CNN backbones

Vlad Shakhuro









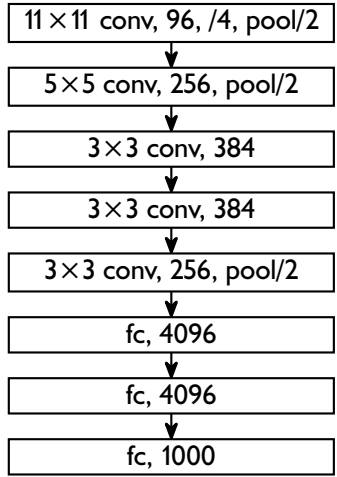


15 October 2026

Outline

1. AlexNet, VGG, Inception
2. ResNet and its improvements
3. Understanding and reusing CNN features
4. How good is ImageNet?

AlexNet



Krizhevsky et al. ImageNet classification with deep convolutional neural networks. NIPS 2012

Applying AlexNet to different resolutions

- Fixed resolution:



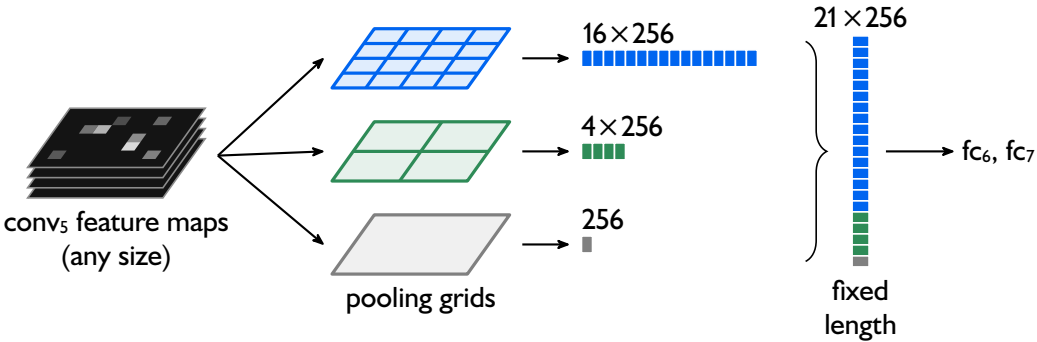
crop



warp

- Sample several random crops, average results
- Scan whole image with fixed size window, average scores

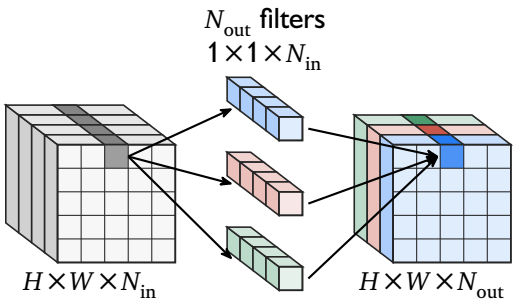
Spatial Pyramid Pooling



Only 1×1 grid — global average pooling

He et al. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. TPAMI 2015

1×1 convolutions

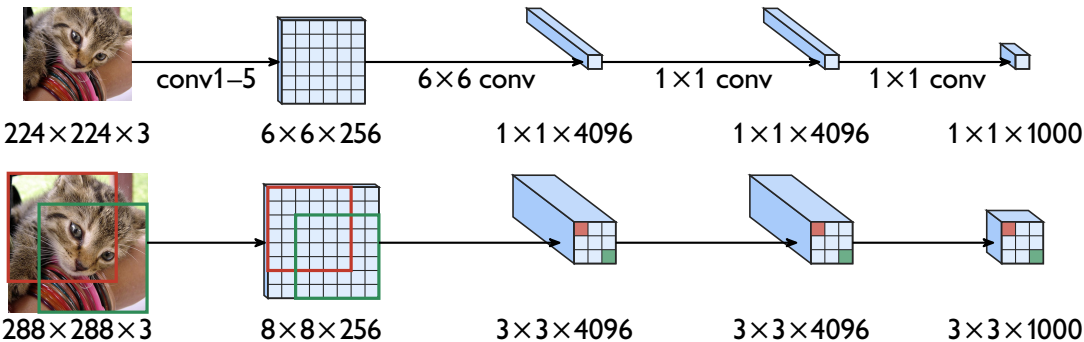


A 1×1 convolution has N_{out} filters of size $1 \times 1 \times N_{in}$. At every pixel it turns the N_{in} channel values into N_{out} weighted sums

Uses:

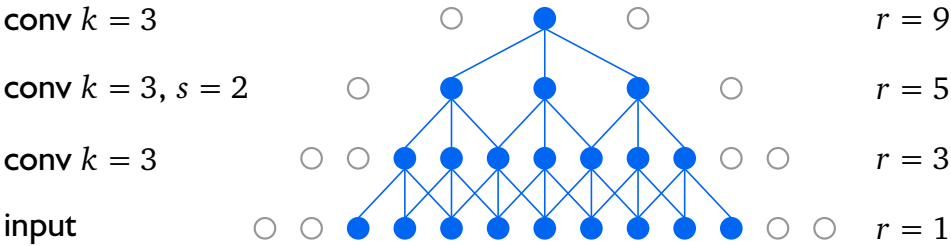
- with ReLU, stacked: an MLP at every pixel, a local classifier
- reduce depth ($N_{in} > N_{out}$)
- expand depth ($N_{in} < N_{out}$)

Fully convolutional AlexNet



- Dense layers as convs, same weights: the first 6×6, the others 1×1
- Larger input: scores of all 224×224 windows, stride 32, in one pass
- Input side $S = 224, 256, 288, \ldots$ gives output side $n = (S - 224)/32 + 1$

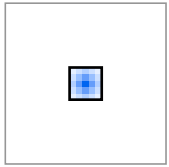
Receptive field



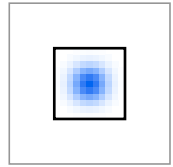
Receptive field (RF) of a neuron is the input region it depends on. A layer with kernel k_l and stride s_l grows it to

$$r_l = r_{l-1} + (k_l - 1) \prod_{i=1}^{l-1} s_i, \quad r_0 = 1$$

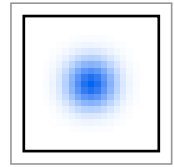
Effective receptive field



$n = 2$, RF 5×5

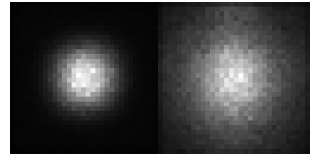


$n = 5$, RF 11×11



$n = 10$, RF 21×21

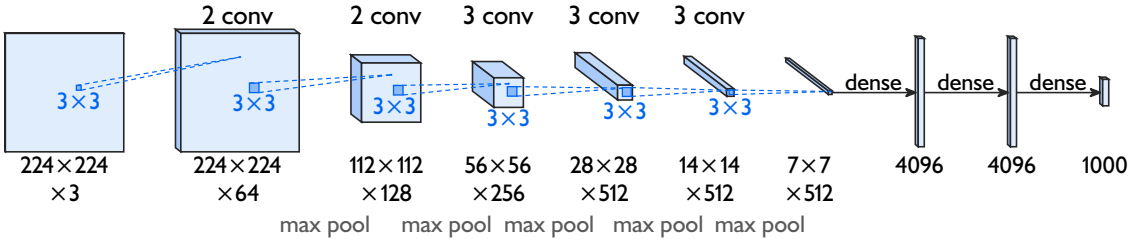
- ERF: pixels x with large $\partial y / \partial x$ for output y. Grows as $\sqrt{n}$
- Training enlarges the ERF



before training | after training

Luo et al. Understanding the Effective Receptive Field in Deep Convolutional Neural Networks. NIPS 2016

VGG

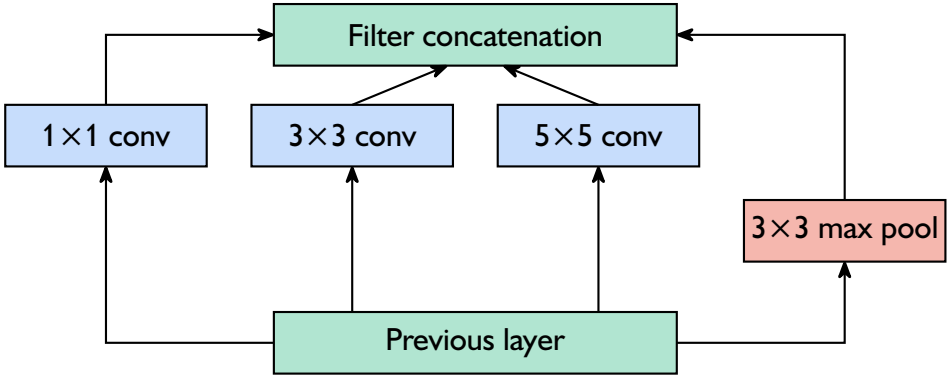


- Use only 3×3 convolutions
- Use only pooling for decreasing resolution

- Increase depth
- Increase #filters in 2 times after pooling

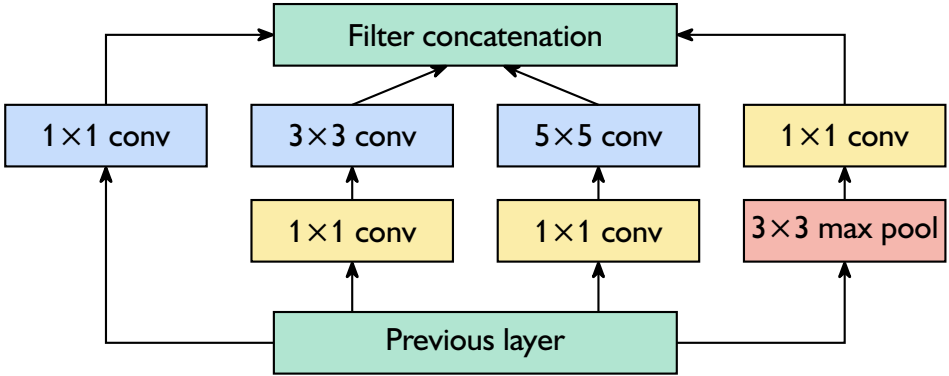
Simonyan, Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR 2015

Inception block



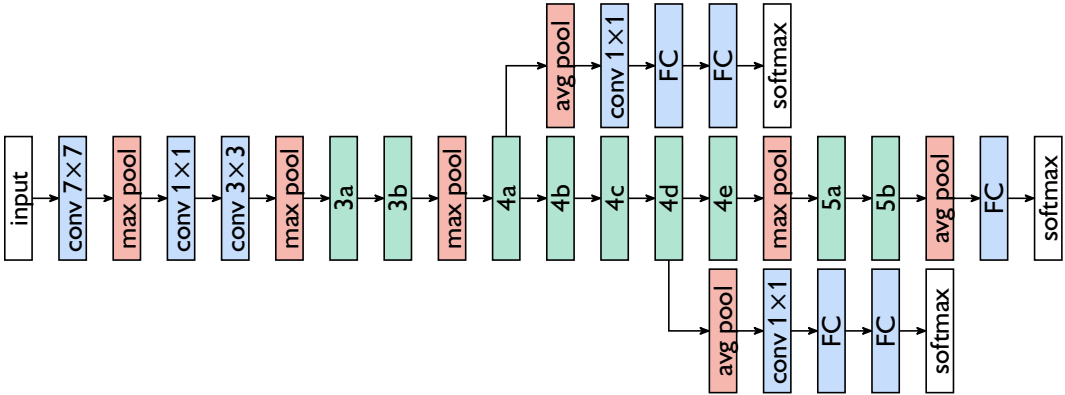
Szegedy et al. Going deeper with convolutions. CVPR 2015

Inception block with dim reduction



Szegedy et al. Going deeper with convolutions. CVPR 2015

Inception architecture



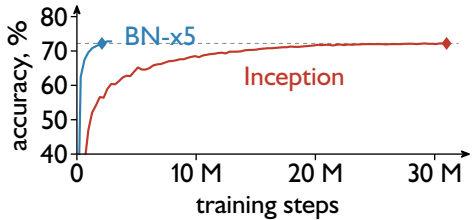
Side branches are extra classifiers, used only in training.
12 times fewer parameters than AlexNet

Szegedy et al. Going deeper with convolutions. CVPR 2015

Batch normalization

Batch norm at every training step normalizes each channel c:

$$\hat{x} = \frac{x - \mu_c}{\sqrt{\sigma_c^2 + \varepsilon}}, \quad y = \gamma_c \hat{x} + \beta_c$$



- As in classic ML, standardized features make the loss "rounder"
- $\mathrm{BN}(aw \circledast u + b) = \mathrm{BN}(w \circledast u)$:
  - Activations can't explode or vanish through depth
  - Gradient is $\perp$ to w, so each step lengthens w and shrinks the effective learning rate $\eta / \|w\|^2$, which tames steps that are too large

Ioffe, Szegedy. Batch normalization. ICML 2015
Santurkar et al. How does batch normalization help optimization? NeurIPS 2018
van Laarhoven. L2 regularization versus batch and weight normalization. arXiv 2017

Weight initialization

Single layer $y = \sum_{i=1}^{n} w_i x_i$:

$\mathrm{Var}[y] = n\ \mathrm{Var}[w]\ \mathrm{Var}[x]$

L layers, explodes or vanishes:

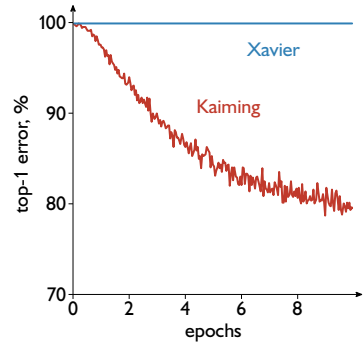
$\mathrm{Var}[y_L] = (n\ \mathrm{Var}[w])^L\ \mathrm{Var}[x]$

Xavier/Glorot:

$\mathrm{Var}[w] = \frac{1}{n_{\text{in}}}$ or $\frac{2}{n_{\text{in}}+n_{\text{out}}}$

Kaiming/He for ReLU activations:

$\mathrm{Var}[w] = \frac{2}{n_{\text{in}}}$



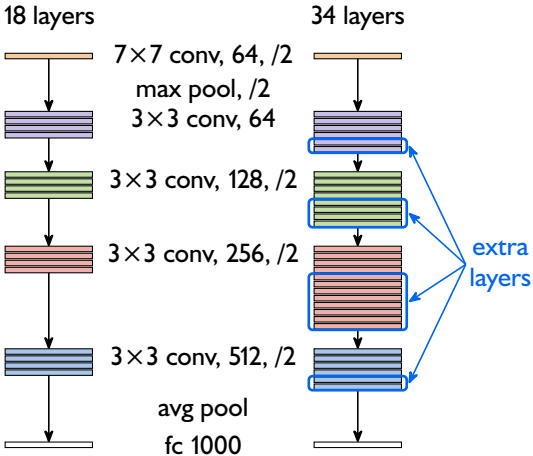
Glorot, Bengio. Understanding the difficulty of training deep feedforward neural networks. AISTATS 2010

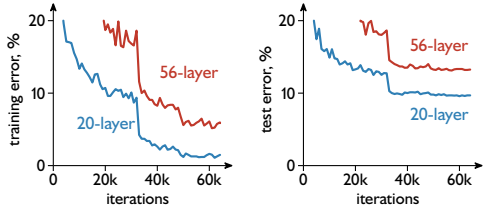
He et al. Delving deep into rectifiers. ICCV 2015

Outline

1. AlexNet, VGG, Inception
2. ResNet and its improvements
3. Understanding and reusing CNN features
4. How good is ImageNet?

Increasing network depth

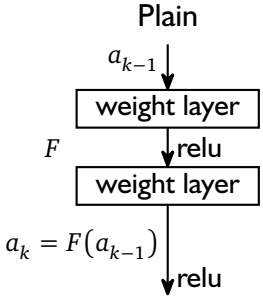


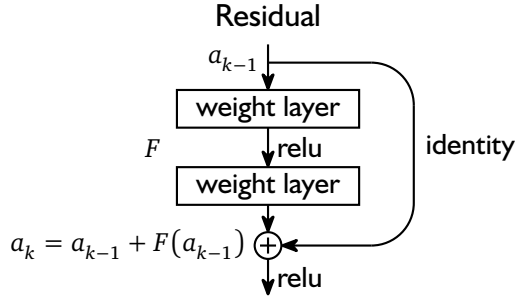


Simply increasing network depth doesn't work. However using identity layers we may obtain neural network of arbitrary depth. Therefore it's training problem

He et al. Deep residual learning for image recognition. CVPR 2016

Residual block

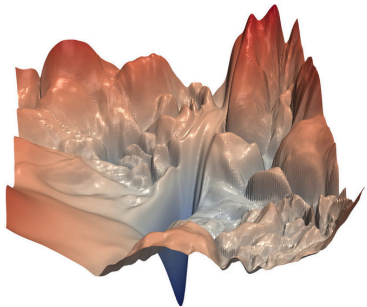




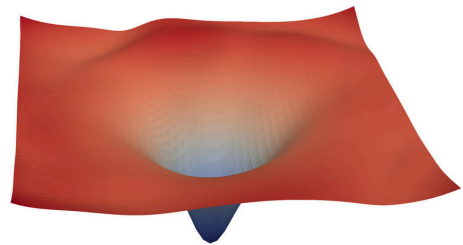
$$\frac{\partial L}{\partial a_k} = \frac{\partial L}{\partial a_n}\frac{\partial a_n}{\partial a_{n-1}}\cdots\frac{\partial a_{k+1}}{\partial a_k}$$

$$\frac{\partial L}{\partial a_k} = \frac{\partial L}{\partial a_n}\left(1 + \frac{\partial}{\partial a_k}\sum_{i=k}^{n-1} F(a_i)\right)$$

Skip connections smooth the landscape



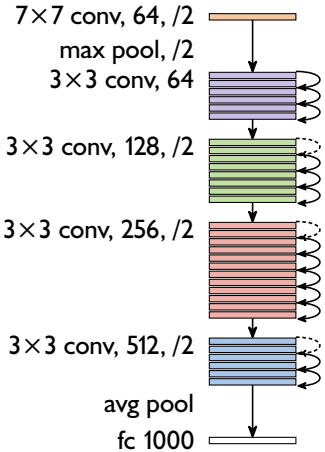
Without skip connections

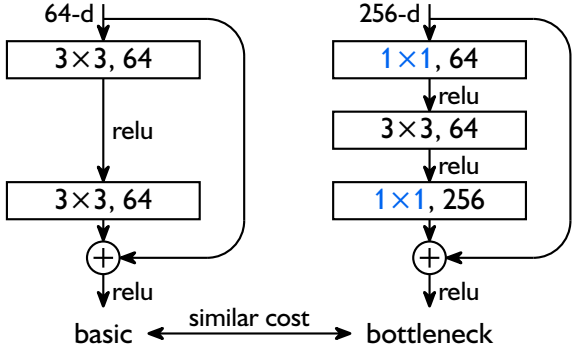


With skip connections

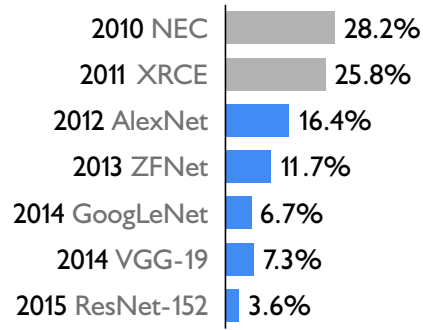
Li et al. Visualizing the loss landscape of neural nets. NeurIPS 2018

ResNet


7×7 conv, 64, /2
max pool, /2
3×3 conv, 64
3×3 conv, 128, /2
3×3 conv, 256, /2
3×3 conv, 512, /2
avg pool
fc 1000


64-d
3×3, 64
relu
3×3, 64
relu
basic
256-d
1×1, 64
relu
3×3, 64
relu
1×1, 256
relu
bottleneck
similar cost

ResNet results

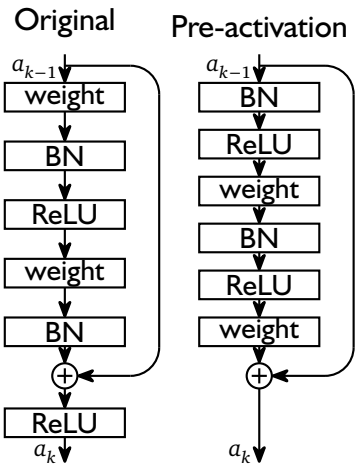


ILSVRC winners, top-5 error



ResNet-152 has less params and FLOPs than VGG-16

Pre-activation residual block



Original:

$$a_k = \mathrm{ReLU}(a_{k-1} + F(a_{k-1}))$$

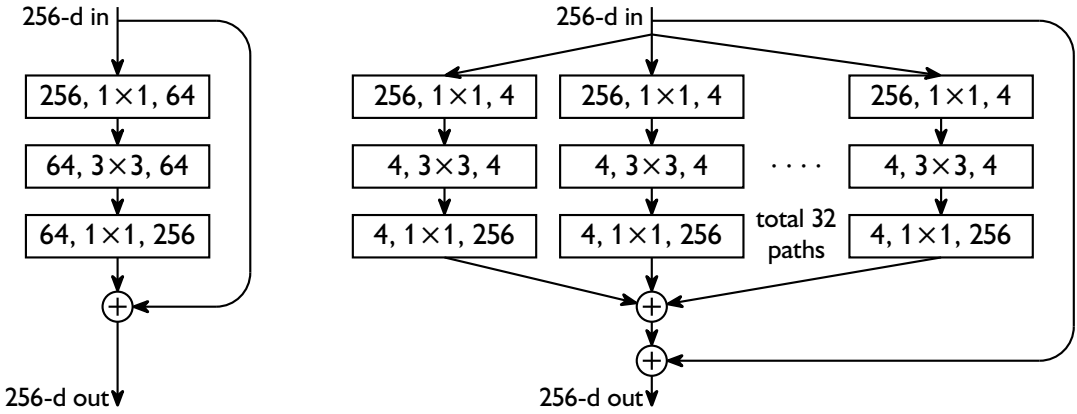
Pre-activation puts BN and ReLU inside F:

$$a_k = a_{k-1} + F(a_{k-1})$$

Test error, %	CIFAR-10		CIFAR-100	
	original	pre-act.	original	pre-act.
ResNet-164	5.9	5.5	25.2	24.3
ResNet-1001	7.6	4.9	27.8	22.7

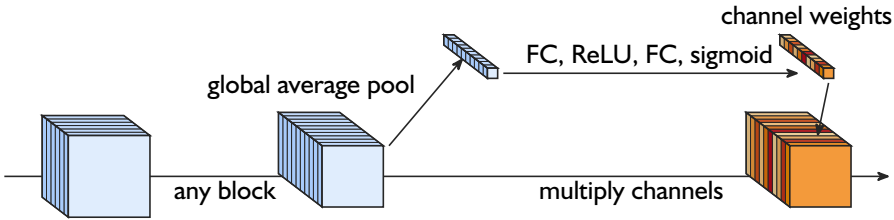
He et al. Identity mappings in deep residual networks. ECCV 2016

ResNeXt



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

Squeeze-and-Excitation

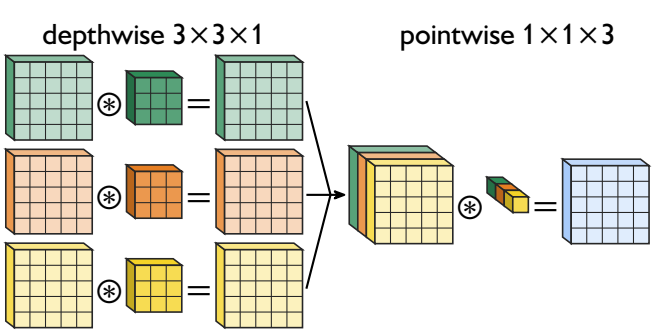


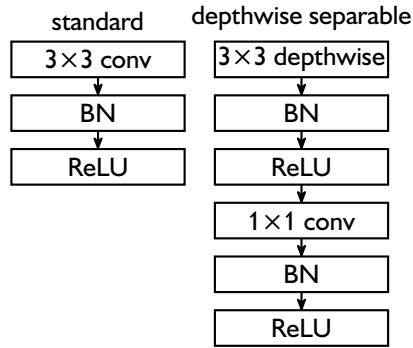
		Baseline		With SE	
Network	GFLOPs	top-1	top-5	top-1	top-5
ResNet-50	3.9	24.8	7.5	23.3	6.6
ResNet-101	7.6	23.2	6.5	22.4	6.1
ResNeXt-50	4.2	22.1	5.9	21.1	5.5
VGG-16	15.5	27.0	8.8	25.2	7.7
Inception-ResNet-v2	11.8	20.4	5.2	19.8	4.8

SE adds less than 1% to GFLOPs

Hu et al. Squeeze-and-Excitation Networks. CVPR 2018

Depthwise separable convolutions

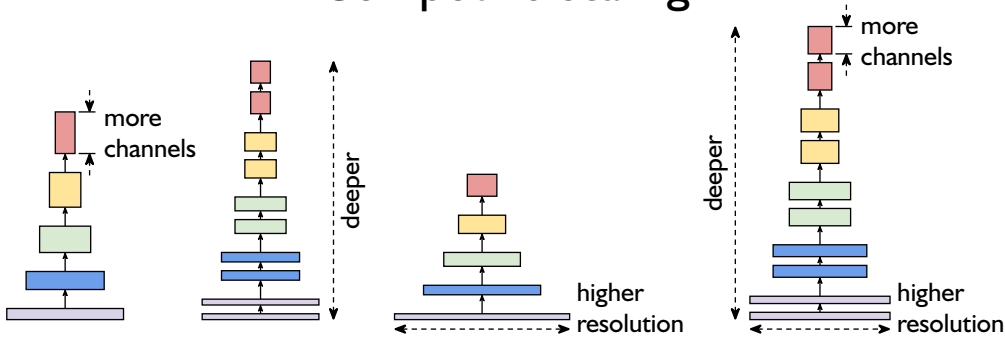




Standard: N filters $3\times3\times M$ filter in space and mix M channels at once
Depthwise: $3\times3\times1$ per channel. Pointwise: N filters $1\times1\times M$

Howard et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. 2017

Compound scaling



Scale by one coefficient ϕ:

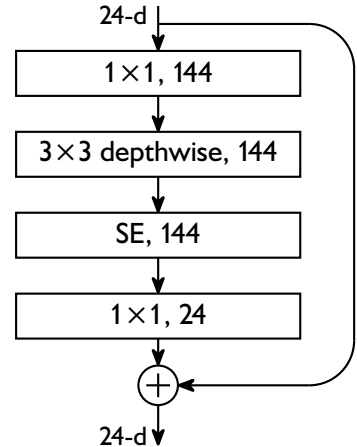
$d = \alpha^\phi$, $w = \beta^\phi$, $r = \gamma^\phi$

$\alpha\beta^2\gamma^2 \approx 2$, FLOPs grow as 2^ϕ

Tan, Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. ICML 2019

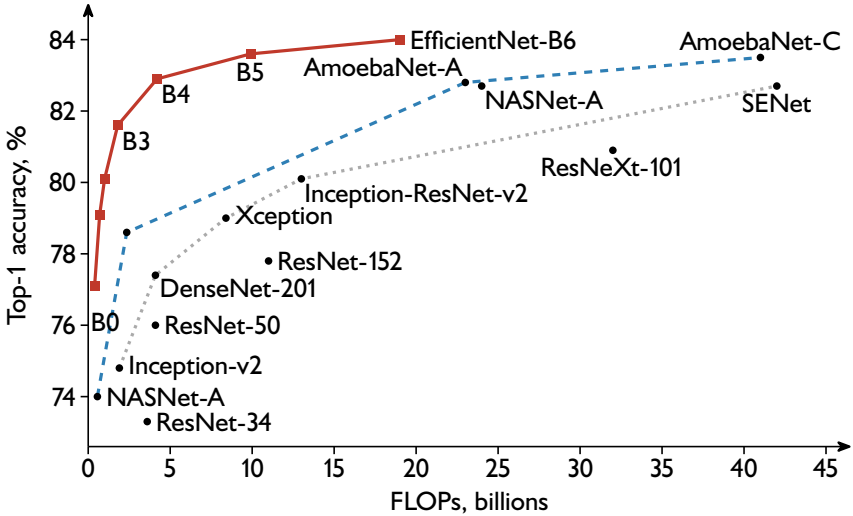
EfficientNet-B0

Operator	Resolution	Channels	Layers
Conv 3×3	224 × 224	32	1
MBConv1, 3×3	112 × 112	16	1
MBConv6, 3×3	112 × 112	24	2
MBConv6, 5×5	56 × 56	40	2
MBConv6, 3×3	28 × 28	80	3
MBConv6, 5×5	14 × 14	112	3
MBConv6, 5×5	14 × 14	192	4
MBConv6, 3×3	7 × 7	320	1
Conv 1×1, pool, FC	7 × 7	1280	1



Sandler et al. MobileNetV2: Inverted Residuals and Linear Bottlenecks. CVPR 2018

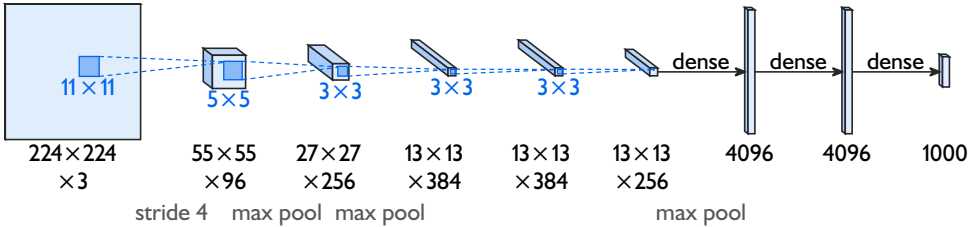
EfficientNet results



Outline

1. AlexNet, VGG, Inception
2. ResNet and its improvements
3. Understanding and reusing CNN features
4. How good is ImageNet?

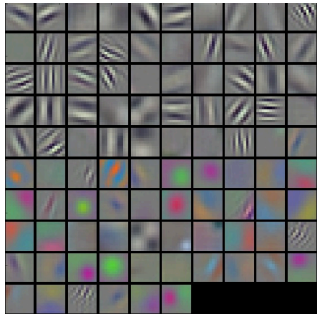
How can we analyze a neural network?



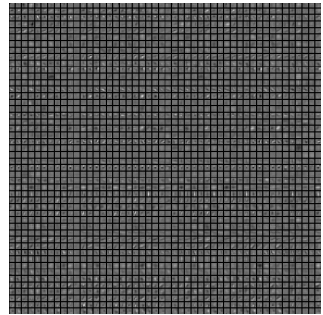
The network maps an image to 1000 class scores. What does each intermediate feature look for? Ways to find out:

- trained weights
- patches a neuron likes most
- activations mapped to pixels
- pixels behind a class score
- images made for a neuron
- features projected to 2D

Visualizing weights



conv1: edges and color blobs

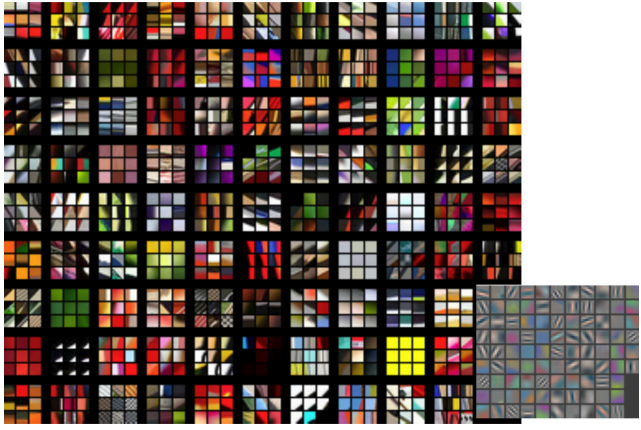


conv2: look like noise

Zeiler, Fergus. Visualizing and Understanding Convolutional Networks. ECCV 2014

Top patches

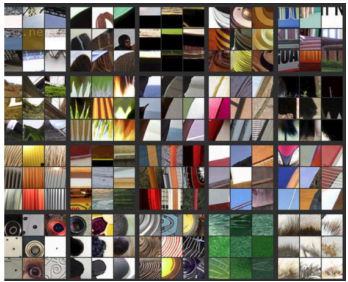
Visualize what part of image influences most on a conv neuron:



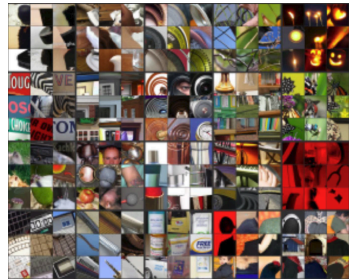
Zeiler, Fergus. Visualizing and Understanding Convolutional Networks. ECCV 2014

Top image patches: layers 2, 3 and 5

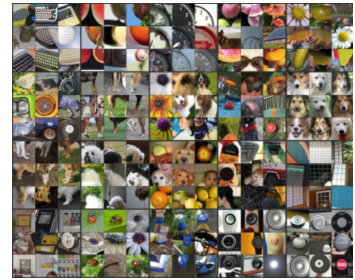
Deeper layers are more class-specific and cover larger areas of the image:



Layer 2: corners, edge and color conjunctions



Layer 3: mesh textures, text, wheels, flames

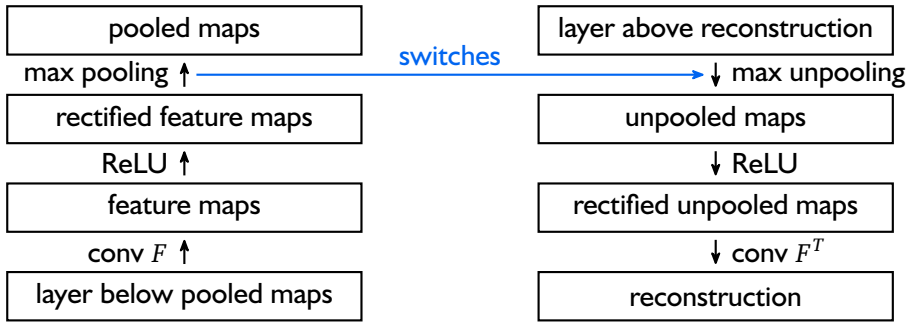


Layer 5: entire objects, large pose variation

Zeiler, Fergus. Visualizing and Understanding Convolutional Networks. ECCV 2014

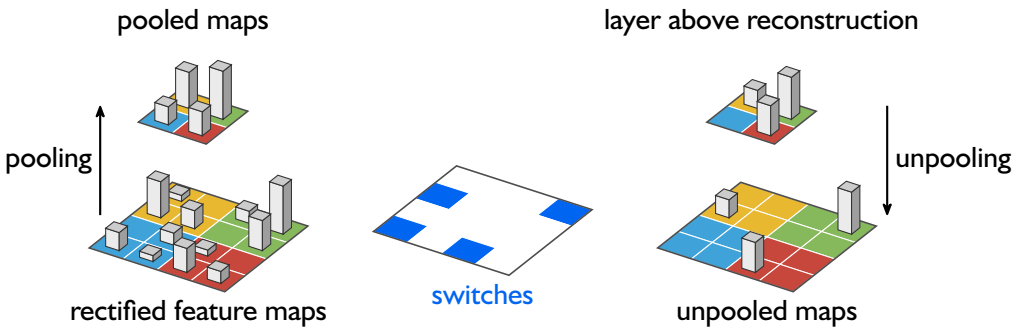
Projecting activations back to pixels

Deconvnet: zero all activations but one, undo each layer down to pixels



Zeiler, Fergus. Visualizing and Understanding Convolutional Networks. ECCV 2014

Unpooling with switches



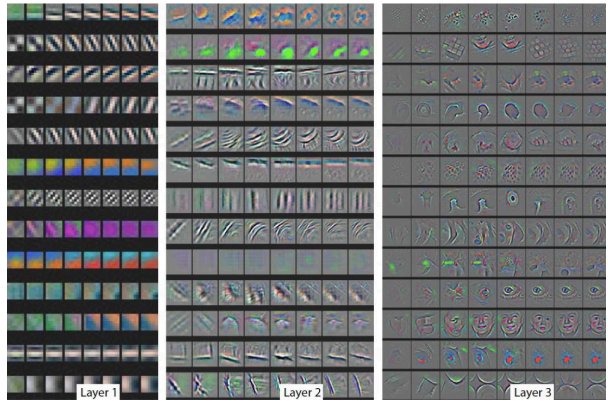
Zeiler, Fergus. Visualizing and Understanding Convolutional Networks. ECCV 2014

Features during training

Deconvnet projection of each feature's strongest activation

Rows: features of layers 1–3. Columns: epochs 1, 2, 5, 10, 20, 30, 40, 64

Lower layers settle in a few epochs, upper ones after 40–50



Zeiler, Fergus. Visualizing and Understanding Convolutional Networks. ECCV 2014

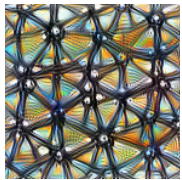
Feature visualization by optimization

Find $x^* = \arg\max_x h(x)$ starting from noise for a fixed net. $h(x)$ may be:



Neuron

a_{ijc}



Channel

$\sum_{i,j} a_{ijc}$



Layer (DeepDream)

$\sum_{i,j,c} a_{ijc}^2$



Class logit

z_k



Class probability

p_k

Olah, Mordvintsev, Schubert. Feature Visualization. Distill 2017, distill.pub/2017/feature-visualization

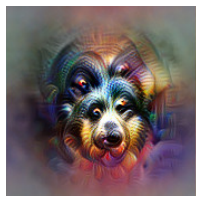
Dataset examples vs optimization





Stripes





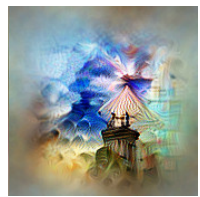
Faces, snouts





Fluffiness





Buildings, sky

Visualizing feature space with t-SNE

Take fc7 features of images: 4096 outputs of the second dense layer of AlexNet

Embed them in 2D with t-SNE: neighbours stay close, distances are not preserved

Draw each image at its point: semantically similar images end up next to each other



van der Maaten, Hinton. Visualizing Data using t-SNE. JMLR 2008

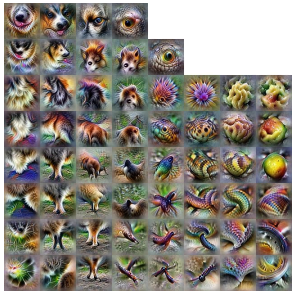
Visualizing feature space with t-SNE



Karpathy. t-SNE visualization of CNN codes. 2014, cs.stanford.edu/people/karpathy/cnnembed

Activation atlas

Activation vectors of 1M images at one layer, laid out with UMAP. A cell shows the feature visualization of its mean vector



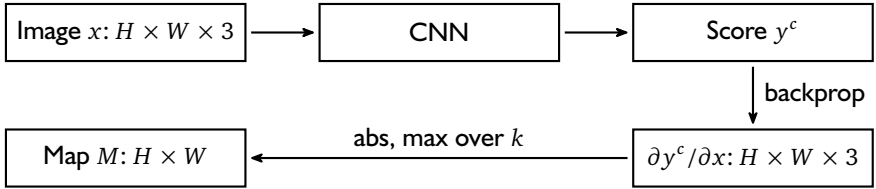
Animals: heads, fur, legs

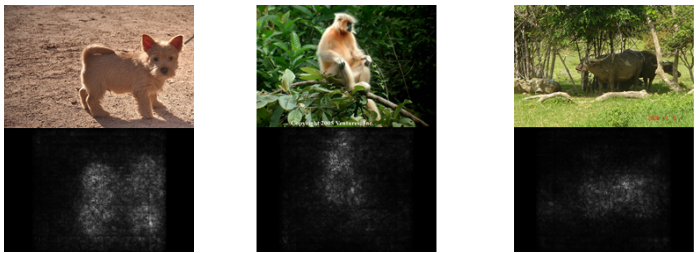


Man-made objects and buildings

Carter et al. Activation Atlas. Distill 2019, distill.pub/2019/activation-atlas

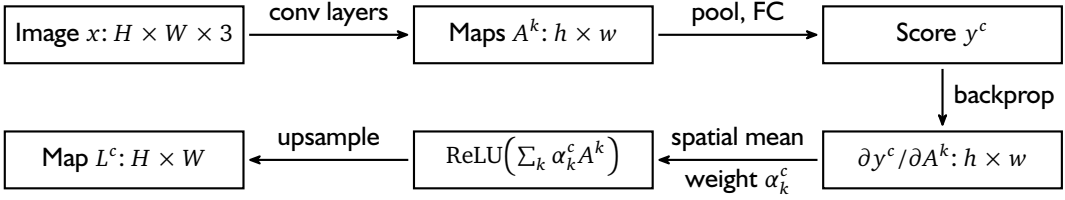
Saliency map





Simonyan, Vedaldi, Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. ICLR Workshop 2014

Grad-CAM

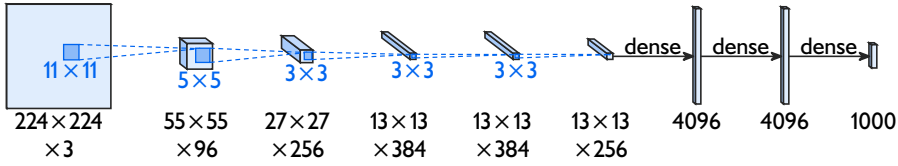




Selvaraju et al. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. ICCV 2017

Reusing features from classification networks

DeCAFn: layer n activations, linear classifier on top



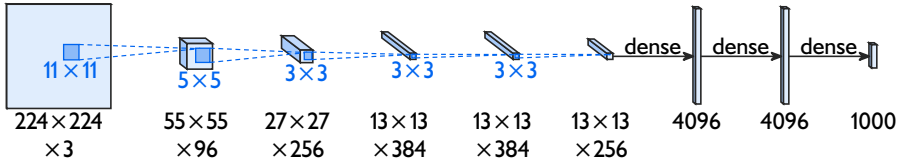
	LogReg	LogReg + dropout	SVM	SVM + dropout
DeCAF$_5$	63.3 ± 6.6	–	77.1 ± 1.1	–
DeCAF$_6$	84.3 ± 1.6	86.1 ± 0.8	84.8 ± 1.2	86.9 ± 0.7
DeCAF$_7$	84.9 ± 0.6	85.7 ± 0.6	83.2 ± 1.2	85.5 ± 0.9

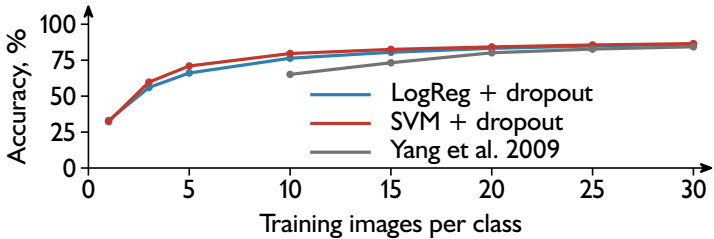
Without ImageNet: Yang et al. 2009 84.3, Jarrett et al. 2009 65.5

Donahue et al. DeCAF: A deep convolutional activation feature for generic visual recognition. ICML 2014

Reusing features from classification networks

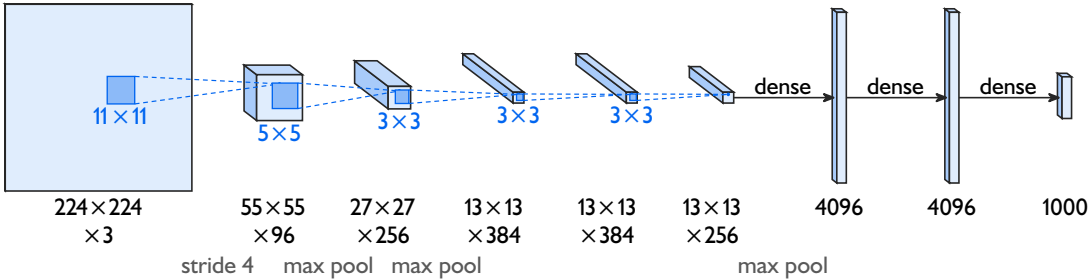
DeCAFn: layer n activations, linear classifier on top





Donahue et al. DeCAF: A deep convolutional activation feature for generic visual recognition. ICML 2014

Finetuning a neural network



Replace the last classifier layer and train the whole network with a smaller learning rate. A small training dataset is enough

Backbone: an architecture pretrained on a large dataset and reused for new tasks

Outline

1. AlexNet, VGG, Inception
2. ResNet and its improvements
3. Understanding and reusing CNN features
4. How good is ImageNet?

Where label errors come from

Ambiguous class	Overlapping classes	Validation differs	Duplicates
			
kitten as “tiger cat”	label “car wheel”	training “canoe”	training “race car”
			
tiger as “tiger cat”	label “sports car”	validation “canoe”	validation “sports car”

Kisel et al. Flaws of ImageNet, Computer Vision’s Favorite Dataset. ICLR Blogposts 2025

Relabelling ImageNet

Several objects, one label



hammer
→ screwdriver, hammer, power drill, carpenter's kit



ant
→ ant, ladybug

One class judged alone



quill
→ feather boa



passenger car
→ school bus

Classes that look the same



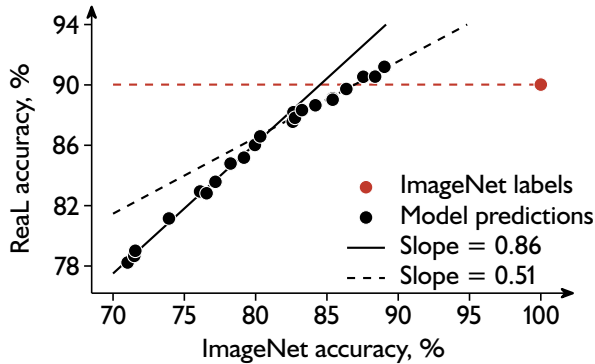
sunglass
→ sunglass, sunglasses



laptop
→ notebook, laptop

Beyer et al. Are we done with ImageNet? 2020

ReaL accuracy



Beyer et al. Are we done with ImageNet? 2020

Training on noisy labels

Training labels have the same flaws. Two fixes:

- Sigmoid loss: a yes or no for each class, so an image can have several
- Clean labels: drop the 10% of images where strong model disagrees

	Model	ImageNet accuracy, %			ReaL accuracy, %		
		90 epochs	270 epochs	900 epochs	90 epochs	270 epochs	900 epochs
ResNet-50	Baseline	76.0	76.9 (+0.9)	75.9 (−0.1)	82.5	82.9 (+0.4)	81.6 (−0.9)
	+ Sigmoid	76.3 (+0.3)	77.8 (+1.8)	76.9 (+0.9)	83.0 (+0.5)	83.9 (+1.4)	82.7 (+0.2)
	+ Clean	76.4 (+0.4)	77.8 (+1.8)	77.4 (+1.4)	82.8 (+0.3)	83.7 (+1.2)	83.3 (+0.8)
	+ Both	76.6 (+0.6)	78.2 (+2.2)	**78.5** (+2.5)	83.1 (+0.6)	**84.3** (+1.8)	84.1 (+1.6)
ResNet-152	Baseline	78.0	78.3 (+0.3)	77.1 (−0.9)	84.1	83.8 (−0.3)	82.3 (−1.8)
	+ Sigmoid	78.5 (+0.5)	78.7 (+0.7)	77.4 (−0.6)	84.6 (+0.5)	84.3 (+0.2)	82.7 (−1.4)
	+ Clean	78.6 (+0.6)	79.6 (+1.6)	79.0 (+1.0)	84.4 (+0.3)	85.0 (+0.9)	84.4 (+0.3)
	+ Both	78.7 (+0.7)	**79.8** (+1.8)	79.3 (+1.3)	84.6 (+0.5)	**85.2** (+1.1)	84.5 (+0.4)

Beyer et al. Are we done with ImageNet? 2020

Types of model errors

not an error | minor | explainable



Missing label
ox
→ barn



Fine-grained
French horn
→ cornet



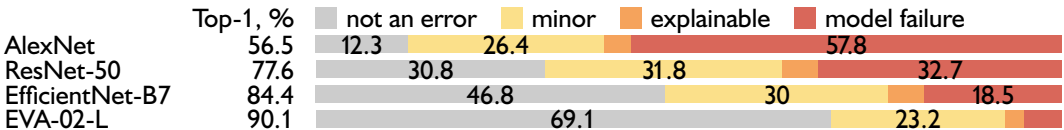
Out of vocabulary
coral reef
→ rock beauty



Non-prototypical
whistle
→ perfume

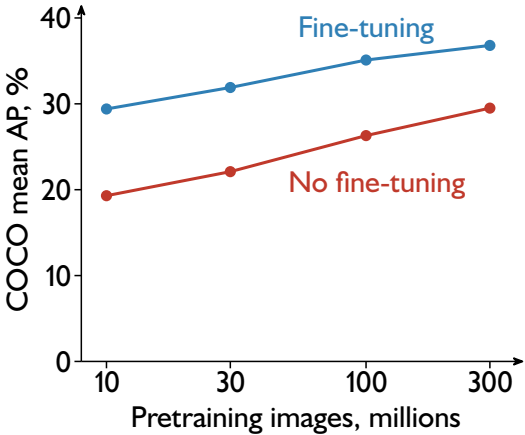


Spurious correlation
ski mask
→ ski



Peychev et al. Automated Classification of Model Errors on ImageNet. NeurIPS 2023

Using larger datasets



- JFT-300M dataset (Google)
- 1B tags, 18291 classes
- 375M tags after filtering, ~20% errors
- Training ResNet-101 on 50 K80 GPUs for a month

Sun et al. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. ICCV 2017

Conclusion

We reviewed following topics:

- using backbones as universal feature extractors
- building deep networks using basic blocks from 3×3 convolutions
- using multiple paths for processing tensors
- skip connections for training very deep networks
- reweighting channels with squeeze-and-excitation
- factorizing convolutions
- ImageNet quality and larger datasets