Systematic evaluation of CNN advances on the ImageNet

crackhopper

Tue Dec 27 21:27:23 2016

Outline

Systematic evaluation of CNN advances on the ImageNet

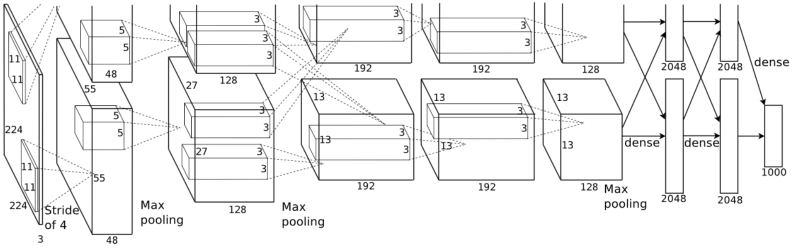
Dataset

- ImageNet-2012
- rescale size to $144 \times N$
- cropping a random 128×128 square
- random mirror
- out $= 0.04(\text{BGR} - (104; 117; 124))$

Network Structure

Modified CaffeNet:

- 96C11/4 -> MP3/2 -> 256G2C5/2 -> MP3/2 -> 384G2C3 -> 384C3 -> 256G2C3 -> MP3/2 -> 2048C3 -> 2048C1 -> 1000C1



Network Structure

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

Systematic evaluation of CNN advances on the ImageNet

Previous Work

ReLU like

- ReLU
- LeakyReLU
- Very Leaky ReLU
- RReLU
- PReLU
- APL
- ELU

Other Ideas

- maxout
- Multi-Bias Non-linear Activation

ReLU

Deep rectifier networks can reach their best performance without requiring any unsupervised pre-training on purely supervised tasks with large labeled datasets. [1]

ReLU

$$y = \max(x, 0)$$

[1] Deep sparse rectifier neural networks

LReLU/VLReLU

Leaky ReLU

Leaky ReLU [a]

$$y = \mathsf{max}(x, \alpha x), \alpha \sim 0.01$$

[a]Rectifier nonlinearities improve neural network acoustic models

Very Leaky ReLU

Very Leaky ReLU [a]

$$y = \mathsf{max}(x, \alpha x), \alpha \in \{0.1, 0.5\}$$

(original paper is about introducing sparsity into the convolutional kernels by learning a basis of kernels, or by PCA decompose which also treat channel as variables)

[a]`https://www.kaggle.com/c/cifar-10/forums/t/10493/`

RReLU/PReLU

Random ReLU

An experimental paper [a]

$$y = \text{max}(x, \alpha x), \alpha = \text{random}(0.1, 0.5)$$

[a] Empirical Evaluation of Rectified Activations in Convolutional Network

Parametric ReLU

Experimental paper, and some inference about ReLU initialization [a]

$$y = \text{max}(x, \alpha x), \alpha \text{is learnable}$$

[a] Delving deep into rectifiers: Surpassing human-level performance on imagenet classification

APL

Adaptive Piecewise Linear Units[2]. Designed a novel form of piecewise linear activation function that is learned independently for each neuron using gradient descent.

$$y = \mathsf{max}(x, 0) + \sum_{s=1}^{S} a_i^s \mathsf{max}(0, -x + b_i^s)$$

where i is the index of activation unit, and S is a pre-defined hyper parameter.

[2]Learning Activation Functions to Improve Deep Neural Networks

ELU

Exponential Linear Units[3].

$$y = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^{-x} - 1) & \text{if} x \leq 0 \end{cases}$$

(The paper is a little hard, but analyze the bias shift effect, it is worthy reading when available)

[3] Fast and Accurate Deep Network Learning by Exponential Linear Units

ELU graph

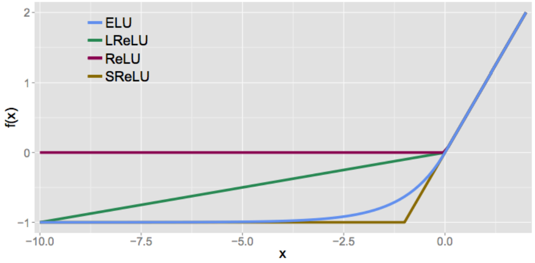


Figure 1: The rectified linear unit (ReLU), the leaky ReLU (LReLU, $\alpha = 0.1$), the shifted ReLUs (SReLUs), and the exponential linear unit (ELU, $\alpha = 1.0$).

Maxout

Maxout Network [4]. In this experiment, Maxout is tested in two modifications: MaxWhaving the same effective network width, which doubles the number of parameters and computation costs because of the two linear pieces, and MaxShaving same computational complexity - with $\sqrt{2}$ thinner each piece.

Maxout

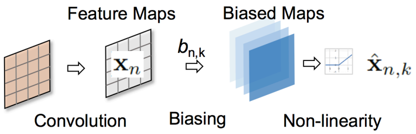
$$y = \max(W_1 x + b_1, W_2 x + b_2)$$

[4] Maxout networks

Multi-Bias Non-linear Activation

MBA: Multi-bias Non-linear Activation

The bias module at a glance:



$$\hat{\mathbf{x}}_{n,k} = \sigma(\mathbf{x}_n + b_{n,k}) \text{ for } k = 1, \ldots, K$$

where *n* is the index of feature maps after convolution and *K* being the number of biases.

- Decouple a feature map into several band/biased maps.
- Enforce different thresholds on the same signal.

Results

The best single performing activation function similar in complexity to ReLU:

- ELU
- PReLU

Surprisingly, Very Leaky ReLU, popular for DCGAN networks and for small datasets, does not outperforms vanilla ReLU.

The Swietojanski[5] et.al hypothesis about maxout power in the final layers is confirmed and combined ELU (after convolutional layers) + maxout (after fully connected layers) shows the best performance among non-linearities with speed close to ReLU. Wide maxout outperforms the rest of the competitors at a higher computational cost.

[5] Investigation of maxout networks for speech recognition

Results

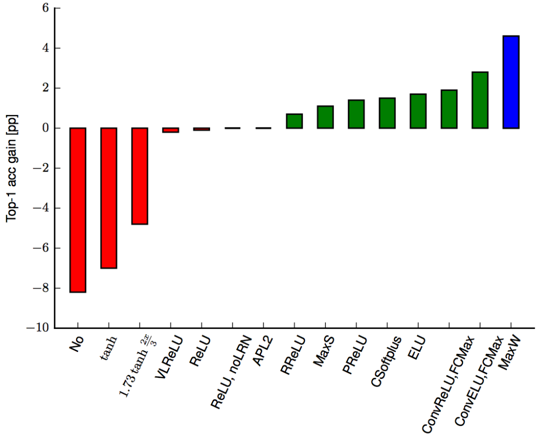


Figure 2: Top-1 accuracy gain over ReLU in the CaffeNet-128 architecture. MaxS stands for "maxout, same compexity", MaxW – maxout, same width, CSoftplus – centered softplus. The baseline, i.e. ReLU, accuracy is 47.1%.

Systematic evaluation of CNN advances on the ImageNet

Previous Work

- max pooling
- average pooling
- Stochastic pooling
- LP-Norm pooling
- Tree-Gated pooling
- sum of average and max poolin
- strided convolutions

max/average pooling

max pooling

$$y = \mathsf{max}_{i,j}^{h,w} x_{i,j}$$

average pooling

$$y = \frac{1}{hw} \sum_{i,j=1}^{h,w} x_{i,j}$$

Stochastic pooling

Select the pooled map response by sampling from a multinomial distribution formed from the activations of each pooling region. [6]

$$p_i = \frac{a_i}{\sum_{k \in R_j} a_k}$$

Using stochastic pooling at test time introduces noise into the networks predictions which we found to degrade performance. Instead, we use a probabilistic form of averaging.

$$s_j = \sum_{i \in R_j} p_i a_i$$

[6] Stochastic Pooling for Regularization of Deep Convolutional Neural Networks

LP-Norm pooling

LP-Norm Pooling[7]

$$y_i = (\frac{1}{N}\sum_{j=1}^{N}|x_j - c_j|^{p_i})^{\frac{1}{p_i}}$$

Both p_i and c_j are model parameters that are learned.

- when activations are all positive, then $p_i = 1$ means average pooling
- when activations are all positive, then $p_i = \infty$ means max pooling
- when $0 < p_i < 1$ the definition is not a norm anymore due to the violation of triangle inequality. In practice, we re-parameterize p_i by $1 + \log(1 + e^{p_i})$ to satisfy this constraint.

[7] Learned-Norm Pooling for Deep Feedforward and Recurrent Neural Networks

Tree-Gated pooling

Gerenalize pooling by sum [8]

"Mixed" max-average pooling

$$f_{mix}(x) = a_l \cdot f_{max}(x) + (1 - a_l) \cdot f_{avg}(x)$$

where the α_l is a learned parameter

"Gated" max-average pooling

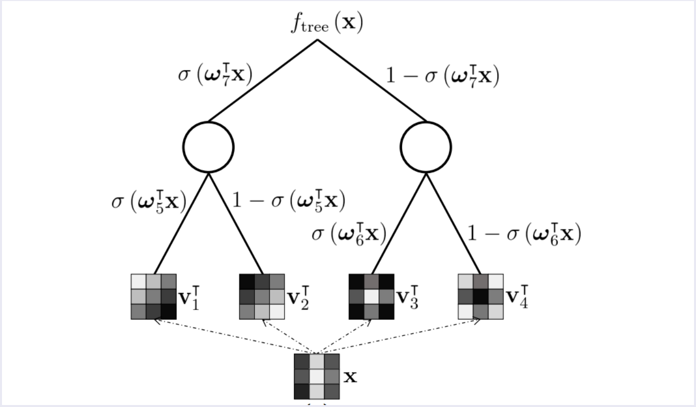
Let $a_l = \sigma(w^T x)$, where w is a parameter to be learned. i.e.

$$f_{gate}(x) = \sigma(w^T x) \cdot f_{max}(x) + (1 - \sigma(w^T x)) \cdot f_{avg}(x)$$

[8] Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree

Tree-Gated pooling

Tree pooling



$f_{\text{tree}}(\mathbf{x})$
$\sigma(\boldsymbol{\omega}_7^\intercal \mathbf{x})$
$1-\sigma(\boldsymbol{\omega}_7^\intercal \mathbf{x})$
$\sigma(\boldsymbol{\omega}_5^\intercal \mathbf{x})$
$1-\sigma(\boldsymbol{\omega}_5^\intercal \mathbf{x})$
$\sigma(\boldsymbol{\omega}_6^\intercal \mathbf{x})$
$1-\sigma(\boldsymbol{\omega}_6^\intercal \mathbf{x})$
$\mathbf{v}_1^\intercal$
$\mathbf{v}_2^\intercal$
$\mathbf{v}_3^\intercal$
$\mathbf{v}_4^\intercal$
$\mathbf{x}$

both w and v are to be learned.

Strided convolutions

max-pooling can simply be replaced by a convolutional layer with increased stride without loss in accuracy on several image recognition benchmarks.[9]

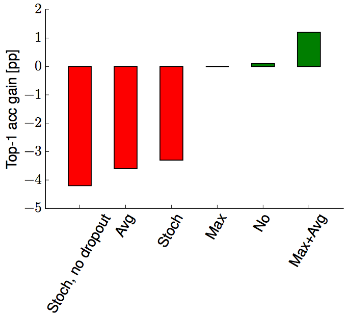
pooling can be removed from a network without abandoning the spatial dimensionality reduction by two means:

1. remove each pooling layer and increase the stride of the convolutional layer that preceded it accordingly.
2. replace the pooling layer by a normal convolution with stride larger than one

The second one do not hurt the performance.

[9]Striving For Simplicity: The All Convolutional Net

Results



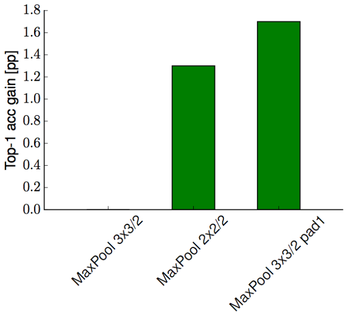
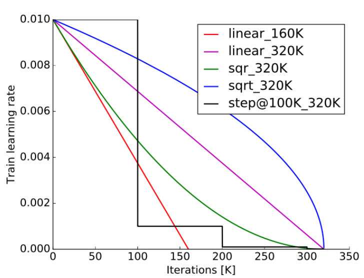


Figure 3: Top-1 accuracy gain over max pooling for the CaffeNet-128 architecture. Left – different pooling methods, right – different receptive field sizes. Stoch stands for stochastic pooling, "stoch no dropout" – for a network with stochastic pooling and turned off drop6 and drop7 layers.

Systematic evaluation of CNN advances on the ImageNet

Learning rate policy



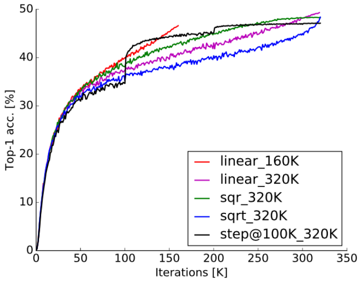
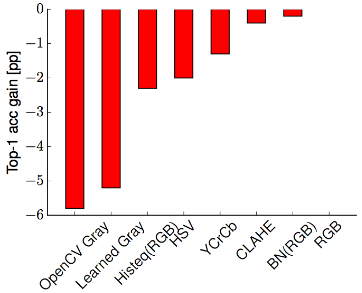


Figure 4: Left: learning rate decay policy, right: validation accuracy. The formulas for each policy are given in Table 5

Image pre-processing

The commonly used input to CNN is raw RGB pixels and the commonly adopted recommendation is not to use any pre-processing.



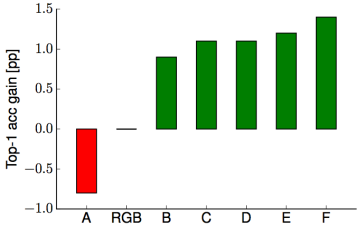


Figure 5: Left: performance of using various colorspaces and pre-processing. Right: learned colorspace transformations. Parameters are given in Table 6.

Image pre-processing

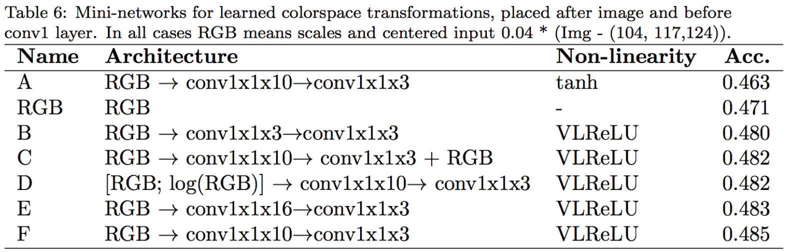


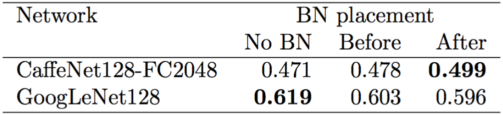
Table 6: Mini-networks for learned colorspace transformations, placed after image and before conv1 layer. In all cases RGB means scales and centered input 0.04 * (Img - (104, 117,124)).

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

Batch normalization

We have not changed any other parameters except using BN, while in the original paper, authors decreased regularization (both weight decay and dropout), changed the learning rate decay policy and applied an additional training set re-shuffling.

Table 7: Top-1 accuracy on ImageNet-128px, batch normalization placement. ReLU activation is used.



Network	BN placement		
	No BN	Before	After
CaffeNet128-FC2048	0.471	0.478	**0.499**
GoogLeNet128	**0.619**	0.603	0.596

Batch normalization with different activation function

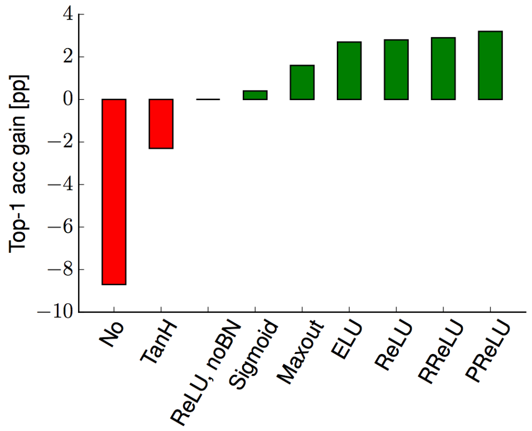


Figure 6: Top-1 accuracy gain over ReLU without batch normalization (BN) in CaffeNet-128 architecture. The baseline – ReLU – accuracy is 47.1%.

Classifier design

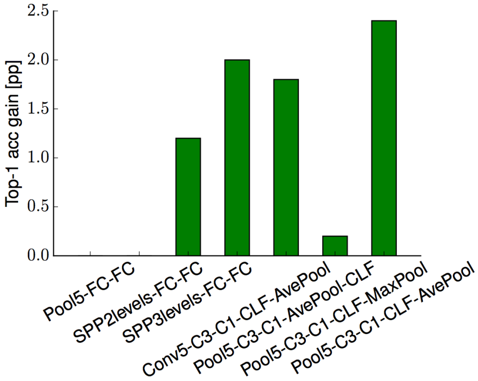


Figure 7: Classifier design: Top-1 accuracy gain over standard CaffeNet-128 architecture.

The best results are get, when predictions are averaged over all spatial positions and MLP layers are treated as convolution.

Batch size and learning rate

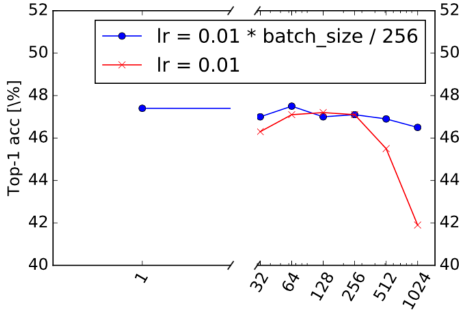


Figure 8: Batch size and initial learning rate impact to the accuracy

Network width

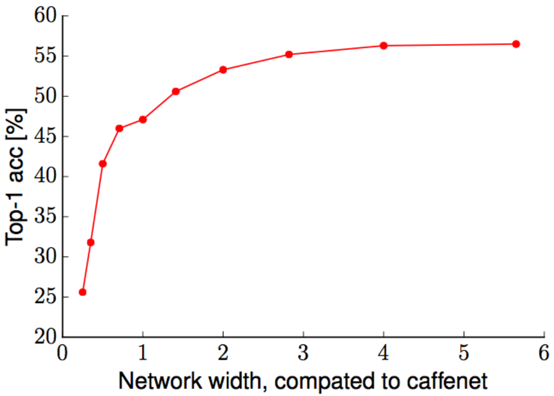


Figure 9: Network width impact on the accuracy.

Input image size

The gain from a large image size mostly (after some minimum value) comes from the larger spatial size of deeper layers than from the unseen image details.(? i doubt this argument)

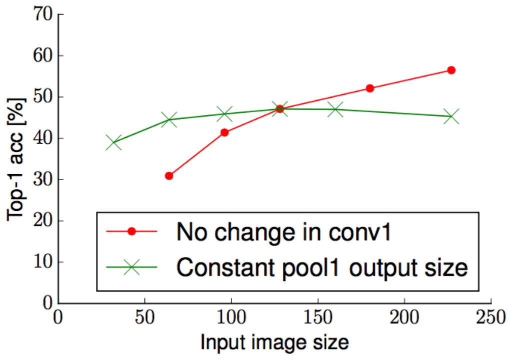


Figure 10: Input image size impact on the accuracy

Dataset size and noisy labels

a clean dataset with 400K images performs on par with 1.2M dataset with 800K correct images

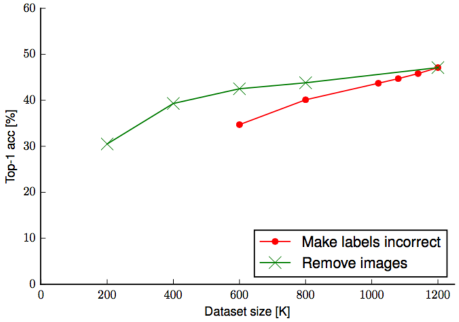


Figure 11: Training dataset size and cleanliness impact on the accuracy

Bias in convolution layers

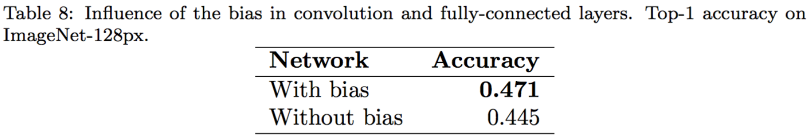


Table 8: Influence of the bias in convolution and fully-connected layers. Top-1 accuracy on ImageNet-128px.

Network	**Accuracy**
With bias	**0.471**
Without bias	0.445

Systematic evaluation of CNN advances on the ImageNet

Best-of-all experiments

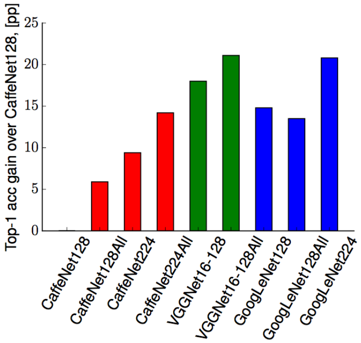


Figure 12: Applying all improvements that do not change feature maps size: linear learning rate decay policy, colorspace transformation "F", ELU nonlinearity in convolution layers, maxout non-linearity in fully-connected layers and a sum of average and max pooling.

Conclusions

- use ELU non-linearity without batchnorm or ReLU with it.
- apply a learned colorspace transformation of RGB.
- 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 then 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.