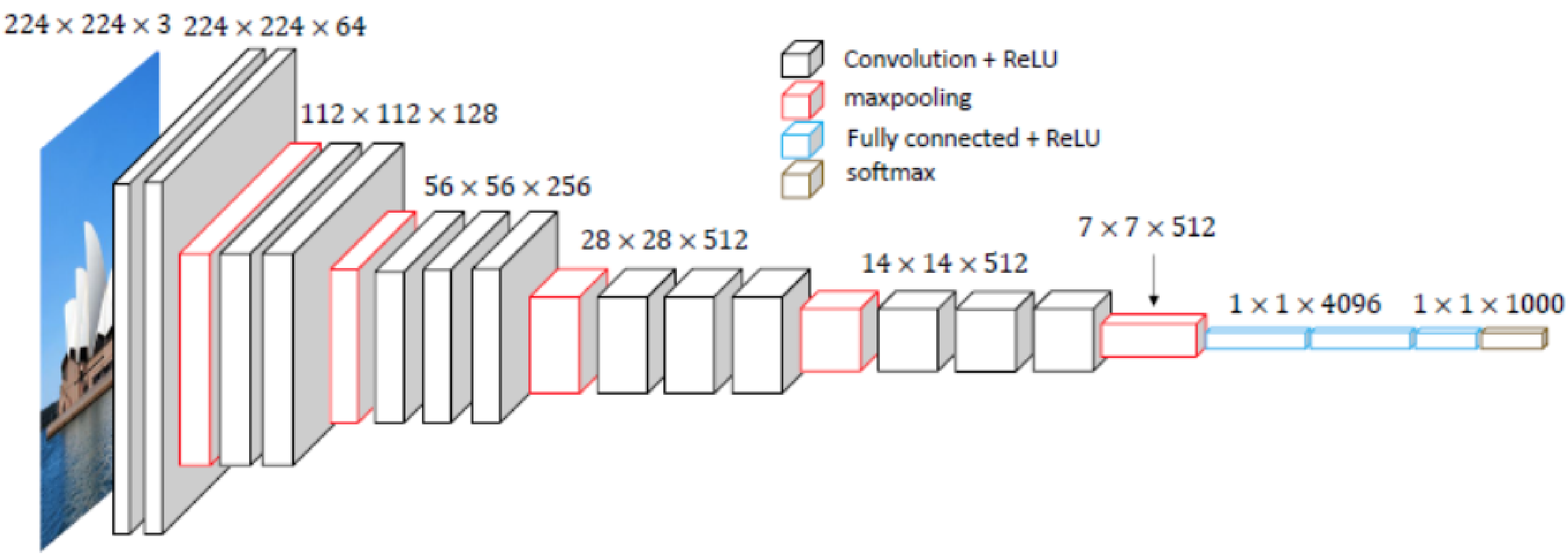
Ketkar "Deep Learning with Python"

[Krizhevsky, Sutskever, Hinton 2012]



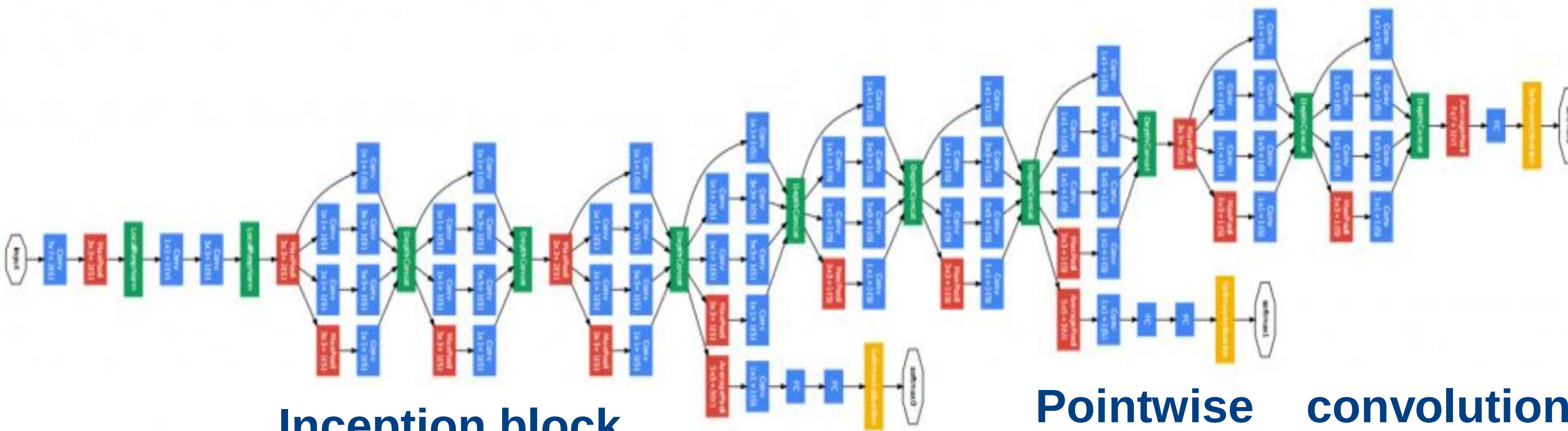
- Deep CNN trained with back propagation using NVIDIA GPU «with all the tricks Yann came up with in the last 20 years, plus dropout" (Hinton, NIPS 2012)»
- Top-5 Error rate: 15%. Previous state of the art: 25%
- Convolutions 11x11, 5x5, 3x3, max pooling, dropout, data augmentation, ReLU, SGD with momentum
- 60M parameters

CNN architectures (2). VGGNet (Oxford visual Geometry Group)

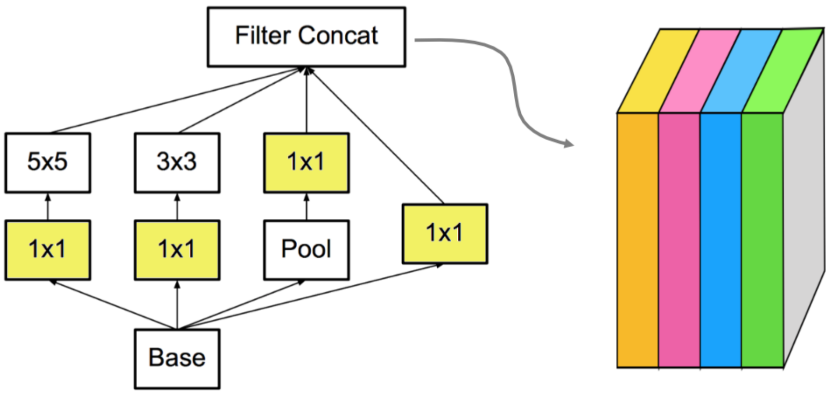


- **Only 3x3 convolutions, but many more filters**
- **ImageNet Top-5 Error rate: 8%**
- **138M parameters**

image
Conv-64
Conv-64
maxpool
Conv-128
Conv-128
maxpool
Conv-256
Conv-256
maxpool
Conv-512
Conv-512
Conv-512
maxpool
Conv-512
Conv-512
Conv-512
maxpool
fc-4096
fc-4096
fc-1000
Softmax

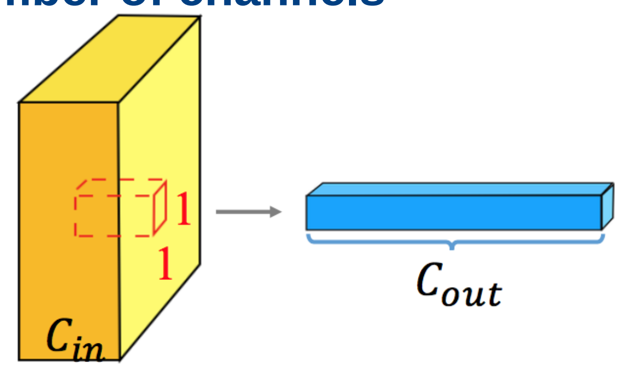


Inception block



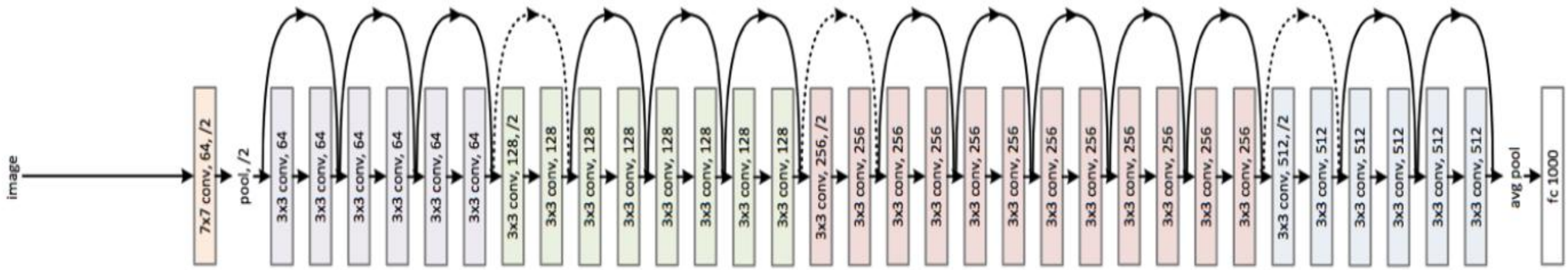
Christian Szegedy, https://arxiv.org/pdf/1512.00567.pdf

Pointwise convolutions: 1x1 convolution to reduce the number of channels



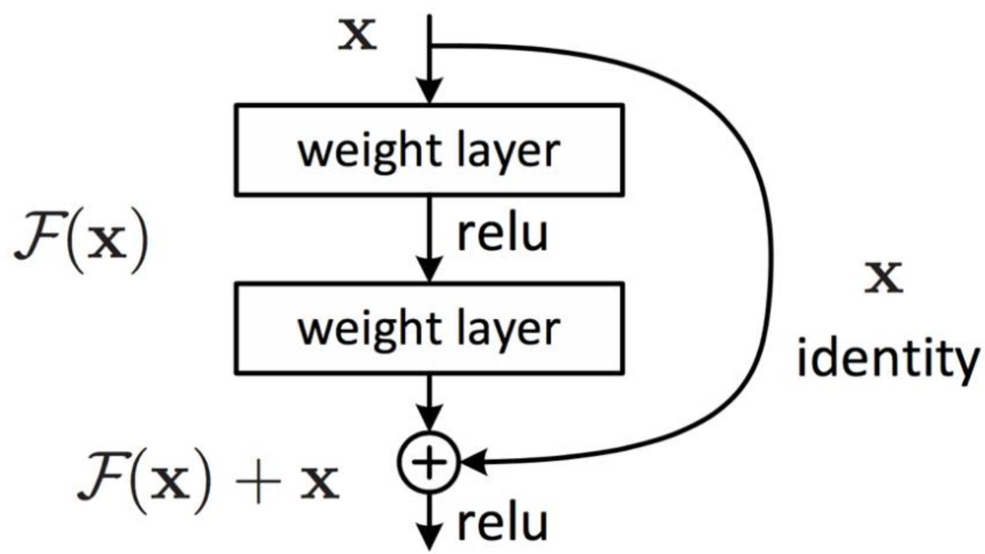
- **Batch norm, RMSProp**
- **ImageNet Top-5 Error rate (v3): 5.6% (single model), 3.6% (ensemble)**
- **25M parameters**

CNN architectures (4). Residual Network (ResNet)



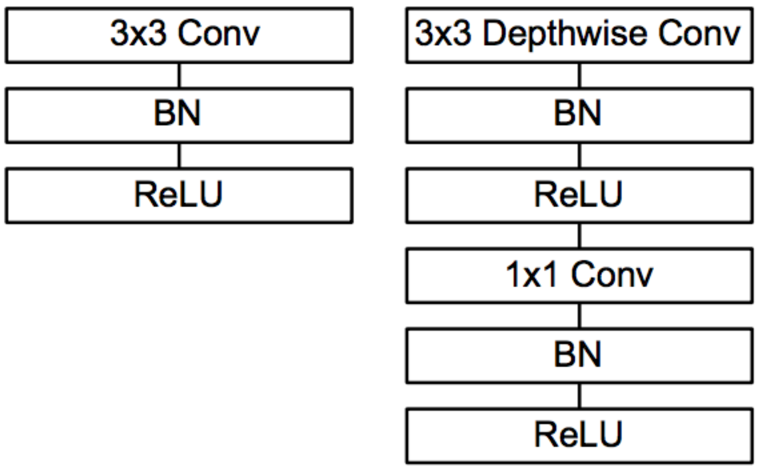
- 50/101/152 layers, several 7x7 convolutions, many 3x3 convolutions, batch norm, max/average pooling
- ImageNet Top-5 Error rate (v3):4.5% (single model), 3.5% (ensemble)
- 60M parameters for ResNet-152

Residual block



CNN architectures (5). MobileNet

Depthwise convolution



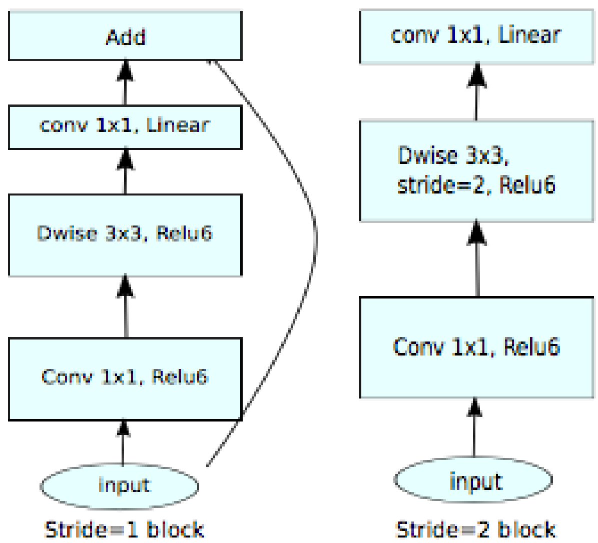
- **28 layers**
- **ImageNet Top-5 Error rate: 12.81%**
- **4.2M parameters**

https://arxiv.org/abs/1704.04861

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

CNN architectures (6). MobileNet v2

Bottleneck residual block



Input	Operator	Output
$h \times w \times k$	1x1 conv2d , ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3x3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	linear 1x1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

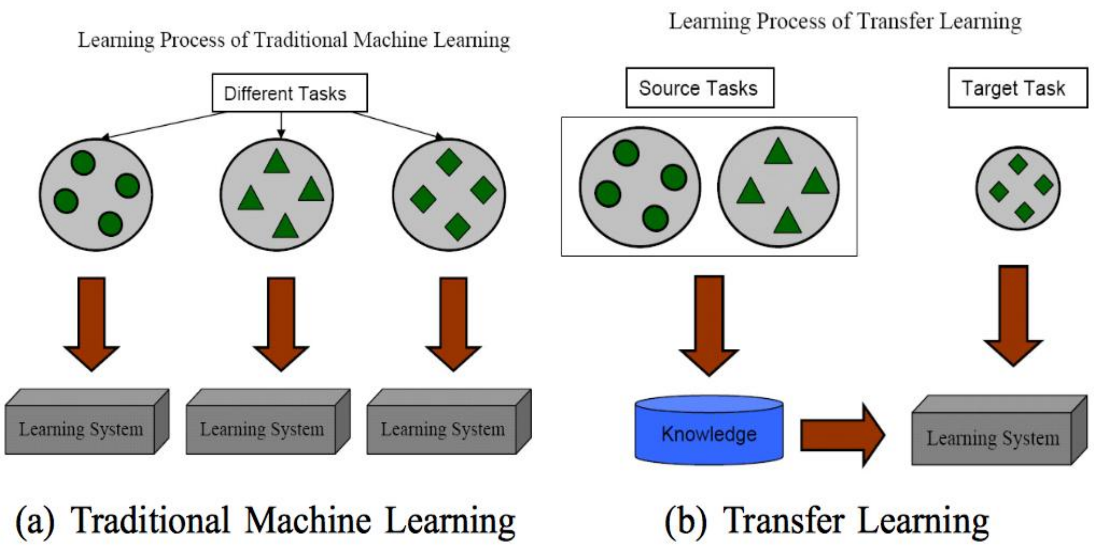
Each line describes a sequence of 1 or more identical (modulo stride) layers, repeated n times.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$28^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times k$	conv2d 1x1	-	k	-	

– **ImageNet Top-1 Accuracy: 74.7% vs 70.6% for MobileNet v1**
– **6..9M parameters**

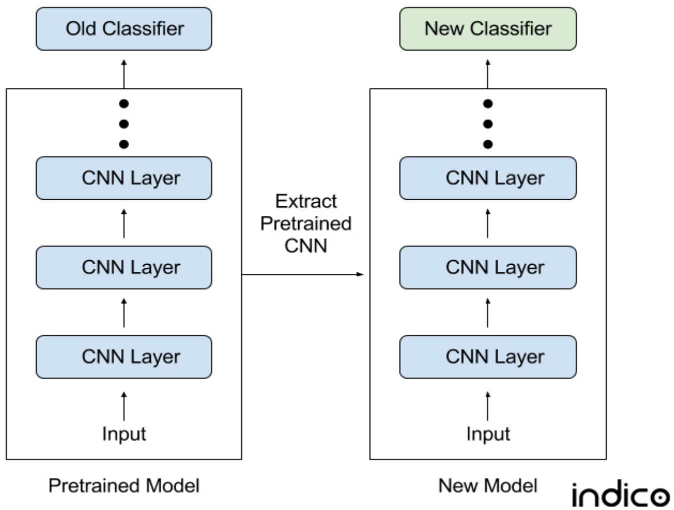
http://arxiv.org/abs/1801.04381

Practical usage: domain adaptation & transfer learning (1)



(a) Traditional Machine Learning (b) Transfer Learning

Fine-tuning



OverFeat features → Trained classifier on other data sets
[Razavian, Azizpour, Sullivan, Carlsson "CNN features off-the-shelf: An astounding baseline for recogniton", CVPR 2014], http://www.csc.kth.se/cvap/cvg/DL/ots/

• image classification	ImageNet LSVRC 2013	competitive	13.6 % error
	Dogs vs Cats Kaggle challenge 2014	state of the art	98.9%
• object localization	ImageNet LSVRC 2013	state of the art	29.9% error
• object detection	ImageNet LSVRC 2013	state of the art	24.3% mAP

Performance optimization in image recognition

- **Modern hardware**
- **Parallel computing**
- **One CNN for many tasks (many outputs)**
- **CNN compression**
- **Fast pattern recognition algorithms:**

➢ Specially designed fast classifiers
➢ Approximate nearest neighbor methods
➢ Sequential analysis

[Han et al. 2016] Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding – **ICLR16 Best paper**

- **Pruning**
  [Han et al. 2016], [Molchanov et al. 2016]
- **Distillation The Knowledge (FitNet)**
  [Hinton et al. 2014], [Romero et al. 2014]
- **Weights Hashing / Quantization**
  [Chen et al. 2015], [Han et al. 2016]
- **Tensor Decompositions**
  [Lebedev et al. 2015], [Kim et al. 2015], [Novikov et al. 2015], [Garipov et al. 2016]
- **Binarization**
  [Courbariaux / Hubara et al. 2016], [Rastegari et al. 2016], [Merolla et al. 2016], [Hou et al. 2016]
- **Architectural tricks**
  [Hong et al. 2016], [Iandola et al. 2016], [Teerapittayanon et al. 2016]

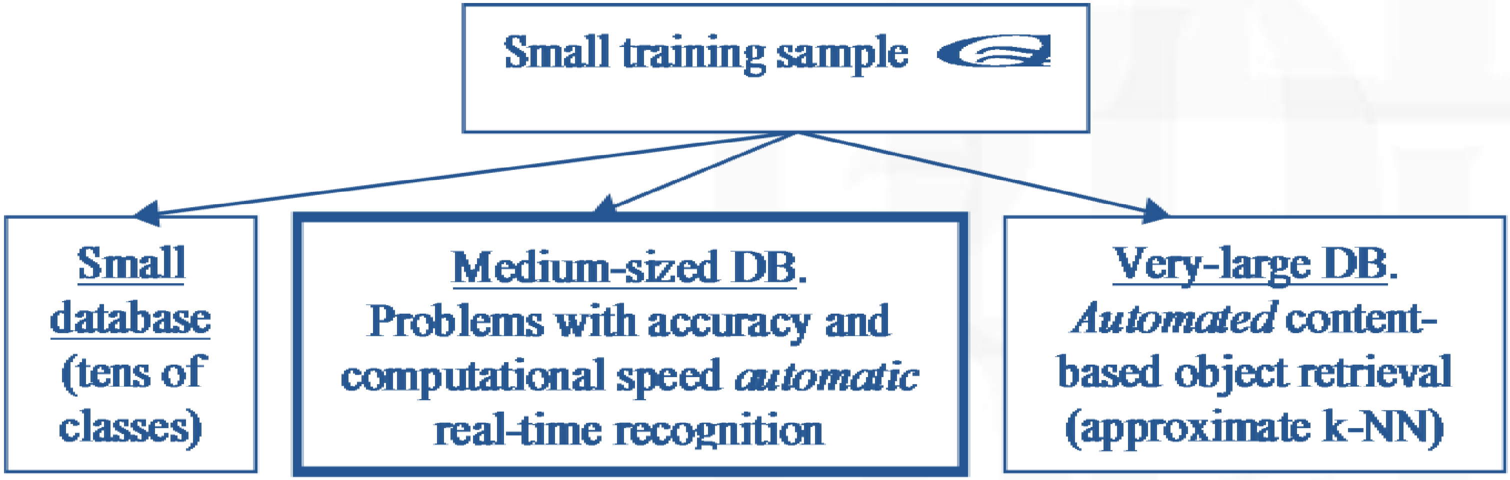
[Rassadin, Savchenko, 2017]

CNN compression (2). Visual emotion recognition (Radboud Faces Database)

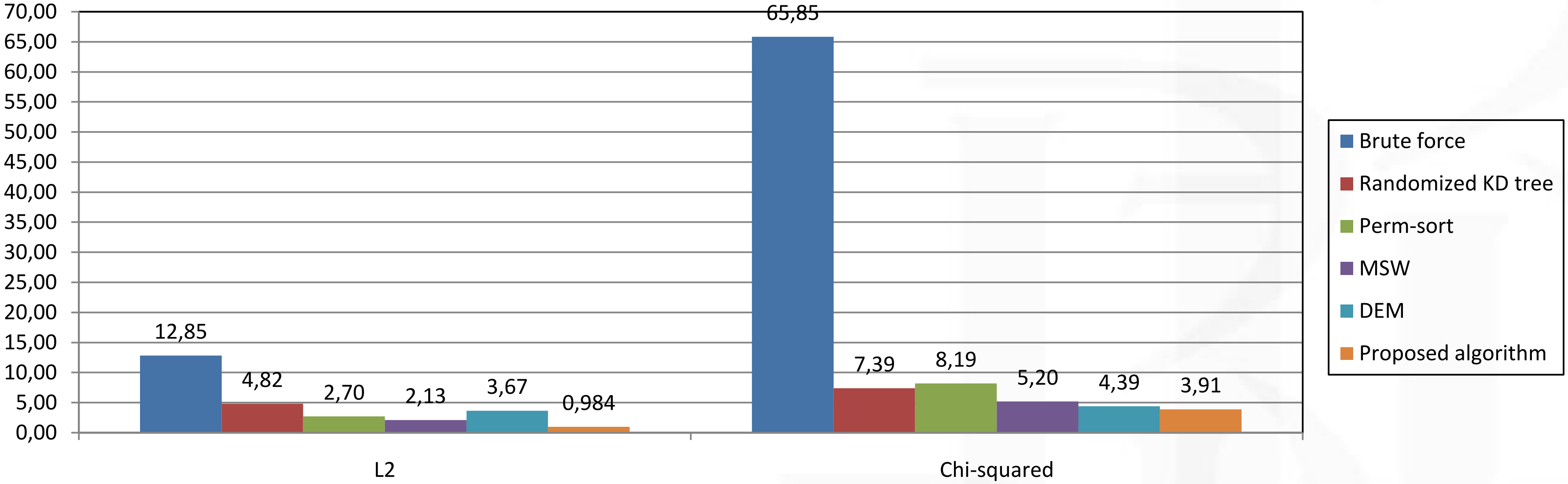
	Training time per epoch, ms	Inference time, ms	Model size, MB	Accuracy, %
VGG-S (baseline)	43.7	33.4	372.2	97.13
SqueezeNet-1.1 (baseline)	**22.94**	**4.94**	2.8	89.14
SqueezeNet-1.1, CP-Decomposition	**22.94**	7.74	**2.1**	87.5
HashedNets	294.8	158.2	68.3	96.31
Binary-Weight-Network (BWN)	83.8	33.5	11.6	**98.57**
XNOR-Net	84.3	34.2	11.6	58.81
XNOR-Net w/o weights activation	43.4	34.1	11.6	88.32

[Rassadin, Savchenko, 2017]

1. **ANN library** (Arya, Mount, etc. // Journal of the ACM, 1998): kd-trees. Only Minkowski distances are supported.
2. Hashing Techniques, i.e. **Google Correlate** (Vanderkam, Schonberger, Rowley, Kumar, Nearest Neighbor Search in Google Correlate, 2013)
3. **FLANN library** (Muja, Lowe // Proc. of VISAPP, 2009)
4. **NonMetricSpaceLib** (Boytsov, Bilegsaikhan // Proc. of SISAP, LNCS, 2013)
5. **Facebook Faiss** (arXiv:1702.08734, 2017)



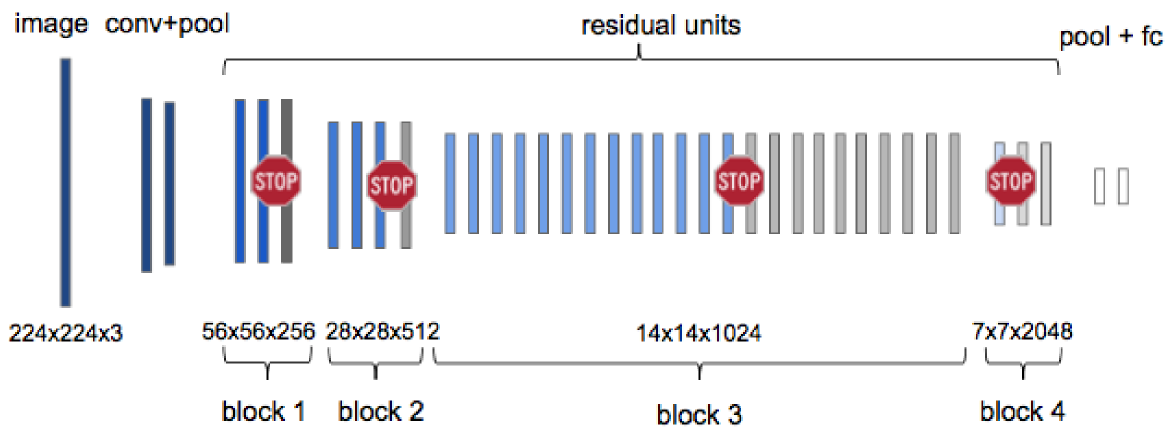
Recognition time, ms

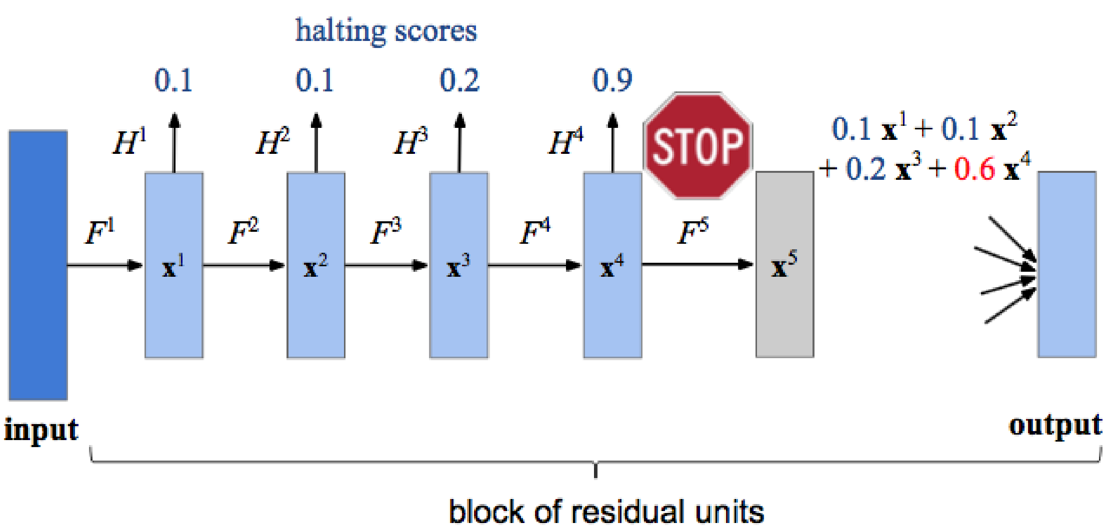


Error rate, %

	VGGNet, L2	VGGNet, chi-squared	LCNN, L2
SVM	49.6	-	28.9
1-NN (Brute force)	10.7	10.2	9.6
Randomized KD tree	11.0	10.6	9.9
Perm-sort	10.9	10.6	9.9
ML-ANN	10.9	10.6	10.5
Proposed algorithm	10.9	10.5	9.9

Sequential analysis (1). Adaptive computation time (ACT)





$$\mathbf{x}^0 = \textbf{input},$$
$$\mathbf{x}^l = F^l(\mathbf{x}^{l-1}) = \mathbf{x}^{l-1} + f^l(\mathbf{x}^{l-1})$$
$$\textbf{output} = \mathbf{x}^L.$$

$$h^l = H^l(\mathbf{x}^l) = \sigma(W^l \operatorname{pool}(\mathbf{x}^l) + b^l).$$

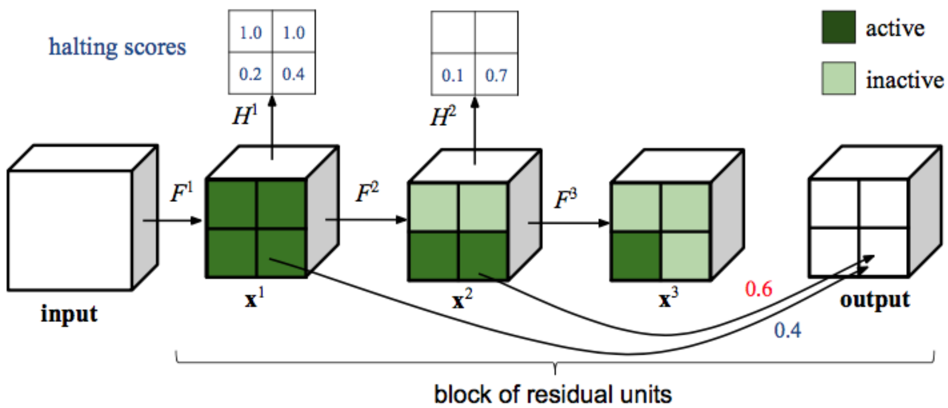
No. of residual units to evaluate:

$$N = \min\left\{n \in \{1 \dots L\} : \sum_{l=1}^{n} h^l \geq 1 - \varepsilon\right\}$$

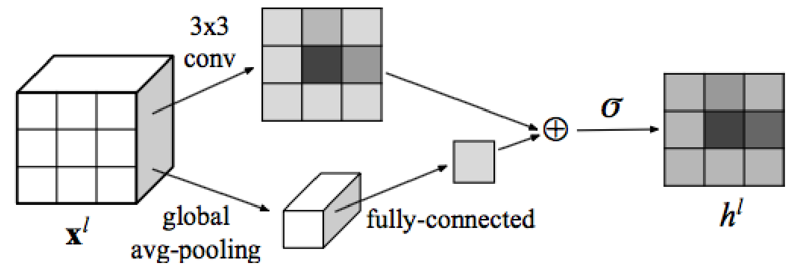
$$\text{output} = \sum_{l=1}^{L} p^l \mathbf{x}^l = \sum_{l=1}^{N} p^l \mathbf{x}^l.$$

https://arxiv.org/abs/1612.02297

Apply ACT to each spatial position of the



Halting score

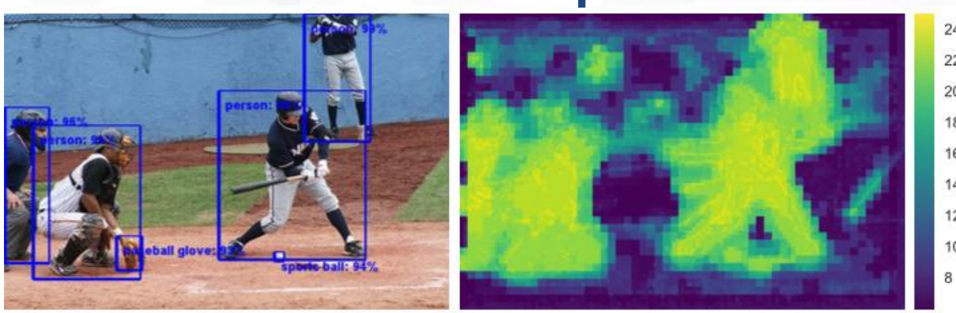


$$H^l(\mathbf{x}) = \sigma(\widetilde{W}^l * \mathbf{x} + W^l \operatorname{pool}(\mathbf{x}) + b^l)$$

COCO val set. Faster R-CNN with SACT

Feature extractor	FLOPs (%)	mAP @ [.5, .95] (%)
ResNet-101 [15]	100	27.2
ResNet-50 (our impl.)	46.6	25.56
SACT $\tau = 0.005$	**56.0 ± 8.5**	**27.61**
SACT $\tau = 0.001$	72.4 ± 8.4	29.04
ResNet-101 (our impl.)	100	29.24

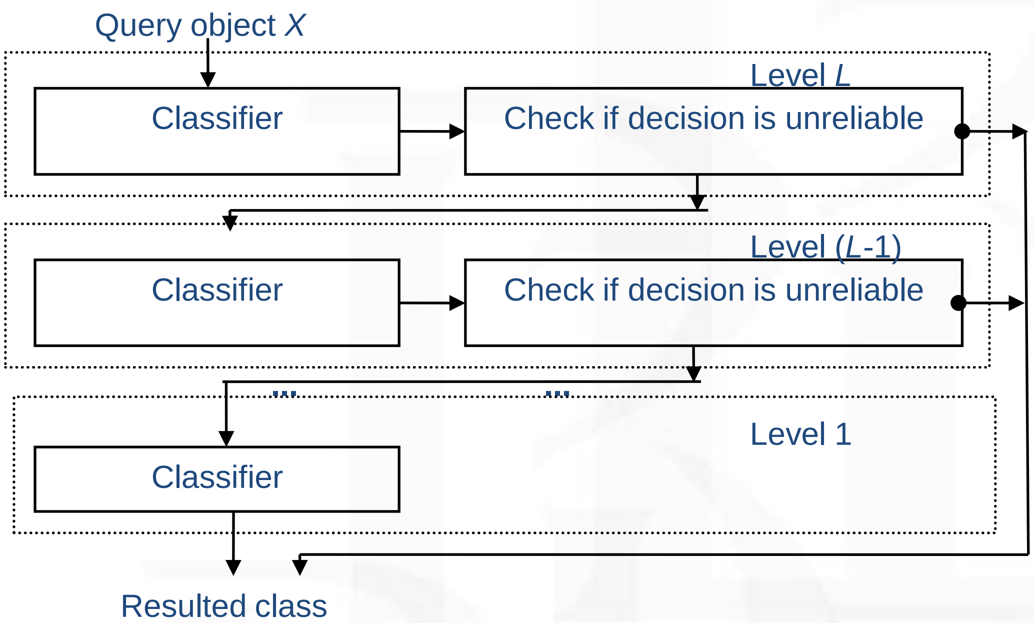
Heat map of computation time



https://arxiv.org/abs/1612.02297

Wald A. Sequential Analysis

Yao Y. Granular Computing and Sequential Three-Way Decisions //Proc. RSKT, 2013: "Objects with a non-commitment decision may be further investigated by using fine-grained granules"



Granules for off-the-shelf CNN features can be defined using PCA (**variable granulation**)

$$\rho\left(\tilde{\mathbf{x}}^{(l+1)}(t), \tilde{\mathbf{x}}_r^{(l+1)}\right) = $$

$$= \rho\left(\tilde{\mathbf{x}}^{(l)}(t), \tilde{\mathbf{x}}_r^{(l)}\right) + \sum_{d=d^{(l)}+1}^{d^{(l+1)}} \rho(\tilde{x}_d(t), \tilde{x}_{r;d})$$

$$\rho_c(\tilde{\mathbf{x}}^{(l)}(t)) = \min_{r \in \{1,\dots,R\}, c(r)=c} \rho(\tilde{\mathbf{x}}^{(l)}(t), \tilde{\mathbf{x}}_r^{(l)})$$

$$c_1^{(l)}(t) = \operatorname{argmin} \rho_c(\tilde{\mathbf{x}}^{(l)}(t))$$

$$C^{(l+1)}(t) = \left\{ c \in C^{(l)}(t) \,\middle|\, \frac{\rho_c(\tilde{\mathbf{x}}^{(l)}(t))}{\rho_{c_1^{(l)}(t)}(\tilde{\mathbf{x}}^{(l)}(t))} \le \delta \right\}$$

Still-to-video recognition results

YTF/LFW

Aggregation	Classifier	ResFace Accuracy (%)	ResFace Performance (ms)	VGGFace Accuracy (%)	VGGFace Performance (ms)	VGGFace2_ft Accuracy (%)	VGGFace2_ft Performance (ms)
None	SVM	18.53±0.5	10213±33	22.49±0.6	19840±40	52.85±1.2	10137±35
Pooling (2)	SVM	22.15±1.2	10269±39	26.90±0.7	20070±37	55.91±1.3	10252±42
None	k-NN (1)	30.92±0.1	12.3±0.0	43.44±0.1	23.3±0.1	74.48±0.1	12.7±0.0
None	MAP (7)	31.49±0.0	12.3±0.0	43.50±0.1	23.7±0.1	74.41±0.1	12.6±0.1
Pooling (2)	k-NN (1)/MAP (7)	31.33±0.2	14.1±0.2	44.23±0.3	25.5±0.2	74.87±0.1	13.9±0.2
None	k-NN (1), 32 PCA	26.28±0.3	1.7±0.0	37.69±0.1	2.2±0.1	58.81±0.0	19±0.1
Pooling (2)	k-NN (1), 32 PCA	26.85±0.1	2.8±0.2	39.03±0.1	3.2±0.2	60.11±0.1	2.5±0.0
None	k-NN (1), 256 PCA	31.21±0.1	2.3±0.0	44.64±0.2	2.9±0.1	75.02±0.0	2.5±0.1
Pooling (2)	k-NN (1), 256 PCA	31.64±0.1	3.3±0.1	45.00±0.1	3.7±0.0	75.09±0.1	2.9±0.2
None	Proposed, k-NN (1)	31.21±0.1	0.8±0.0	44.64±0.1	1.1±0.1	74.77±0.1	0.7±0.0
None	Proposed, MAP (7)	31.52±0.2	0.9±0.1	45.46±0.1	1.2±0.1	74.87±0.2	0.7±0.0
Pooling (2)	Proposed, k-NN (1)	31.58±0.3	1.3±0.2	45.03±0.1	1.8±0.0	73.16±0.0	1.1±0.0

IJB-A

Aggregation	Classifier	ResFace Accuracy (%)	ResFace Performance (ms)	VGGFace Accuracy (%)	VGGFace Performance (ms)	VGGFace2_ft Accuracy (%)	VGGFace2_ft Performance (ms)
None	SVM	55.98±0.3	117.3±0.9	70.88±0.3	230.3±1.6	84.12±0.4	117.8±1.1
Pooling (2)	SVM	57.17±0.5	114.7±0.7	73.62±0.0	233.0±4.7	87.06±0.1	124.4±1.4
None	k-NN (1)	54.41±0.0	12.9±0.0	68.88±0.3	24.9±0.2	85.67±0.2	12.8±0.1
None	MAP (7)	55.16±0.2	13.3±0.3	70.50±0.8	25.0±0.0	87.09±0.1	12.8±0.1
Pooling (2)	k-NN (1)/MAP (7)	50.35±0.2	13.7±0.0	66.91±0.6	26.3±0.3	85.69±0.1	13.3±0.1
None	k-NN (1), 32 PCA	48.49±0.4	1.2±0.1	66.26±0.3	1.9±0.2	77.91±0.2	1.3±0.1
Pooling (2)	k-NN (1), 32 PCA	45.86±0.4	2.0±0.1	63.80±0.2	2.3±0.1	78.61±0.5	1.7±0.1
None	k-NN (1), 256 PCA	53.91±0.3	1.9±0.1	70.94±0.1	2.5±0.1	85.48±0.1	2.1±0.0
Pooling (2)	k-NN (1), 256 PCA	50.50±0.5	2.6±0.1	69.07±0.5	2.8±0.1	85.44±0.3	2.2±0.1
None	Proposed, k-NN (1)	54.12±0.2	0.8±0.1	70.94±0.1	1.2±0.0	85.65±0.2	0.8±0.0
None	Proposed, MAP (7)	56.01±0.1	0.8±0.1	73.06±0.5	1.2±0.1	86.89±0.1	0.8±0.1
Pooling (2)	Proposed, k-NN (1)	50.79±0.6	1.2±0.1	68.80±0.3	1.6±0.0	85.17±0.4	0.9±0.1

Image categorization results, Caltech-101 dataset

Caltech-101

Classifier	Inception v1		VGGNet-19	
	α, %	$\bar{t}$, ms	α, %	$\bar{t}$, ms
SVM	8.76	3.89	8.9	16.46
RF	13.43	0.25	15.69	0.3
NN, all	13	3.39	14.94	11.88
NN, 256 features	13.54	0.86	14.04	0.87
NN, 64 PCA features	14.5	0.29	16.65	0.29
TWD (15), (16), posteriors (4)	13.7	0.51	14.4	0.64
TWD (15), (16), BF (8)	12.12	0.77	13.92	0.74
TWD (15), (16), DF (22)	13.52	0.45	14.91	0.48
Proposed TWD, $m = 32$	13.84	0.25	14.4	0.25
Proposed TWD, $m = 64$	12.02	0.34	13.74	0.33

Caltech-256

Classifier	Inception v1		VGGNet-19	
	α, %	$\bar{t}$, ms	α, %	$\bar{t}$, ms
SVM	34.4	6.45	41.55	12.8
RF	48.12	0.42	59.82	0.28
NN, all	34.66	4.94	46.89	13.6
NN, 256 features	34.22	1.19	39.75	0.7
NN, 64 PCA features	36.56	0.36	43.12	0.28
TWD (15), (16), posteriors (4)	34.28	1.37	39.78	1.26
TWD (15), (16), BF (8)	34.37	1.63	39.86	1.22
TWD (15), (16), DF (22)	34.68	0.88	40.1	0.63
Proposed TWD, $m = 32$	34.56	0.34	40.39	0.34
Proposed TWD, $m = 64$	34.21	0.45	39.72	0.44

1. **State-of-the-art results are obtained with modern deep CNNs, but sometimes their usage is restricted due to performance issues**
2. **CNN compression is still under active research**
3. **The usage of CNNs to extract off-the-shelf features allows implementing fast classifiers including approximate nearest neighbor methods**
4. **Sequential analysis of CNN features/layers can potentially provide high performance without losses in accuracy**

NATIONAL RESEARCH
UNIVERSITY

Thank you!