CNNs
from the Basics to Recent Advances

Dmytro Mishkin
Center for Machine Perception
Czech Technical University in Prague
ducha.aiki@gmail.com

Outline

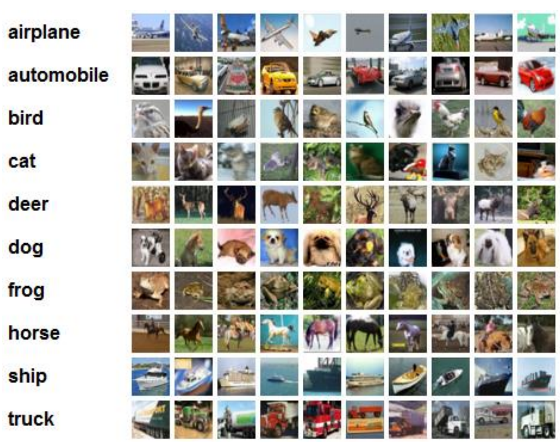
- Short review of the CNN design
- Architecture progress

  AlexNet → VGGNet → ResNet → now…

  → GoogLeNet
- VGGNet is an universal design?
- Automatic architecture search
- Design choices

DATASETS USED IN PRESENTATION: IMAGENET AND CIFAR-10

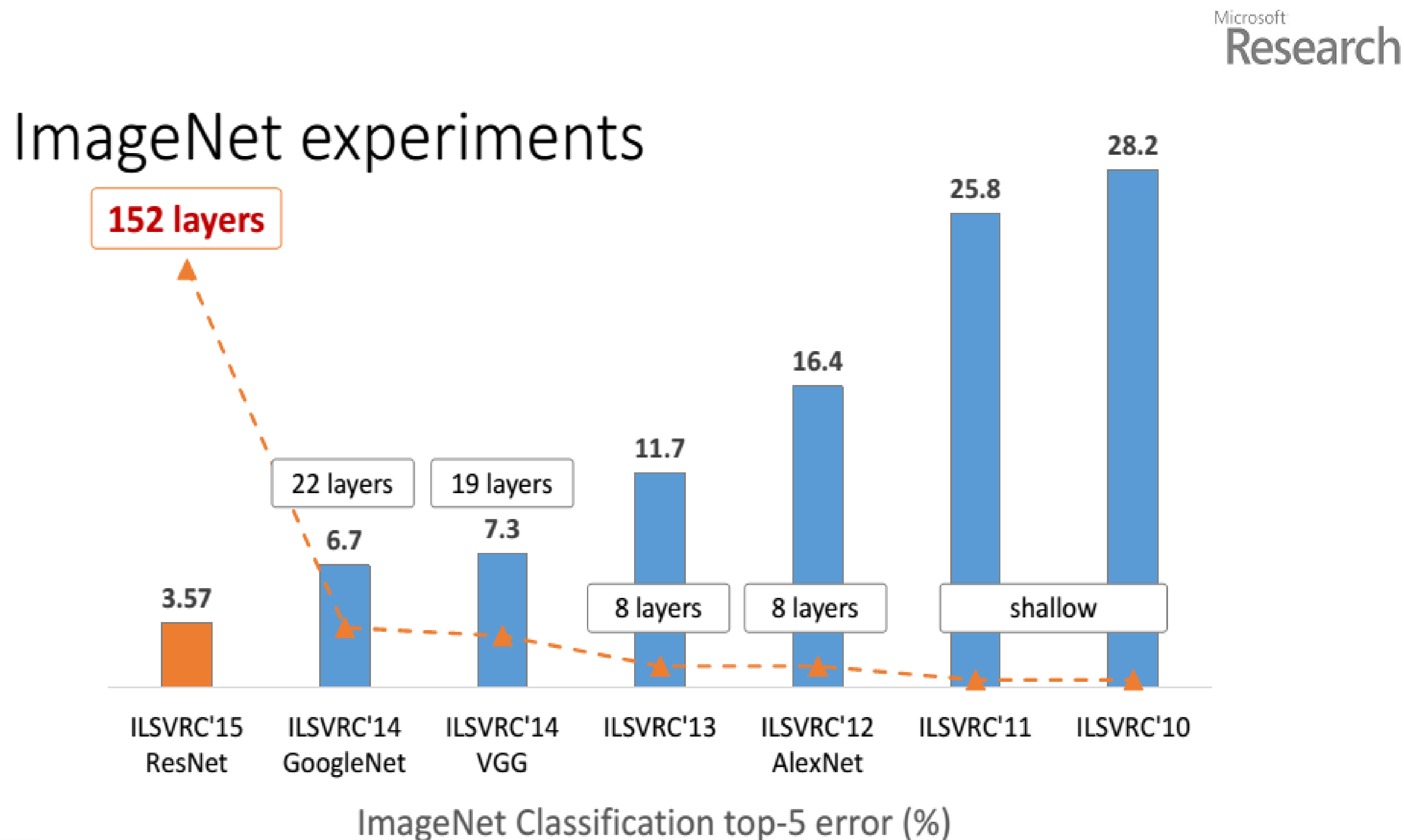
ImageNet:
1.2M training images
(~ 256 x 256px)
50k validation images
1000 classes



CIFAR-10:
50k training images
(32x32 px)
10k validation images
10 classes

Russakovskiy et.al, ImageNet Large Scale Visual Recognition Challenge, 2015
Krizhevsky, Learning Multiple Layers of Features from Tiny Images, 2009

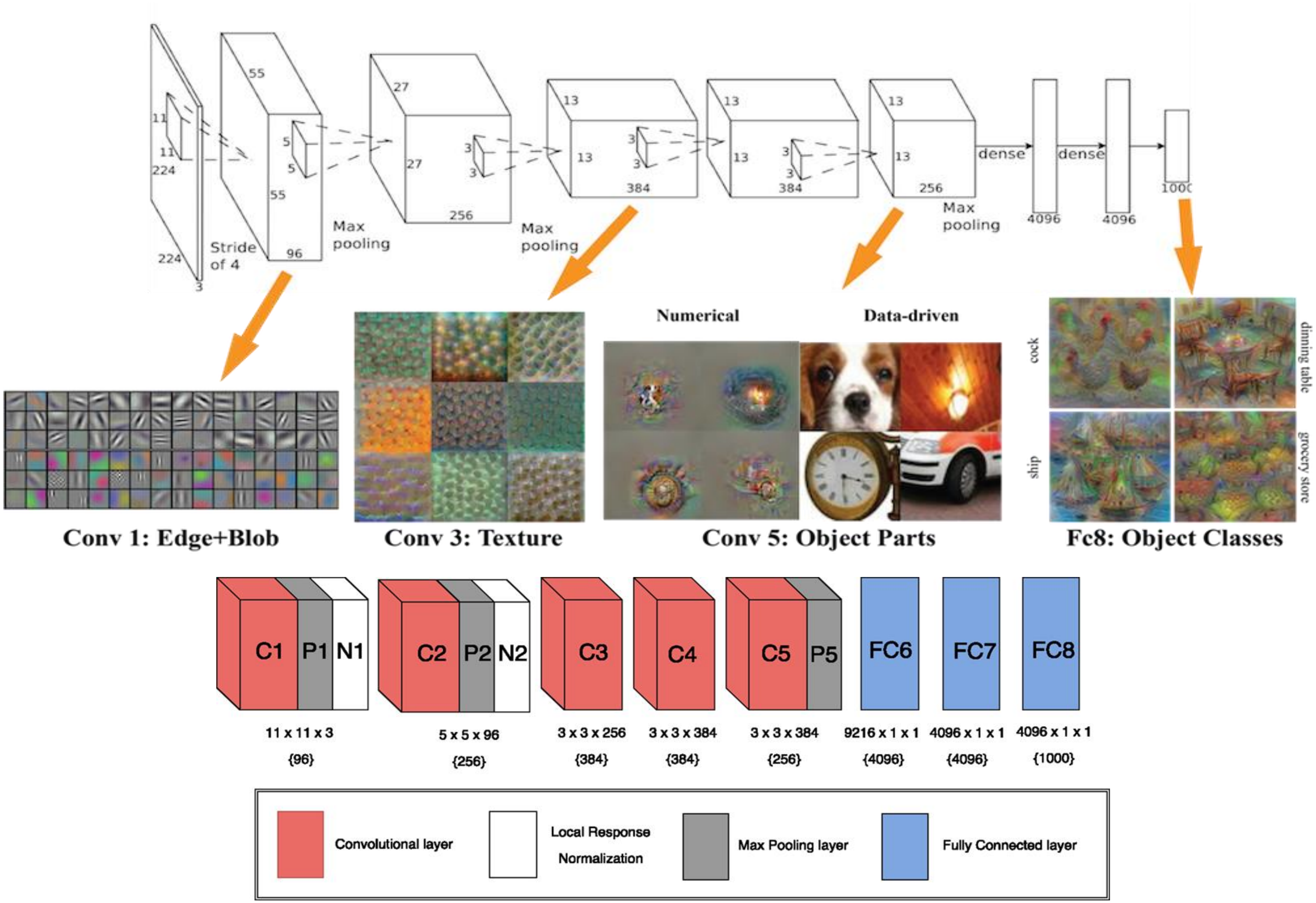
IMAGENET WINNERS



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

ICCV15 International Conference on Computer Vision

CAFFENET ARCHITECTURE



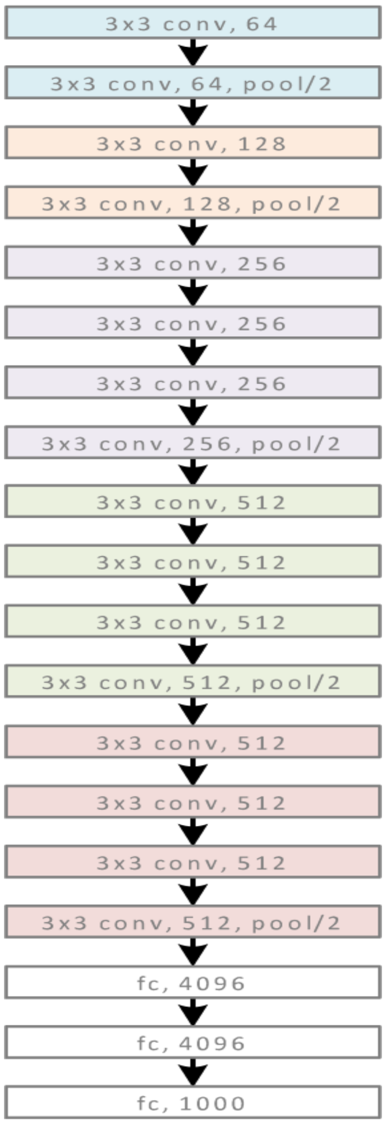
AlexNet (original):Krizhevsky et.al., ImageNet Classification with Deep Convolutional Neural Networks, 2012.
CaffeNet: Jia et.al., Caffe: Convolutional Architecture for Fast Feature Embedding, 2014.
Image credit: Roberto Matheus Pinheiro Pereira, "Deep Learning Talk".
Srinivas et.al "A Taxonomy of Deep Convolutional Neural Nets for Computer Vision", 2016.

CaffeNet architecture

input	image 128x128 px, random crop from 144xN, random mirror
pre-process	out = 0.04 (BGR - (104; 117; 124))
conv1 pool1	conv 11x11x96, stride 4 ReLU max pool 3x3, stride 2
conv2 pool2	conv 5x5x256, stride 2, pad 1, group 2 ReLU max pool 3x3, stride 2
conv3 conv4 conv5 pool5	conv 3x3x384, pad 1 ReLU conv 3x3x384, pad 1, group 2 ReLU conv 3x3x256, pad 1, group 2 ReLU max pool 3x3, stride 2
fc6 drop6 fc7 drop7	fully-connected 4096 ReLU dropout ratio 0.5 fully-connected 4096 ReLU dropout ratio 0.5
fc8-clf	softmax-1000

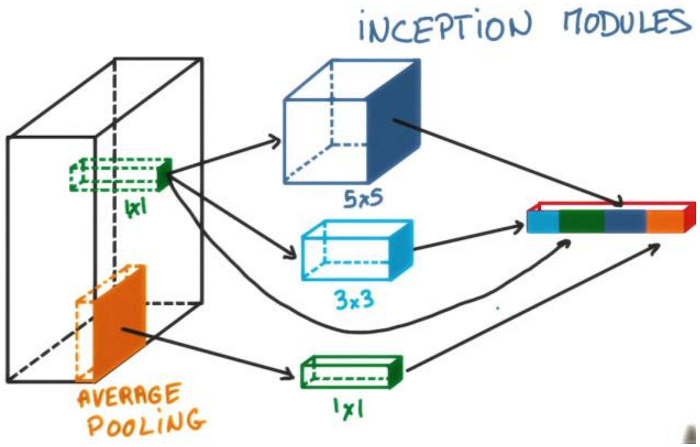
Mishkin et.al. Systematic evaluation of CNN advances on the ImageNet, arXiv 2016

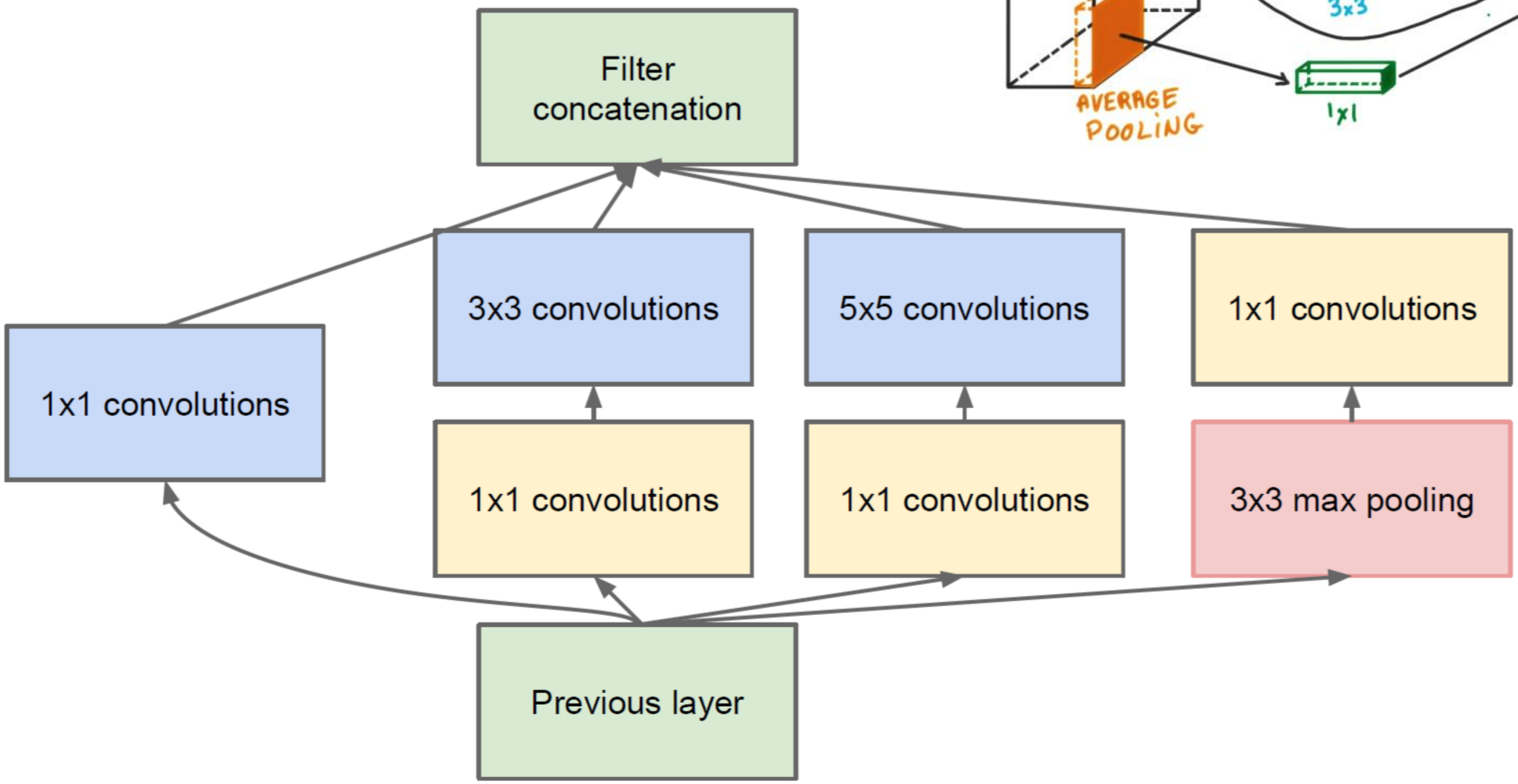
VGGNet architecture



All convolutions are 3x3
Good performance,
but slow

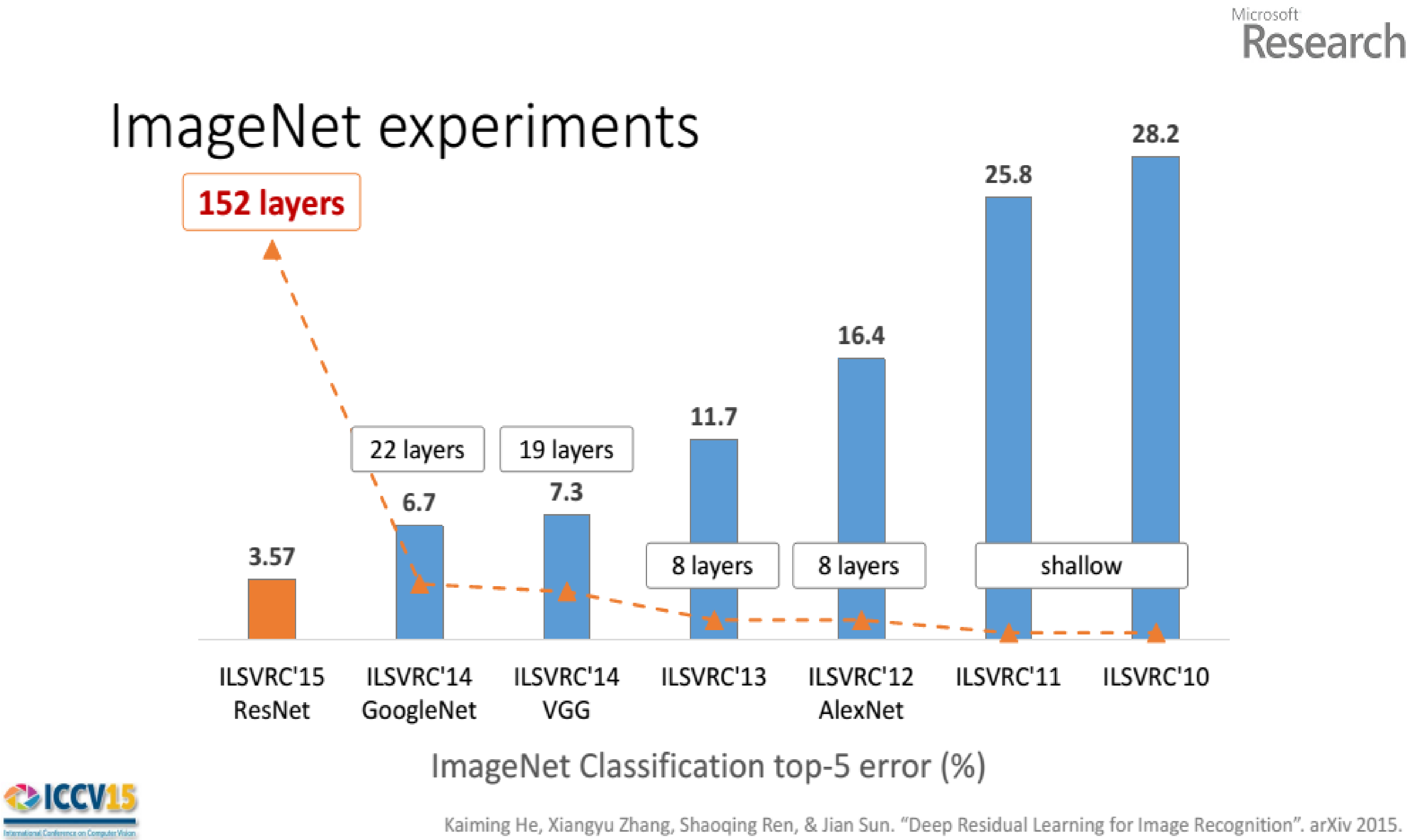
INCEPTION (GOOGLENET): BUILDING BLOCK



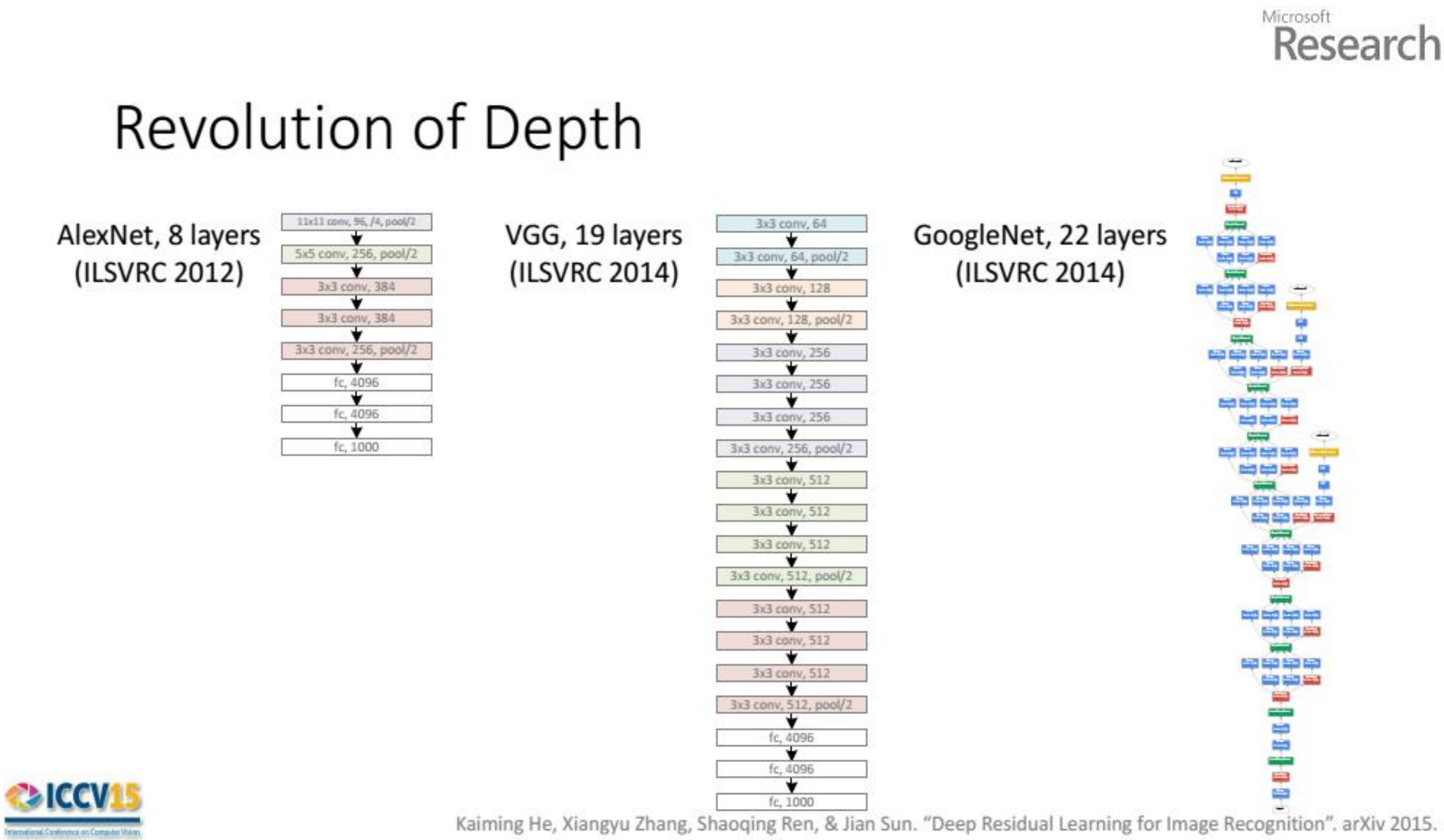


Szegedy et.al. Going Deeper with Convolutions. CVPR, 2015
Image credit: https://www.udacity.com/course/deep-learning--ud730

Depth limit not reached yet in deep learning

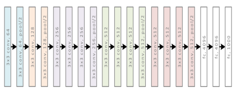


Depth limit not reached yet in deep learning

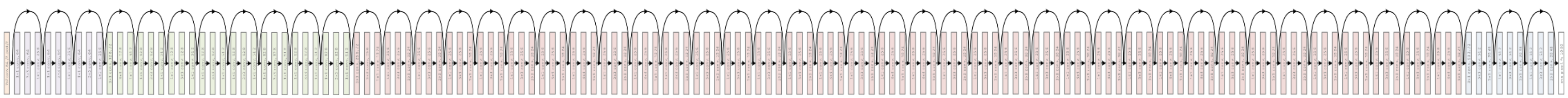


DEPTH LIMIT NOT REACHED YET IN DEEP LEARNING

VGGNet – 19 layers, 19.6 billion FLOPS. Simonyan et.al., 2014



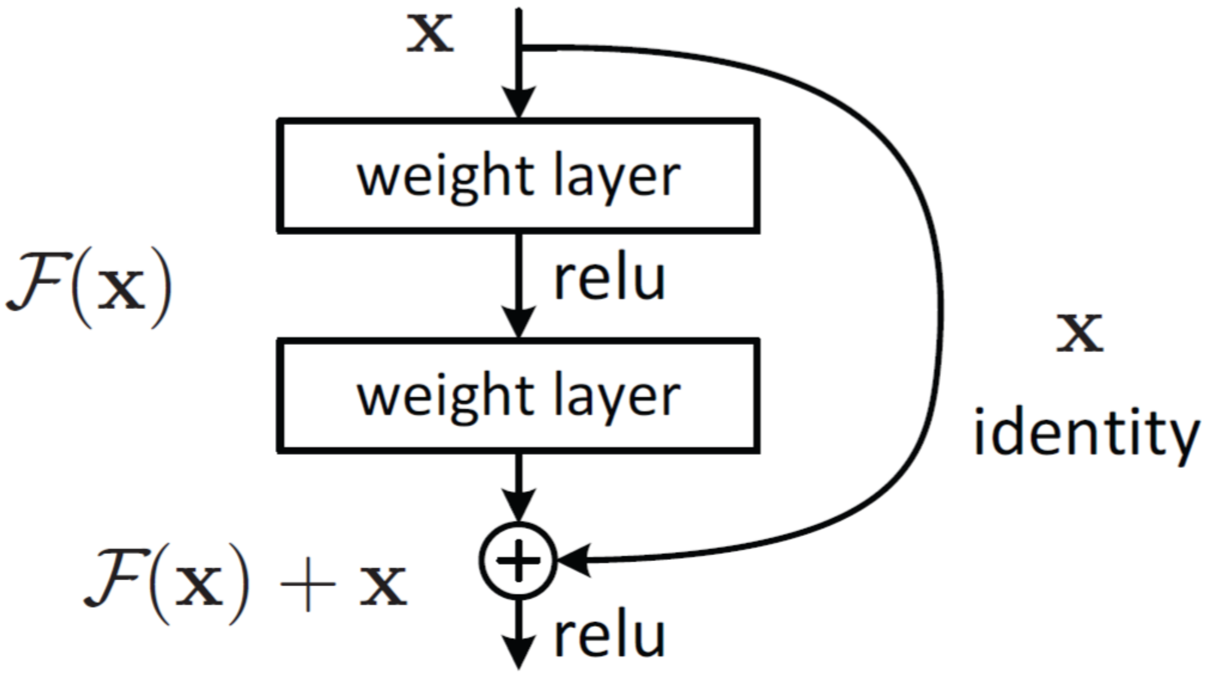
ResNet – 152 layers, 11.3 billion FLOPS. He et.al., 2015



Stochastic ResNet – 1200 layers.
Huang et.al., Deep Networks with Stochastic Depth, 2016

Slide credit Šulc et.al. Very Deep Residual Networks with MaxOut for Plant Identification in the Wild, 2016.

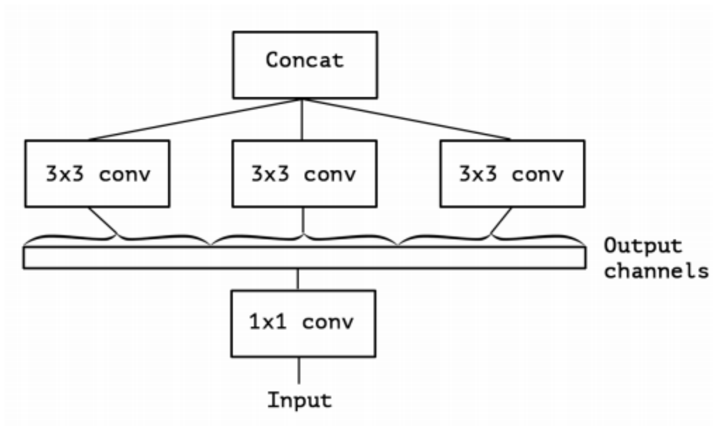
ResNet: Residual block



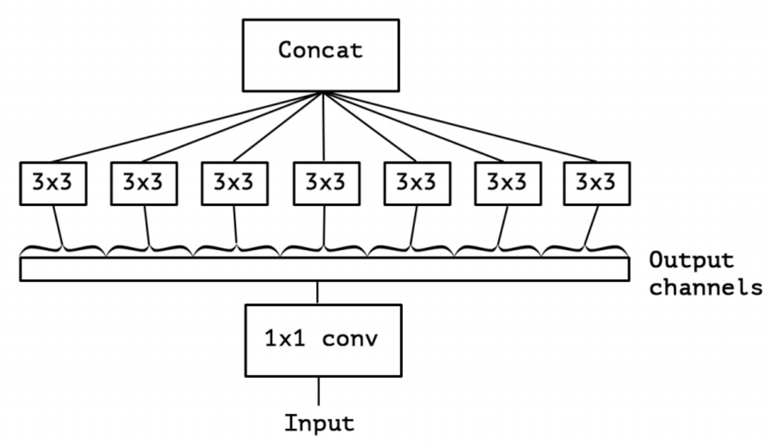
He et.al. Deep Residual Learning for Image Recognition, ICCV 2015

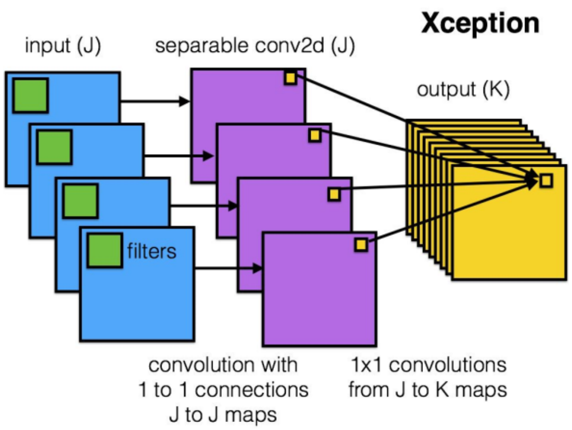
Xception, ResNeXt: depth-sep conv

Inception



Xception

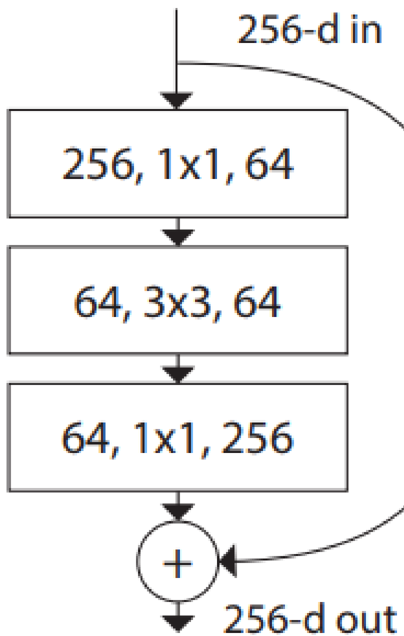




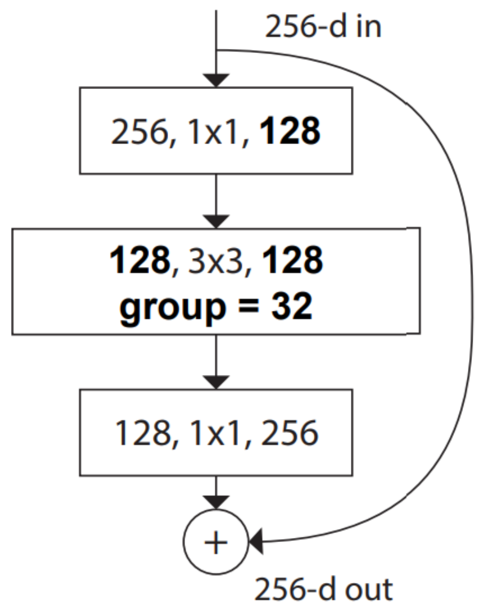
F.Chollet. Xception: Deep Learning with Depthwise Separable Convolutions, arXiv 2016

XCEPTION, RESNEXT: DEPTH-SEP CONV

ResNet block

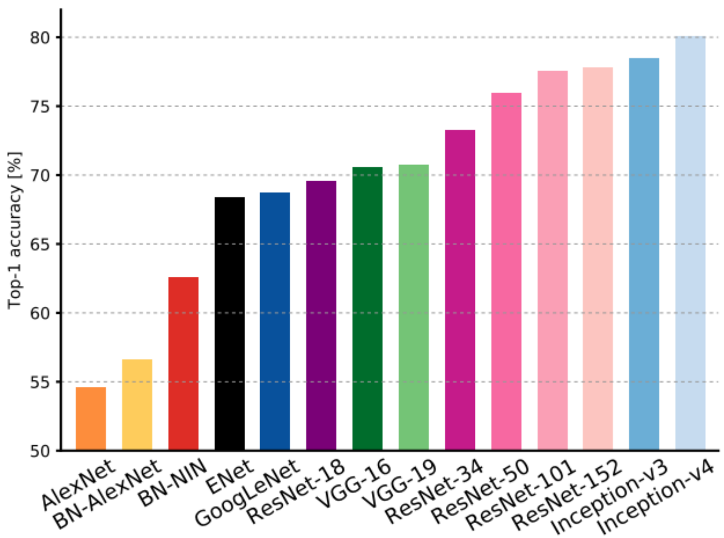


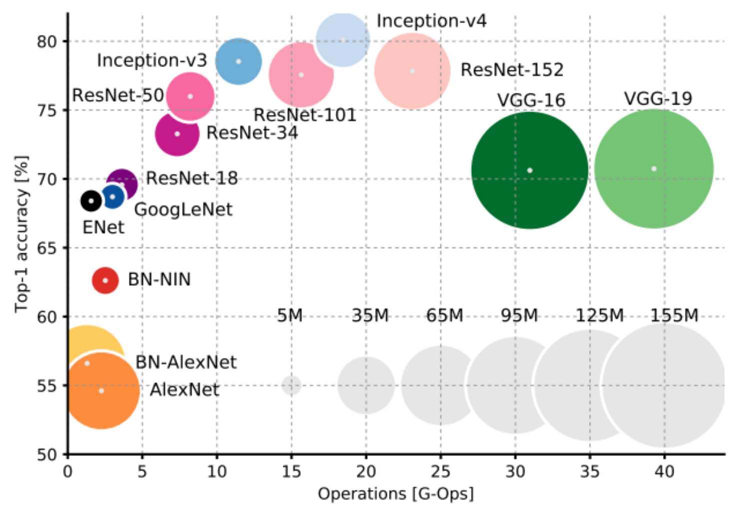
ResNeXt block



Xie et.al. Aggregated Residual Transformations for Deep Neural Network, CVPR 2017

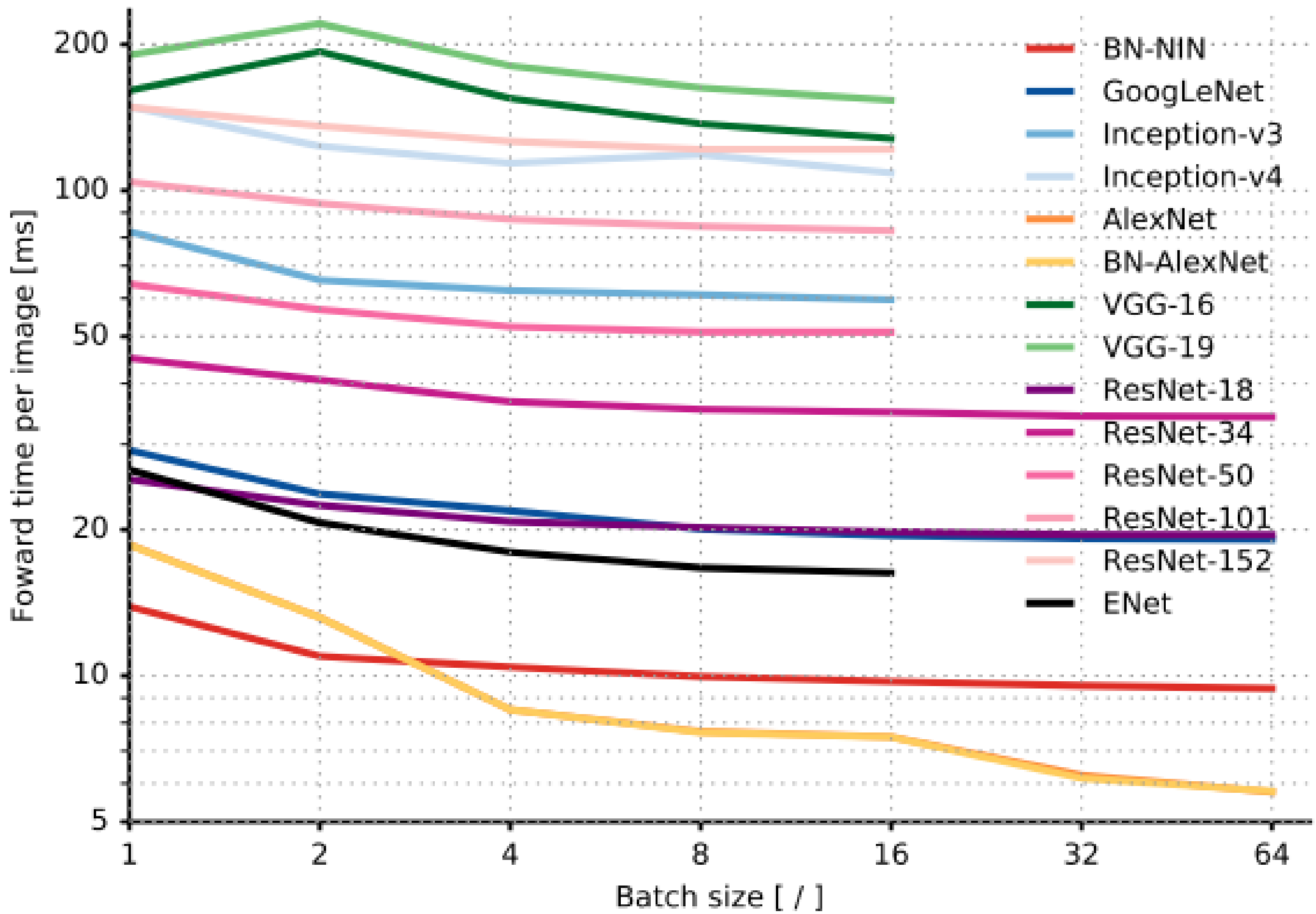
Accuracy-complexity trade-off





Canziani et.al. An analysis of deep neural network models for practical applications, 2016

Accuracy-complexity trade-off

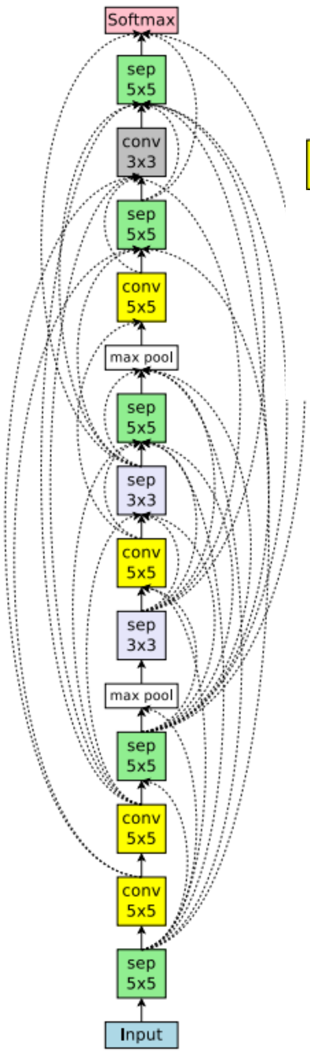


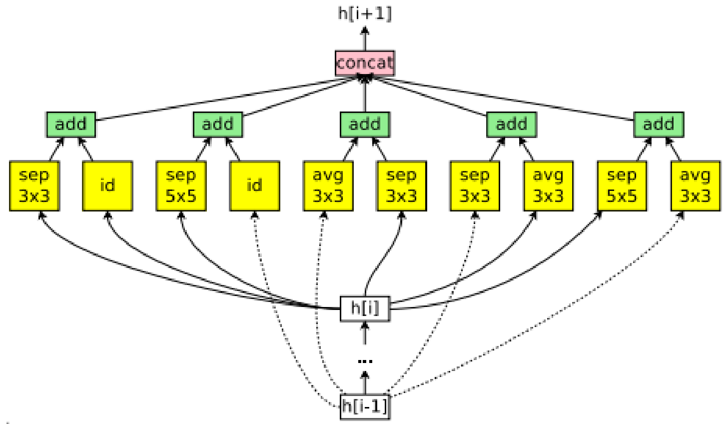
Canziani et.al. An analysis of deep neural network models for practical applications, 2016

Residual Networks: hot topic

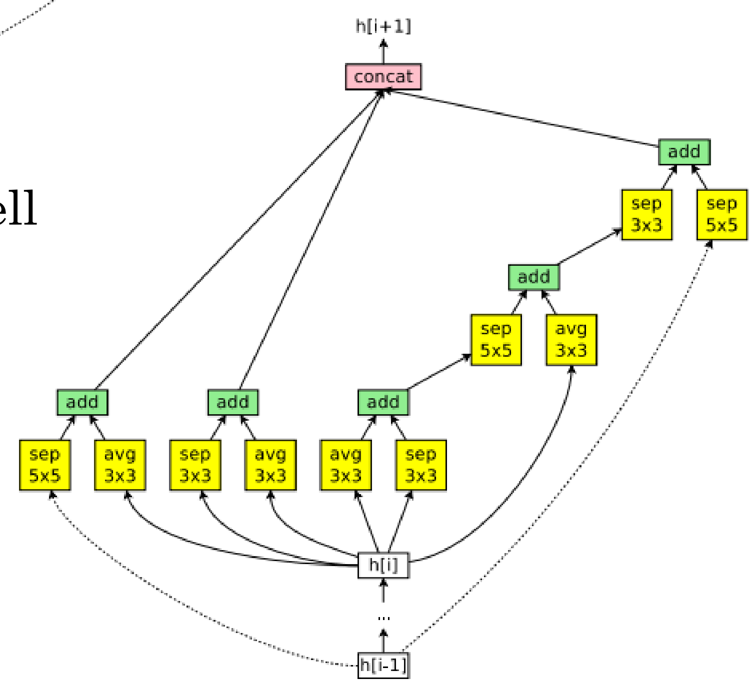
- Identity mapping https://arxiv.org/abs/1603.05027
- Wide ResNets https://arxiv.org/abs/1605.07146
- Stochastic depth https://arxiv.org/abs/1603.09382
- Residual Inception https://arxiv.org/abs/1602.07261
- ResNets + ELU http://arxiv.org/pdf/1604.04112.pdf
- ResNet in ResNet http://arxiv.org/pdf/1608.02908v1.pdf
- DC Nets http://arxiv.org/abs/1608.06993
- Weighted ResNet http://arxiv.org/pdf/1605.08831v1.pdf

Automatic architecture search: Reinforcement Learning





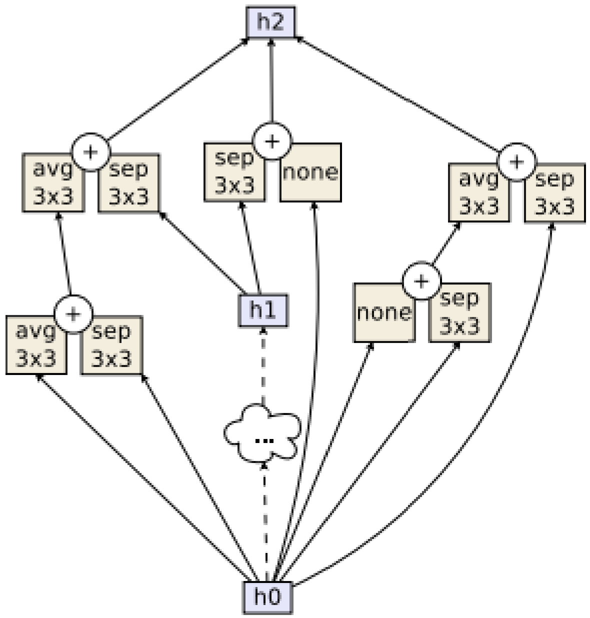
ENASNet Conv cell



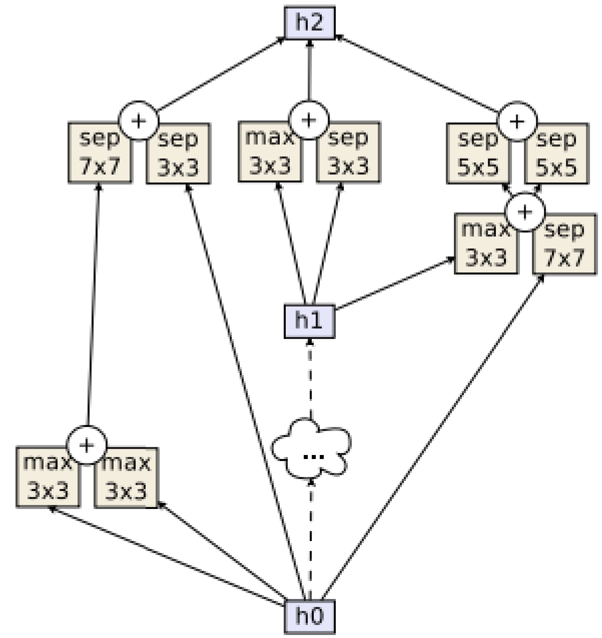
ENASNet Reduction cell

Pham et.al. Efficient Neural Architecture Search via Parameter Sharing, 2018

AUTOMATIC ARCHITECTURE SEARCH: EVOLUTION



AmoebaNet Conv cell

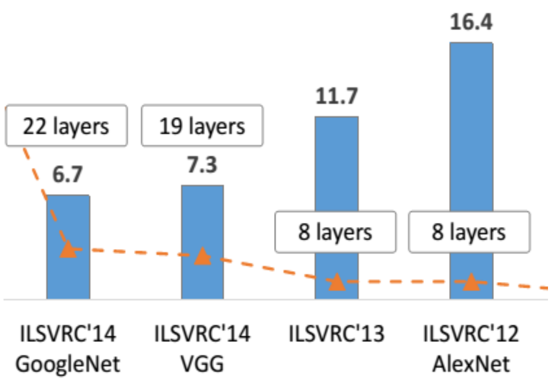


AmoebaNet Reduction cell

Real et.al. Regularized Evolution for Image Classifier Architecture Search, 2018

Too much arch optimization may hurt

ImageNet classification:



PASCAL semantic segmentation

	FCN-AlexNet	FCN-VGG16	FCN-GoogLeNet[4]
mean IU	39.8	**56.0**	42.5
forward time	50 ms	210 ms	59 ms
conv. layers	8	16	22
parameters	57M	134M	6M
rf size	355	404	907
max stride	32	32	32

Long et.al. Fully Convolutional Networks for Semantic Segmentation, CVPR 2015

Too much arch optimization may hurt

Performance on ESPGame dataset, semantic metrics

	Visual ST	Visual MEN	Multimodal ST	Multimodal MEN
AlexNet	0.018	0.448	0.208	0.686
GoogLeNet	0.063	0.487	0.243	0.700
VGGNet	0.125	0.506	0.269	0.708

Kiela et.al. Comparing Data Sources and Architectures for Deep Visual Representation Learning in Semantics

Now small design choices

CaffeNet architecture

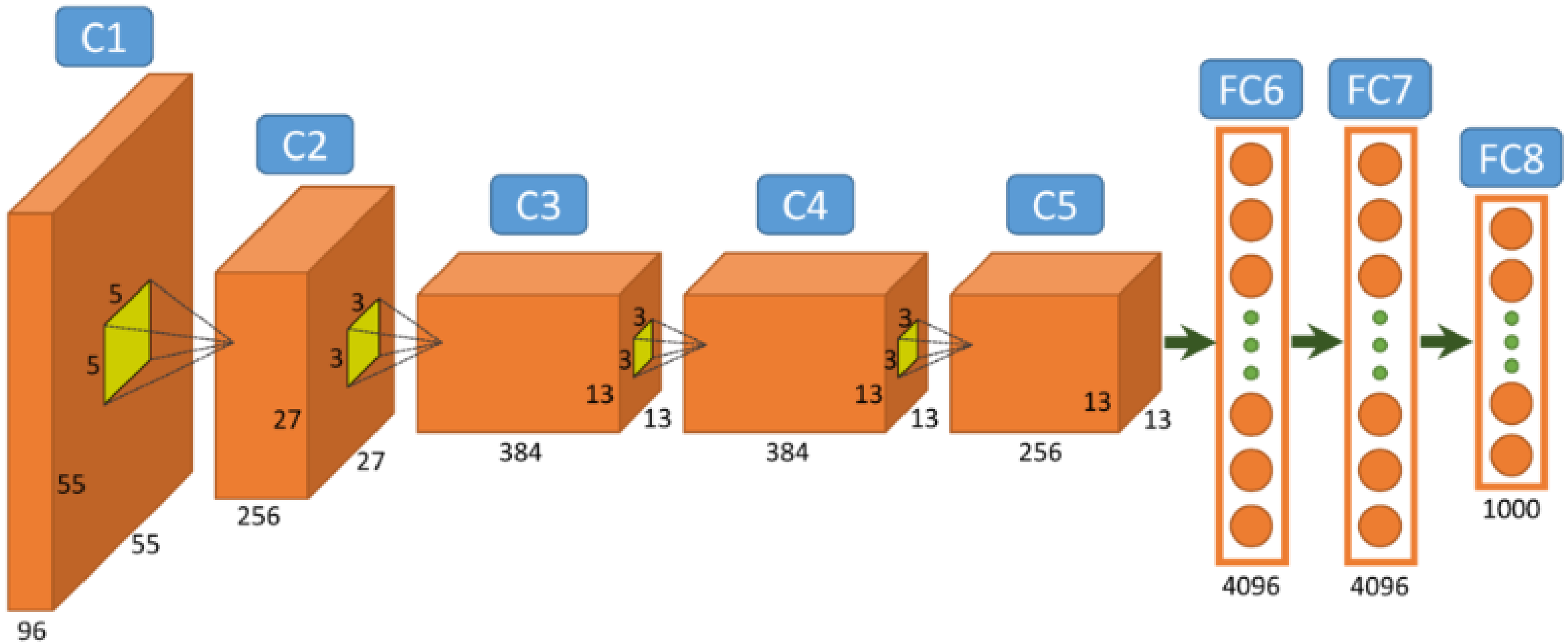
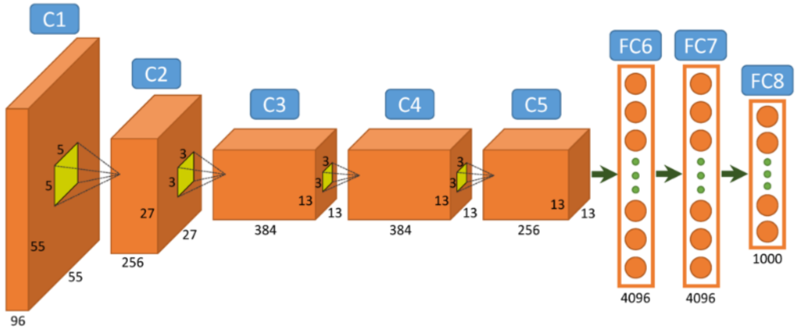


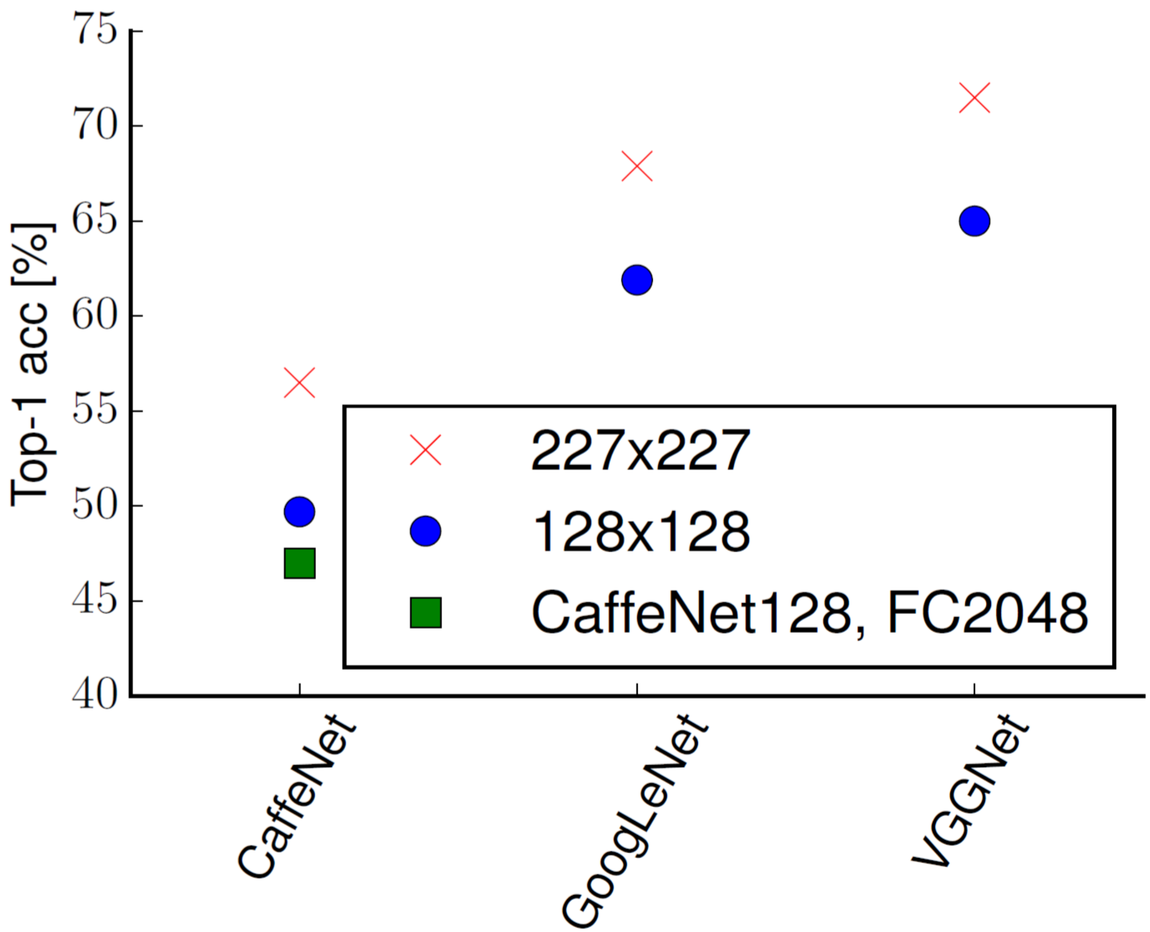
Image credit: Hu et.al, 2015
Transferring Deep Convolutional Neural Networks for the Scene Classification of High-Resolution Remote Sensing Imagery

List of hyper-parameters tested

Hyper-parameter	Variants
Non-linearity	linear, tanh, sigmoid, ReLU, VLReLU, RReLU, PReLU, ELU, maxout, APL, combination
Batch Normalization (BN)	before non-linearity. after non-linearity
BN + non-linearity	linear, tanh, sigmoid, ReLU, VLReLU, RReLU, PReLU, ELU, maxout
Pooling	max, average, stochastic, max+average, strided convolution
Pooling window size	3x3, 2x2, 3x3 with zero-padding
Learning rate decay policy	step, square, square root, linear
Colorspace & Pre-processing	RGB, HSV, YCrCb, grayscale, learned, CLAHE, histogram equalized
Classifier design	pooling-FC-FC-clf, SPP-FC-FC-clf, pooling-conv-conv-clf-avepool, pooling-conv-conv-avepool-clf
Network width	$1/4$, $1/2\sqrt{2}$, $1/2$, $1/\sqrt{2}$, 1 ,$\sqrt{2}$, 2, $2\sqrt{2}$, 4, $4\sqrt{2}$
Input image size	64, 96, 128, 180, 224
Dataset size	200K, 400K, 600K, 800K, 1200K(full)
Batch size	1, 32, 64, 128, 256, 512, 1024
Percentage of noisy data	0, 5%, 10%, 15%, 32%
Using bias	yes/no

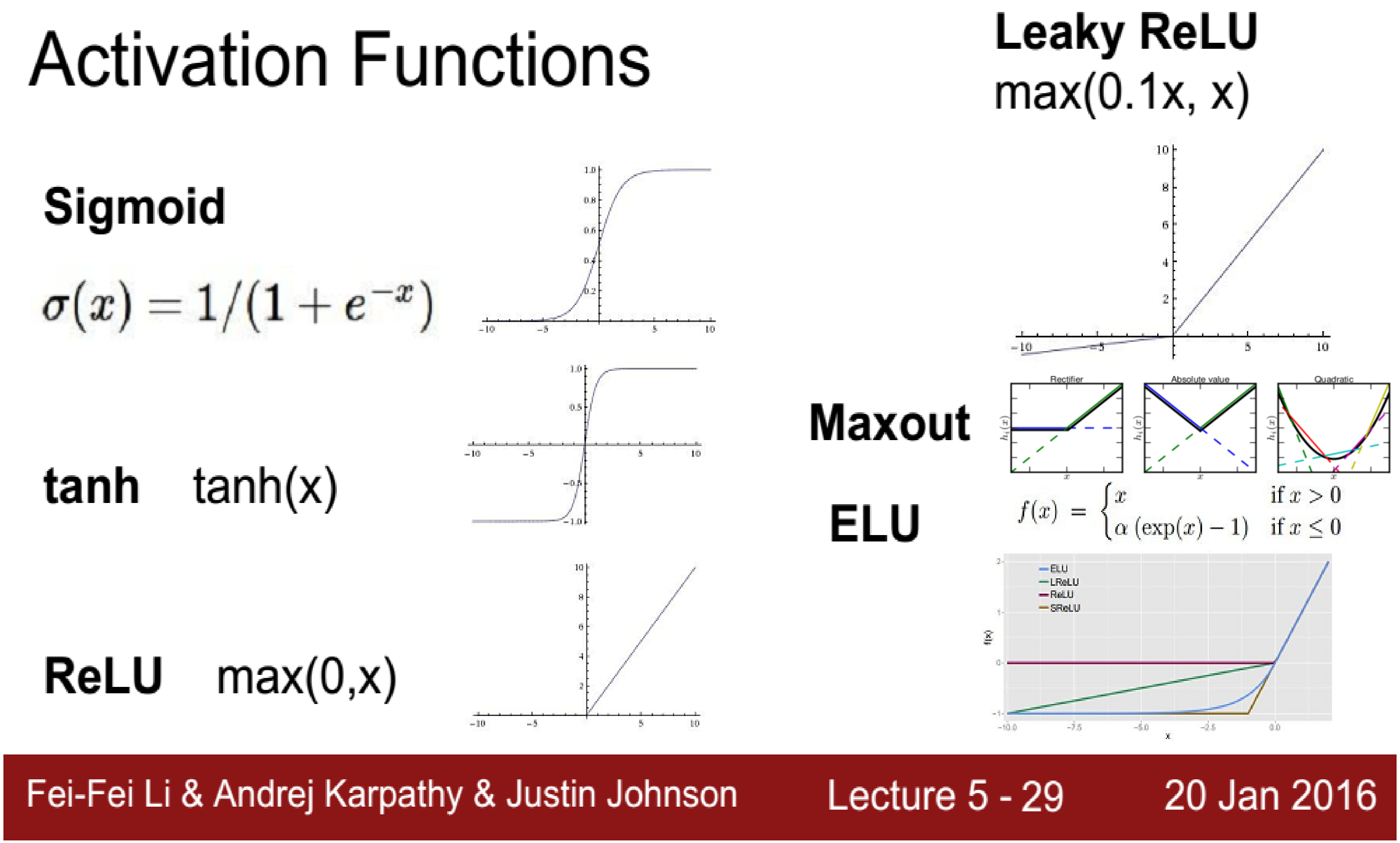


Reference methods: image size sensitive



Mishkin et.al. Systematic evaluation of CNN advances on the ImageNet, arXiv 2016

Choice of non-linearity



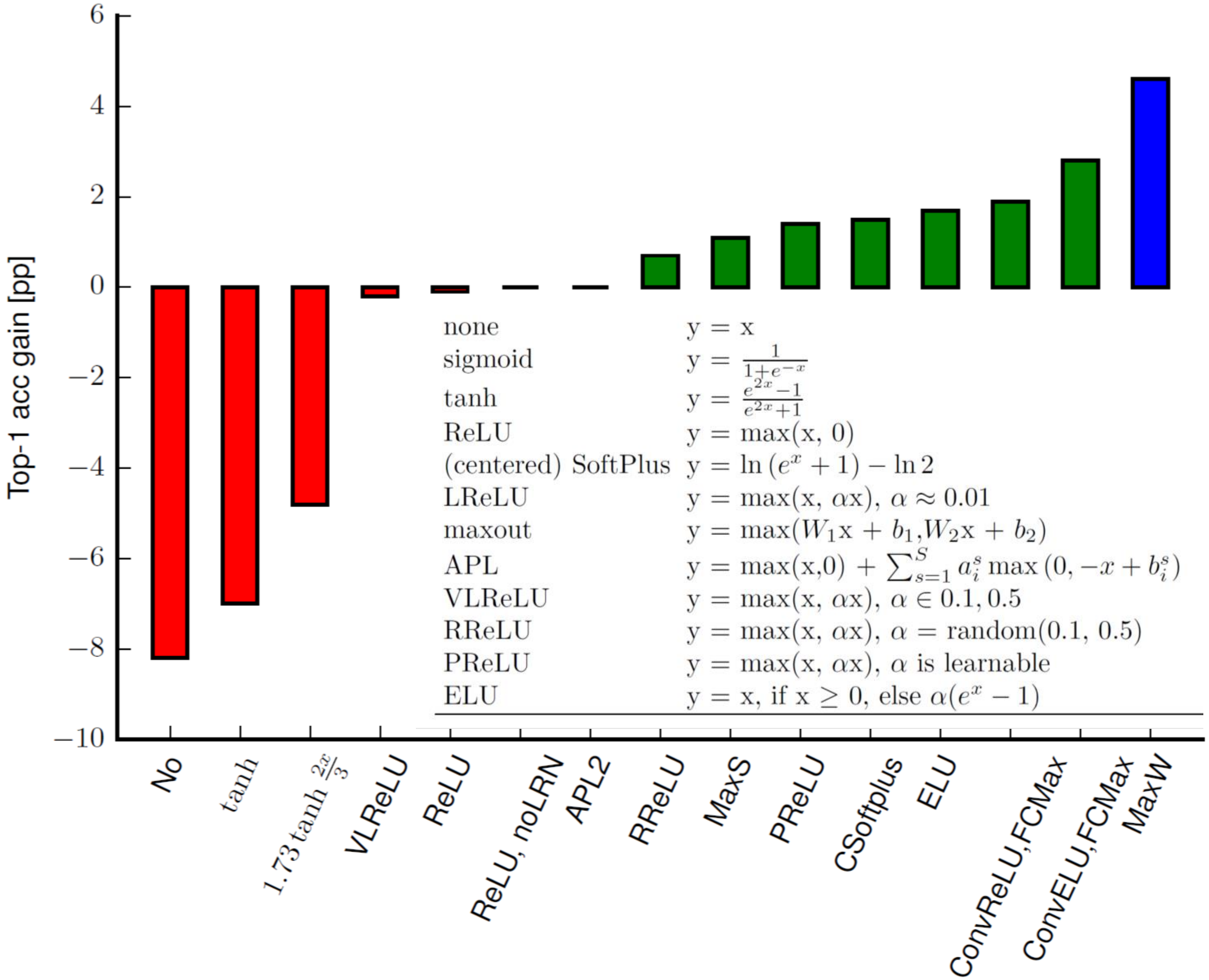
Choice of non-linearity

Table 3: Non-linearities tested.

Name	Formula	Year
none	y = x	-
sigmoid	$y = \frac{1}{1+e^{-x}}$	1986
tanh	$y = \frac{e^{2x}-1}{e^{2x}+1}$	1986
ReLU	y = max(x, 0)	2010
(centered) SoftPlus	$y = \ln\left(e^{x}+1\right) - \ln 2$	2011
LReLU	y = max(x, αx), $\alpha \approx 0.01$	2011
maxout	y = max(W_1x + b_1,W_2x + b_2)	2013
APL	y = max(x,0) + $\sum_{s=1}^{S} a_i^s \max\left(0, -x + b_i^s\right)$	2014
VLReLU	y = max(x, αx), $\alpha \in 0.1, 0.5$	2014
RReLU	y = max(x, αx), α = random(0.1, 0.5)	2015
PReLU	y = max(x, αx), α is learnable	2015
ELU	y = x, if x $\geq$ 0, else $\alpha(e^{x}-1)$	2015

Mishkin et.al. Systematic evaluation of CNN advances on the ImageNet, arXiv 2016

Non-linearities on CaffeNet



Mishkin et.al. Systematic evaluation of CNN advances on the ImageNet, arXiv 2016

Batch normalization (after every convolution layer)

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

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

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

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

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

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

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

Ioffe et.al, ICML 2015

BATCH NORMALIZATION: WHERE, BEFORE OR AFTER NON-LINEARITY?

ImageNet, top-1 accuracy

Network	No BN	Before ReLU	After ReLU
CaffeNet128-FC2048	47.1	47.8	**49.9**
GoogLeNet128	**61.9**	60.3	59.6

CIFAR-10, top-1 accuracy,
FitNet4 network

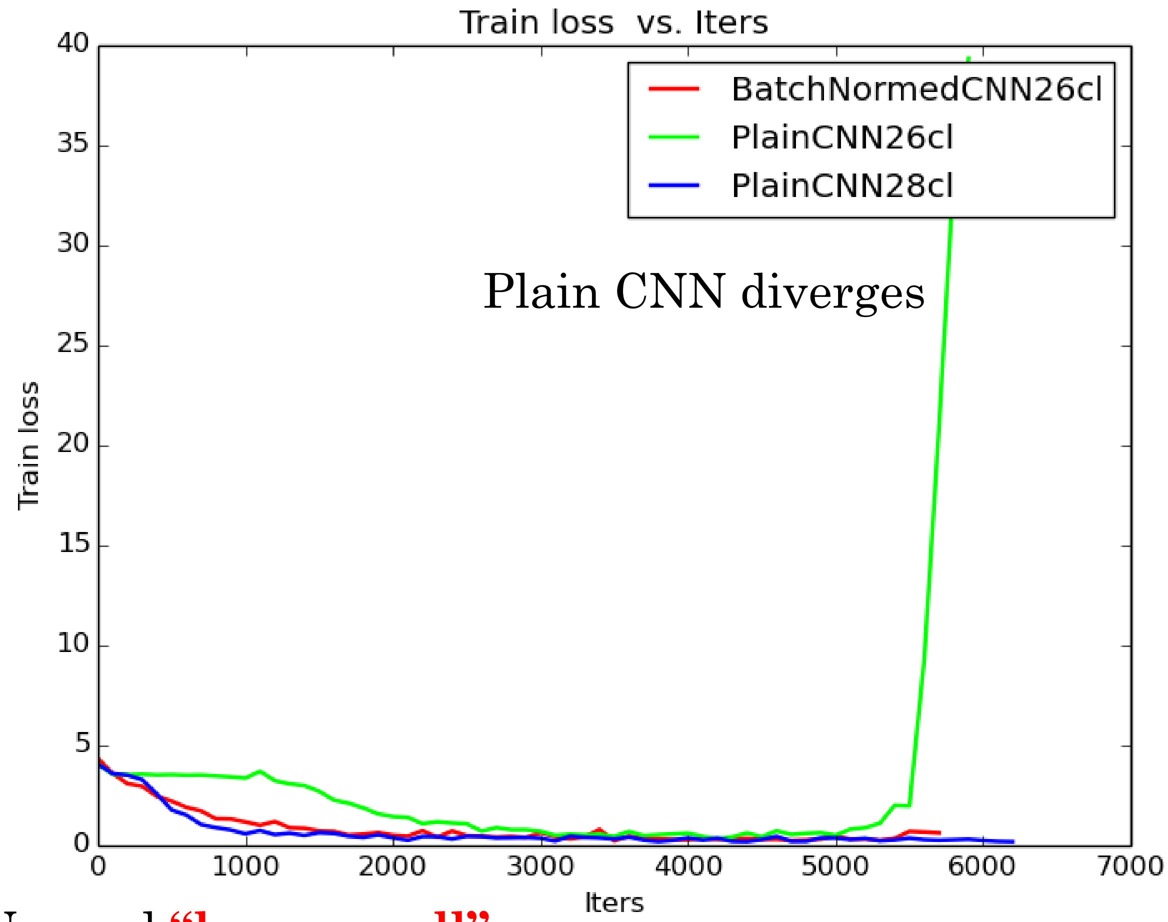
Non-linearity	BN Before	BN After
TanH	88.1	**89.2**
ReLU	**92.6**	92.5
MaxOut	92.3	**92.9**

In short:
better to test
with
your architecture
and dataset :)

Mishkin and Matas. All you need is a good init. ICLR, 2016
Mishkin et.al. Systematic evaluation of CNN advances on the ImageNet, arXiv 2016

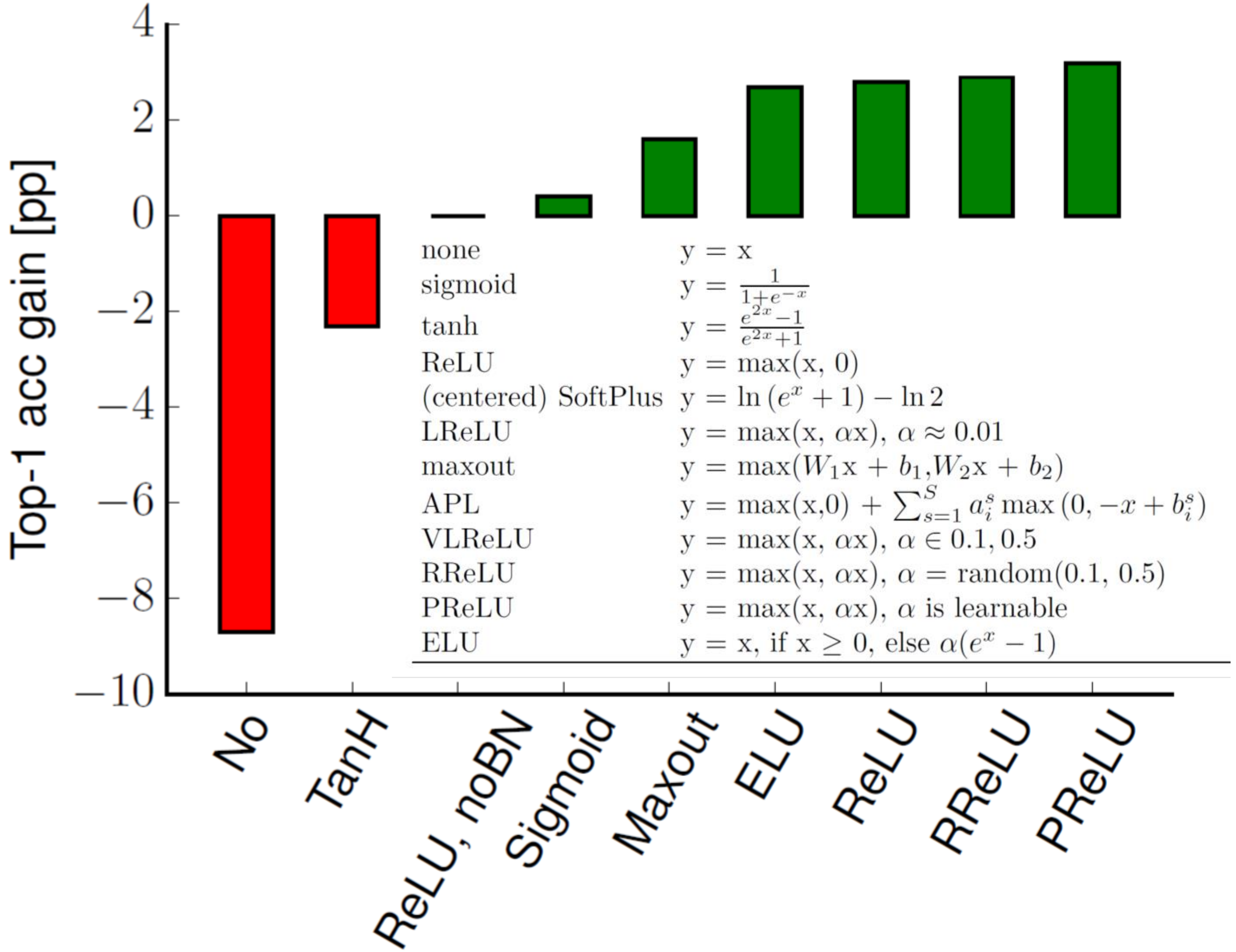
Batch normalization
Sometimes works too good and hides problems

Case:
CNN has less number outputs (just typo),
than classes in dataset: 26 vs. 28



BatchNormed **"learns well"**

Non-linearities on CaffeNet, with Batch Normalization



Mishkin et.al. Systematic evaluation of CNN advances on the ImageNet, arXiv 2016

Non-linearities: take away message

- Use ELU without batch normalization
- Or ReLU + BN
- Try maxout for the final layers
- Fallback solution (if something goes wrong) – ReLU

Mishkin et.al. Systematic evaluation of CNN advances on the ImageNet, arXiv 2016

But in small data regime (~50K images) try Leaky or Randomized ReLU

- Accuracy [%], Network in Network architecture

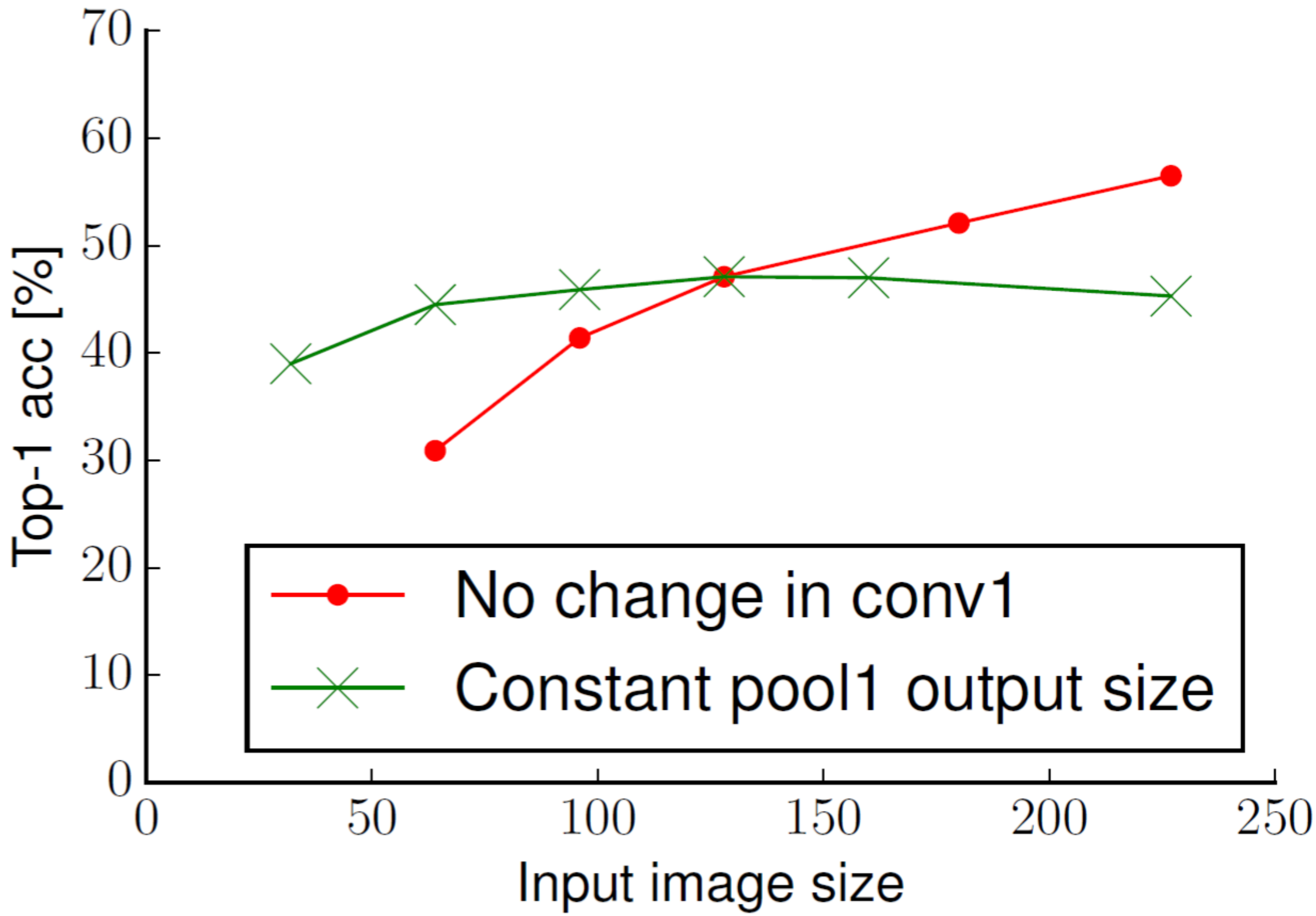
	ReLU	VLReLU	RReLU	PReLU
CIFAR-10	87.55	**88.80**	**88.81**	88.20
CIFAR-100	57.10	**59.60**	**59.80**	58.4

- LogLoss, Plankton VGG architecture

	ReLU	VLReLU	RReLU	PReLU
KNDB	0.77	**0.73**	**0.72**	0.74

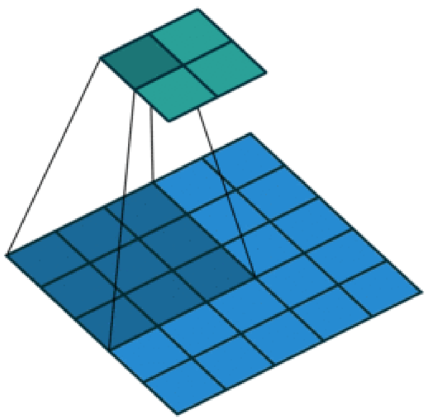
Xu et.al. Empirical Evaluation of Rectified Activations in Convolutional Network ICLR 2015

Input image size

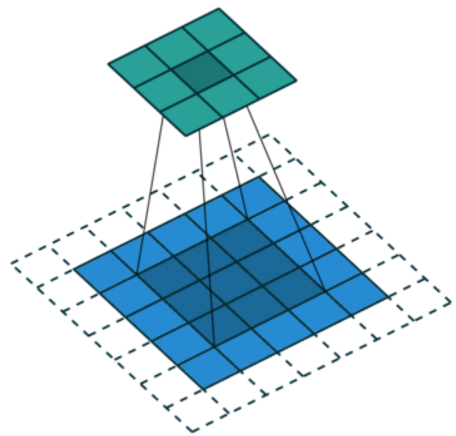


Mishkin et.al. Systematic evaluation of CNN advances on the ImageNet, arXiv 2016

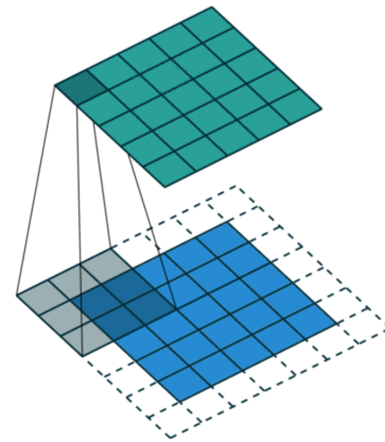
Padding types



No padding, stride = 2



Zero padding, stride = 2



Zero padding, stride = 1

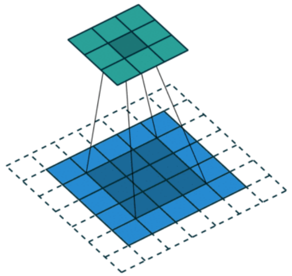
Dumoulin and Visin. A guide to convolution arithmetic for deep learning. ArXiv 2016

PADDING

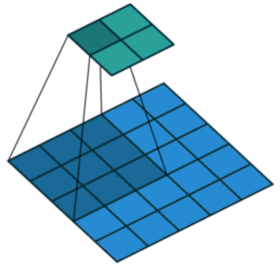
- Zero-padding:
  - Preserving spatial size, not "washing out" information
  - Dropout-like augmentation by zeros

Caffenet128

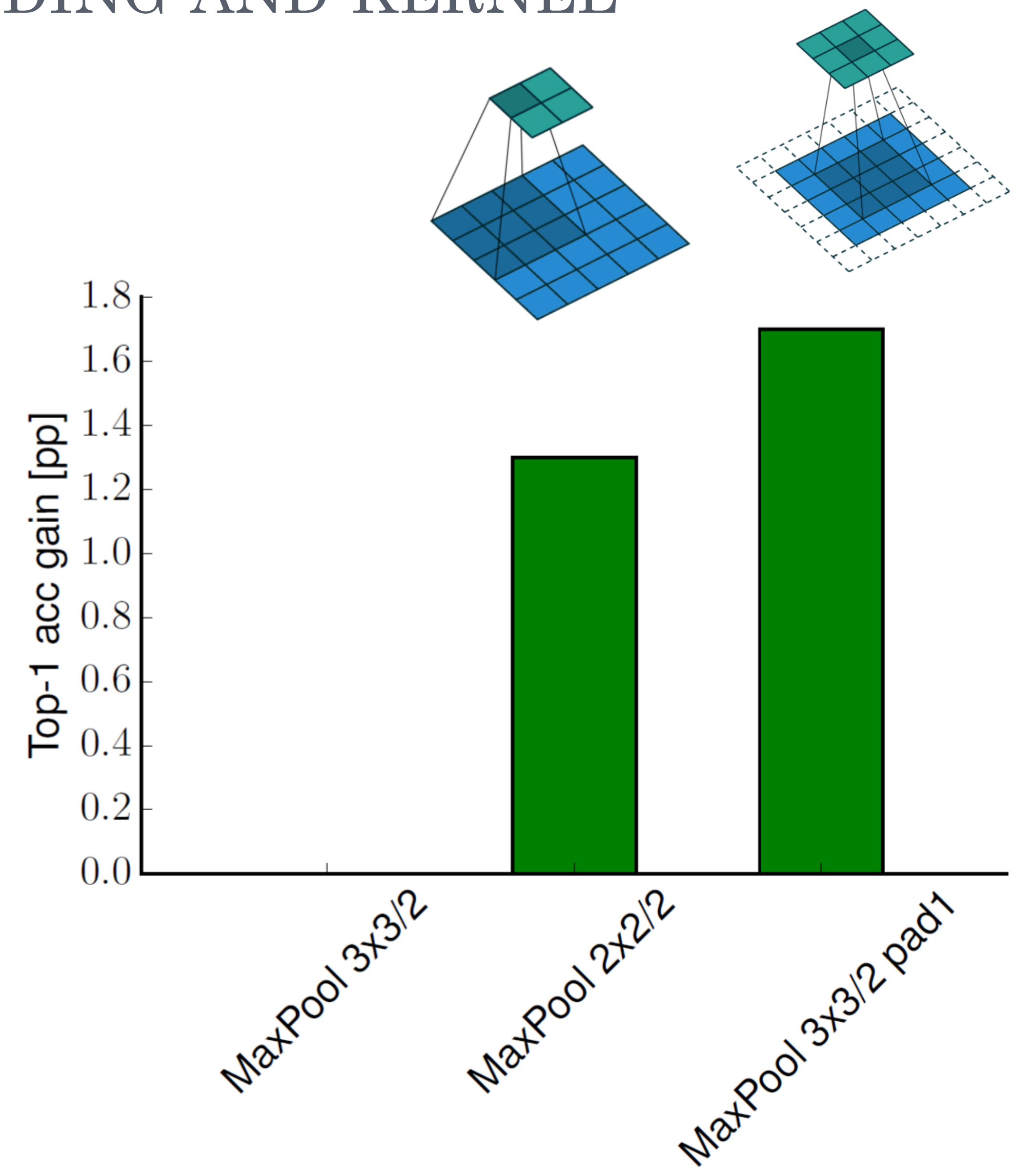
- with conv padding: **47**% top-1 acc



- w/o conv padding: **41**% top-1 acc.



MAX POOLING: PADDING AND KERNEL



Mishkin et.al. Systematic evaluation of CNN advances on the ImageNet, arXiv 2016

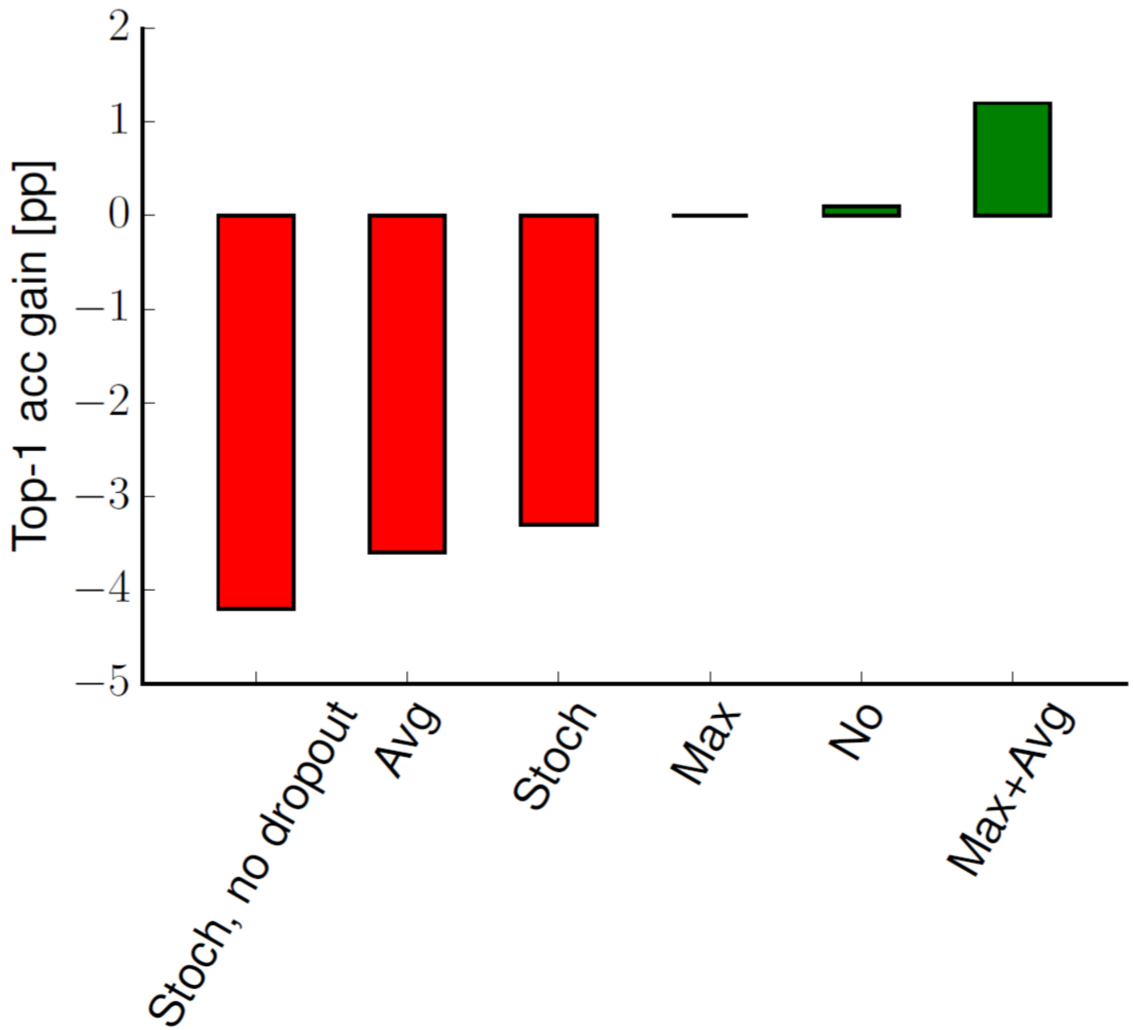
POOLING METHODS

Table 4: Poolings tested.

Name	Formula	Year
max	$y = \max_{i,j=1}^{h,w} x_{i,j}$	1989
average	$y = \frac{1}{hw}\sum_{i,j=1}^{h,w} x_{i,j}$	1989
stochastic	$y = x_{i,j}$ with prob. $\frac{x_{i,j}}{\sum_{i,j=1}^{h,w} x_{i,j}}$	2013
strided convolution	–	2014
max + average	$y = \max_{i,j=1}^{h,w} x_{i,j} + \frac{1}{hw}\sum_{i,j=1}^{h,w} x_{i,j}$	2015

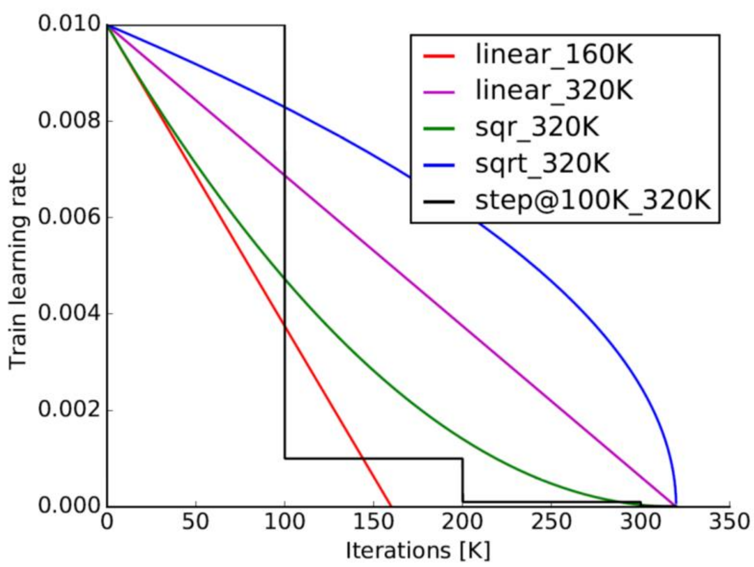
Mishkin et.al. Systematic evaluation of CNN advances on the ImageNet, arXiv 2016

Pooling methods



Mishkin et.al. Systematic evaluation of CNN advances on the ImageNet, arXiv 2016

LEARNING RATE POLICY: LINEAR



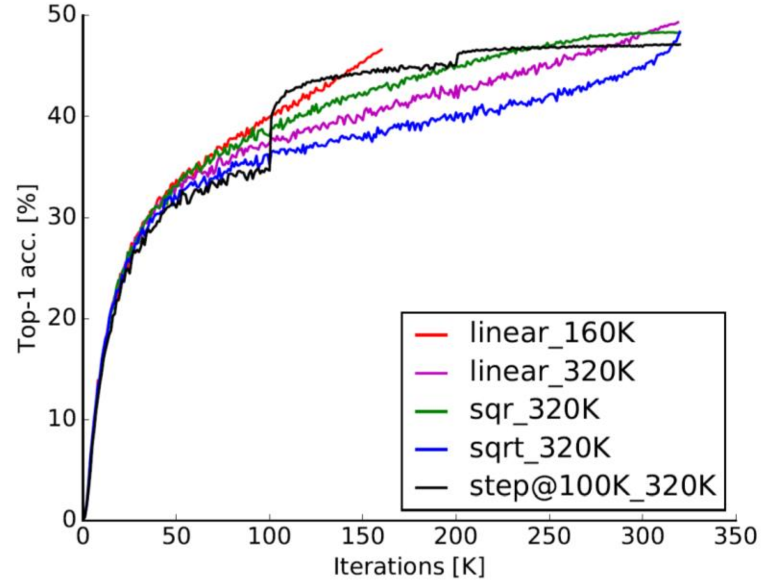


Table 5: Learning rate decay policies, tested in paper. L_0 – initial learning rate, M = number of learning iterations, i – current iteration, S – step iteration. γ – decay coefficient .

Name	Formula	Parameters	Accuracy
step	$\mathrm{lr} = L_0 \gamma^{\mathrm{floor}(i/S)}$	$S = 100\mathrm{K}, \gamma = 0.1, M = 320K$	0.471
square	$\mathrm{lr} = L_0(1 - i/M)^2$	$M = 320K$	0.483
square root	$\mathrm{lr} = L_0\sqrt{1 - i/M}$	$M = 320K$	0.483
linear	$\mathrm{lr} = L_0(1 - i/M)$	$M = 320K$	0.493
		$M = 160K$	0.466

Mishkin et.al. Systematic evaluation of CNN advances on the ImageNet, arXiv 2016

Learning rate policy: Linear cosine

- Bello et.al performed large scale reinforcement-learning search of learning rate schedule:

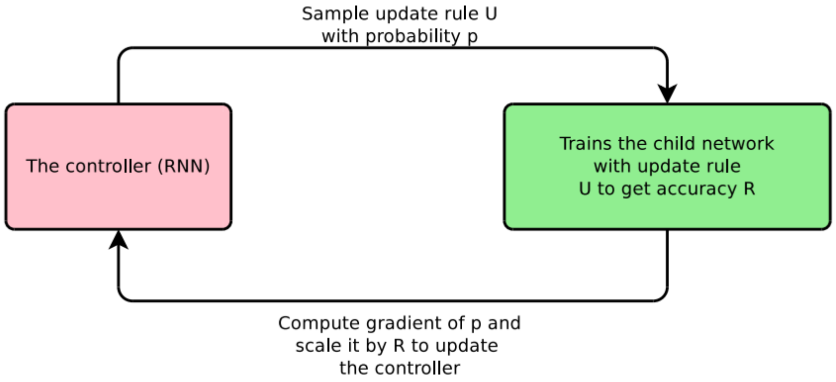


Figure 1. An overview of Neural Optimizer Search.

"Interestingly, we also found that the ***linear cosine decay*** *generally allows for a larger initial learning rate and leads to faster convergence"*

Bello et.al. Neural Optimizer Search with Reinforcement Learning, arXiv 2017

Image preprocessing

- Subtract mean pixel (training set), divide by std.
- RGB is the best (standard) colorspace for CNN
- Do nothing more…
- …unless you have specific dataset.

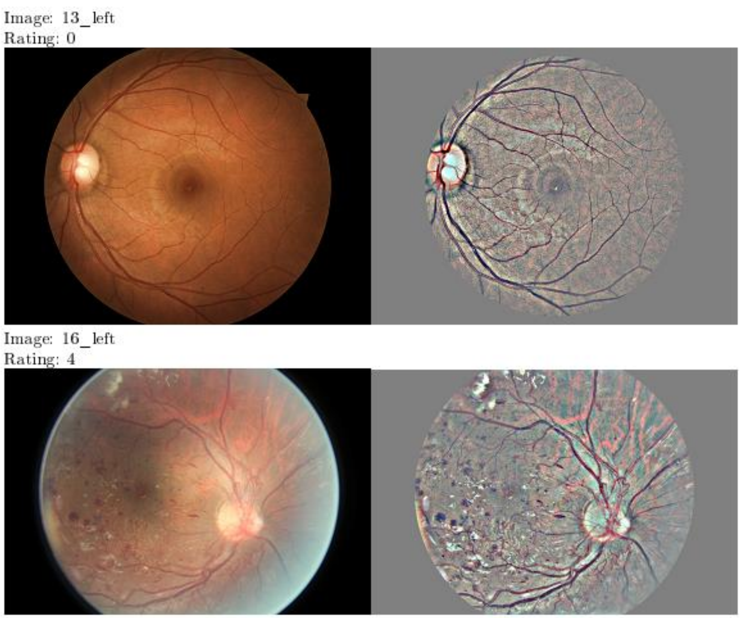
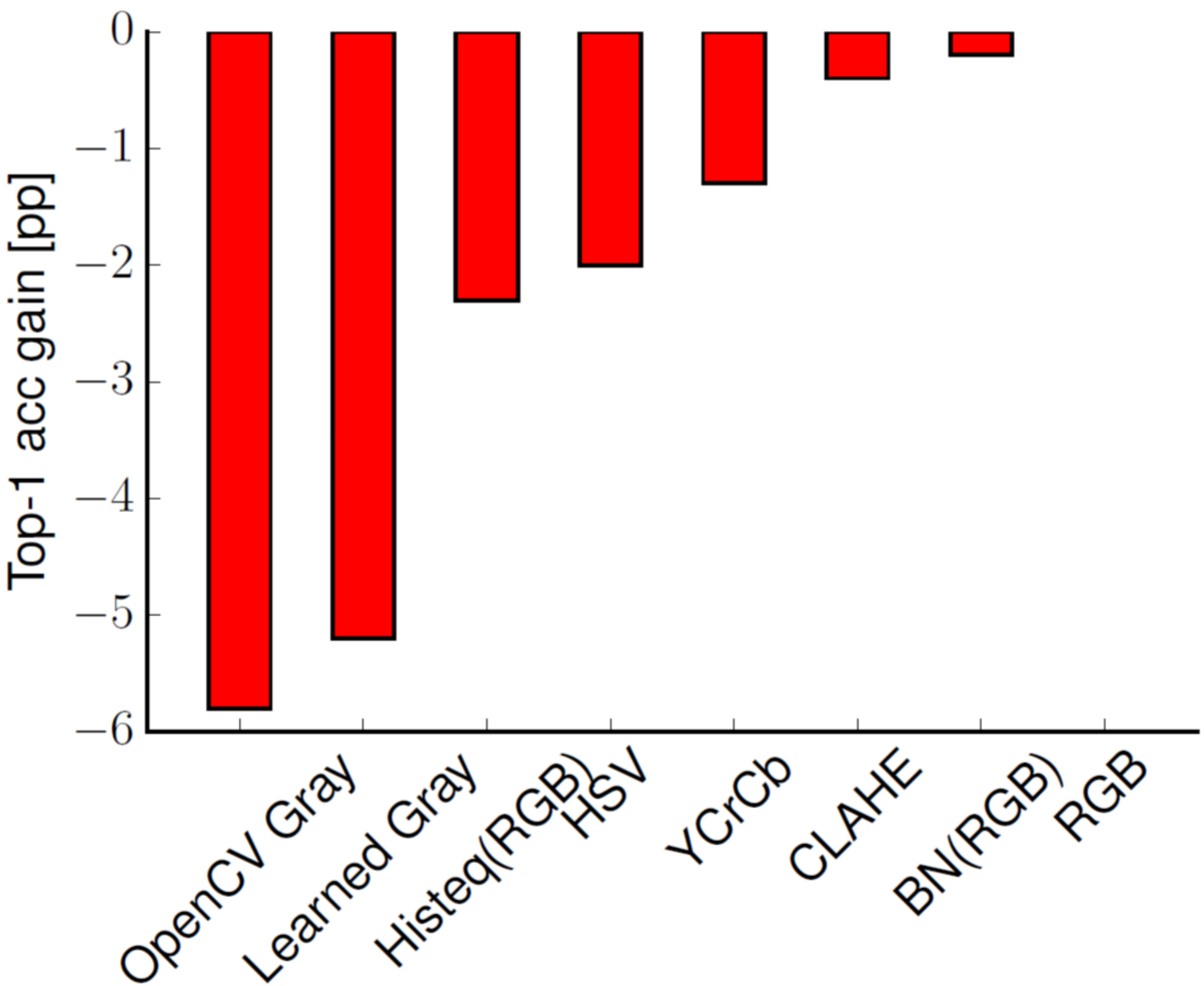


Figure 4: Two images from the training set. Original images on the left and preprocessed images on the right.

Subtract local mean pixel
B.Graham, 2015
Kaggle Diabetic Retinopathy Competition report

Image preprocessing: what doesn`t work

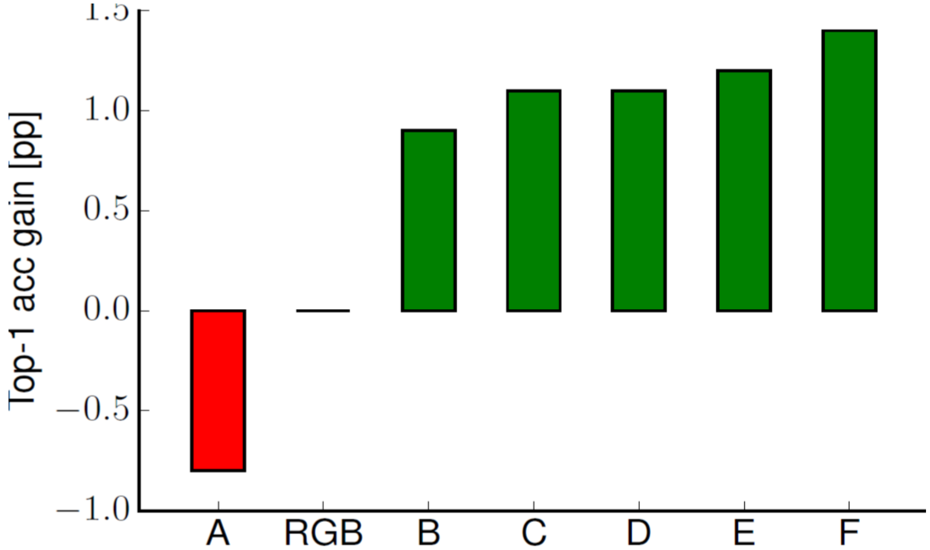


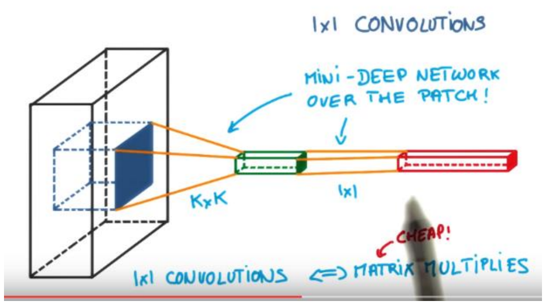
Mishkin et.al. Systematic evaluation of CNN advances on the ImageNet, arXiv 2016

Image preprocessing: Let`s learn the colorspace

Table 6: Mini-networks for learned colorspace transformations, placed after image and before conv1 layer. In all cases RGB means scales and centered input 0.04 * (Img - (104, 117,124)).

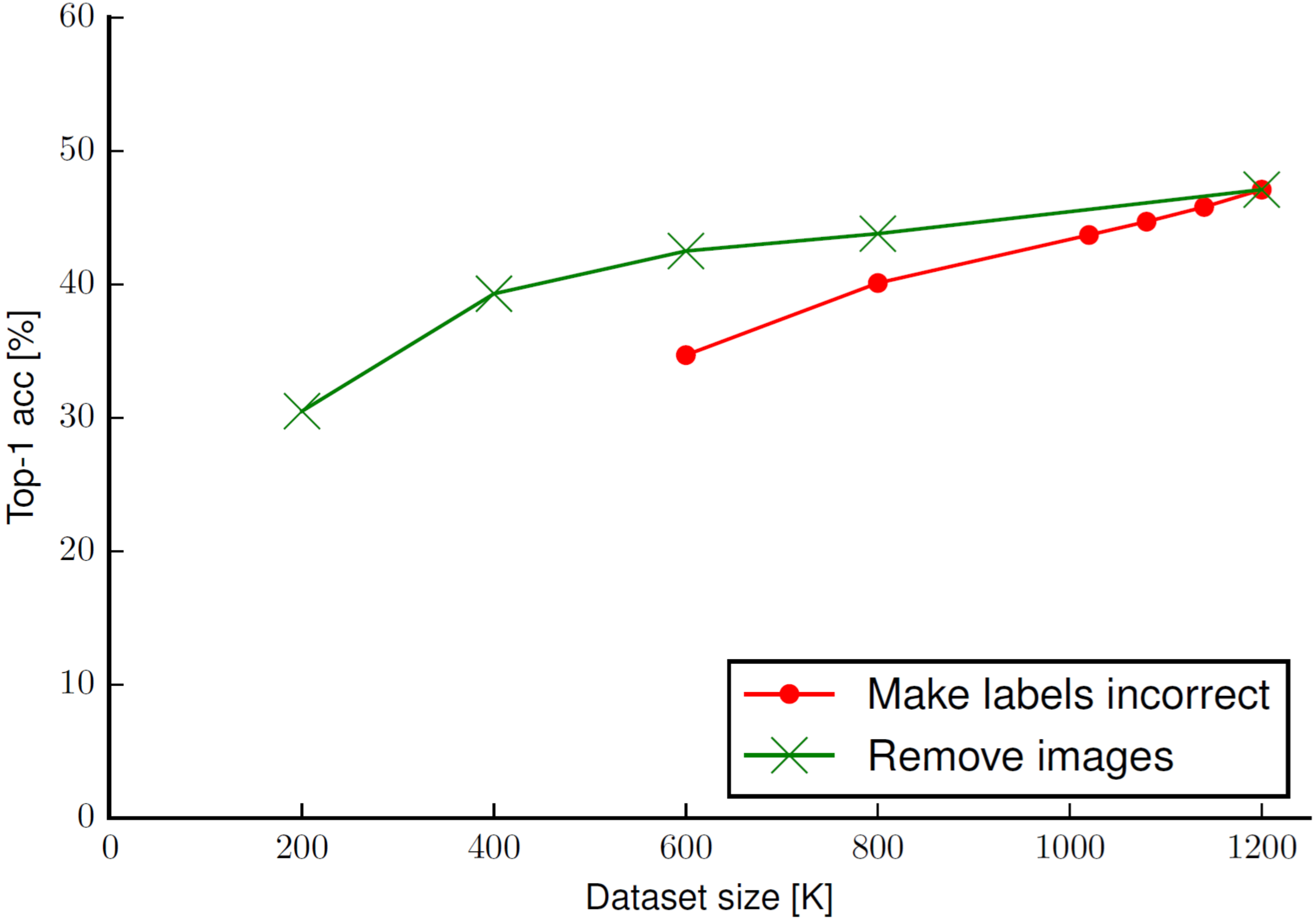
Name	Architecture	Non-linearity	Acc.
A	RGB → conv1x1x10→conv1x1x3	tanh	0.463
RGB	RGB	-	0.471
B	RGB → conv1x1x3→conv1x1x3	VLReLU	0.480
C	RGB → conv1x1x10→ conv1x1x3 + RGB	VLReLU	0.482
D	[RGB; log(RGB)] → conv1x1x10→ conv1x1x3	VLReLU	0.482
E	RGB → conv1x1x16→conv1x1x3	VLReLU	0.483
F	RGB → conv1x1x10→conv1x1x3	VLReLU	0.485





Mishkin et.al. Systematic evaluation of CNN advances on the ImageNet, arXiv 2016
Image credit: https://www.udacity.com/course/deep-learning--ud730

Dataset quality and size



Mishkin et.al. Systematic evaluation of CNN advances on the ImageNet, arXiv 2016

NETWORK WIDTH: SATURATION AND SPEED PROBLEM

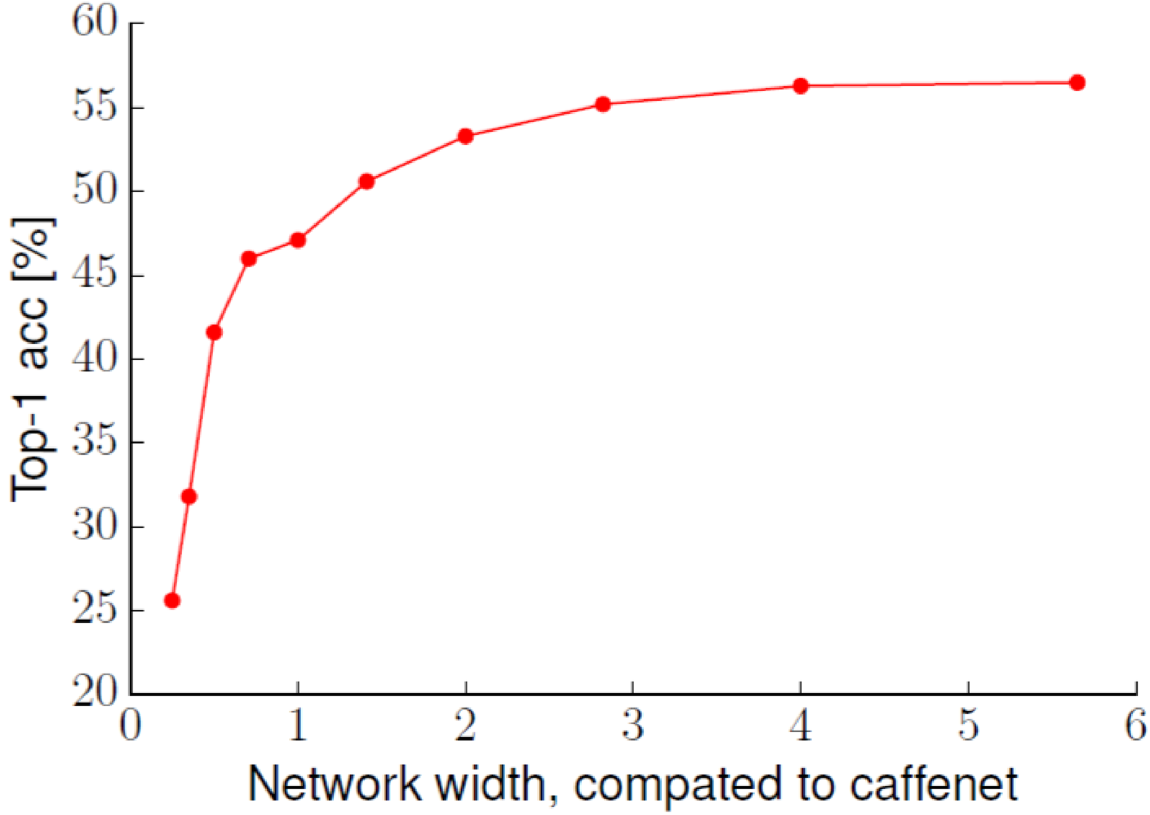
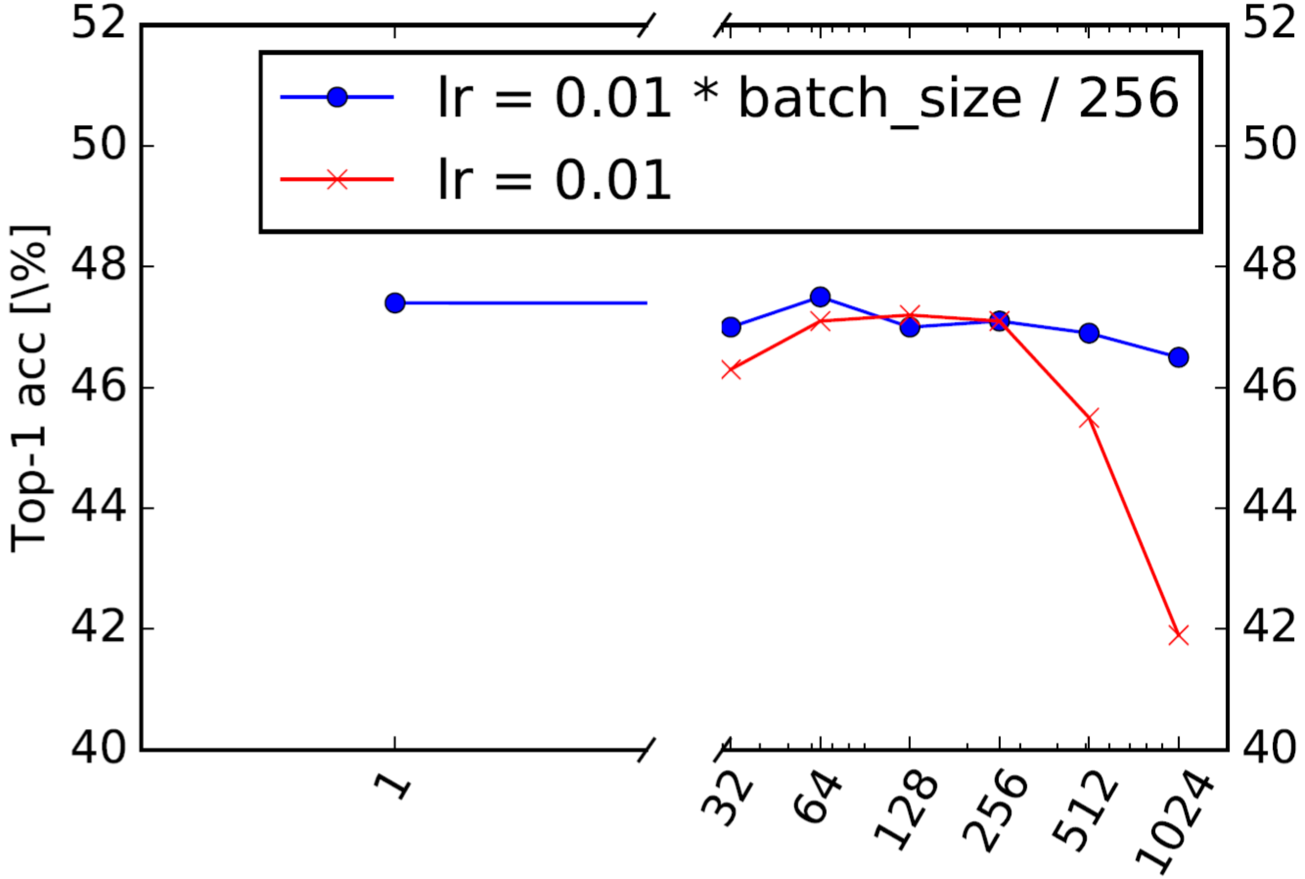


Figure 9: Network width impact on the accuracy.

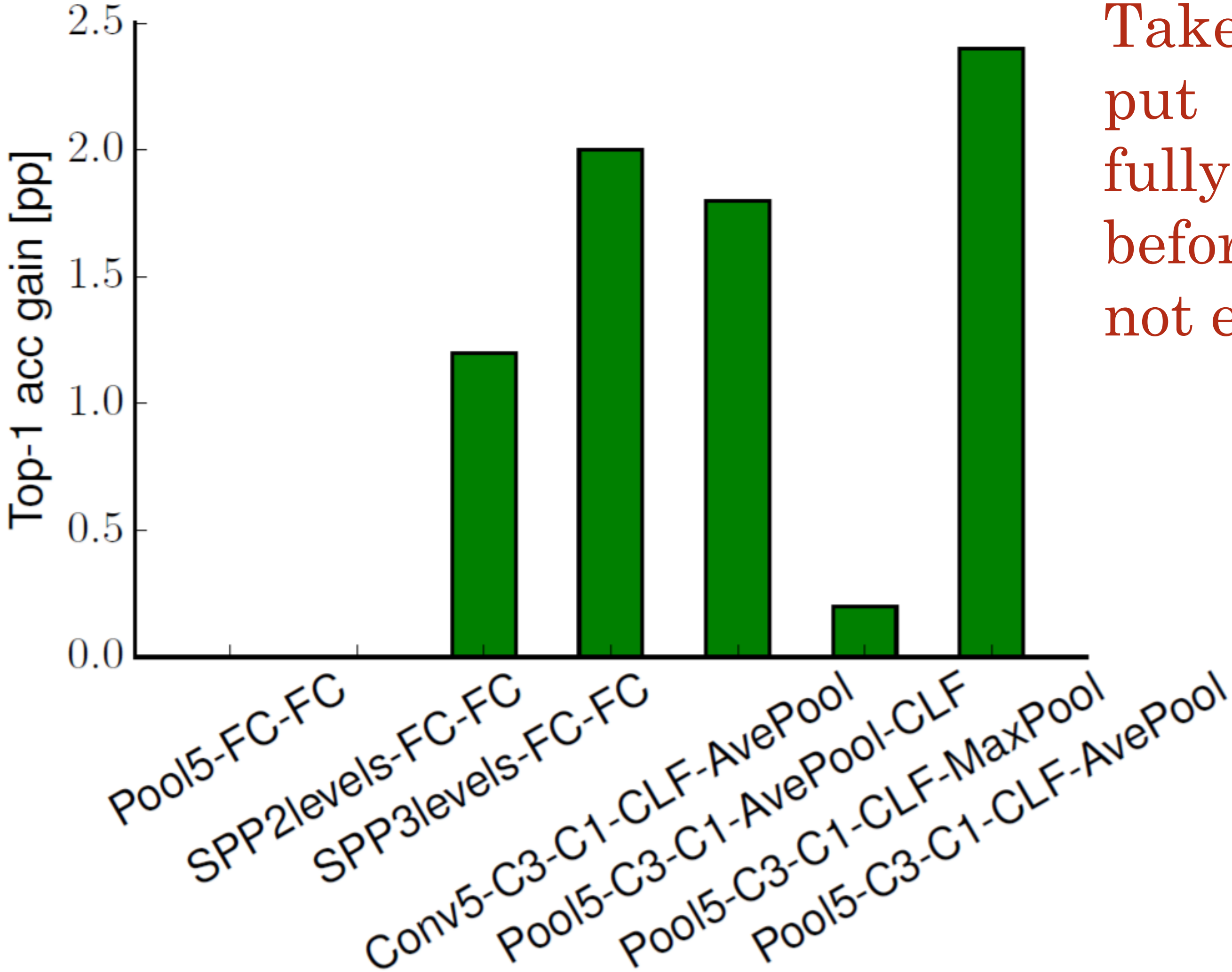
Mishkin et.al. Systematic evaluation of CNN advances on the ImageNet, arXiv 2016

BATCH SIZE AND LEARNING RATE



Mishkin et.al. Systematic evaluation of CNN advances on the ImageNet, arXiv 2016

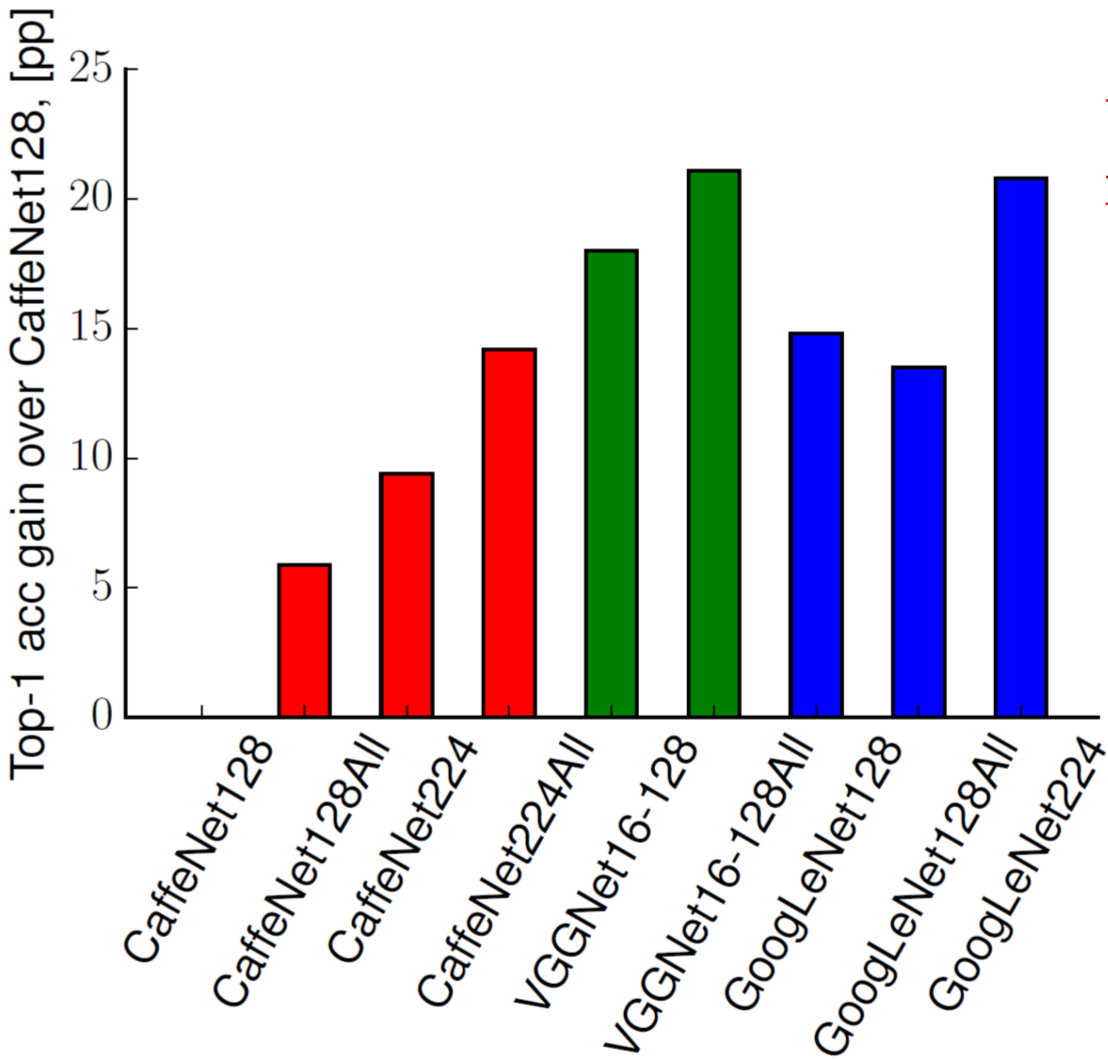
Classifier design



Take home:
put
fully-connected
before final layer,
not earlier

Mishkin et.al. Systematic evaluation of CNN advances on the ImageNet, arXiv 2016
Ren et.al Object Detection Networks on Convolutional Feature Maps, arXiv 2016

Applying altogether



>5 pp. additional top-1 accuracy for free.

Mishkin et.al. Systematic evaluation of CNN advances on the ImageNet, arXiv 2016

Take home messages

- use **ELU if without batchnorm** or **ReLU with BN**.
- apply a **learned colorspace** transformation of RGB (2 layers of 1x1 convolution).
- use the **linear learning rate** decay policy.
- use a **sum of the average and max pooling** layers.
- use mini-batch size around 128 or 256. If this is too big for your GPU, decrease the learning rate proportionally to the batch size.
- use **fully-connected layers as convolutional** and average the predictions for the final decision.
- when investing in increasing training set size, check if a plateau has not been reach.
- **cleanliness of the data is more important than the size.**
- if **you cannot increase the input image size, reduce the stride** in the consequent layers, it has roughly the same effect.
- if **your network has a complex and highly optimized architecture**, like e.g. GoogLeNet, **be careful** with modifications.

Thank you for the attention

- Any questions?
  - All logs, graphs and network definitions: https://github.com/ducha-aiki/caffenet-benchmark
    Feel free to add your tests
  - The paper is here: https://arxiv.org/abs/1606.02228

ducha.aiki@gmail.com
mishkdmy@cmp.felk.cvut.cz

ARCHITECTURE

- Use as small filters as possible
  - 3x3 + ReLU + 3x3 + ReLU > 5x5 + ReLU.
  - 3x1 + 1x3 > 3x3.
  - 2x2 + 2x2 > 3x3
- Exception: 1st layer. Too computationally ineffective to use 3x3 there.

	top-1	top-5	d	stage 1	pool	stage 2	pool	stage 3	pool	stage 4	comp.
A	37.4	15.9	5	$(7, 64)_{/2}$	$3_{/3}$	(5, 128)	$2_{/2}$	(3, 256)×3			1
B	35.7	14.9	8	$(7, 64)_{/2}$	$3_{/3}$	(5, 128)	$2_{/2}$	(2, 256)×6			0.96
C	35.0	14.3	6	$(7, 64)_{/2}$	$3_{/3}$	(3, 128)×2	$2_{/2}$	(3, 256)×3			1.02
D	34.5	13.9	9	$(7, 64)_{/2}$	$3_{/3}$	(3, 128)×2	$2_{/2}$	(2, 256)×6			0.98
E	<u>33.8</u>	<u>13.3</u>	11	$(7, 64)_{/2}$	$3_{/3}$	(2, 128)×4	$2_{/2}$	(2, 256)×6			0.99
F	35.5	14.8	8	$(7, 64)_{/2}$	$3_{/3}$	(5, 128)	$2_{/2}$	(3, 160)×5+(3, 256)			1
G	35.5	14.7	11	$(7, 64)_{/2}$	$3_{/3}$	(5, 128)	$2_{/2}$	(3, 128)×8+(3, 256)			1
H	34.7	14.0	8	$(7, 64)_{/2}$	$3_{/3}$	(3, 64)×3+(3, 128)	$2_{/2}$	(3, 256)×3			0.97
I	<u>33.9</u>	<u>13.5</u>	11	$(7, 64)_{/2}$	$3_{/3}$	(3, 64)×3+(3, 128)	$2_{/2}$	(2, 256)×6			0.93
J	**32.9**	**12.5**	11	$(7, 64)_{/2}$	$3_{/3}$	(2, 128)×4	$2_{/2}$	(2, 256)×4	$3_{/3}$	(2, 2304)+(2, 256)	0.98

Table 1. **Configurations of the models under constrained time complexity**. The notation (s, n) represents the filter size and the number of filters. "/2" represents stride = 2 (default 1). "$\times k$" means the same layer configuration is applied k times (not sharing weights). "+" means another layer is followed. The numbers in the pooling layer represent the filter size and also the stride. All convolutional layers are with ReLU. The feature map size of stage 2 is dominantly 36×36, of stage 3 is 18×18, and of stage 4 (if any) is 6×6. The top-1/top-5 errors (at 75 epochs) are on the validation set. The "comp." is the theoretical time complexity (relative to A) computed using Eqn.(1).

Convolutional Neural Networks at Constrained Time Cost. He and Sun, CVPR 2015

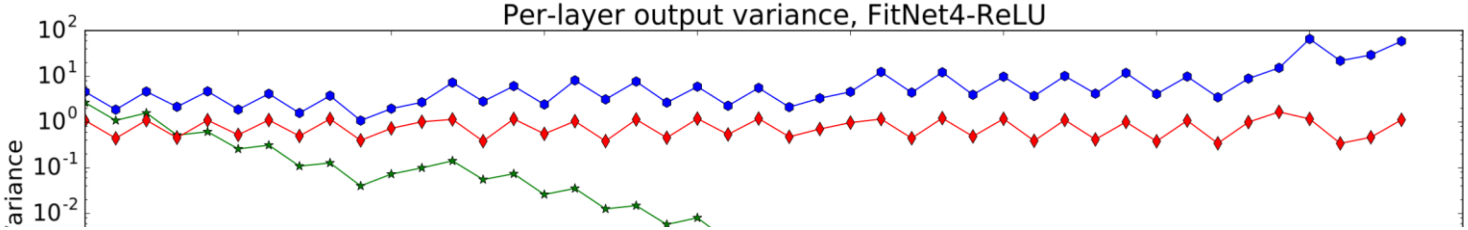
WEIGHT INITIALIZATION FOR A VERY DEEP NET

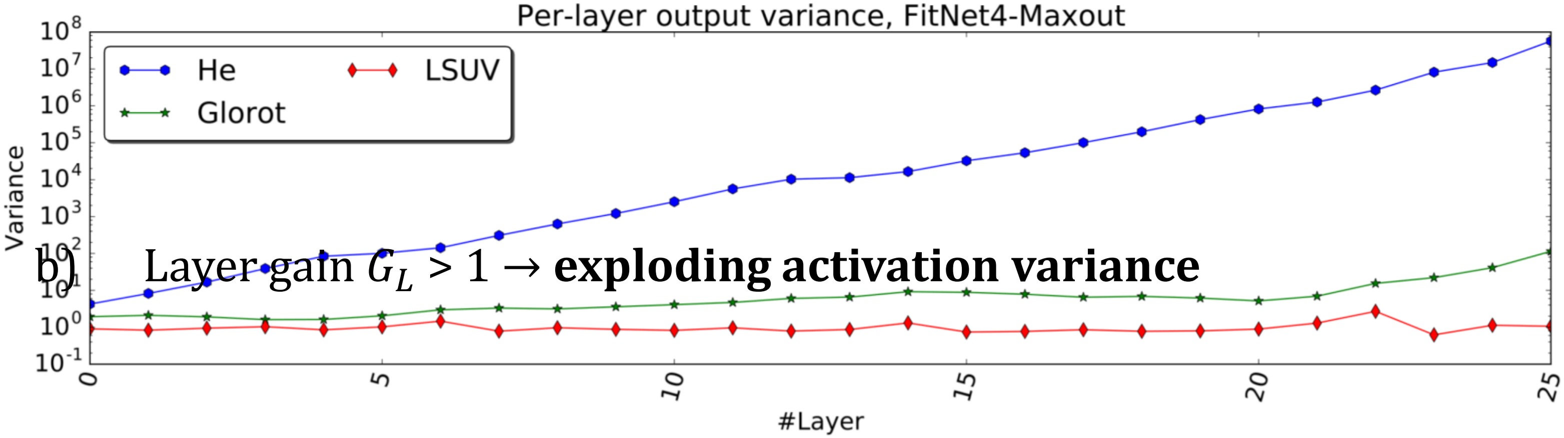
- Gaussian noise with variance.
  - $var(\omega_l) = 0.01$ (AlexNet, Krizhevsky et.al, 2012)
  - $var(\omega_l) = 1/n_{inputs}$ (Glorot et.al. 2010)
  - $var(\omega_l) = 2/n_{inputs}$ (He et.al. 2015)
- Orthonormal: (Saxe et.al. 2013)
  Glorot → SVD → ω_l = V
- Data-dependent: LSUV (Mishkin et.al, 2016)

Mishkin and Matas. All you need is a good init. ICLR, 2016

WEIGHT INITIALIZATION INFLUENCES ACTIVATIONS

a) Layer gain $G_L < 1 \rightarrow$ **vanishing activations variance**

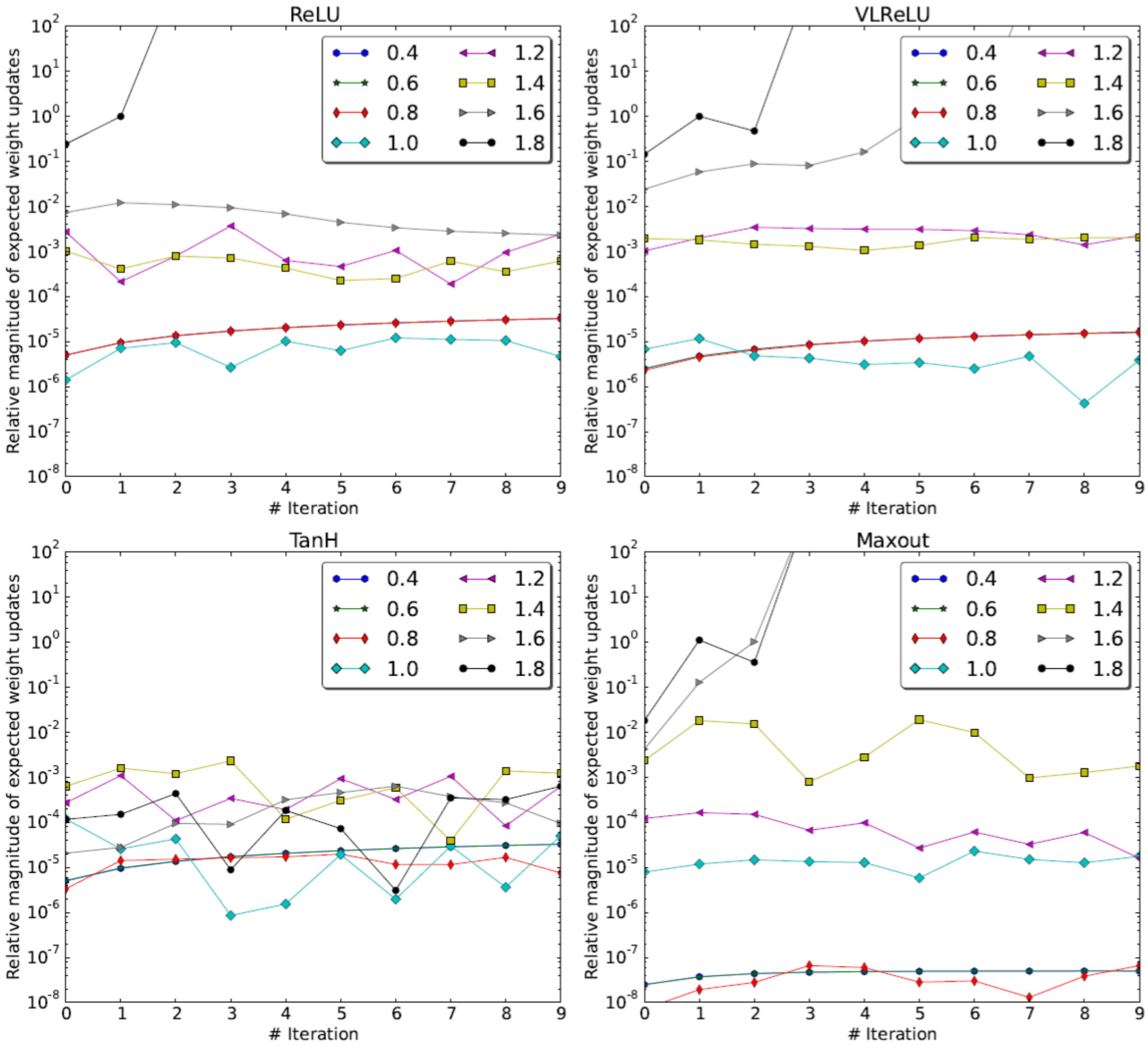




b) Layer gain $G_L > 1 \rightarrow$ **exploding activation variance**

Mishkin and Matas. All you need is a good init. ICLR, 2016

ACTIVATIONS INFLUENCES MAGNITUDE OF GRADIENT COMPONENTS



Mishkin and Matas. All you need is a good init. ICLR, 2016

Layer-sequential unit-variance orthogonal initialization

Algorithm 1. Layer-sequential unit-variance orthogonal initialization.
$\boldsymbol{L}$ − convolution or fully-connected layer, W_L − its weights, O_L − layer output,
ε − variance tolerance,
T_i − iteration number, T_{max} − max number of iterations.

Pre-initialize network with orthonormal matrices as in Saxe et.al. (2013)
for each convolutional and fully-connected layer $\boldsymbol{L}$ **do**
    do forward pass with mini-batch
        calculate $var(O_L)$
        $W_L^{i+1} = W_L^i / \sqrt{var(O_L)}$
    until $|var(O_L) - 1.0| < \varepsilon$ **or** $(T_i > Tmax)$
end for

*The LSUV algorithm does not deal with biases and initializes them with zeros

Mishkin and Matas. All you need is a good init. ICLR, 2016

Comparison of the initializations for different activations

- CIFAR-10 FitNet, accuracy [%]

Init method	Maxout	ReLU	VLReLU	tanh
LSUV	**93.94**	**92.11**	92.97	89.28
OrthoNorm	93.78	91.74	92.40	89.48
Xavier	91.75	90.63	92.27	**89.82**
MSRA	n/c†	90.91	92.43	89.54
OrthoNorm MSRA-scaled	–	91.93	**93.09**	–

- CIFAR-10 FitResNet, accuracy [%]

Init method	maxout	ReLU	VLReLU	tanh
LSUV	**94.16**	**92.82**	**93.36**	89.17
OrthoNorm	n/c	91.42	n/c	89.31
Xavier	n/c	92.48	**93.34**	**89.62**
MSRA	n/c	n/c	n/c	88.59
–	–	–	–	–

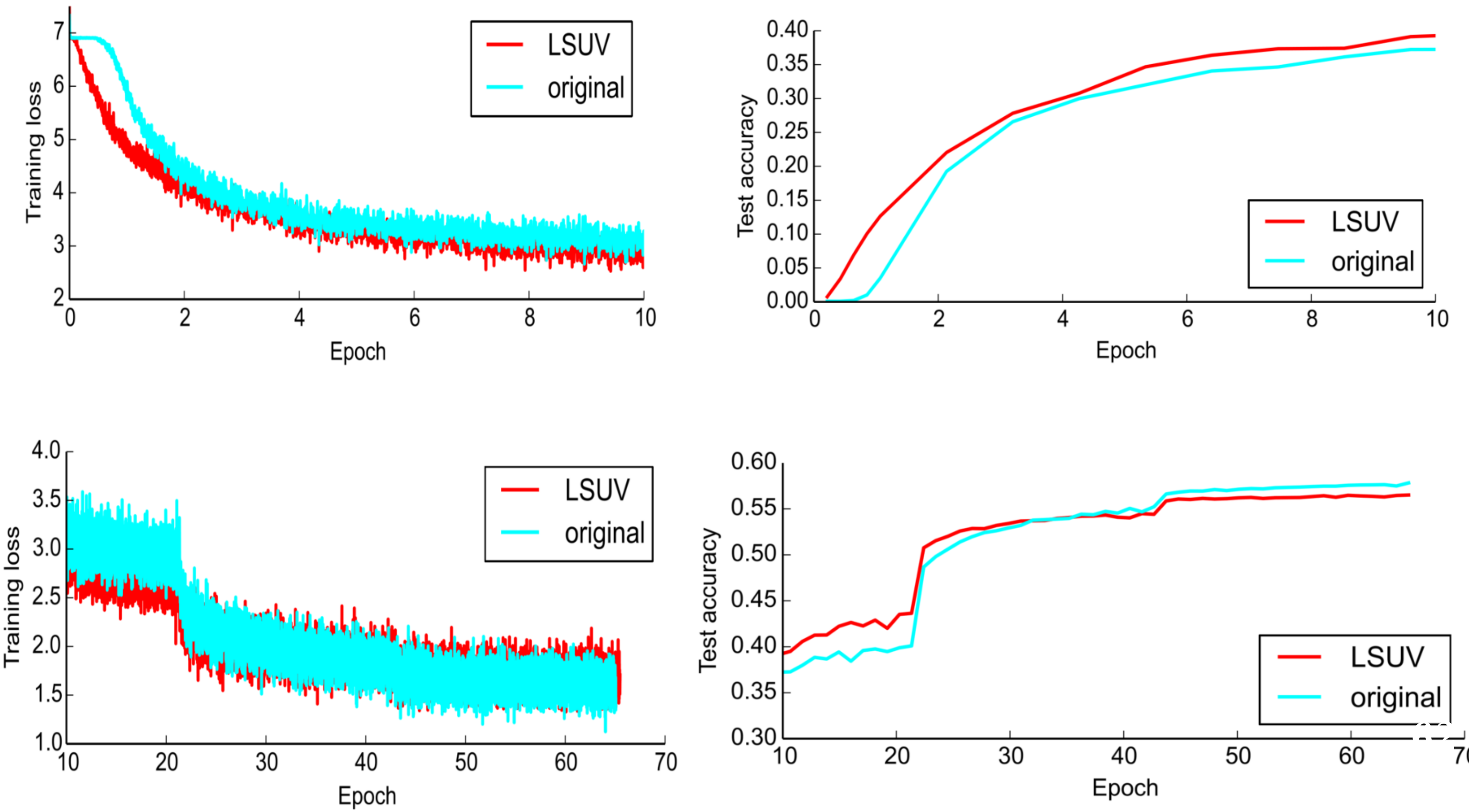
Mishkin and Matas. All you need is a good init. ICLR, 2016

LSUV initialization implementations

- Caffe https://github.com/ducha-aiki/LSUVinit
- Keras https://github.com/ducha-aiki/LSUV-keras
- Torch https://github.com/yobibyte/torch-lsuv

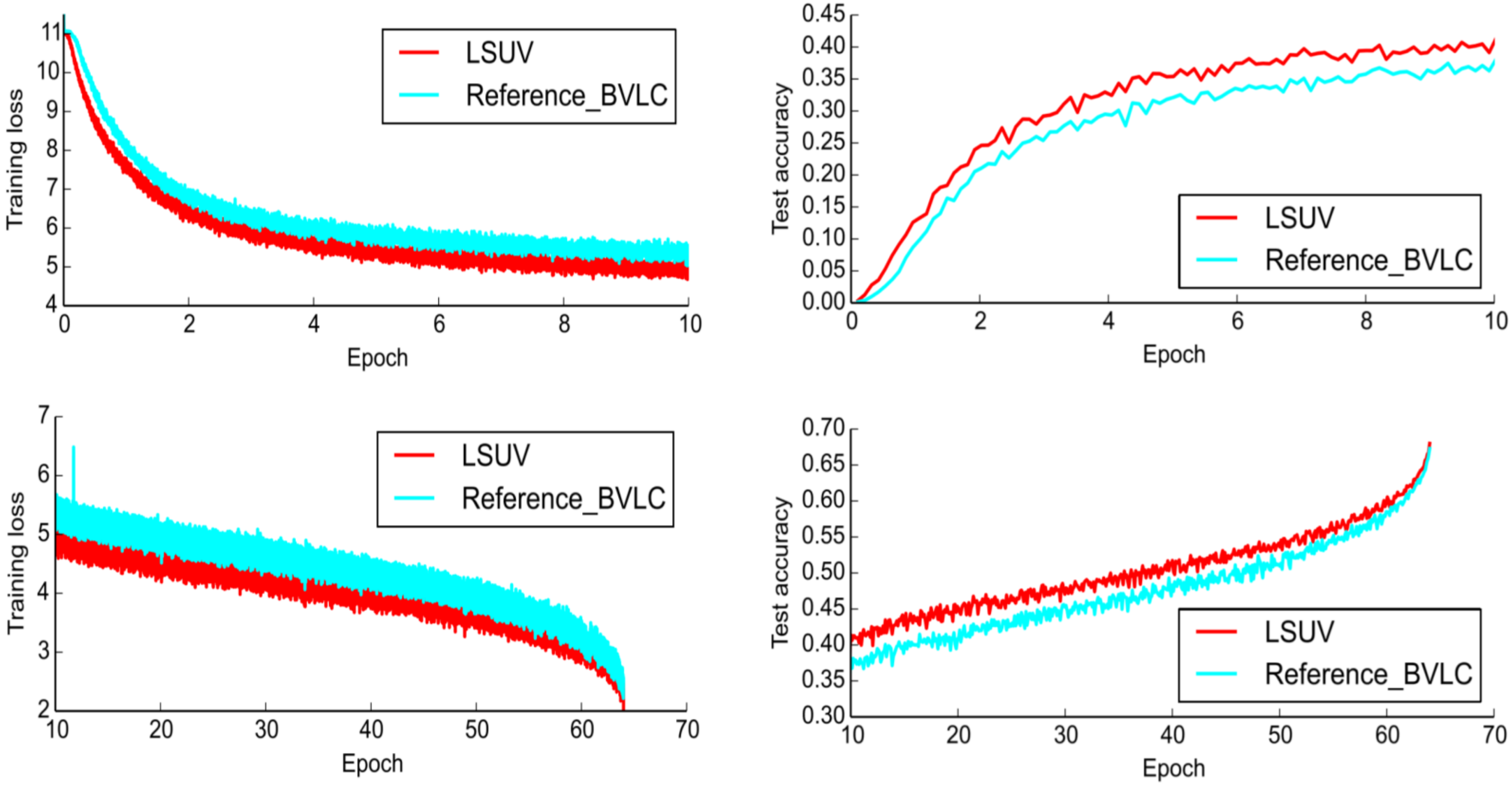
Mishkin and Matas. All you need is a good init. ICLR, 2016

CaffeNet training



Mishkin and Matas. All you need is a good init. ICLR, 2016

GoogLeNet Training



Mishkin and Matas. All you need is a good init. ICLR, 2016