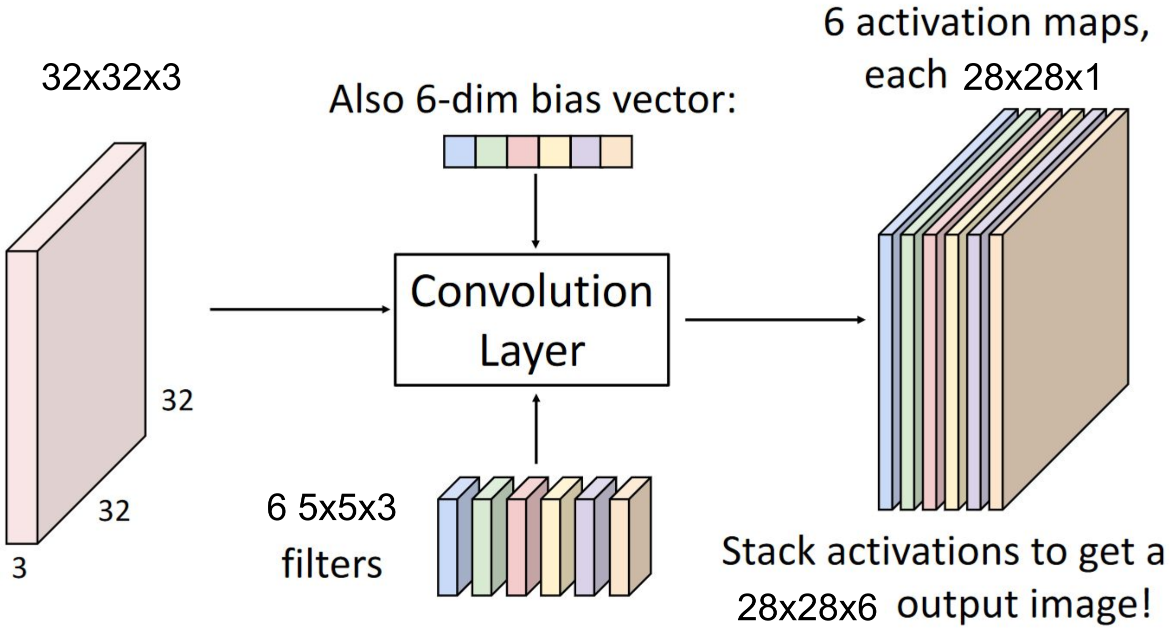
CSE428: Image Processing

Lecture 13

Convolutional Neural Networks: Part 2

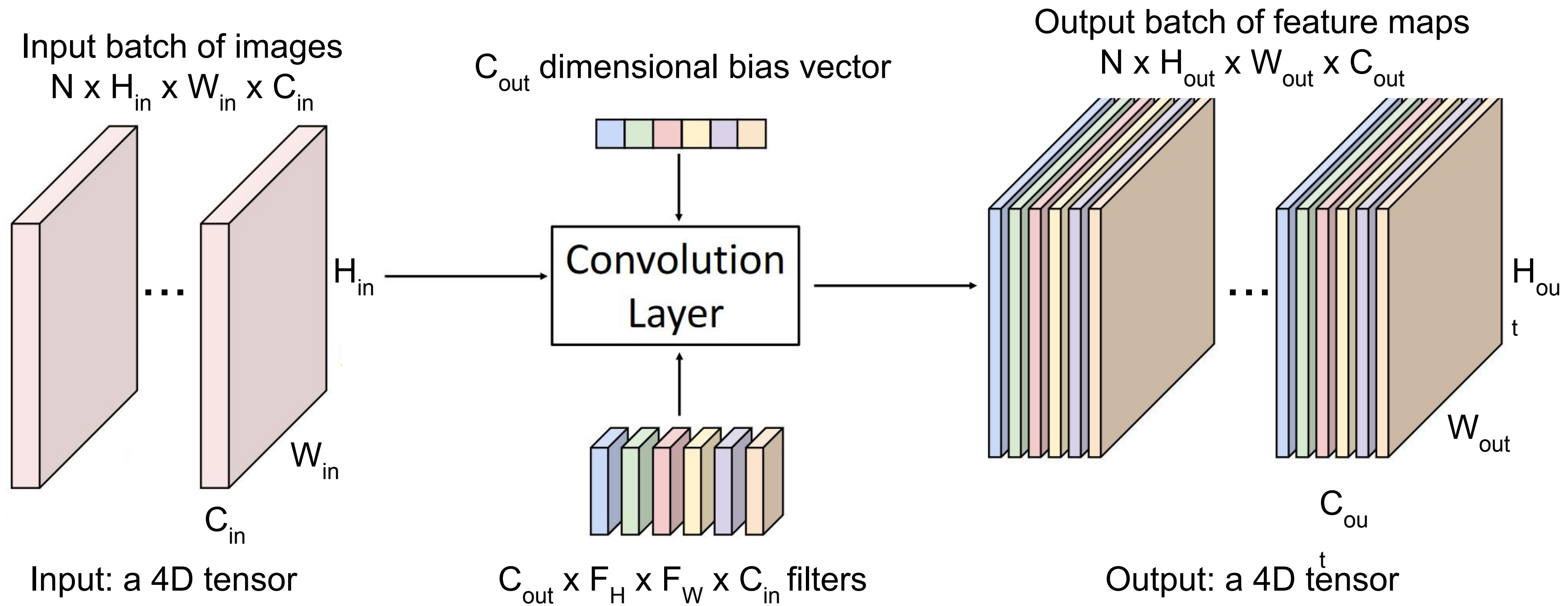
Convolution Layer

Convolution layer input: a *single* image



Batch Convolution

Convolution layer input: a *batch* of images



Batched Training

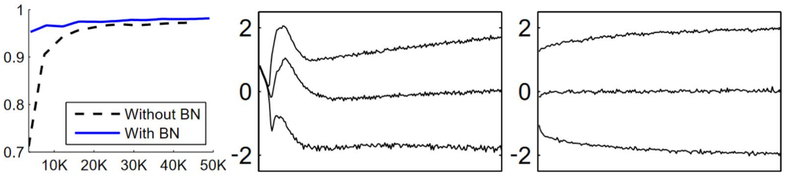
```
fit(
    x=None, y=None, batch_size=None, epochs=1, verbose='auto',
    callbacks=None, validation_split=0.0, validation_data=None, shuffle=True,
    class_weight=None, sample_weight=None, initial_epoch=0, steps_per_epoch=None,
    validation_steps=None, validation_batch_size=None, validation_freq=1,
    max_queue_size=10, workers=1, use_multiprocessing=False
)
```

Number of training samples per batch, usually in powers of 2 for hardware efficiency: 2, 4, 8, 16, 32, 64, ..

Batch Normalization

Idea: “Normalize” the output batch of a layer so they have zero mean and unit variance

Batch Normalization makes the distribution more stable and reduces the internal covariate shift



(a) (b) Without BN (c) With BN

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

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

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

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

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

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

https://arxiv.org/abs/1502.03167

Batch Normalization

Problem: depends on the batch or the distribution statistics (mean, variance) which is neither suitable nor obtainable at test time

Result: different behavior at train time and test time

> **Train**: use the batch mean and variance
>
> **Test**: use the running average of mean and variance seen at training phase

https://arxiv.org/abs/1502.03167

Batch Normalization

During training: for each channel being normalized, the layer returns gamma * (batch - mean(batch)) / sqrt(var(batch) + epsilon) + beta, where:

- epsilon is small constant
- gamma is a learned scaling factor (initialized as 1)
- beta is a learned offset factor (initialized as 0)

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

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

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

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

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

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

https://arxiv.org/abs/1502.03167

Batch Normalization

During inference: for each channel, the layer returns gamma * (batch - self.moving_mean) / sqrt(self.moving_var + epsilon) + beta.

self.moving_mean and self.moving_var are non-trainable variables that are updated each time the layer in called in training mode

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

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

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

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

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

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

https://arxiv.org/abs/1502.03167

Batch Normalization Layer

- Output has the same shape as the input
- Only the distribution of the input changes
- Usually stacked after convolution layer/fully connected layer, before non linearity
- Learnable params: γ, β

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

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

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

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

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

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

https://arxiv.org/abs/1502.03167

tf.keras.layers.BatchNormalization

```
tf.keras.layers.BatchNormalization(
    axis=-1, momentum=0.99, epsilon=0.001, center=True, scale=True,
    beta_initializer='zeros', gamma_initializer='ones',
    moving_mean_initializer='zeros',
    moving_variance_initializer='ones', beta_regularizer=None,
    gamma_regularizer=None, beta_constraint=None, gamma_constraint=None, **kwargs
)
```

https://www.tensorflow.org/api_docs/python/tf/keras/layers/BatchNormalization

ImageNet Large Scale Visual Recognition Challenge

The ImageNet Large Scale Visual Recognition Challenge or **ILSVRC** for short is an annual competition helped between 2010 and 2017 in which challenge tasks use subsets of the ImageNet *dataset*.

- **ImageNet dataset**: 1,000 classes, 1,281,167 training images, 50,000 validation images and 100,000 test images
- **ILSVRC challenge**: Image classification, Object localization (classification+bounding box)

https://www.image-net.org/

ImageNet Large Scale Visual Recognition Challenge

In this task, given an image an algorithm will produce 5 class labels $c_i, i = 1, \dots 5$ in decreasing order of confidence and 5 bounding boxes $b_i, i = 1, \dots 5$, one for each class label. The quality of a labeling will be evaluated based on the label that best matches the ground truth label for the image. The idea is to allow an algorithm to identify multiple objects in an image and not be penalized if one of the objects identified was in fact present, but not included in the ground truth.

The ground truth labels for the image are $C_k, k = 1, \dots n$ with n class labels. For each ground truth class label C_k, the ground truth bounding boxes are $B_{km}, m = 1 \dots M_k$, where M_k is the number of instances of the k^{th} object in the current image.

Let $d(c_i, C_k) = 0$ if $c_i = C_k$ and 1 otherwise. Let $f(b_i, B_k) = 0$ if b_i and B_k have more than 50% overlap, and 1 otherwise. The error of the algorithm on an individual image will be computed using two metrics:

- Classification-only:

$$e = \frac{1}{n} \cdot \sum_k \min_i d(c_i, C_k)$$

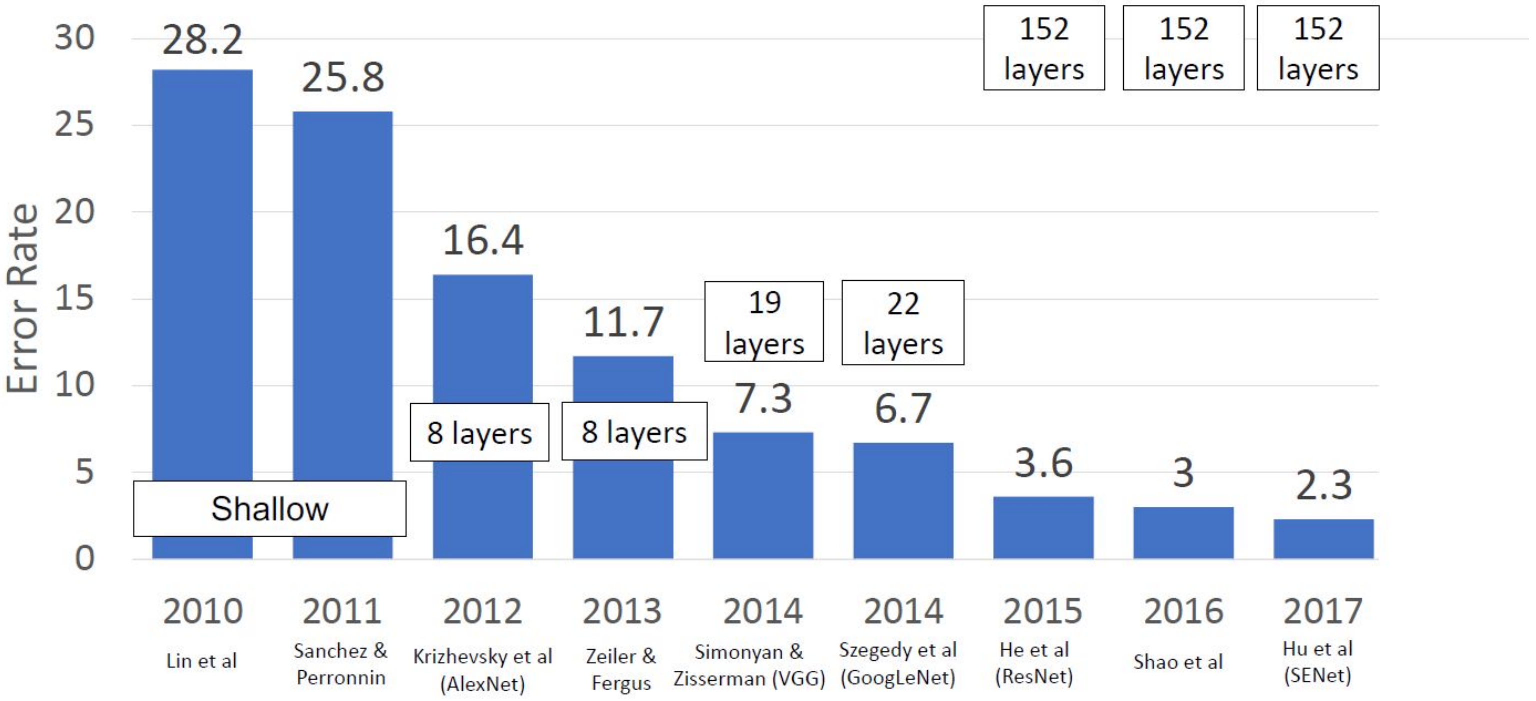
- Classification-with-localization:

$$e = \frac{1}{n} \cdot \sum_k min_i min_m max\{d(c_i, C_k), f(b_i, B_{km})\}$$

The classification-only and classification-with-localization errors of the algorithm is the average corresponding error across all test images.

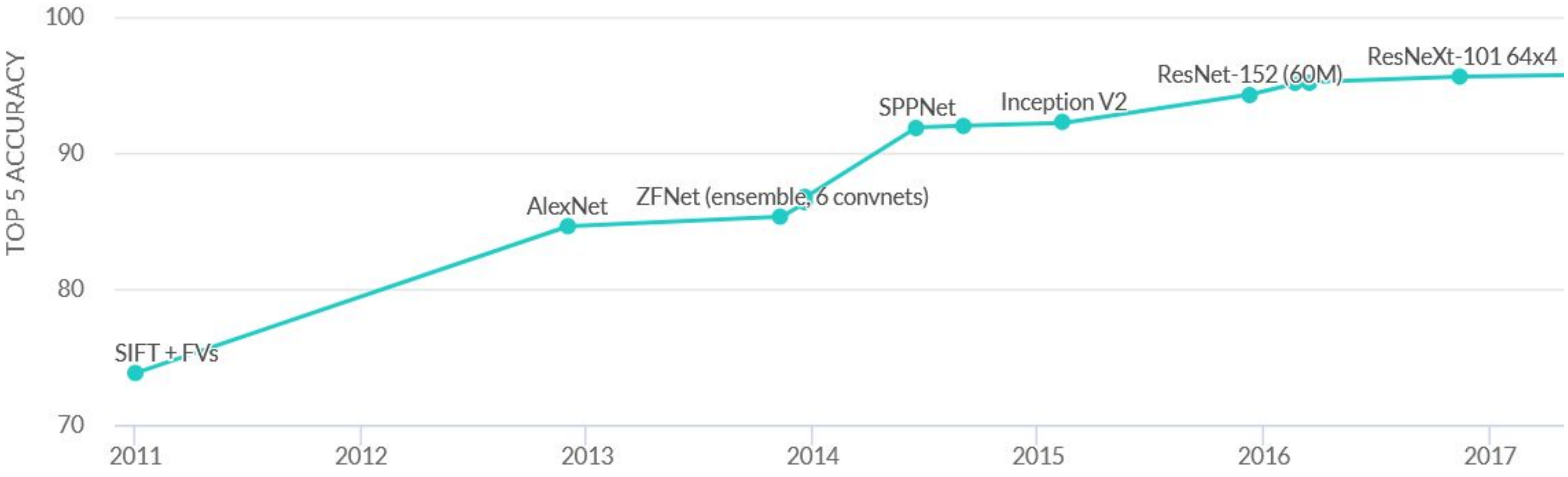
https://www.image-net.org/

ImageNet Classification Challenge Results (2010-2017)



https://web.eecs.umich.edu/~justincj/slides/eecs498/FA2020/598_FA2020_lecture08.pdf

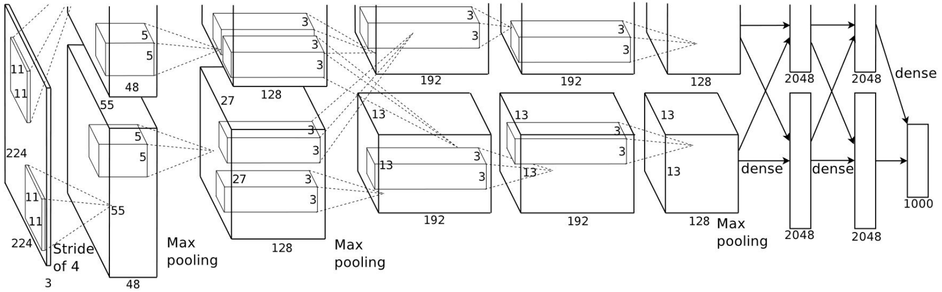
ImageNet Classification Challenge Results (2010-2017)



https://paperswithcode.com/sota/image-classification-on-imagenet

AlexNet

Architecture



- Input 224x224x3 cropped images
- 5 convolution (conv) layer
- 3 max-pooling layer
- 3 fully connected (FC) layer
- ReLU nonlinearity
- Total number of layers: **8**
- ILSVRC **2012** Top-5 accuracy: **84.6%**

[PDF] **Imagenet classification** with **deep convolutional neural networks**

A Krizhevsky, I Sutskever… - Advances in **neural** …, 2012 - proceedings.neurips.cc

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is …

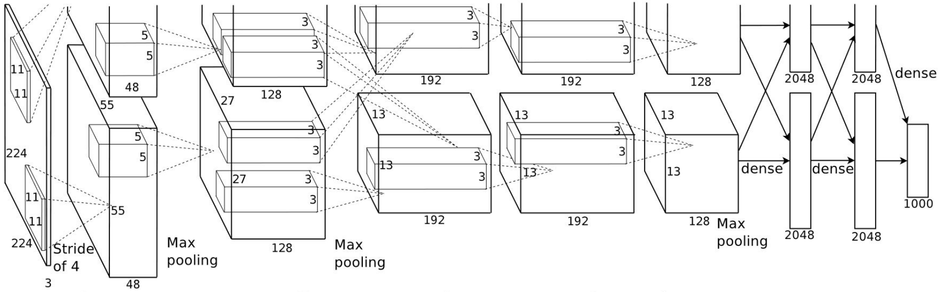
Cited by 85837 Related articles All 121 versions

25 Aug, 2021

http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

AlexNet

Architecture



Layer	Input Volume			Hyperparameters				Output Volume			Memory (KB)	Learnable parameters (K)	FLOPs (M)
	Hin	Win	Cin	Filters	Size	Stride	Pad	Hout	Wout	Cout			
conv1	227	227	3	64	11	4	2	56	56	64	784	23	73
pool1													
conv2													
pool2													
conv3													
conv4													
conv5													
pool3													
flatten													
fc1													
fc2													
fc3 (output)													
Total													

Output size
Hout (= Wout)
=(Hin-F+2*P)/S +1
=(227-11+2*2)/4 +1

Memory (Byte)
=output volume storage
=(Hout*Wout*Cout*4Bytes)
=(56*56*64*4)/1024 KByte

Conv layer parameters
=Cout*F*F*Cin + Cout
=(64*11*11*3)/1000 K

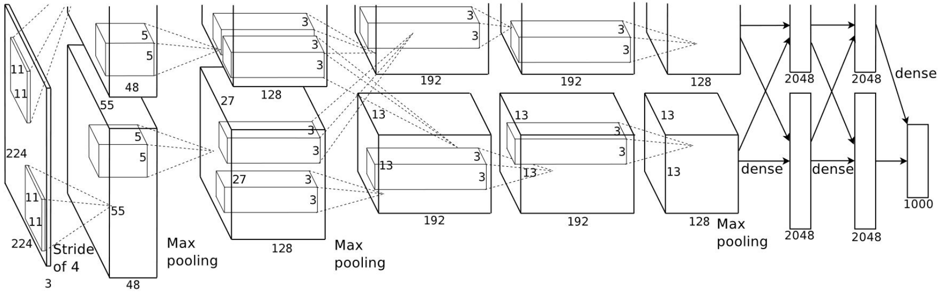
FC layer parameters
=Cout*Cin + Cin

POOL layer parameters
=0

Floating Point Operations FLOPs (Mul+Add)
=output volume * filter volume FLOPS
=(56*56*64)*(11*11*3)/10^6 M FLOPS

https://docs.google.com/spreadsheets/d/1ZVbWQYhqk5SDhPXMj-uvqKeNp7tH00I0MJHCTXPhAk0/edit?usp=sharing

AlexNet



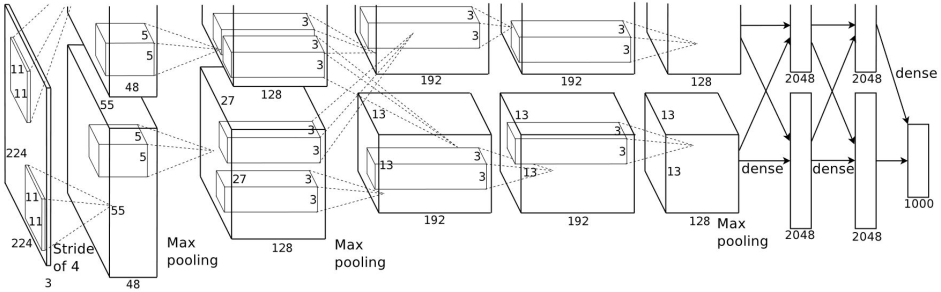
Architecture details

- Spatial size *reduces* & depth *increases*
- FC layers have far more parameters than the conv layers
- Conv layers require more FLOPs

Layer	Input Volume			Hyperparameters				Output Volume			Memory (KB)	Learnable parameters (K)	FLOPs (M)
	Hin	Win	Cin	Filters	Size	Stride	Pad	Hout	Wout	Cout			
conv1	227	227	3	64	11	4	2	56	56	64	784	23	73
pool1	56	56	64		3	2	0	27	27	64	182	0	0
conv2	27	27	64	192	5	1	2	27	27	192	547	307	224
pool2	27	27	192		3	2	0	13	13	192	127	0	0
conv3	13	13	192	384	3	1	1	13	13	384	254	664	112
conv4	13	13	384	256	3	1	1	13	13	256	169	885	150
conv5	13	13	256	256	3	1	1	13	13	256	169	590	100
pool3	13	13	256		3	2	0	6	6	256	36	0	0
flatten	6	6	256							256	36	0	0
fc1			9216	4096						4096		37758	38
fc2			4096	4096						4096		16781	17
fc3 (output)			4096	1000						1000		4100	4
Total											2304	61108	718

https://docs.google.com/spreadsheets/d/1ZVbWQYhqk5SDhPXMj-uvqKeNp7tH00I0MJHCTXPhAk0/edit?usp=sharing

AlexNet



Impact

- First CNN based winner!
- First use of ReLU in practice
- Trained on 2 GTX 580 GPUs
  - (3GB memory/gpu)
- Dropout
- Data augmentation, L2 regularization
- Highly cited paper in the field
- Top 5 error reduction 25.8% -> 16.4%

[PDF] **Imagenet classification** with **deep convolutional neural networks**

A Krizhevsky, I Sutskever… - Advances in **neural** …, 2012 - proceedings.neurips.cc

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is …

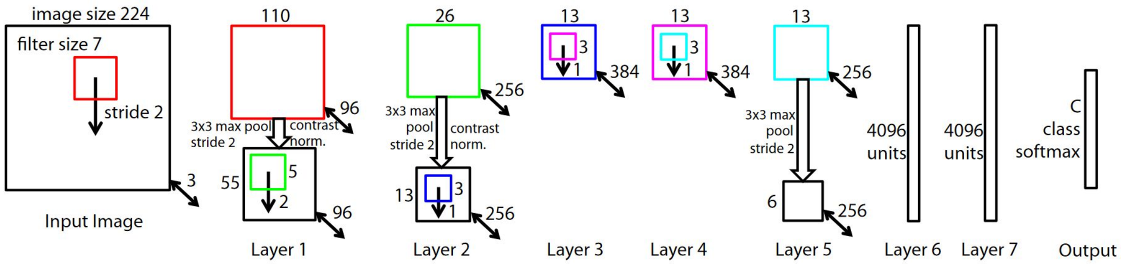
Cited by 85837 Related articles All 121 versions

25 Aug, 2021

http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

ZF Net

Architecture



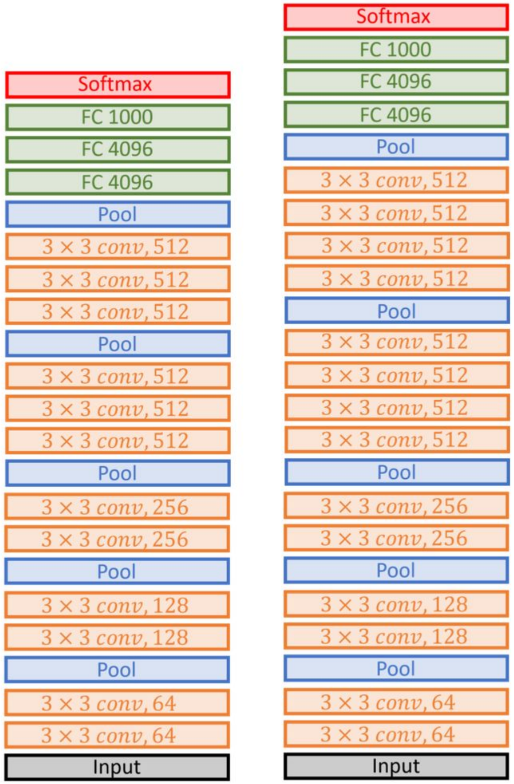
- Modified AlexNet
- Improved AlexNet by tweaking hyperparameters
- Smaller filter size (11x11 -> 7x7) in the earlier layers
- Total number of layers: **8**
- ILSVRC **2013** Top-5 accuracy: **85.3%**
- Top 5 error reduction 16.4% -> 11.7%

http://arxiv.org/abs/1311.2901

VGG

Architecture

- Regular design
  - F=3, S=1, P=1 conv layers
  - F=2, S=2 max-pooling lakers
  - Double # of filters after every pooling layer
- Can be broken into stages
  - - [conv layer] x 2/3/4 - [pooling layer] -
- Total number of layers: **16, 19** (VGG16, VGG19)
- ILSVRC **2014** Top-5 accuracy: **92.0%** (runner-up)



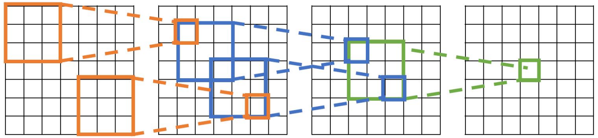
VGG16 VGG19

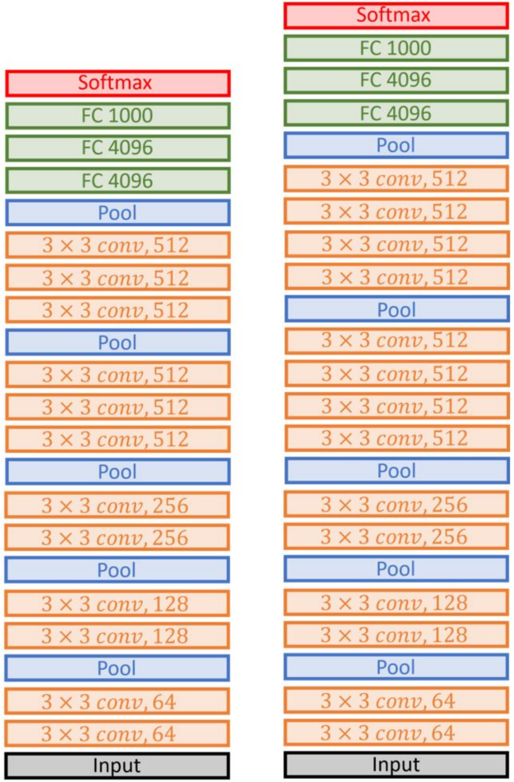
https://arxiv.org/abs/1409.1556, http://datahacker.rs/deep-learning-vgg-16-vs-vgg-19/

VGG

Architecture

- Smaller convolutions
    - The receptive field of **2 stacked 3x3 conv layers** is the same as **1 single 5x5 conv layer**, but has fewer parameters
    - For C_{in}=C_{out}=C number of channels, 2 stacked 3x3 conv layers have 2*(C*3*3*C)=**18C²** whereas 1 single 5x5 conv layer has C*5*5*C=**25C²**
- Spatial resolution is preserved after every convolution layer and downsampling only occurs at max-pooling stages



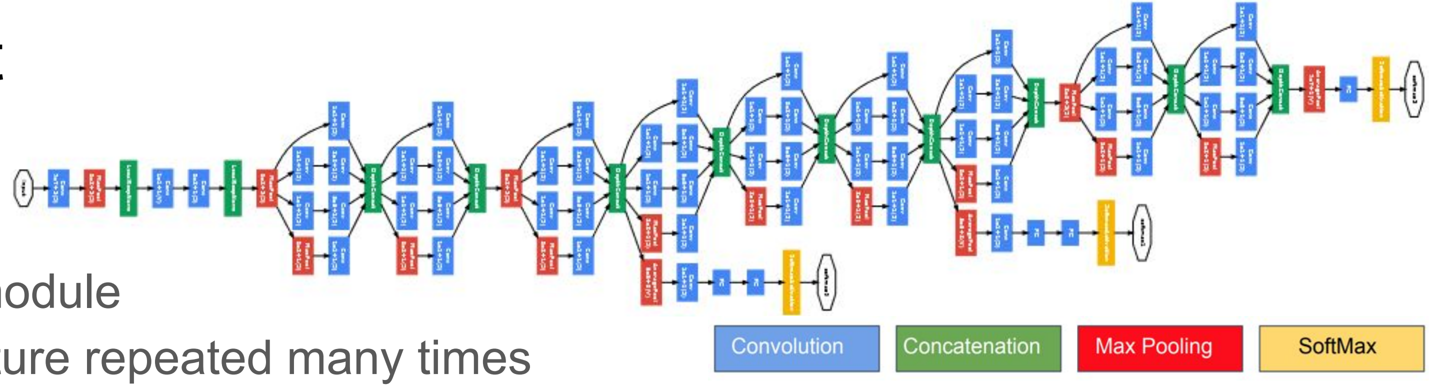


VGG16 VGG19

https://arxiv.org/abs/1409.1556, http://datahacker.rs/deep-learning-vgg-16-vs-vgg-19/

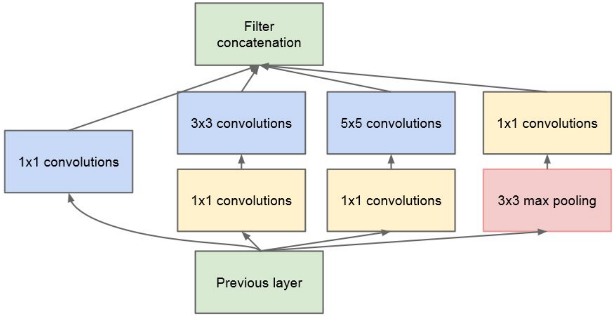
GoogLeNet

Architecture



- Inception module
- Local structure repeated many times
- Starts with a “stem” network
- Utilizes 1x1 convolution layers
- No large FC layers, more efficient
- Only 1 FC layer at the end to classify
- Total number of layers: **22** (Inception V2)
- ILSVRC **2014** Top-5 accuracy: **92.2%** (winner)

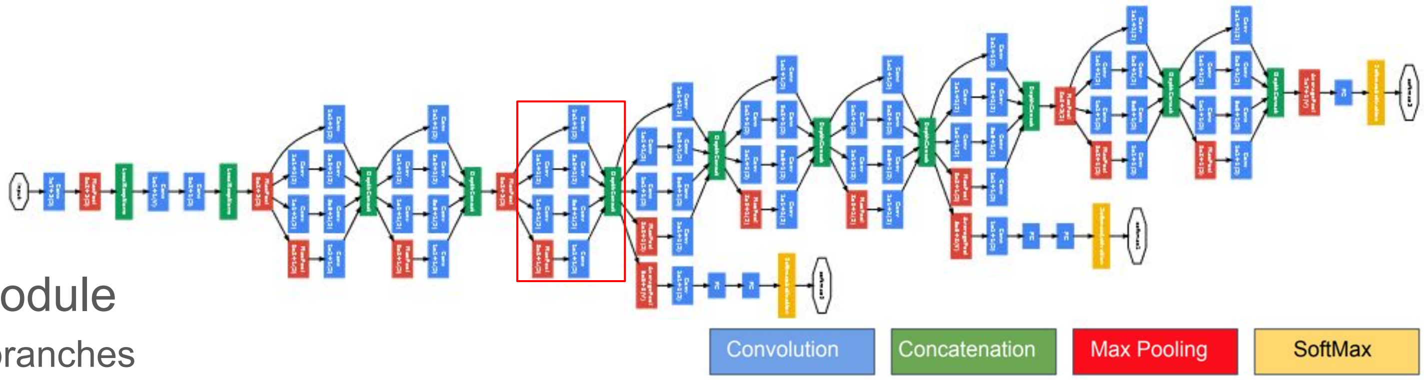
Inception module



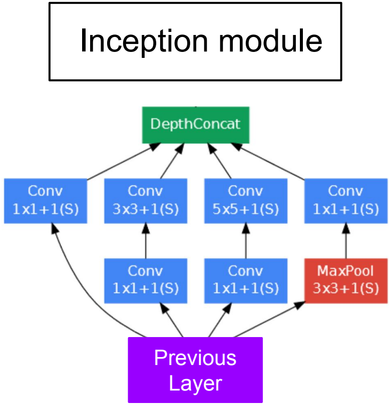
https://arxiv.org/abs/1409.4842

GoogLeNet

Architecture



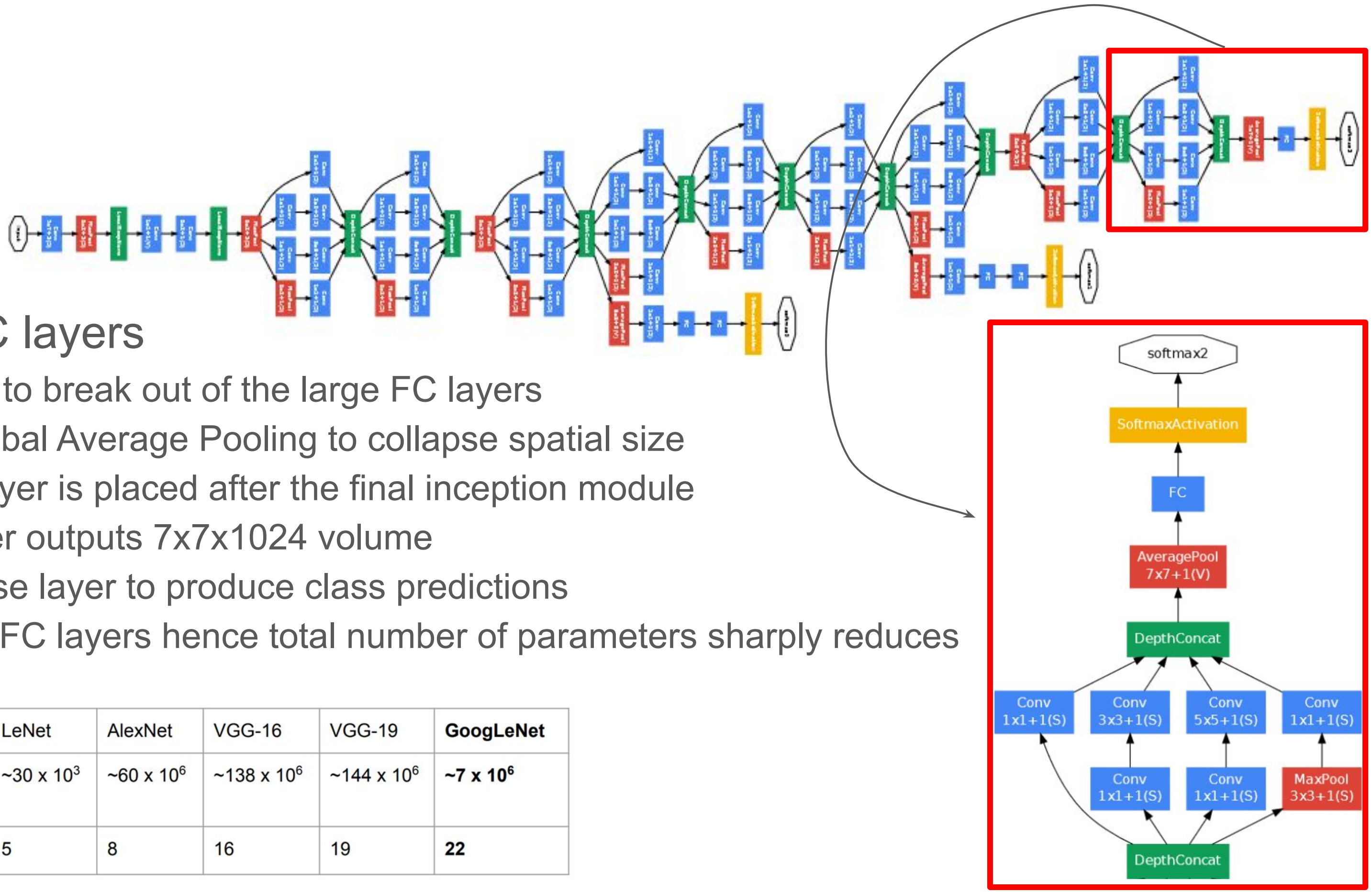
- Inception module
  - Parallel branches
  - Parallel convolutions of 1x1, 3x3, 5x5
  - 1x1 “bottleneck” layers used to reduce number of channels
  - Concatenate output of each branch depth wise
  - Repeat this structure throughout the network!



https://arxiv.org/abs/1409.4842

GoogLeNet

Architecture



- No large FC layers
  - First one to break out of the large FC layers
  - Uses Global Average Pooling to collapse spatial size
  - A GAP layer is placed after the final inception module
  - GAP layer outputs 7x7x1024 volume
  - One dense layer to produce class predictions
  - No huge FC layers hence total number of parameters sharply reduces

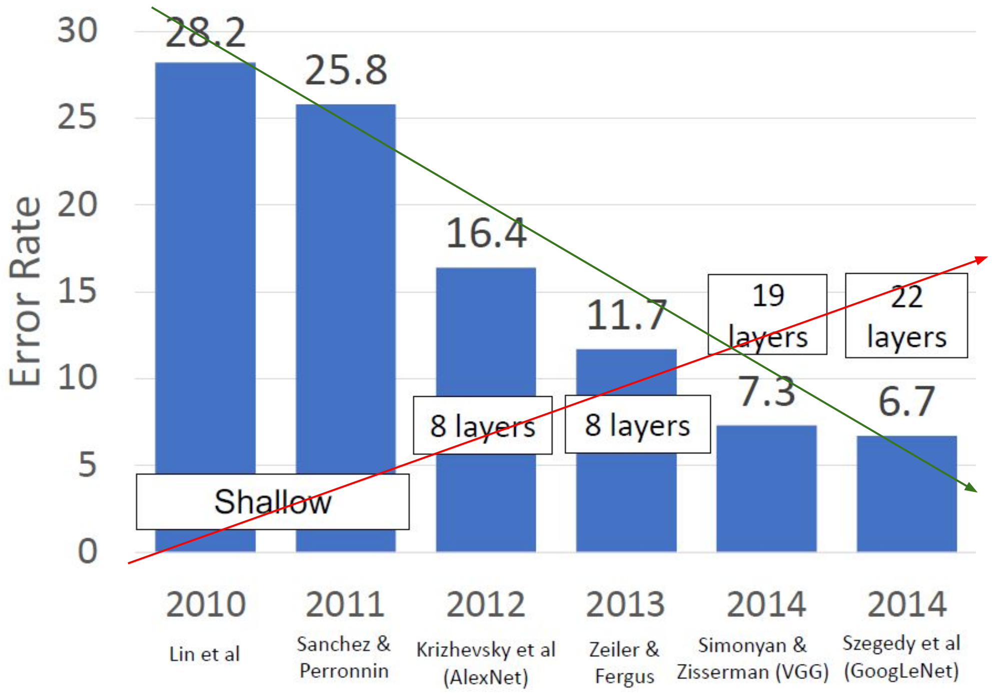
	LeNet	AlexNet	VGG-16	VGG-19	**GoogLeNet**
Parameters	~30 x 10^3	~60 x 10^6	~138 x 10^6	~144 x 10^6	**~7 x 10^6**
Layers	5	8	16	19	**22**

https://arxiv.org/abs/1409.4842

Are Deeper Networks Better?

At this point researcher started to think: deeper network ⇒ better accuracy?

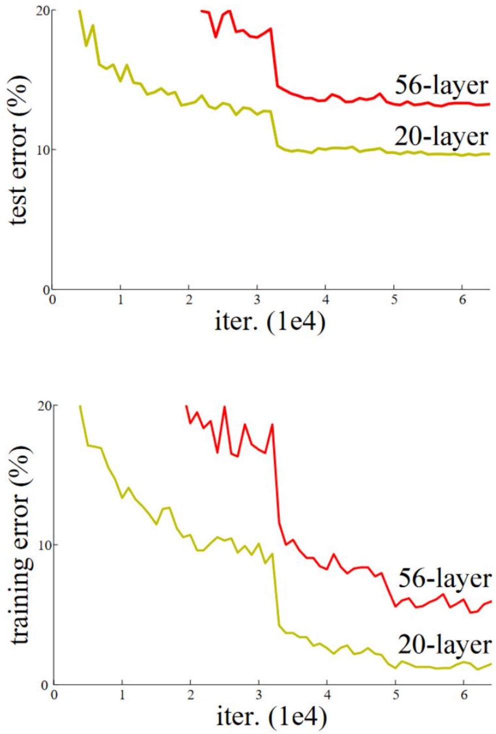
- Since there is a very obvious trend in the graph shown!



Are Deeper Networks Better?

Problems with deeper network

- **Training**: Very hard to train (optimize) because gradients get smaller and smaller as they back propagate (vanishing gradient)
- **Performance**: Worse than shallow models! Why?

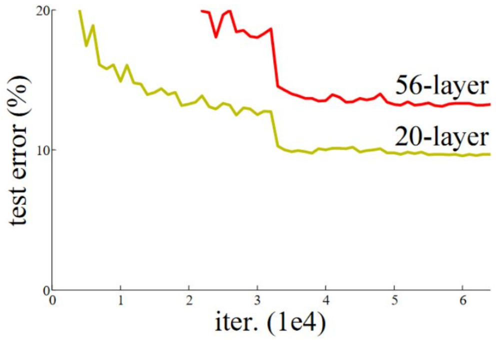


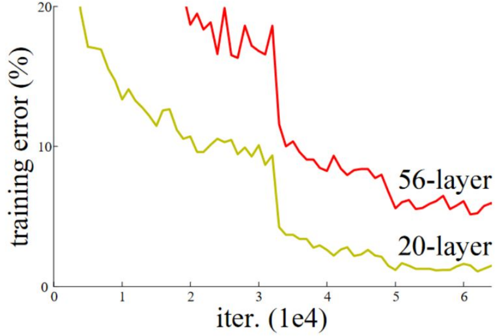
https://arxiv.org/abs/1512.03385

Residual Networks Motivation

Performance: Worse than shallow models! Why?

- Initially it people thought because of overfitting! (test)
- But the training profile shows it is actually underfitting!
- A larger model should easily be able to emulate shallower models, by adding extra layers and learning the identity function
- Maybe deeper models are suffering from not being able to learn the *identity* function
- Solution: a layer that can learn identity function easily!

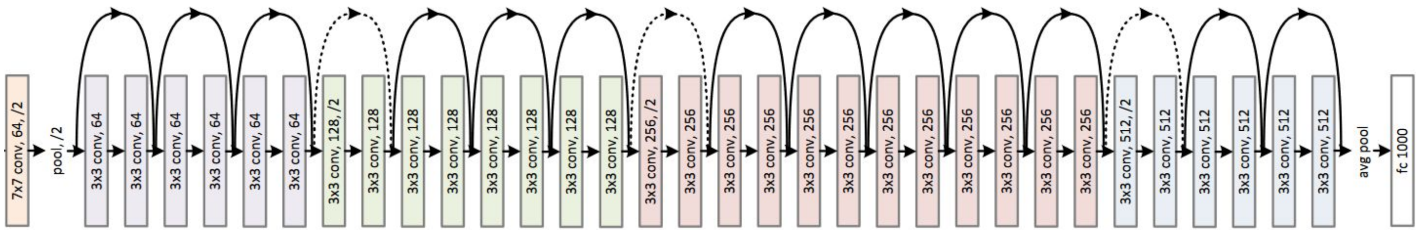




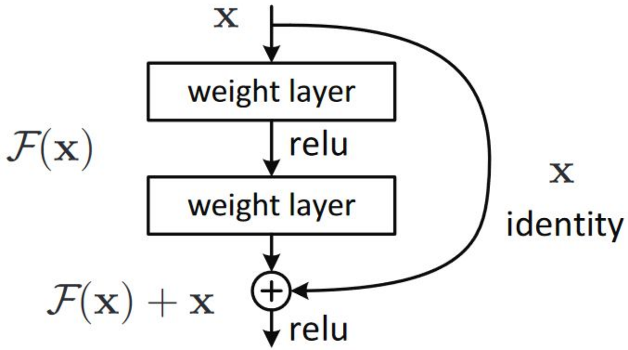
https://arxiv.org/abs/1512.03385

ResNet

Architecture

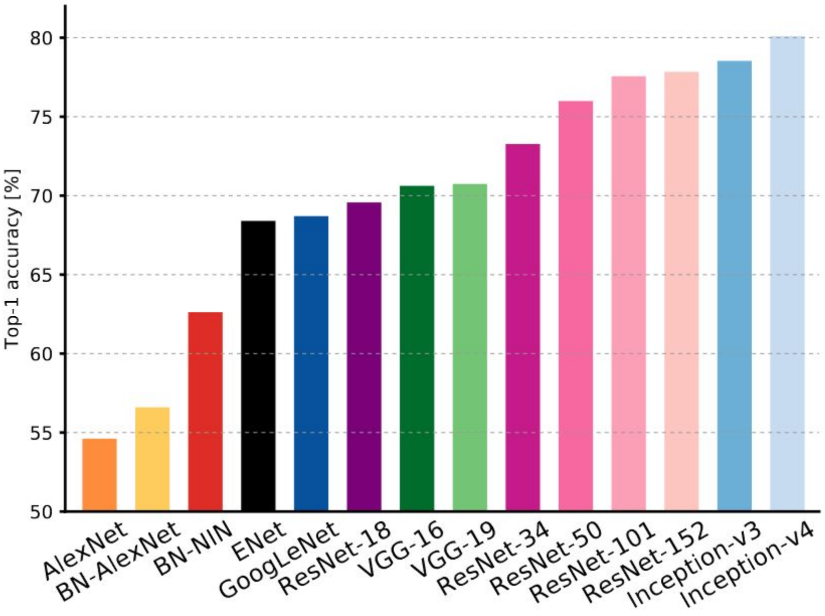


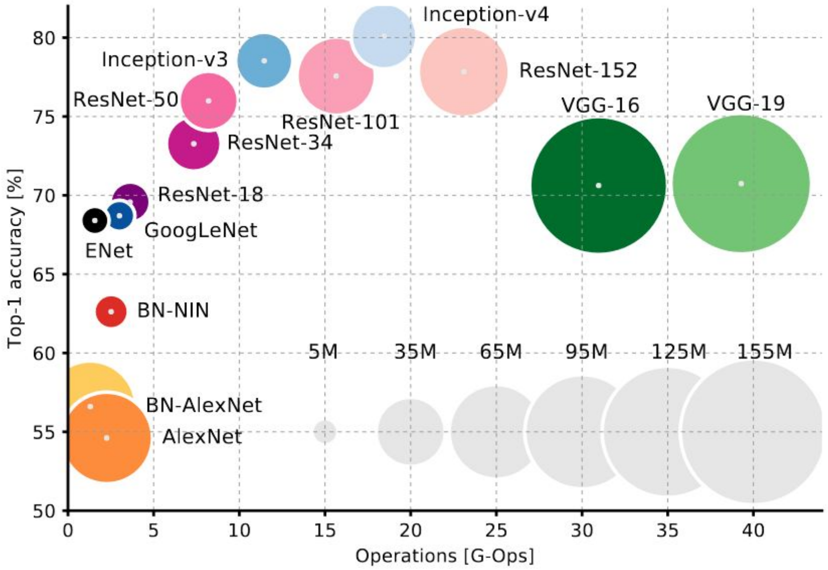
- Skip connections: residual block
- Structure is like VGG, with residual blocks
- Network is divided into stages
- No pooling, only strided downsampling
- GAP+Linear layer for final classification
- Able to train very, very deep networks
- Total number of layers: **152** (ResNet-152)
- ILSVRC **2015** Top-5 accuracy: **94.29%** (winner across 5 major competitions)



https://arxiv.org/abs/1512.03385

Model Complexity Comparison





https://arxiv.org/abs/1605.07678

Other Popular Architectures

Network in Network

ResNeXt

DenseNet

SqueezeNet

MobileNet

Summary

AlexNet and VGG started showing the power of CNNs in computer vision tasks over very large datasets

Inception and ResNet enabled to build even deeper CNNs and helped them train efficiently

ILSVRC has now moved on to Kaggle

For starters, most common problems can be solved by using any of the off-the-shelf architectures, no need to build a new one

Later we will see how to modify these networks for different tasks!

Resources

1. https://cs231n.github.io/convolutional-networks/
2. https://web.eecs.umich.edu/~justincj/teaching/eecs498/FA2020/
3. https://www.tensorflow.org/api_docs/python/tf/keras
4. Deep Learning with Python Book by François Chollet
5. https://www.deeplearningbook.org/
6. Hands-on Computer Vision with TensorFlow 2 by Eliot Andres & Benjamin Planche (Packt Pub.)

Colab Notebook

1. A simple toy ResNet model using keras functional API

   https://colab.research.google.com/drive/1oh5HYjFoi4WVnIbyme3rdCknVQZ36ob6?usp=sharing

https://www.tensorflow.org/guide/keras/functional