Object Recognition in Images and Video: CNNs

http://www.micc.unifi.it/bagdanov/obrec

Prof. Andrew D. Bagdanov

Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Firenze
andrew.bagdanov AT unifi.it

27 April 2017

Outline

Overview

Overview

- In this lesson I will first do a quick overview of the Deformable Part Models detector.
- Then, I will present four works that radically changed the face of object recognition.
- These new models are based on a specific type of Neural Network.
- Specifically, they are based on Convolutional Neural Networks (CNNs).
- I will begin the discussion of CNNs with some motivating examples that demonstrate the basic building blocks of CNNs.
- We will see how this model (sort of) arises naturally from the Bag-of-Words model.
- And we will see how it has radically changed the landscape of object recognition.

Deformable Part Models

Deformable Part Models

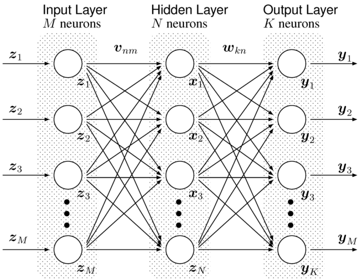
- [OTHER PRESENTATION]

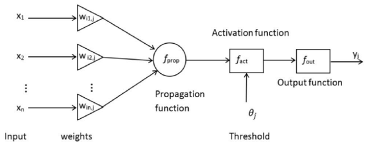
Setting the Stage

Setting the Stage

- After ten years of incremental improvements of the Bag-of-Words model, we found ourselves reaching a sort of asymptote.
- It was unclear what would be the Next Big Thing in object recognition.
- The state-of-the-art in 2012 was: Fisher Vectors + Multiple Cues + Late Fusion (summing scores).
- Meanwhile, since the 1990s there was a competing paradigm for object recognition based on Neural Networks.
- This method for object recognition had been developed in the 1990s for character recognition.
- However, it never really broke through onto the object recognition scene.
- This is due to a number of factors that are important to understand.

- Let's look at a simple Neural Network architecture known as the Multilayer Perceptron (MLP):



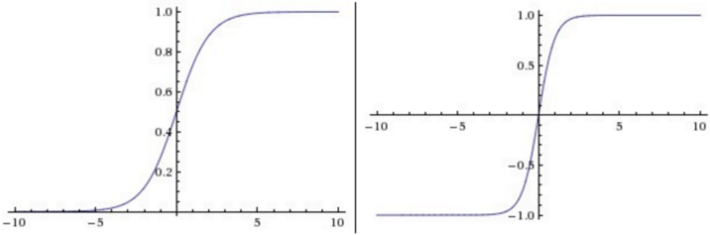


Setting the Stage

- The MLP equation (one hidden layer):

$$\hat{\mathbf{y}}(\mathbf{x}) = \sigma(\mathbf{w}_2^T \sigma(\mathbf{w}_1^T \mathbf{z} + b_1) + b_2)$$

- Except for the activation function σ, this is a linear system.
- Common activation functions (elementwise):
  - $\sigma(\mathbf{x}) = \tanh(\mathbf{x})$
  - $\sigma(\mathbf{x}) = (1 + e^{-\mathbf{x}})^{-1}$
  - $\sigma(\mathbf{x}) = \frac{\exp(\mathbf{x})}{\sum_i e^{x_i}}$



- How do you train a model?
- Decide on a loss function:

$$L(\mathbf{y}, \hat{\mathbf{y}}(\mathbf{x})) = \frac{1}{C} \sum_i y_i \log(\hat{y}_i)$$

- And perform gradient descent w.r.t. all model parameters:

$$\begin{aligned} \boldsymbol{\theta}_{n+1} &= \boldsymbol{\theta}_n - \varepsilon \nabla_{\boldsymbol{\theta}} L(\mathbf{y}, \hat{\mathbf{y}}(\mathbf{x})) \\ \boldsymbol{\theta}_{n+1} &= \boldsymbol{\theta}_n - \varepsilon \sum_{i=1}^{N} \frac{1}{N} \nabla_{\boldsymbol{\theta}} L(\mathbf{y}, \hat{\mathbf{y}}(\mathbf{x}_i)) \end{aligned}$$

- Where ε is the learning rate.
- The standard algorithm for this is known as backpropagation and it is very clever and efficient.

- Problems with this approach:
  - Model size: many, many parameters for even small-sized images. This leads to memory and efficiency problems.
  - Overfitting: many parameters (and limited training data) mean that it is easy to overfit the model to your training set.
  - Undergeneralization: overfitting means that a trained model is unlikely to generalize to new data.
  - Vanishing gradients: a known problem with backpropagation (due to application of the chain rule) leads to very small gradient values near the beginning of the network.
  - Saturating units: traditional activation functions can lead to saturated units (outputs near 1 or 0 (or -1)), which have near-zero derivatives.
- These problems (and others) led the community to largely ignore the potential of these models for decades.

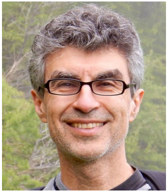
Setting the Stage

- Nonetheless, there were staunch supporters of this paradigm.
- Then, one day in 2010 (or 2011), the following conversation took place. . .

J. Malik

G. Hinton

Y. LeCunn



Y. Bengio

- Then, at the ImageNet competition workshop at ECCV 2012 (right here in Florence!):

Task 1

Team name	Filename	Error (5 guesses)	Description
SuperVision	test-preds-141-146.2009-131-137-145-146.2011-145f.	0.15315	Using extra training data from ImageNet Fall 2011 release
SuperVision	test-preds-131-137-145-135-145f.txt	0.16422	Using only supplied training data
ISI	pred_FVs_wLACs_weighted.txt	0.26172	Weighted sum of scores from each classifier with SIFT+FV, LBP+FV, GIST+FV, and CSIFT+FV, respectively.

CNNs: AlexNet

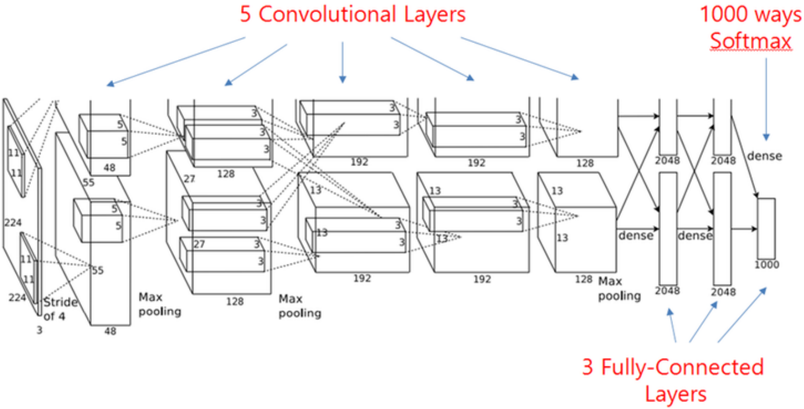
AlexNet: Introduction

- We will now take a look at the International Large Scale Visual Recognition Competition (ILSVRC) submission that changed everything:

ImageNet Classification with Deep Convolutional Neural Networks. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. In: Proceedings of NIPS, 2012.

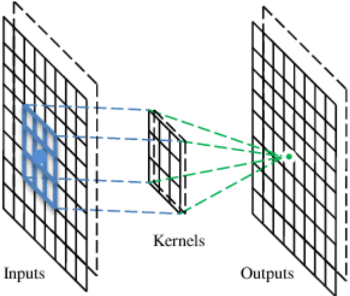
- In this paper the authors defined a convolutional network architecture that became the New Standard.
- This architecture systematically addresses most of the problems with training large network architectures.
- It is a Convolutional Neural Network (CNN) that is universally called AlexNet.
- It is also a Deep Network because it has many hidden layers.
- Hence the term Deep Learning.

- Let's look first at the overall architecture and then analyze in detail how each component addresses specific problems.
- It is also helpful to examine how data flows through the network.



AlexNet: Sharing Weights

- The early layers of the network are convolutional.
- This means that the weights are shared across locations of the image.
- The input of size $w \times h \times d$ is transformed into an output of size $w \times h \times d'$.
- The outputs are called feature maps and they are derived by convolving the image with a 3D tensor of size $u \times v \times d'$.
- So, the number of parameters is "merely" $u * v * d' + d'$.
- The output feature maps can be very large however.



AlexNet: Pooling Features

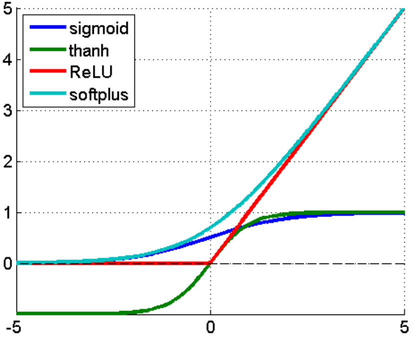
- Like in the Bag-of-Words model, we can pool local features.
- In AlexNet, they authors use 3×3 pooling regions with a stride of 2 pixels.
- This means that after some convolutional layers the feature map size is reduced by a factor of 2.
- They use max pooling: in each feature map, keep the maximum value in each overlapping 3×3 pooling region.
- This helps to contain the size of feature maps propagated through the network.
- And it also helps to build higher-level representations of the image.
- This is because, halving the image resolution is the same as doubling the size of subsequent convolutions.

AlexNet: Unit Saturation

- Another innovation in AlexNet is the use of the Rectified Linear Unit (ReLU) activation function.

$$\sigma(\mathbf{x}) = max(0, \mathbf{x})$$

- This activation function does not saturate like sigmoids.
- The result is a 6x speedup in training.



AlexNet: Reducing Overfitting

- Even with convolutional weight sharing, AlexNet still has 60M parameters.
- To reduce overfitting, the authors use two extra (now standard) tricks:
  - Data augmentation: random translations and reflections of input images are generated, plus random variation in principal directions of RGB space.
  - Dropout: an advanced trick from the Neural Network community which randomly removes half of the inputs to select layers at training time.

AlexNet: More Tricks

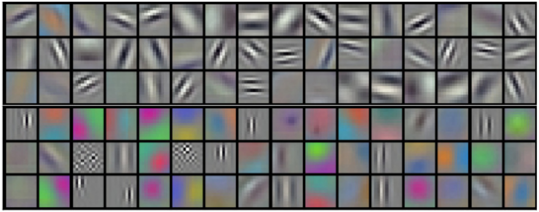
- The AlexNet paper is an excellent resource because it explains all of the tricks necessary to get a deep network to learn:
  - Local response normalization: keep local variation in feature maps under control (section 3.3).
  - Momentum: limits the "skateboard" effect when following valleys in the loss surface, equivalent to L1 (or L2) regularization of weights (section 5).
  - Mini-batch Stochastic Gradient Descent (SGD): with 1.2M training samples, we cannot consider the entire dataset in one batch; instead, randomly sample mini-batches of 128 images (section 5).
  - Multiple GPUs: AlexNet was too big to fit in a single GPU (in 2012), so feature maps are split over two GPUs (section 3.2).
  - Model averaging: state-of-the-art results are obtained by training multiple CNNs and averaging outputs.

AlexNet: Results

- The proof is in the pudding:

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [7]	—	—	26.2%
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

- And in the representations the network learns:



AlexNet: BOW vs. DCNNs

- Feature detection:
  - In BOW, we use handcrafted features as input to pooling and finally classification layers.
  - In DCNNs, we learn convolutional features, which are then pooled, and then shoved into classification layers.
- Local feature pooling:
  - In BOW we use spatial pooling to add structure to our final representation (Spatial Pyramids).
  - In DCNNs, we use max pooling to reduce feature map size and create higher-level features.
- Global feature pooling:
  - In BOW, we compute a global image representation via pooling.
  - In DCNNs, we compute a global image representation via fully-connected (sometimes called dense) layers.
- Training:
  - In BOW, we use handcrafted representations, followed by shallow classifier learning (e.g. an SVM).
  - In DCNNs, we perform end-to-end training of the entire architecture.

AlexNet: Reflections

- AlexNet took the object recognition world by storm.
- Many of the elements of the model are not really new.
- However, this was the first work to convincingly demonstrate how state-of-the-art object recognition systems can be trained end-to-end on real problems.
- This was made possible by a number of confluent development:
  - The availability of enormous amounts of annotate data (ImageNet, with 1.2M training images).
  - Modern GPUs, which make convolutions super fast.
  - Decades of persistent theoretical development (ReLUs, fast backprop, dropout, etc).

CNNs: Very Deep Networks

Very Deep Networks

- Very soon after the ILSVRC 2012 results, the community began experimenting with newer, deeper architectures for CNNs.
- We will look at an architecture that became (and still is) a standard one.

Very Deep Convolutional Networks for Large-Scale Image Recognition. Karen Simonyan and Andrew Zisserman. In: arXiv preprint arXiv:1409.1556, 2014.

- In this paper the authors performed a thorough exploration of the architectural parameter space.
- They varied the hyperparameters (e.g. number of layers, size of convolutions, etc).
- And established a new baseline for CNN-based object recognition.
- These networks are known as VGG16 and VGG19 (VGG = Visual Geometry Group from Oxford).

VGG: The Setup

- Input to the networks is a fixed, $224 \times 224 \times 3$ image tensor.
- The mean RGB value is first subtracted from all training images to center the data.
- All convolutions are $3 \times 3 \times d$ or $1 \times 1 \times d$ in size (d is an arbitrary number of feature maps) with a stride of 1.
- The idea is: if you need larger convolutions, just go deeper.
- Max pooling is done over non-overlapping 2×2 windows with a stride of 2 (2x reduction in size).
- All hidden layer use a ReLU activation function, but do not do local response normalization.

VGG: The Configurations

- The following configurations were considered:

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

VGG: Training

- The training procedure is similar to AlexNet:
  - The training is carried out by optimising the multinomial logistic regression objective using mini-batch gradient descent.
  - The batch size is 256, with momentum of 0.9
  - Training was regularised by weight decay (the L2 penalty multiplier set to 10^4 and dropout on the first two fully-connected layers.
  - The learning rate was initially set to 10^2 and decreased by a factor of 10 when the validation set accuracy stopped improving.
  - Learning was stopped after 370K iterations (74 epochs).
- Initialization:
  - CNNs are extremely sensitive to initialization of the weights.
  - For training VGG networks, the authors use a combination of random initialization and pre-training.

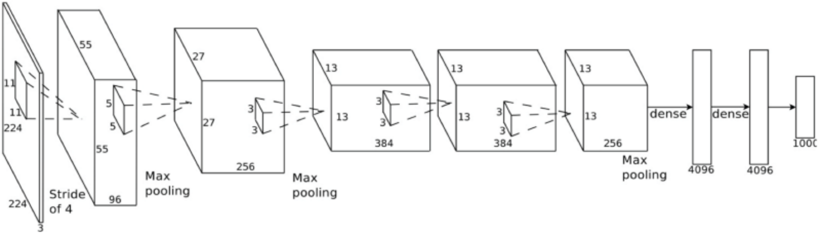
VGG: Training Image Size

- Note that training images are all scaled to 224×224 pixels before passing them through the network.
- This is the same as AlexNet, and clearly can affect the image content by introducing artifacts (consider portrait images).
- In VGG networks, images are isotropically scaled so that the smallest dimension has fixed size.
- Then subimage of size 224×224 is randomly cropped from the scaled image.
- The authors evaluated randomly scaling to between 256 and 384 pixels for the smallest dimension.

VGG: Testing Image Size

- At testing time, there are five strategies for image scaling evaluated (and their combinations):
  - Dense: the network is fully convolutionalized (I will explain this on the next slide), evaluated densely on the input image, and results are globally pooled.
  - Single-scale: a single isotropic scale is used.
  - Multi-scale: like at training time, images are scaled to three discrete isotropic scales. of th
  - Multi-crop: multiple, random crops are taken from the fully-convolutional output for average pooling.

- Below is a diagram of a typical ConvNet.
- How can we make it independent of the image size?



VGG: Results

- Even when only considering a single input scale, the results are impressive.
- Note: deeper is better, LRN doesn't help, scale jittering at test time does.

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
	[256;512]	384	**25.5**	**8.0**

- Using multiple scales leads to even better performance:

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
B	256	224,256,288	28.2	9.6
C	256	224,256,288	27.7	9.2
	384	352,384,416	27.8	9.2
	[256; 512]	256,384,512	26.3	8.2
D	256	224,256,288	26.6	8.6
	384	352,384,416	26.5	8.6
	[256; 512]	256,384,512	**24.8**	**7.5**
E	256	224,256,288	26.9	8.7
	384	352,384,416	26.7	8.6
	[256; 512]	256,384,512	**24.8**	**7.5**

- As does fusing multiple cropping strategies:

ConvNet config. (Table 1)	Evaluation method	top-1 val. error (%)	top-5 val. error (%)
D	dense	24.8	7.5
	multi-crop	24.6	7.5
	multi-crop & dense	**24.4**	**7.2**
E	dense	24.8	7.5
	multi-crop	24.6	7.4
	multi-crop & dense	**24.4**	**7.1**

UNIVERSITÀ DEGLI STUDI FIRENZE

- Finally, model averaging over multi-scale, multi-crop models leads to state-of-the-art performance:

Method	top-1 val. error (%)	top-5 val. error (%)	top-5 test error (%)
VGG (2 nets, multi-crop & dense eval.)	**23.7**	**6.8**	**6.8**
VGG (1 net, multi-crop & dense eval.)	24.4	7.1	7.0
VGG (ILSVRC submission, 7 nets, dense eval.)	24.7	7.5	7.3
GoogLeNet (Szegedy et al., 2014) (1 net)	-	7.9	
GoogLeNet (Szegedy et al., 2014) (7 nets)	-	**6.7**	
MSRA (He et al., 2014) (11 nets)	-	-	8.1
MSRA (He et al., 2014) (1 net)	27.9	9.1	9.1
Clarifai (Russakovsky et al., 2014) (multiple nets)	-	-	11.7
Clarifai (Russakovsky et al., 2014) (1 net)	-	-	12.5
Zeiler & Fergus (Zeiler & Fergus, 2013) (6 nets)	36.0	14.7	14.8
Zeiler & Fergus (Zeiler & Fergus, 2013) (1 net)	37.5	16.0	16.1
OverFeat (Sermanet et al., 2014) (7 nets)	34.0	13.2	13.6
OverFeat (Sermanet et al., 2014) (1 net)	35.7	14.2	-
Krizhevsky et al. (Krizhevsky et al., 2012) (5 nets)	38.1	16.4	16.4
Krizhevsky et al. (Krizhevsky et al., 2012) (1 net)	40.7	18.2	-

VGG: Analysis

- The VGG16 and VGG19 networks are still in common use today (though used in novel ways).
- In this paper the authors significantly improved over the previous generation by going deeper.
- Again, most of the ideas are not new, but systematic exploration of the design space led to significant improvements.
- Note that the networks are deeper, but have a smaller memory footprint at training time due to carefully balancing the size of feature maps.
- Dense evaluation of the network at test time can also increase performance, leading to fully convolutional networks that are independent of input image size.
- The architecture is still a classical ConvNet.

CNNs: Even Deeper Networks

- Of course, Google had to jump into the game.
- We will now consider a different architecture for CNNs, one that will allow us to go even deeper:

Going Deeper with Convolutions. C Szegedy, W Liu, Y Jia, P Sermanet, S Reed, D Anguelov, D Erhan, V Vanhoucke, and A Rabinovich. In: Proceedings of CVPR 2015.

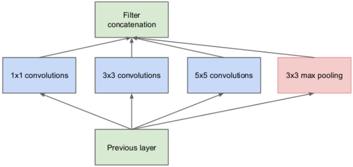
- This will use the idea of fully convolutional networks to both go deeper and to limit size of intermediate feature maps.

GoogLeNet: Observations

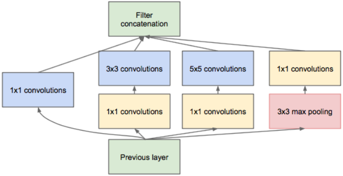
- Trend: bigger and deeper networks (and multiple models at that).
- Problems:
  - Problem 1: bigger networks need a lot more annotated training data, which is extremely expensive to collect.
  - Problem 2: hardware resources are finite, including memory and CPU cycles.
  - CPU hog: convolutional layers over many feature maps.
- Ideas:
  - Use multiple, multi-resolution convolutions at each layer to better capture local structure.
  - Use fully-convolutionalř layers (i.e. $1 \times 1 \times d$) convolutions to *reduce feature map dimensionality before expensive convolutions.

GoogLeNet: Inception

- Thus, the Inception Module was born:



Naive



With dimension reduction

- The naive module concatenates multiple feature map representations at each level.
- This includes expensive 5x5 and 3x3 convolutions.
- The full Inception Module applies 1x1 convolutions to reduce dimensionality by first convolving with 1x1 filters.

UNIVERSITÀ DEGLI STUDI FIRENZE

- The GoogLeNet name is an homage to the first ConvNet proposed by Yann LeCun in 1989.

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

- Let's take a detailed look at the monster of a figure 3.
- First the stem.
- Then a cascade of Inception Modules.
- Then the auxiliary loss layers.
- The final output layers.

GoogLeNet: Training

- The training procedure of GoogLeNet is a complete mess.
- This is typical of the type of training that happens leading up to a competition.
- They experimented with many configurations, keeping some, discarding others.
- In any case, they use SGD on CPUs (they're Google, they have CPUs at their disposal).
- Final results are based on a combination of 7 trained GoogLeNet variants.
- At test time they pass 144 224×224 RGB images through the network and average the outputs.

- The progress in two years was significant:

Team	Year	Place	Error (top-5)	Uses external data
SuperVision	2012	1st	16.4%	no
SuperVision	2012	1st	15.3%	Imagenet 22k
Clarifai	2013	1st	11.7%	no
Clarifai	2013	1st	11.2%	Imagenet 22k
MSRA	2014	3rd	7.35%	no
VGG	2014	2nd	7.32%	no
GoogLeNet	2014	1st	6.67%	no

- And the effect of all of the tricks were also significant:

Number of models	Number of Crops	Cost	Top-5 error	compared to base
1	1	1	10.07%	base
1	10	10	9.15%	-0.92%
1	144	144	7.89%	-2.18%
7	1	7	8.09%	-1.98%
7	10	70	7.62%	-2.45%
7	144	1008	6.67%	-3.45%

GoogLeNet: Reflections

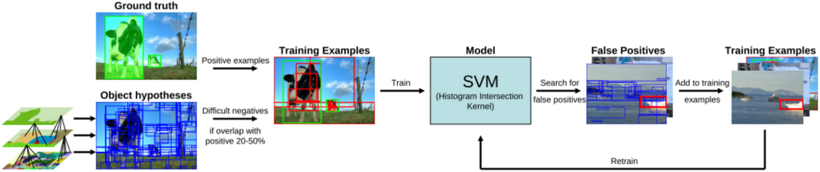
- The GoogLeNet architecture has 12x fewer parameters than AlexNet.
- And it makes less than 1/3 of the errors.
- This architecture demonstrates that CNNs can be tamed in size complexity.
- In the long run, this is important for deployment on limited hardware.
- GoogLeNet also popularized the fully convolutional layer architecture, which has been used (for example) for object segmentation.

CNNs: Fast-RCNN

- We will now look at a network designed for object detection rather than just recognition.

Fast-RCNN. R Girshick. In: Proceedings of ICCV 2015.

- This is a detection technique that actually uses a method from the pre-CNN revolution (Selective Search):



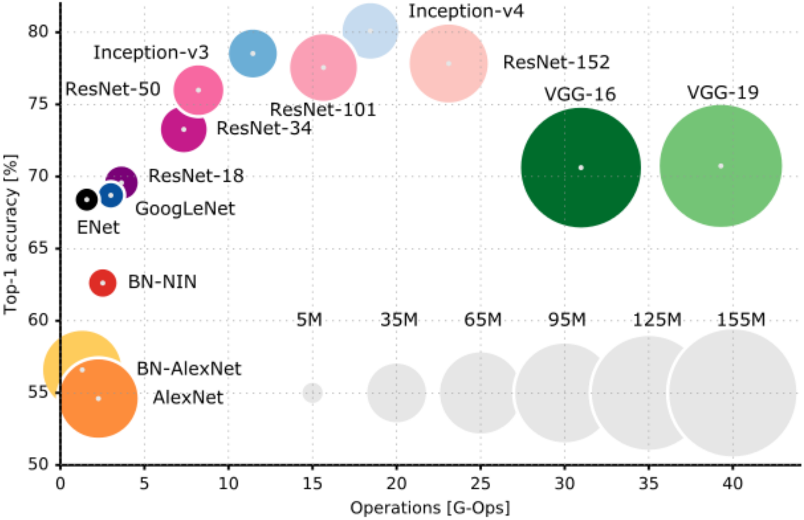
- Main idea: propose likely object locations in a class-independent way, using only image content; classify each proposal.

Fast-RCNN: The Idea

- Problem: how to do this efficiently?
- [SWITCH PRESENTATION]

Deeper, Bigger, and Better

Deeper, Bigger, and Better



Deeper, Bigger, and Better

- We have come a long way in five short years.
- However, though we have left behind (for the most part) the era of handcrafted features, we have entered the era of handcrafted architectures.
- It can be hard to make sense out of the confusing array of CNN architectures out there.
- It is even harder to optimally train these networks.

Discussion

Discussion

- Discuss