

Fundamentals of Machine Learning:

Convolutional Neural Networks

Prof. Andrew D. Bagdanov ([andrew.bagdanov AT unifi.it](mailto:andrew.bagdanov@unifi.it))



UNIVERSITÀ
DEGLI STUDI
FIRENZE
DINFO
DIPARTIMENTO DI
INGEGNERIA
DELL'INFORMAZIONE

Outline

A Critique of Pure Reason

Connectionism: From MLPs to CNNs

AlexNet: The Shot Heard 'Round the World

The Family of Very Deep Networks

Deep Residual Networks: Deeper and Deeper

CNNs: Tricks of the Trade

Discussion

A Critique of Pure Reason

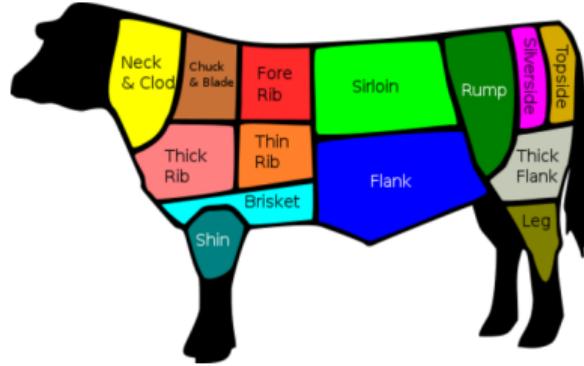
What makes a cow, a *cow*?



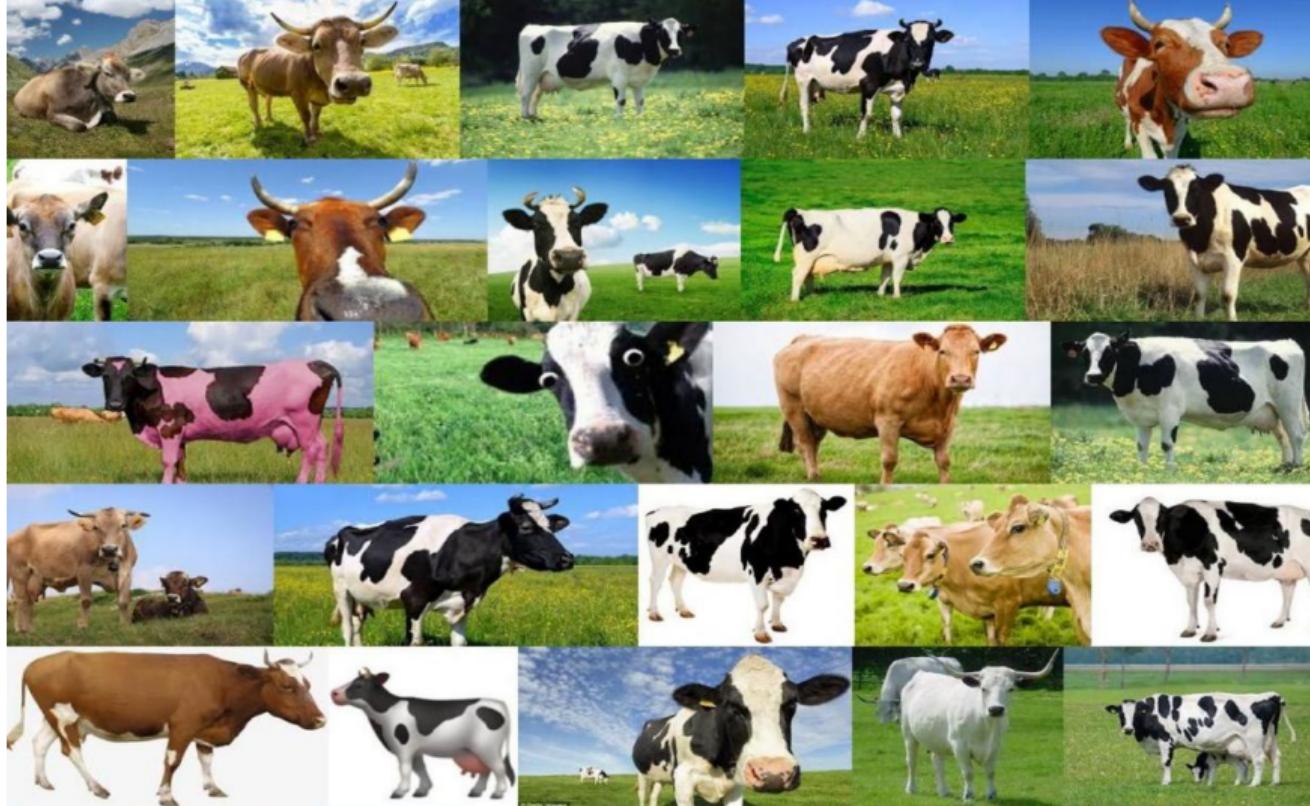
What makes a cow, a *cow*?



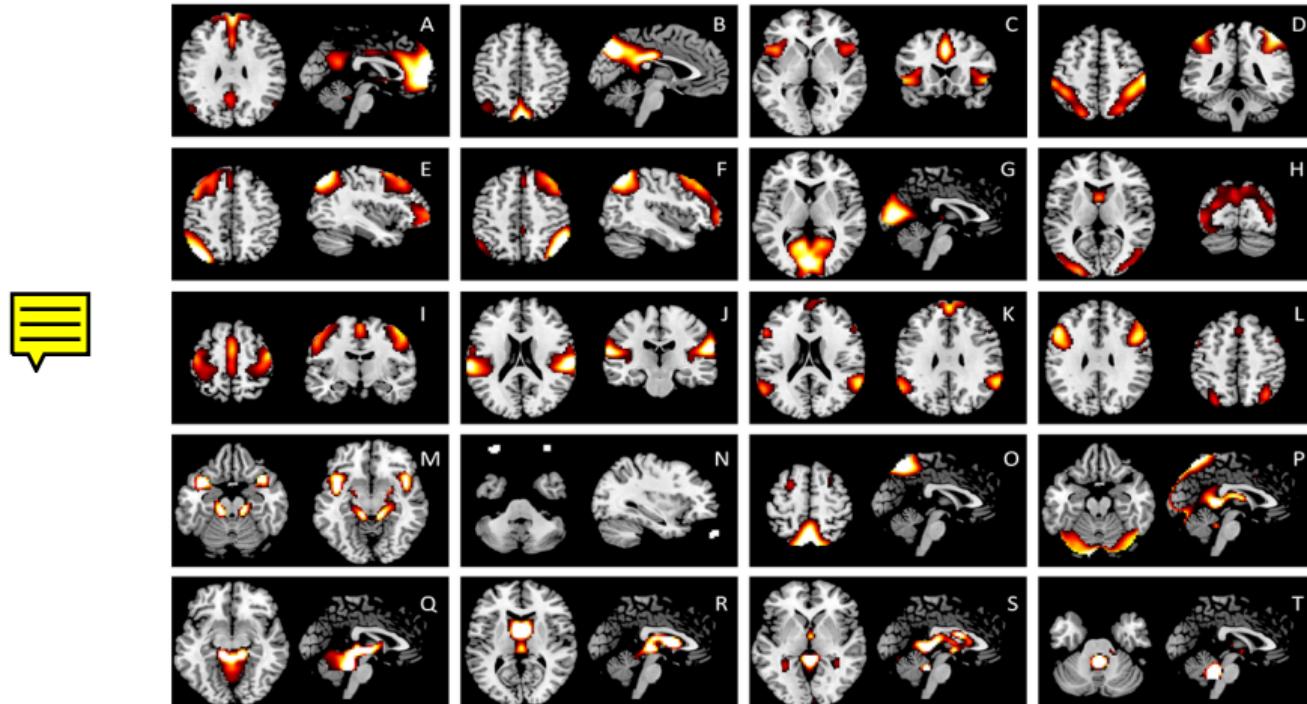
What makes a cow, a cow?



What makes a cow, a cow?

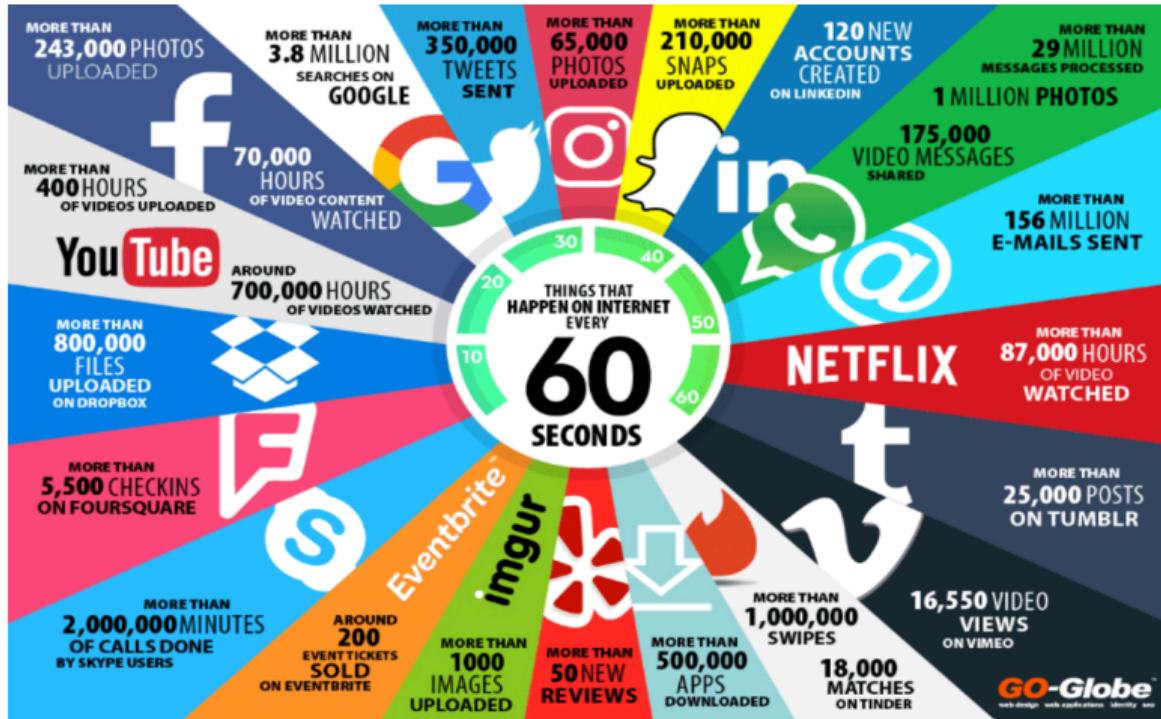


A picture is worth a thousand words...



[Cabral et al., 2013]

Content crash: why do we care?

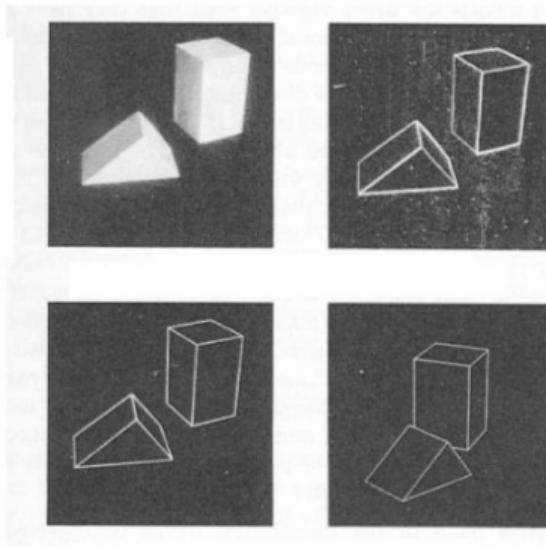


Visual recognition: explicit models

- Early works on visual recognition used (very) **explicit** models [[Roberts, 1963](#)].
- They are significant as first steps and appeal to our **analytic** beliefs about image understanding.

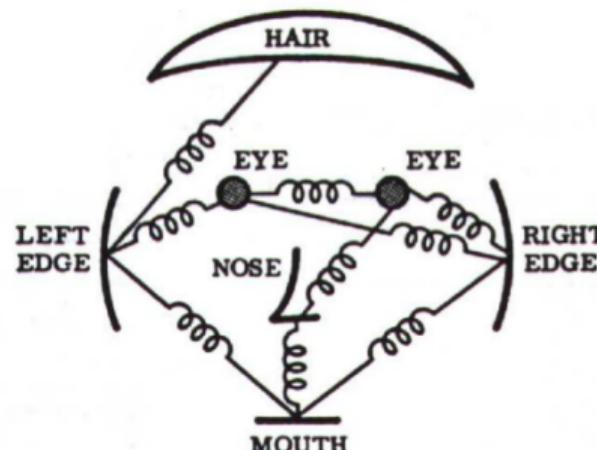


L. Roberts



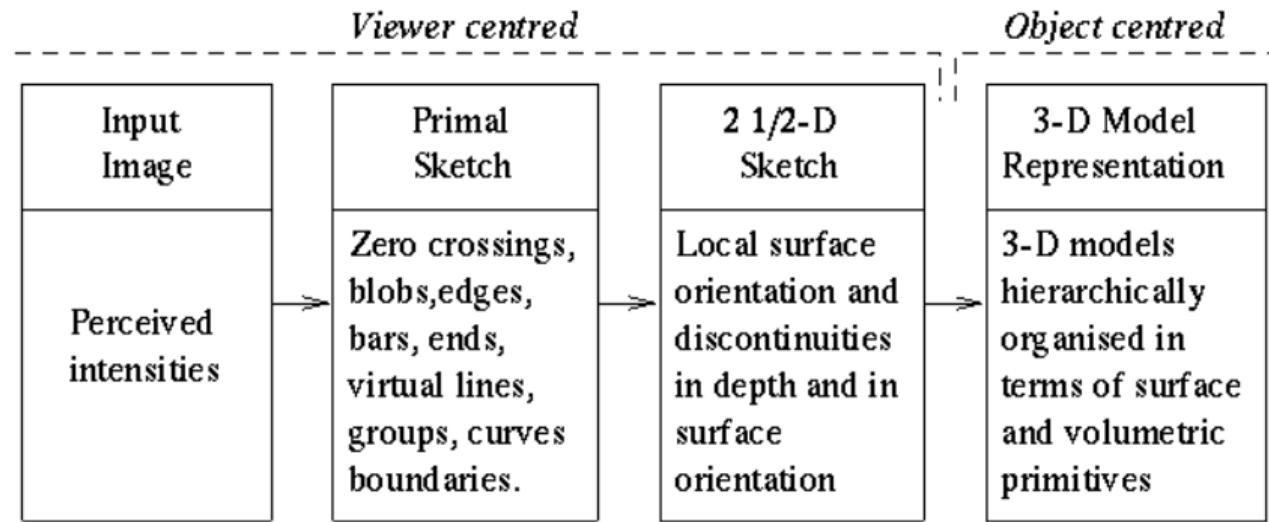
Visual recognition: explicit models

- This type of **explicit** model of recognition gave way to part-based representations [Fischler and Elschlager, 1973].
- An object was represented by a set of **parts** arranged in an **elastic** configuration.
- The **trend**: move away from **models** and move towards the **image**.



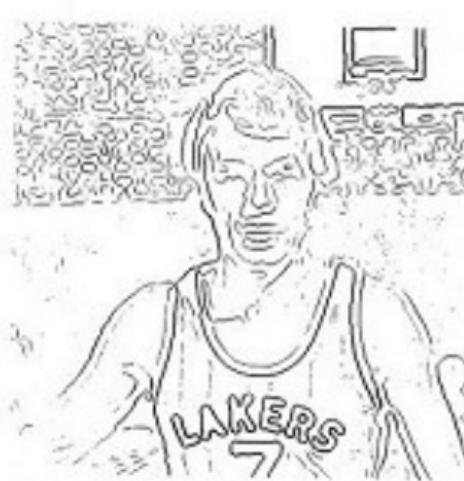
Visual recognition: Marr's Vision

- In his book, Marr developed a **modular** framework for computervision [[Marr, 1982](#)].
- This framework consists of **three representations** that are created, maintained, and interpreted by the process of vision:



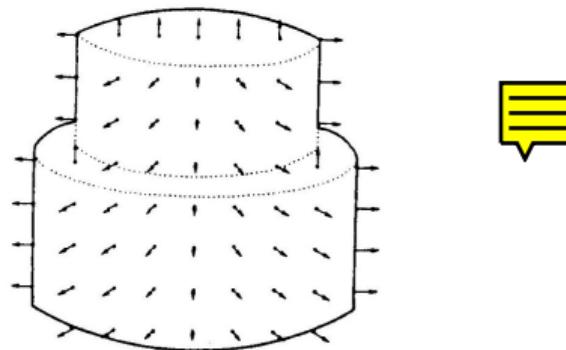
Visual recognition: Marr's Vision

- The Primal Sketch is a description of the intensity changes in the image and their local geometry.
- It is based on the assumption that intensity variations are likely to correspond to physical realities like object boundaries.



Visual recognition: Marr's Vision

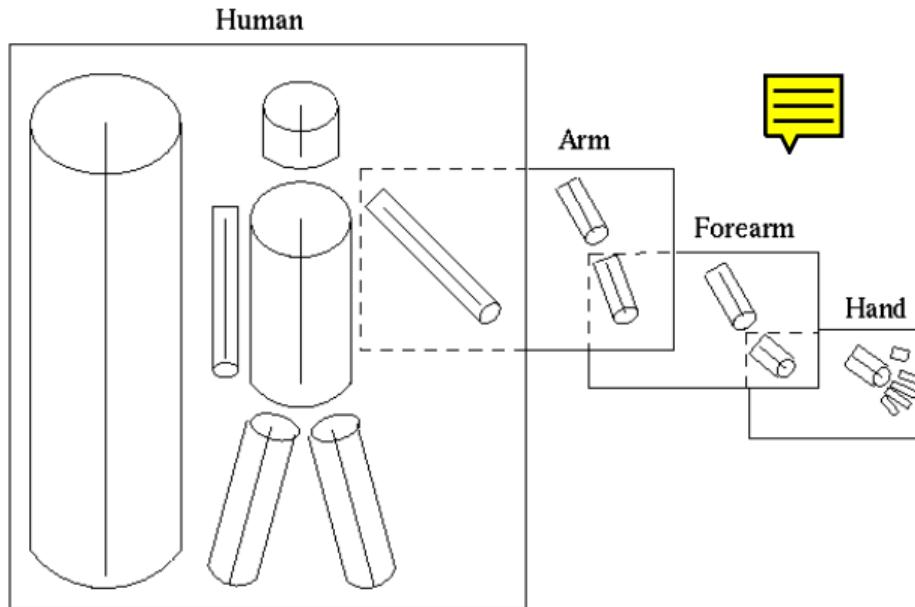
- The **2.5D Sketch** is a **viewer-centric** representation of orientation and depth of visible surfaces drawing from the primal sketch.
- Note that **no grouping** is done yet: we are only associating **weak geometry** to image elements.
- Hence the metaphor **2.5D sketch**.



Vision: A computational investigation into the human representation and processing of visual information

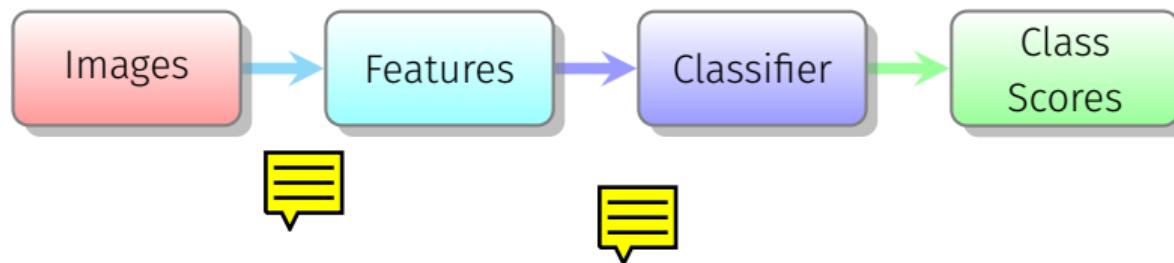
Visual recognition: Marr's Vision

- The **3D Model** is an object-centric representation of 3D objects in the image.
- The goal of this model is to enable object **manipulation** and **recognition**.



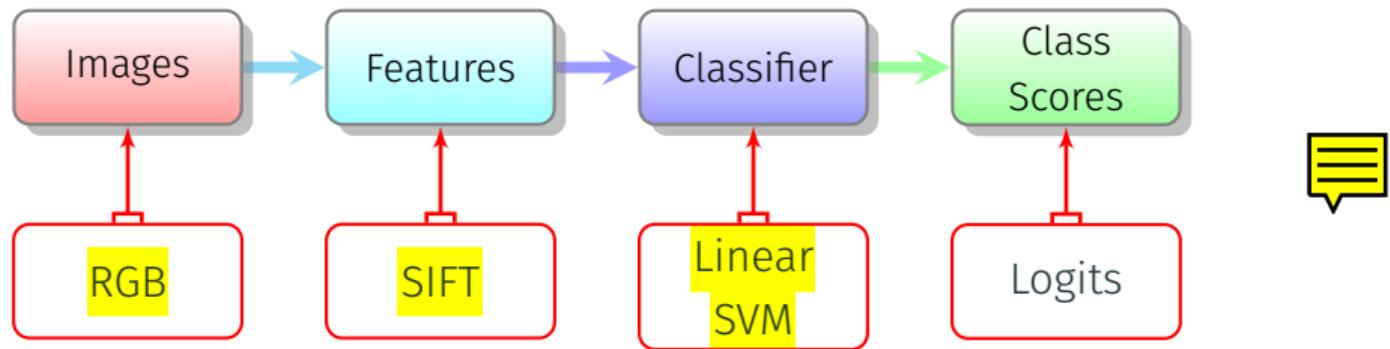
Visual recognition: implicit models

- Let's consider a **more-or-less** Standard setup of supervised learning for visual classification.
- We can imagine a **simple pipeline** like below.
- Each stage has it's own **design space** and critical choices to be made.
- This appeals to the **computer scientist** in us since we are effectively **dividing**, **modularizing**, and (hopefully) conquering.



Visual recognition: implicit models

- Let's consider a **more-or-less** Standard setup of supervised learning for visual classification.
- We can imagine a **simple pipeline** like below.
- Each stage has it's own **design space** and critical choices to be made.
- This appeals to the **computer scientist** in us since we are effectively **dividing**, **modularizing**, and (hopefully) conquering.



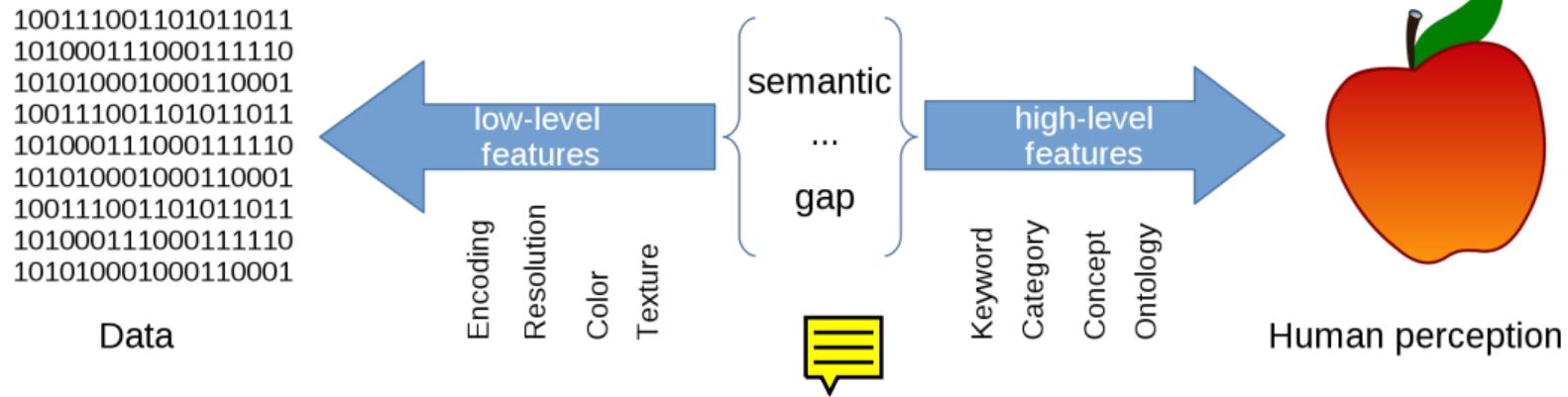
Visual recognition: why is this hard?

- A paper appeared in 2000 that summarized the state-of-the-art in visual recognition [Smeulders et al., 2000].
- It introduced **sensory gap** into the conversation on visual recognition:
The sensory gap is the gap between the object in the world and the information in a (computational) description derived from a recording of that scene.
- **Think about this for a moment:** we are always working with an **imperfect** reconstruction of the real world.
- Images have finite resolution, they are subject to noise, they are acquired with a sensor which is **another** free object in the world.
- This **sensory gap** must be surpassed in order to render object recognition **invariant to scene-incidental artifacts.**

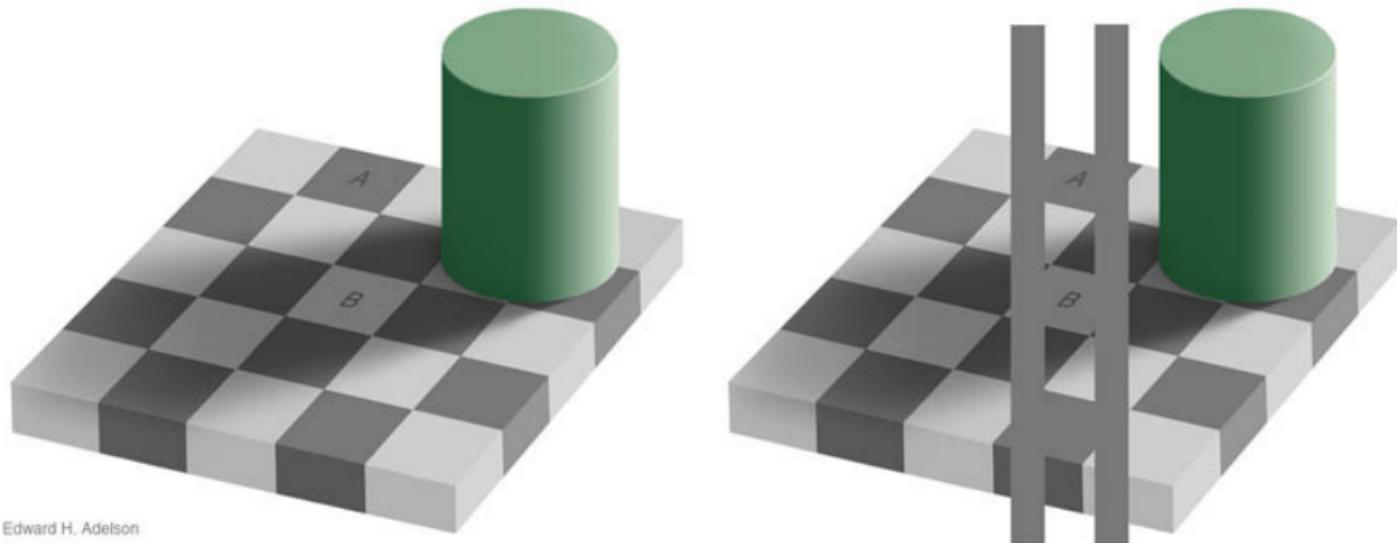
The semantic gap

- The other key concept is the **semantic gap**:

The semantic gap is the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation.



The semantic gap

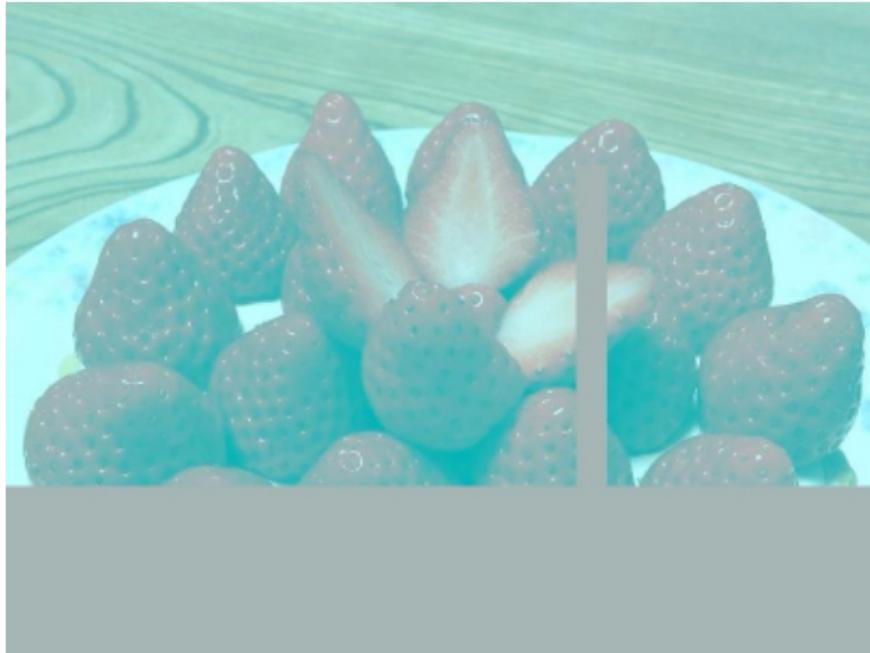


Edward H. Adelson

The semantic gap

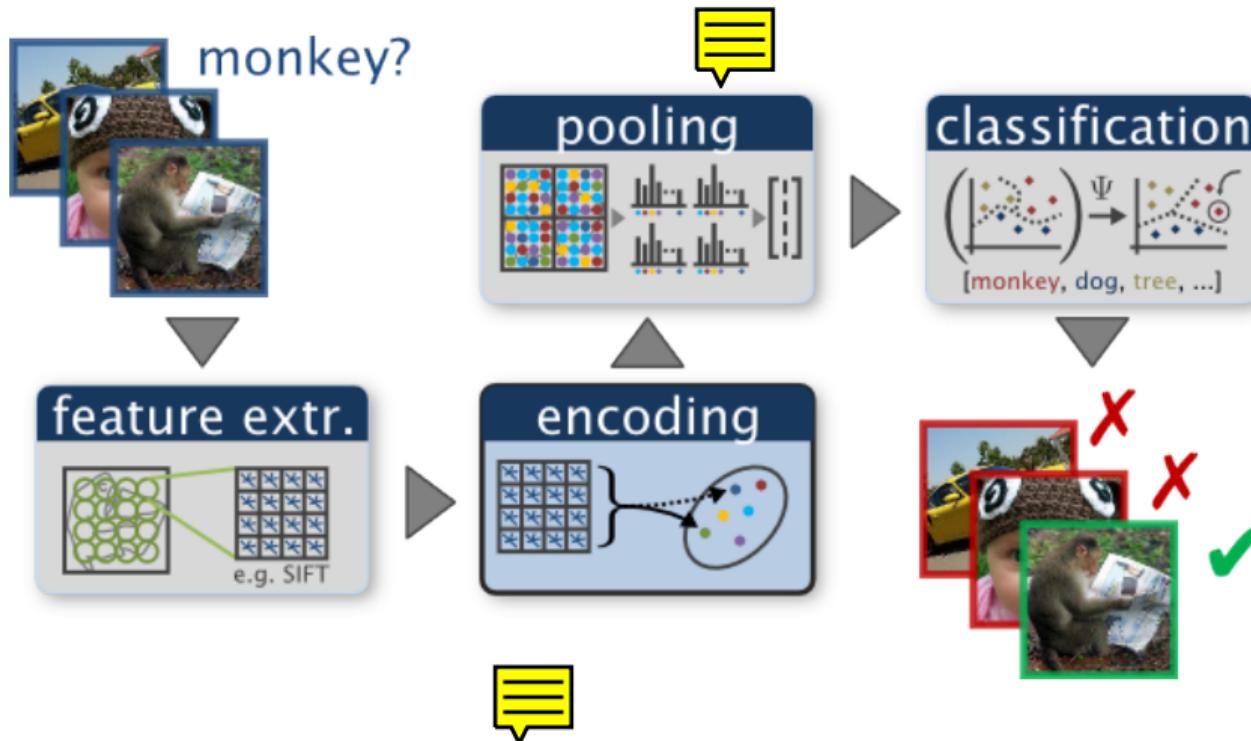


The semantic gap

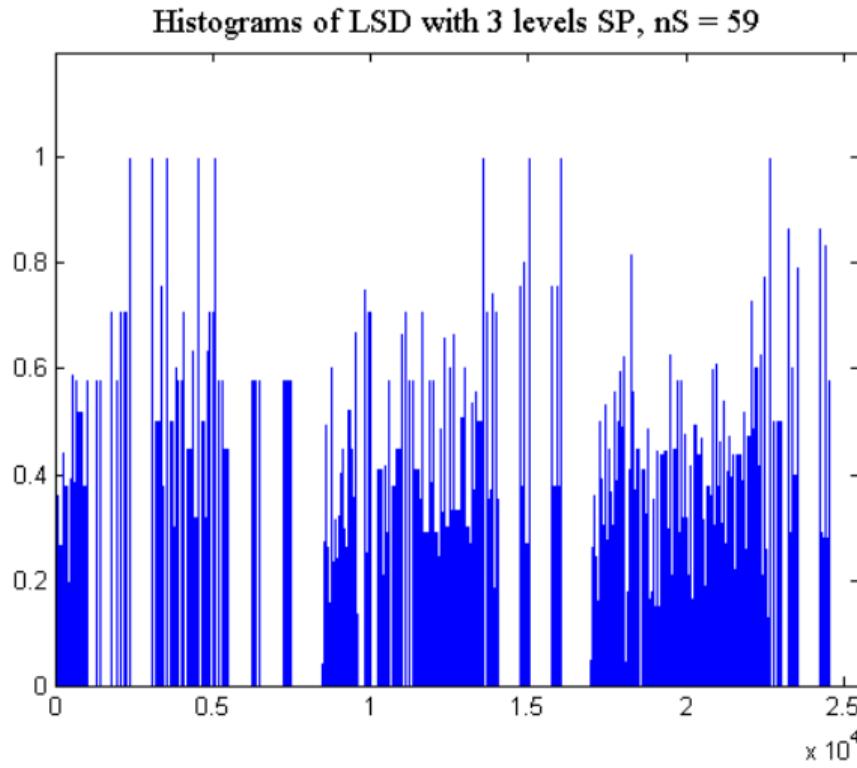


Historical context: Bags of Features

- This was the state-of-the-art in 2011:

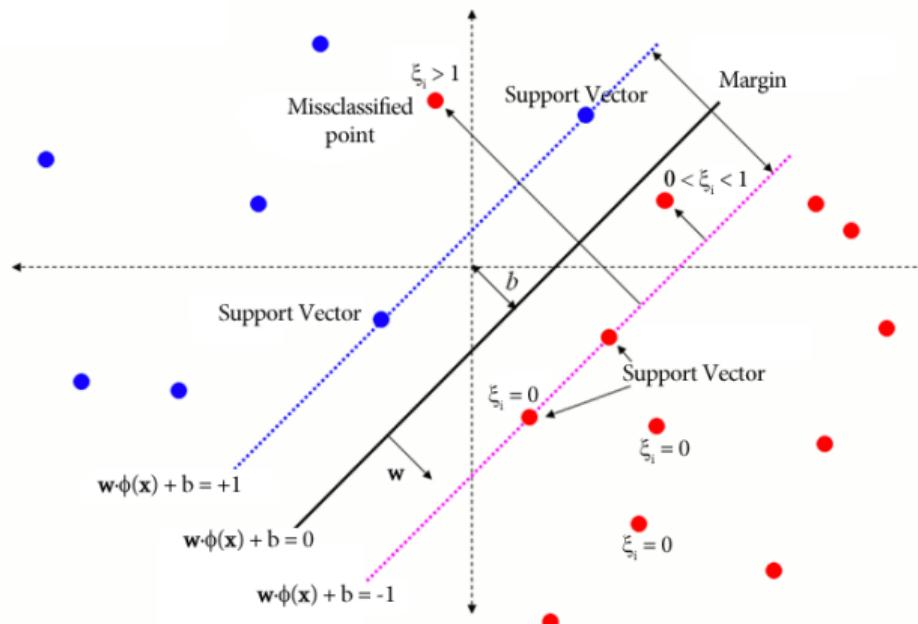


Historical context: This was a “cat”



Historical context: Now “learn”

- To each image representation was associated one (or more) **labels**.
- Then we feed these into a multi-class SVM.



Historical context: A recipe

- Then returned the **optimal** decision boundaries between all classes (and all the others).
- The **process** is important:
 1. **First:** extract a **handcrafted** representation of **fiducial points**.
 2. **Then:** encode these into a **global** image representation.
 3. **Then:** fit an SVM (with or without kernel).
- **Pro:** the actual **learning** has few hyperparameters (**usually just one**).
- **Con:** many **handcrafted** elements with **many** (basically infinitely many) **hyperparameters**.
- **Con:** **learning** is separate from **representation**.

Lecture objectives

At the end of the lecture you should:

- Understand the historical motivations of the *general problem of computer vision* and the central role played by the *semantic gap*.
- Understand the *differences* and *similarities* between the *Multilayer Perceptron (MLP)* and *Convolutional Neural Network (CNN)* model architectures.
- Understand some of the recent *historical* developments and *technological innovations* that made the *Deep Learning Renaissance* possible.
- Understand the two *fundamental families* of CNN architectures and what *distinguishes* them from each other.
- Understand some of the “*tricks*” needed to make CNNs effective in *practice*.

Connectionism: From MLPs to CNNs

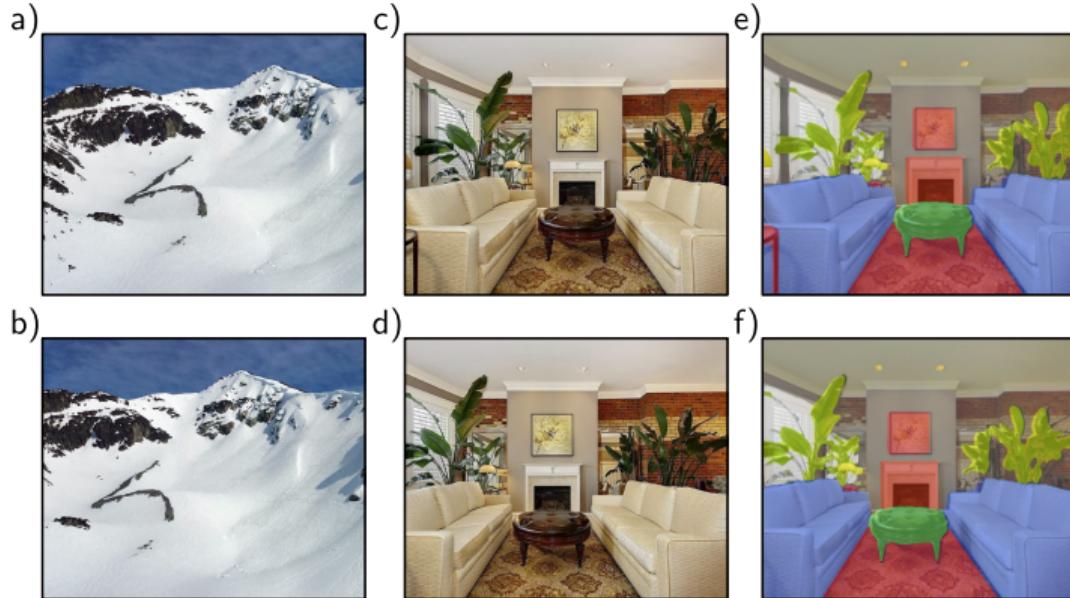
The attractiveness of end-to-end learning

- MLPs have a number extremely attractive features:
 - It is an **end-to-end** model: we can train **everything** in the model using a single optimization algorithm.
 - MLPs learn **representations** of input **and** classifier.
- Why can't we just use this model for **image recognition** problems?
- An MLP should be able to **learn** feature representations that are in turn **good** representations for **classification**.
- Why is this problematic?
- **NOTE:** The treatment and many figures in this section I have taken from the excellent book *Understanding Deep Learning*, by Simon Prince.
- I **highly** recommend this book.



On invariance and equivariance

- A function $f[x]$ is **invariant** to transformation $t[x]$ if $f[t[x]] = f[x]$.
- A function $f[x]$ is **equivariant** to transformation $t[x]$ if $f[t[x]] = t[f[x]]$.



One dimensional convolutions

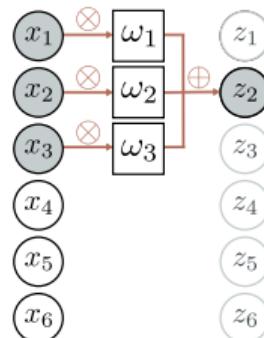
- The convolution operator computes a weighted sum of neighboring values:

$$h_i = \sigma[\beta_i + w_1x_{i-1} + w_2x_i + w_3x_{i+1}]$$

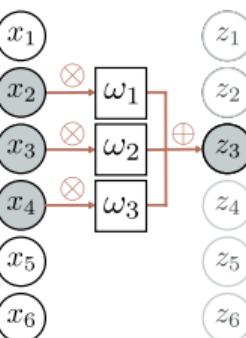
☞

$$= \sigma[\beta + \sum_{j=1}^3 w_j x_{i+j-1}]$$

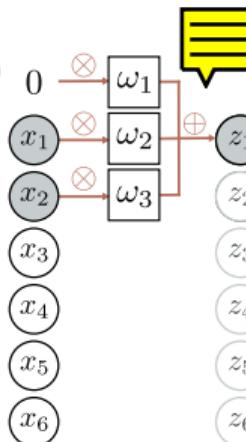
a)



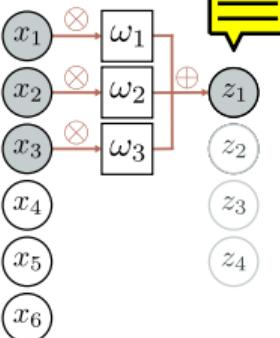
b)



c)

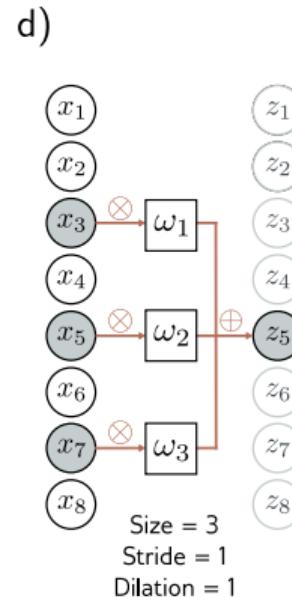
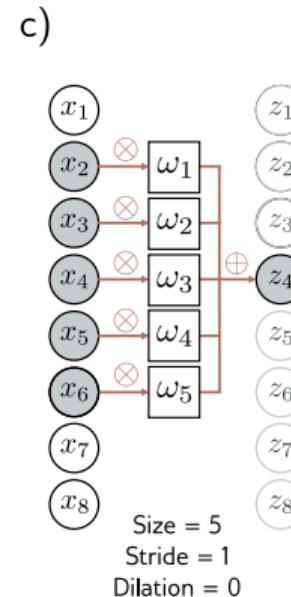
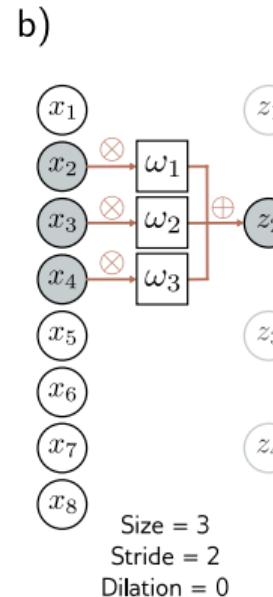
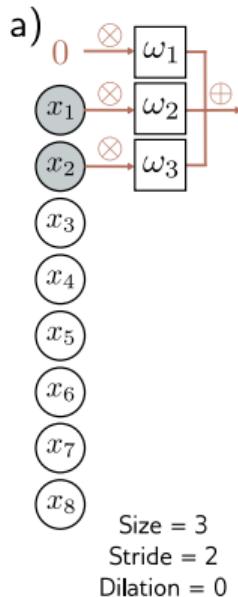


d)



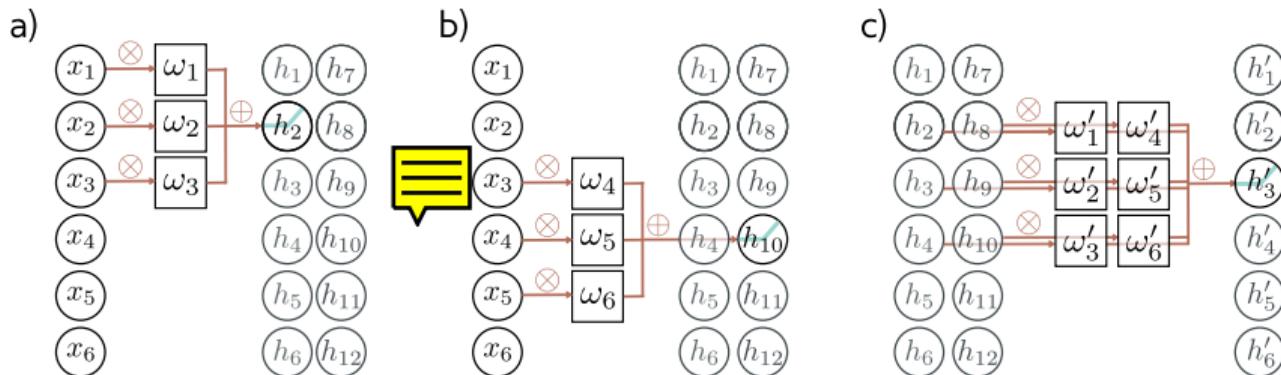
One dimensional convolution (continued)

- The convolution is **very** flexible (at the cost of **hyperparameters**):



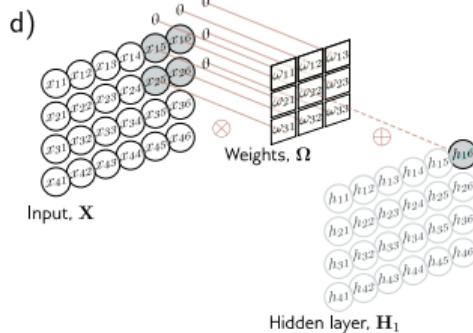
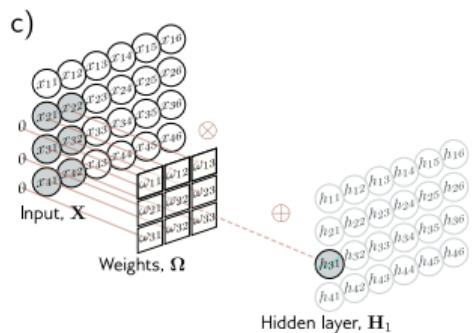
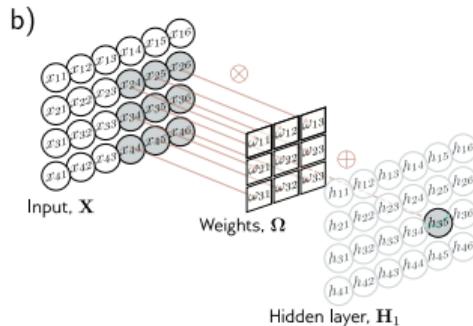
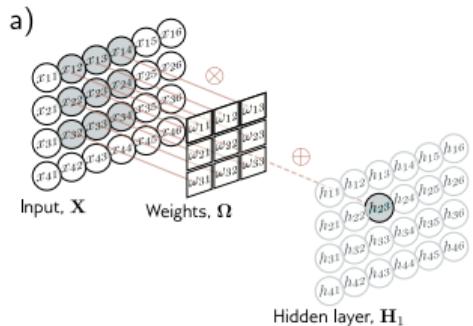
Multichannel one dimensional convolutions

- We can compute **multiple** convolutions and convolve **multiple** inputs:



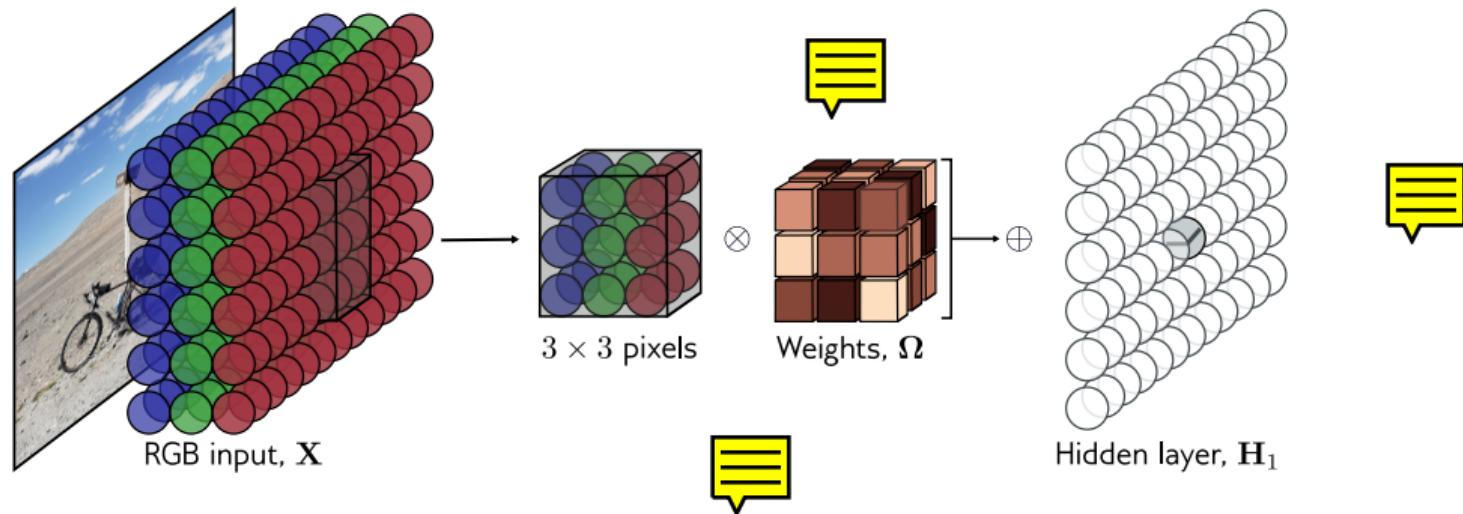
Two dimensional convolution

- Exactly the same operation is naturally extended to **two** spatial dimensions:



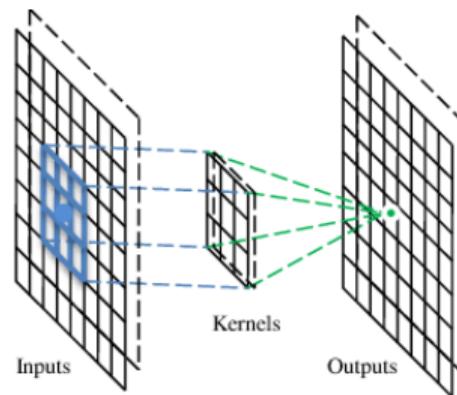
Multichannel two dimensional convolutions

- And to multichannel images (or feature maps):



From MLPs to CNNs (which are really still **MLPs**)

- A **convolutional layer** is a **fully-connected layer** with weights *shared across locations of the image*.
- The input of size $w \times h \times c$ is transformed into an output of size $w \times h \times c'$.
- The outputs are called **feature maps** and they are derived by convolving the image with d' 3D **tensors** of size $u \times v \times d$.
- So, the number of parameters is “merely” $(u * v * c' * c) + c'$.



A CNN is really an MLP (in disguise)

- That sure doesn't **look** like an MLP...

Image to column operation (im2col)

Slide the input image like a convolution but each patch become a column vector.

Input Image [4x4x3]

	33	34	35	36
17	18	19	20	21
24	25	26	27	28
31	32	33	34	35

Kernel Width:2
Kernel Height:2
Stride:1.
Padding:0

$$W_{out} = (W_{in} - kW + 2^*P)/S + 1$$
$$H_{out} = (H_{in} - kH + 2^*P)/S + 1$$

$$W_{out} = (4-2)/1+1=3$$
$$H_{out} = (4-2)/1+1=3$$

2x2x3 column vector
[2x2] R, [2x2] G, [2x2] B

Result: [12x9]

1	2	3	5	6	7	9	10	11
2	3	4	6	7	8	10	11	12
5	6	7	9	10	11	13	14	15
6	7	8	10	11	12	14	15	16
17	18	19	21	22	23	25	26	27
18	19	20	22	23	24	26	27	28
21	22	23	25	26	27	29	30	31
22	23	24	26	27	28	30	31	32
33	34	35	37	38	39	41	42	43
34	35	36	38	39	40	42	43	44
37	38	39	41	42	43	45	46	47
38	39	40	42	43	44	46	47	48

9 possible
Sliding window
positions

im2col

We can multiply this result matrix [12x9]
with a kernel [1x12].

result = kernel x matrix

The result would be a row vector [1x9].

We need another operation that will convert
this row vector into a image [3x3].

$\text{Result: } [1 \times 9]$

X

$\text{Conv out: } [3 \times 3]$



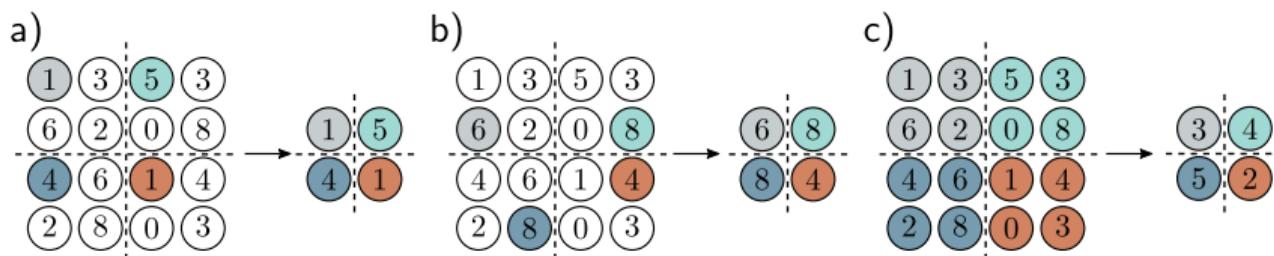
col2im

1	2	3	5	6	7	9	10	11
2	3	4	6	7	8	10	11	12
5	6	7	9	10	11	13	14	15
6	7	8	10	11	12	14	15	16
17	18	19	21	22	23	25	26	27
18	19	20	22	23	24	26	27	28
21	22	23	25	26	27	29	30	31
22	23	24	26	27	28	30	31	32
33	34	35	37	38	39	41	42	43
34	35	36	38	39	40	42	43	44
37	38	39	41	42	43	45	46	47
38	39	40	42	43	44	46	47	48

Consider col2im as a row major
reshape.

The final ingredient: Pooling Layers

- Representation learning works by creating an **semantic bottleneck**.
- We **force** the network to extract **meaningful** representations that capture **semantic features**.
- In CNNs we do this by **Subsampling, Max Pooling, and Mean Pooling**:



- We will now look at some **actual** CNN architectures which will make this clearer.



AlexNet: The Shot Heard 'Round the World

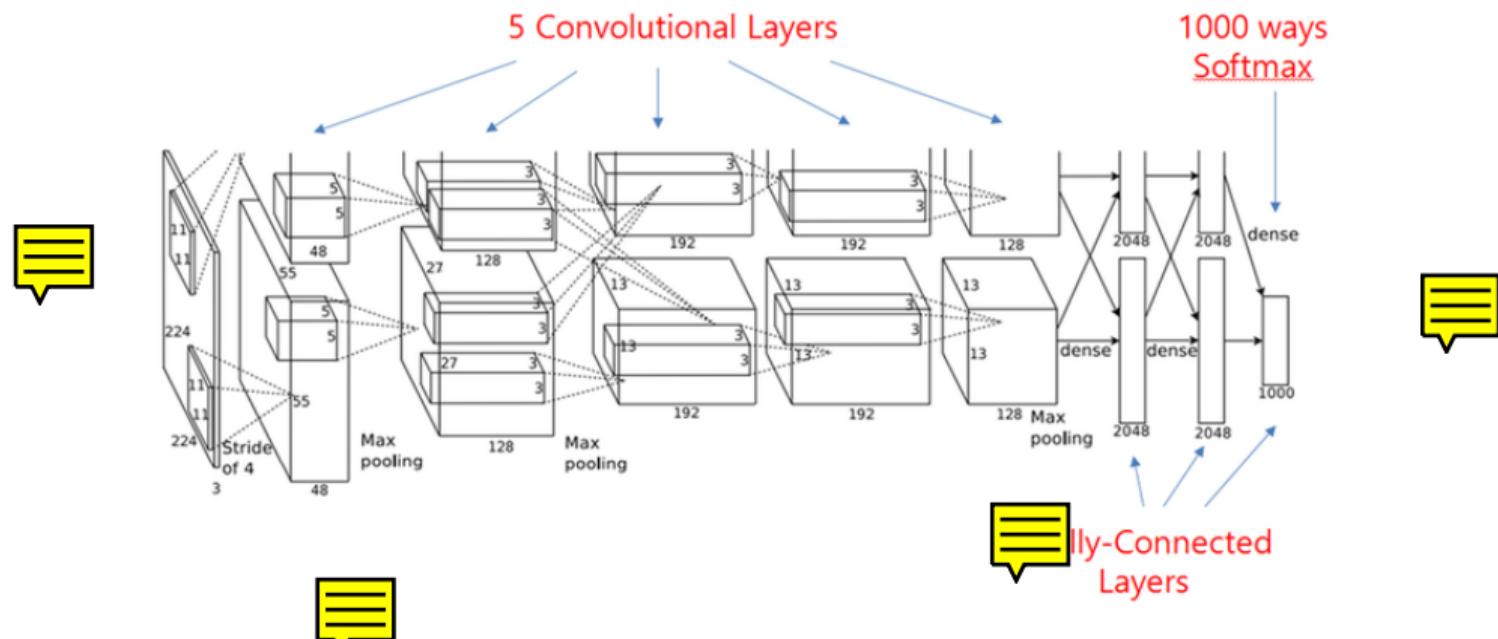
AlexNet: Context

- We will now take a look at the International Large Scale Visual Recognition Competition (ILSVRC) submission that **changed everything** [Krizhevsky et al., 2012].
- This architecture systematically addresses **most** of the problems with training large network architectures on large datasets.
- It is a **Convolutional Neural Network (CNN)** that is universally called **AlexNet**.
- It is also a **Deep Network** because it has **many** hidden layers.

ImageNet Classification with Deep Convolutional Neural Networks

AlexNet: The Architecture

- 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: Pooling Features

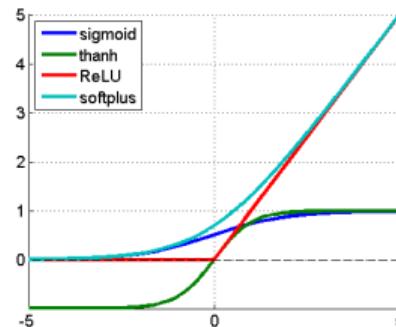
- Like in the Bag-of-Words model, we can **pool** local features.
- AlexNet uses 3×3 **pooling regions** with a **stride** of 2 pixels.
 - This means that after some convolutional layers the feature map size is **reduced by a factor of 2**.
 - They use **max pooling**: in each feature map, keep the **maximum value** in each overlapping 3×3 pooling region (in **each** feature map).
 - 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.

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

- 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:



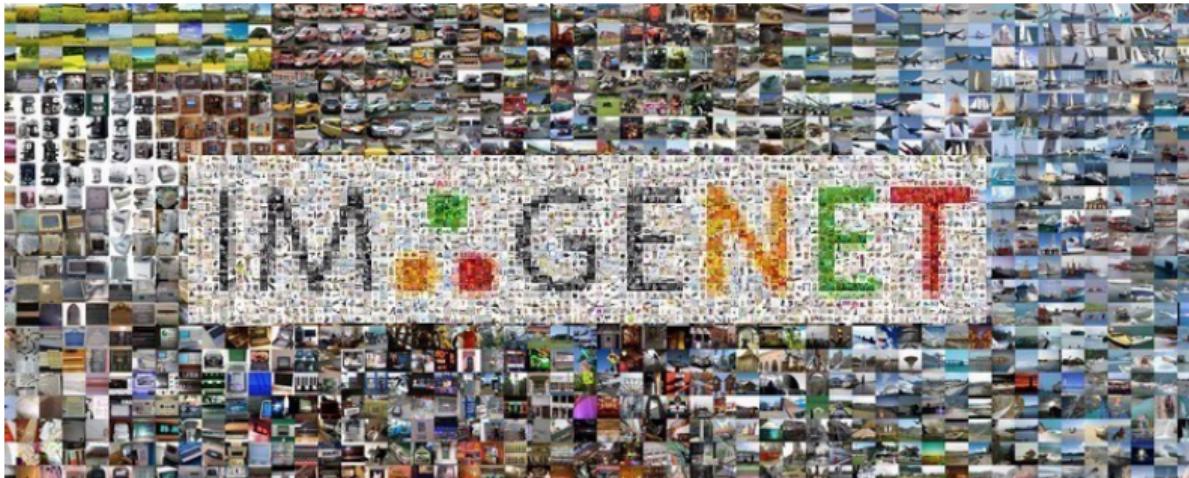
AlexNet: Reflections

- 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).



Reflections: CNNs are really big

- One of the first observations one can make about CNNs is that they have a **HUGE** number of parameters.
- Even modestly-sized, state-of-the-art networks can have on the order of **150 million trainable parameters**.
- Fitting such models requires **massive** amounts of **labeled training data**.



The Family of Very Deep Networks

VGG: Small convolutions, more depth

- Now that we have AlexNet as a point of reference, we can look at some other state-of-the-art CNN architectures.
- The VGG Family of Networks (VGG = Visual Geometry Group from Oxford) is a refinement of the AlexNet style of CNN [Simonyan and Zisserman, 2014].
- In this work the authors performed a thorough exploration of the architectural parameter space.
- They varied hyperparameters (e.g. number of layers, size of convolutions, etc).
- And established a new baseline for CNN-based object recognition.

Very deep convolutional networks for large-scale image recognition

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 layers use a ReLU activation function, but do not do local response normalization.

Very deep convolutional networks for large-scale image recognition

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 optimizing the multinomial logistic regression objective using mini-batch gradient descent.
 - Training was regularised by weight decay 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.

Very deep convolutional networks for large-scale image recognition

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 a 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.

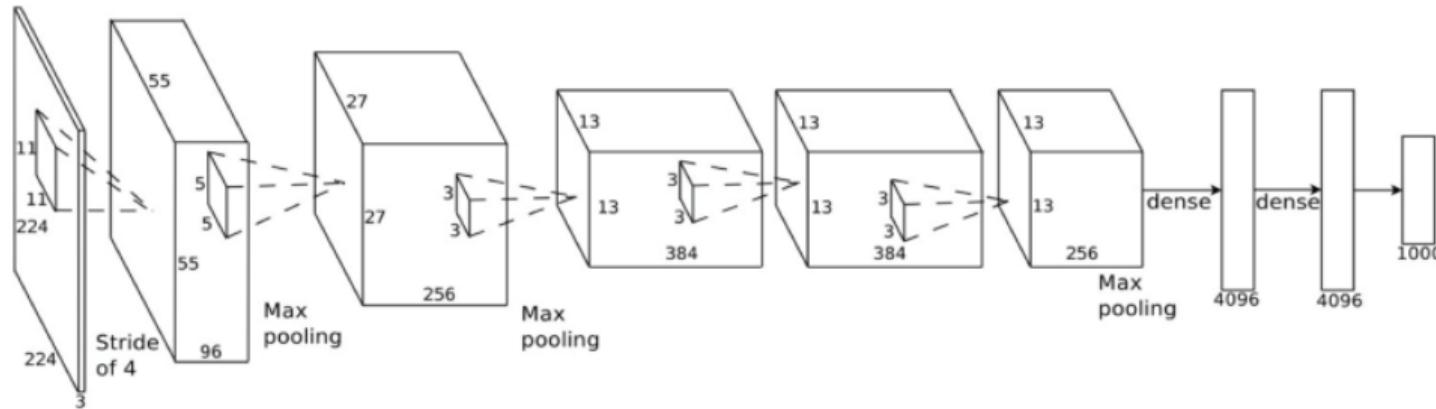
Very deep convolutional networks for large-scale image recognition

VGG: Testing image size

- At **testing** time, **five** strategies for image scaling are evaluated:
 - **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.
 - **Multi-crop**: multiple, random crops are taken from the fully-convolutional output for average pooling.

VGG: Fully-convolutionalization

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



Very deep convolutional networks for large-scale image recognition

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

Very deep convolutional networks for large-scale image recognition

VGG: Results

- 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
	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
C	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
D	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
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

VGG: Us versus Them

- 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

- 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**.

Deep Residual Networks: Deeper and Deeper

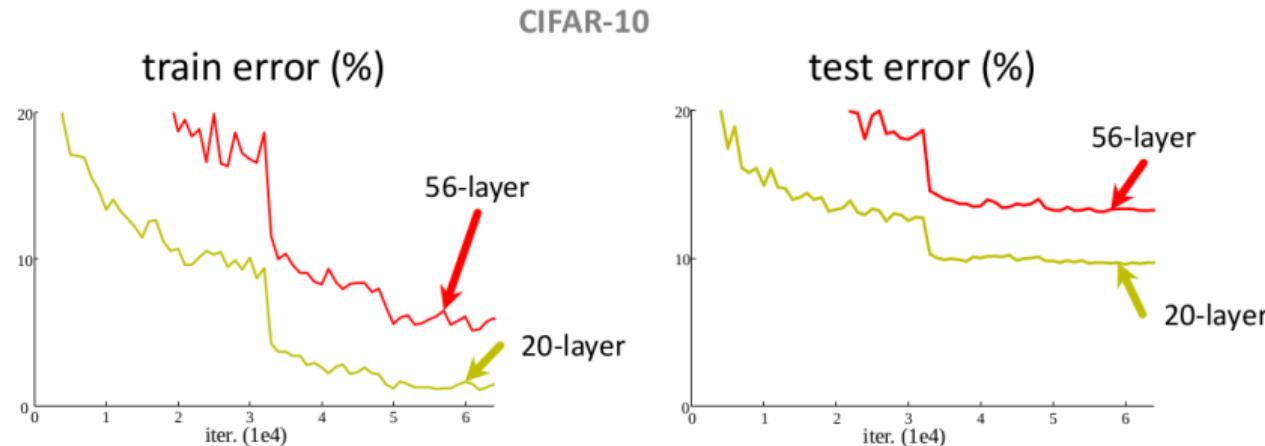
ResNets: Introduction

- From what we have seen so far, it seems like **deeper** networks **generalize** better.
- So, can we just keep **stacking** more and more layers onto the end of our CNNs?
- Aside from the **computational** complications (GPU memory is **finite**), this seems like it should “just work”.
- We will now look at our last state-of-the-art network architecture (known as **ResNet**) which looks at this question in detail [[He et al., 2016](#)].

Deep residual learning for image recognition

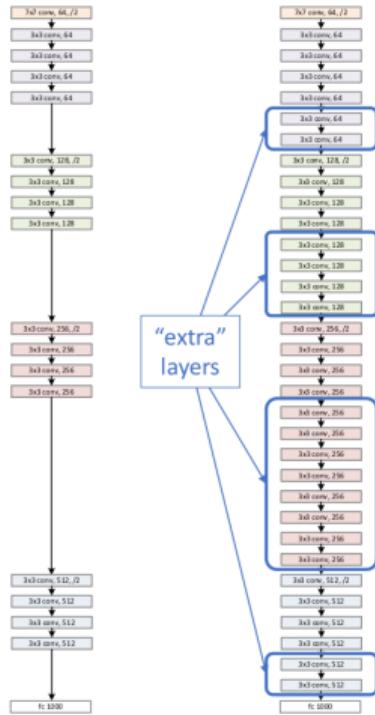
ResNets: Deeper isn't better?

- Pre-ResNet Thinking: *Deeper networks should always perform better – at least on the *training* data.*



Deep residual learning for image recognition

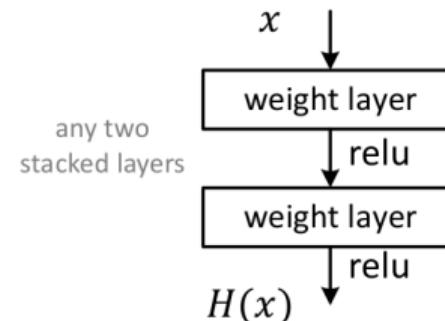
ResNets: Wait, shouldn't training error be lower?



- Using an **artificial** construction, we see that the training error at least shouldn't **increase** with depth.
- Just copy pre-trained weights from **plain** network into a **deeper** network with new, randomly initialized weights.

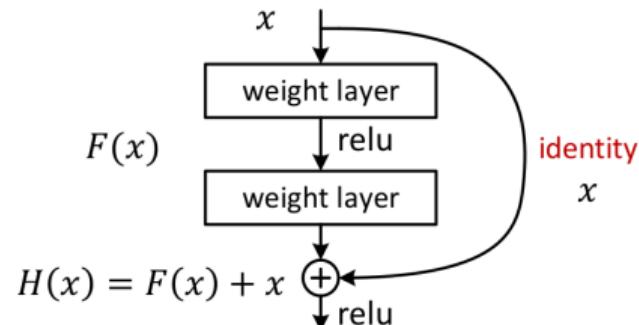
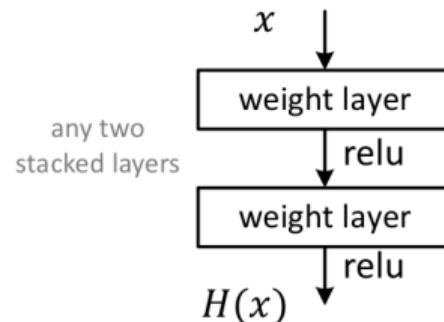
ResNets: Targets and Residuals

- Let's say that the network is **learning** towards some optimal feature representation $H(x)$.
- The **compositional** and **feed-forward** nature of the CNN isn't really helping.



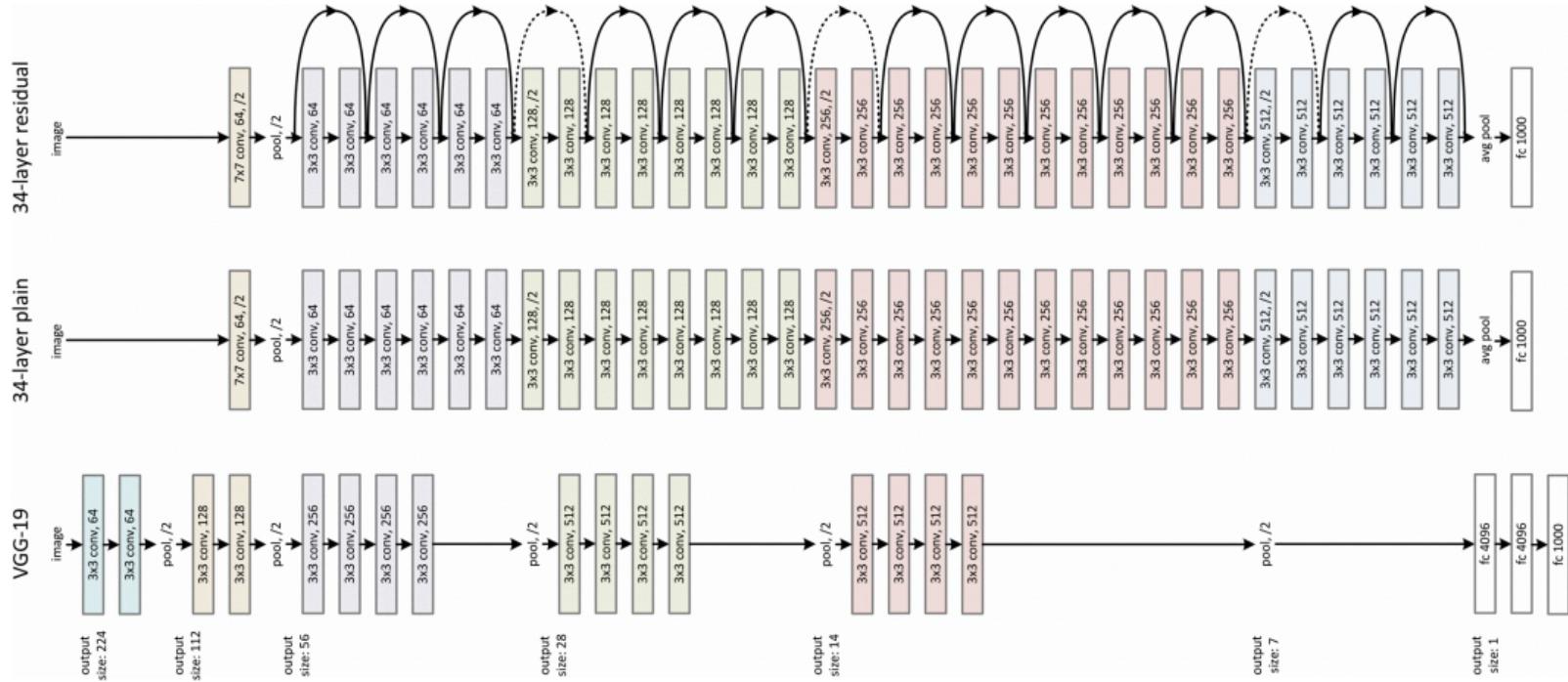
ResNets: Targets and Residuals

- Let's say that the network is **learning** towards some optimal feature representation $H(x)$.
- The **compositional** and **feed-forward** nature of the CNN isn't really helping.
- Instead, we can **help** the network out by not requiring it to **pass through** as much information.
- Pass x forward and **add** it to the output of the **residual block** – now we “only” need to learn $H(x) - x$.



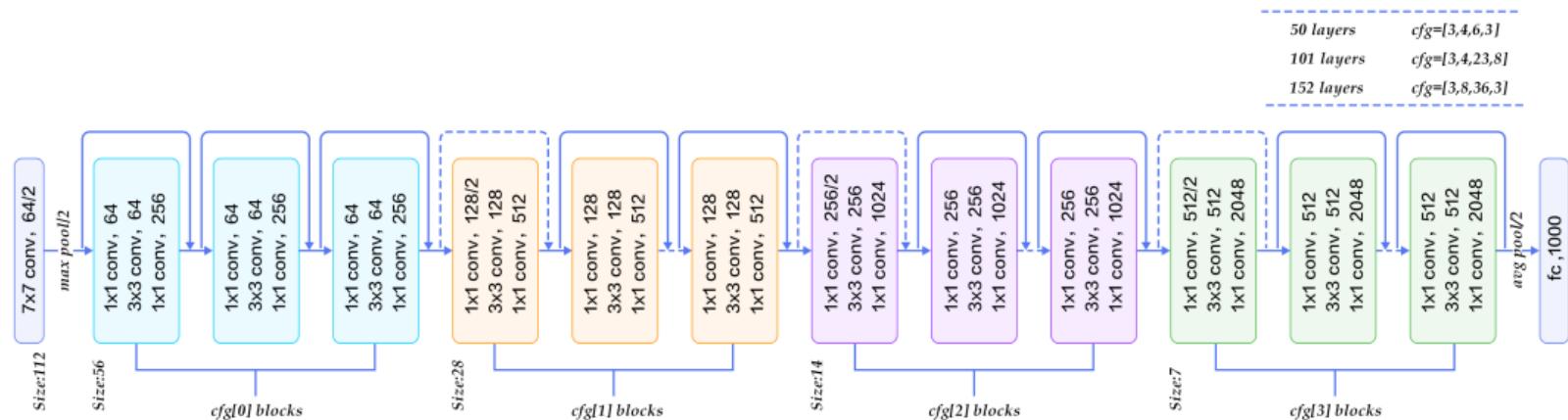
ResNets: Comparison

- Here is a comparison of VGG19 and ResNet-34:



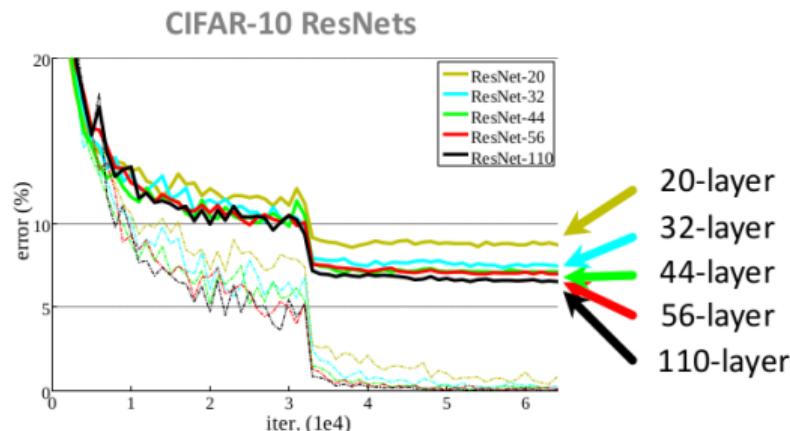
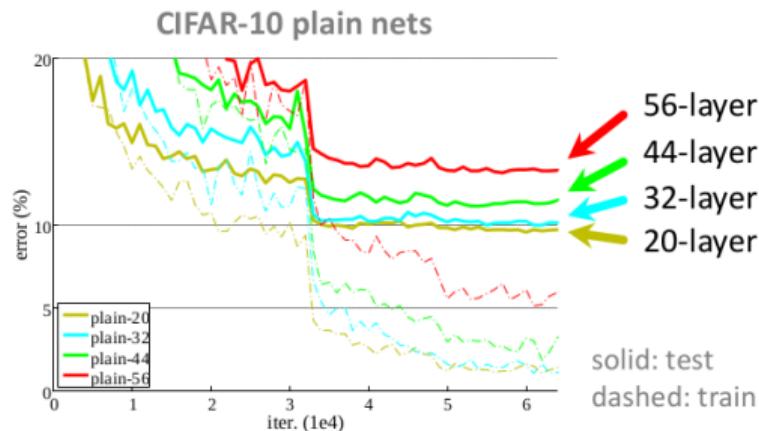
ResNets: Parametric Modularity

- And this is a common way of parametrically representing the various ResNet configurations:



ResNets: Results

- Residual connections are **extremely effective**:

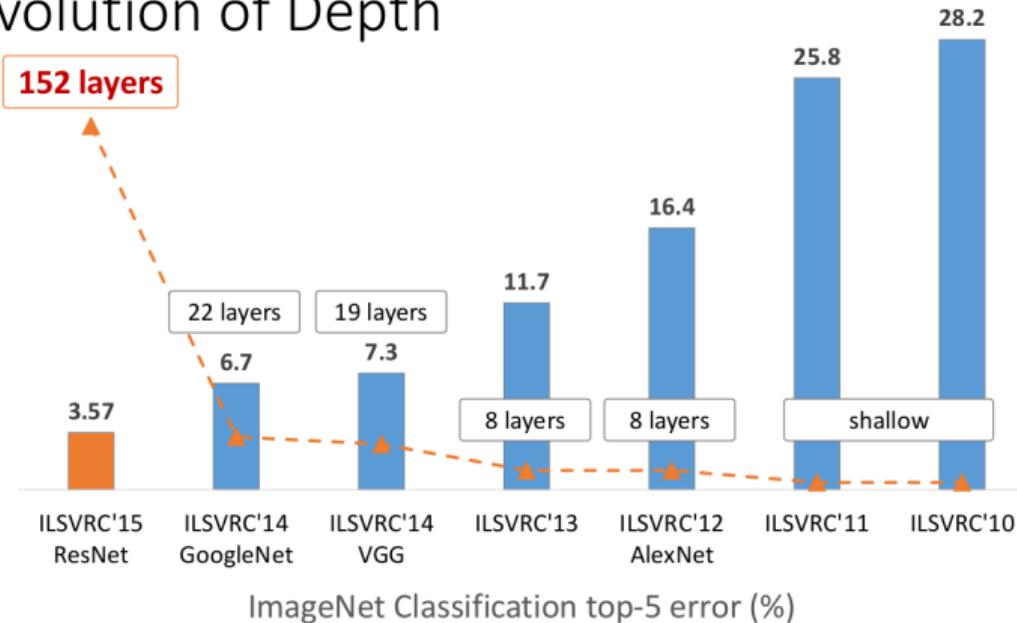


Deep residual learning for image recognition

ResNet: Results

- And the proof, as always, is in the *pudding*:

Revolution of Depth



CNNs: Tricks of the Trade

CNNs: How do you train CNNs?

- CNNs work great, but it's not always **sunshine and lollipops** trying to get them to work.
- The community has developed a number of tricks, techniques, and heuristics that are **proven** to help.
- Let's look at a few of them.

CNNs: Batch Normalization

- Attention to the **data distribution** (through the whole network) and **normalization** are critical [Ioffe and Szegedy, 2015]:

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1 \dots m\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{mini-batch mean}$$
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{mini-batch variance}$$
$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{normalize}$$
$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{scale and shift}$$

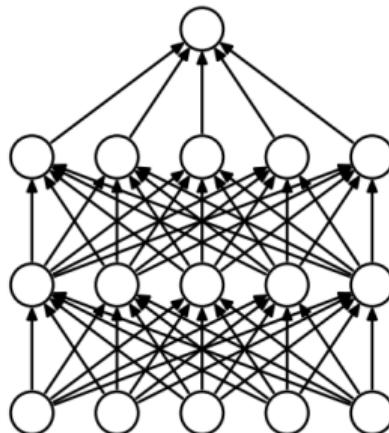
Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Input: Network N with trainable parameters Θ ;
subset of activations $\{x^{(k)}\}_{k=1}^K$
Output: Batch-normalized network for inference, $N_{\text{BN}}^{\text{inf}}$

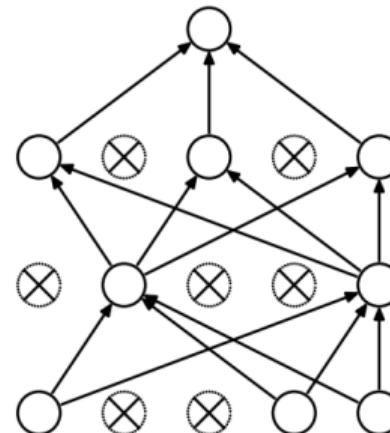
- 1: $N_{\text{BN}}^{\text{tr}} \leftarrow N \quad // \text{Training BN network}$
- 2: **for** $k = 1 \dots K$ **do**
- 3: Add transformation $y^{(k)} = \text{BN}_{\gamma^{(k)}, \beta^{(k)}}(x^{(k)})$ to $N_{\text{BN}}^{\text{tr}}$ (Alg. 1)
- 4: Modify each layer in $N_{\text{BN}}^{\text{tr}}$ with input $x^{(k)}$ to take $y^{(k)}$ instead
- 5: **end for**
- 6: Train $N_{\text{BN}}^{\text{tr}}$ to optimize the parameters $\Theta \cup \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^K$
- 7: $N_{\text{BN}}^{\text{inf}} \leftarrow N_{\text{BN}}^{\text{tr}} \quad // \text{Inference BN network with frozen parameters}$
- 8: **for** $k = 1 \dots K$ **do**
- 9: // For clarity, $x \equiv x^{(k)}, \gamma \equiv \gamma^{(k)}, \mu_{\mathcal{B}} \equiv \mu_{\mathcal{B}}^{(k)}$, etc.
- 10: Process multiple training mini-batches \mathcal{B} , each of size m , and average over them:
$$\text{E}[x] \leftarrow \text{E}_{\mathcal{B}}[\mu_{\mathcal{B}}]$$
$$\text{Var}[x] \leftarrow \frac{m}{m-1} \text{E}_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$$
- 11: In $N_{\text{BN}}^{\text{inf}}$, replace the transform $y = \text{BN}_{\gamma, \beta}(x)$ with
$$y = \frac{\gamma}{\sqrt{\text{Var}[x]+\epsilon}} \cdot x + \left(\beta - \frac{\gamma \text{E}[x]}{\sqrt{\text{Var}[x]+\epsilon}}\right)$$
- 12: **end for**

Algorithm 2: Training a Batch-Normalized Network

CNNs: Dropout (another “crazy” idea)



(a) Standard Neural Net



(b) After applying dropout.

With unlimited computation, the best way to “regularize” a fixed-sized model is to average the predictions of all possible settings of the parameters, weighting each setting by its posterior probability given the training data.

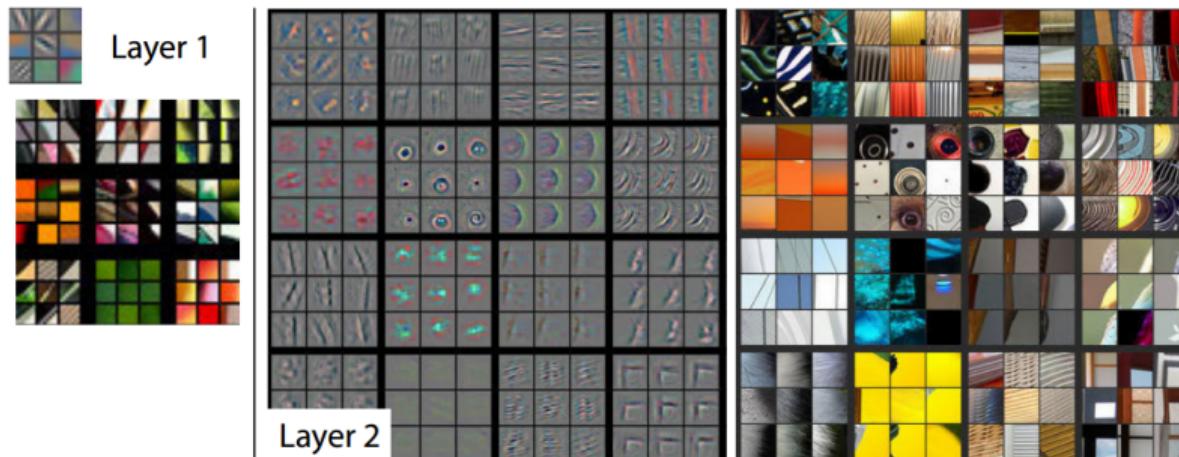
– Srivastava et al. [2014]

CNNs: What's Going On?

- Remember earlier I mentioned that one of the **biases** against using neural networks was that lack of **interpretability**.
- As soon as the spectacular results of CNNs on object recognition started coming in, researchers began inventing new ways to **interpret** the innards of these **huge** networks.
- Nowadays there are many **tools** and **tricks** you can use to understand what the network has learned.

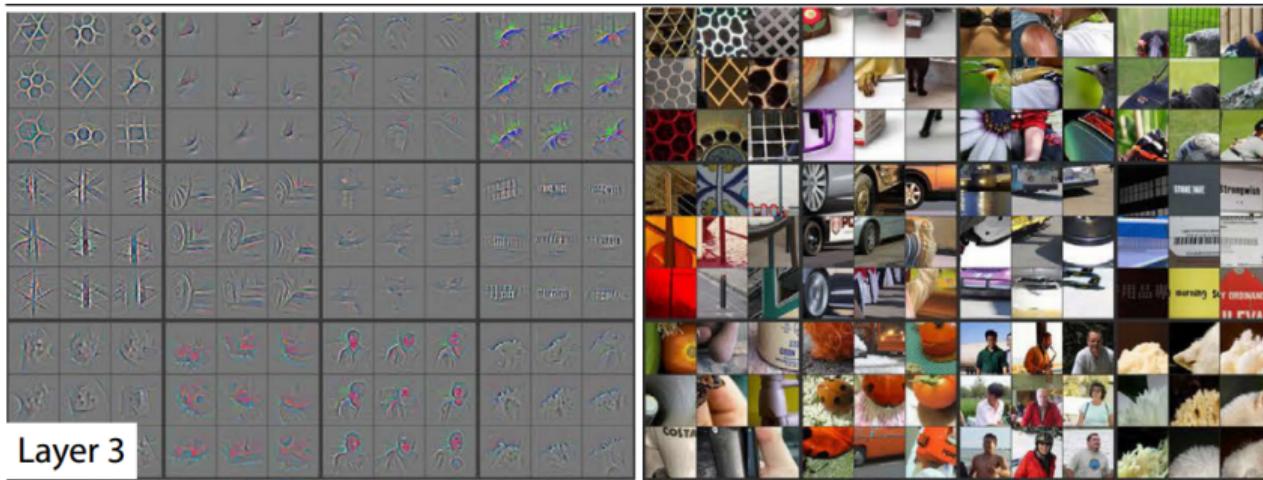
CNNs: What's Going On

- An early work looked at just this problem and the paper has a **ton** of interesting analysis of how these networks work [[Zeiler and Fergus, 2014](#)].
- I am only going to talk about how **visualizations** of feature map activations demonstrate **what's going on**.



CNNs: What's Going On

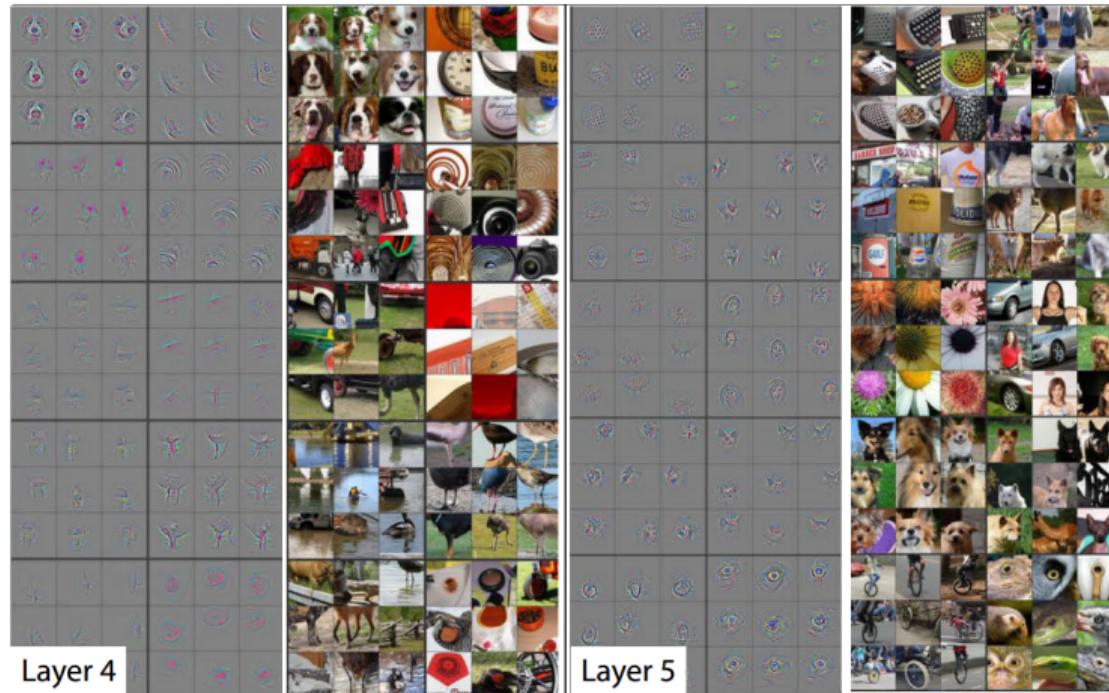
- As we go **deeper** into the network, feature activations correspond to **higher-level semantics**.



Visualizing and understanding convolutional networks

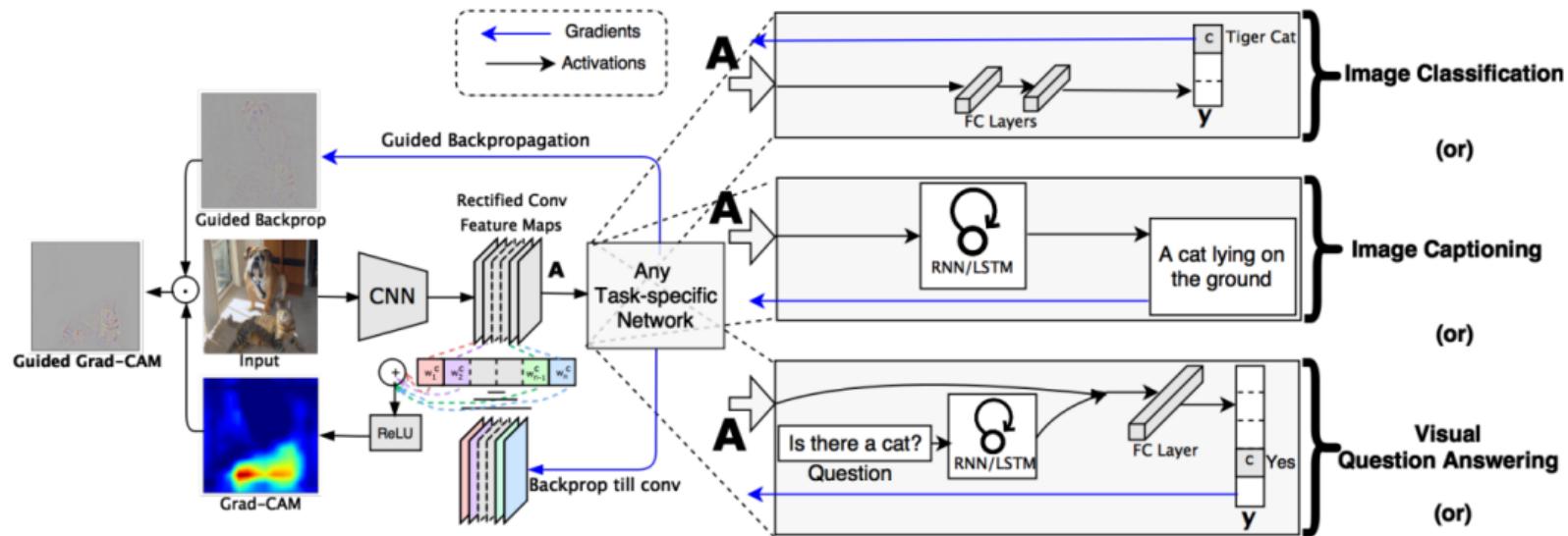
CNNs: What's Going On

- Until features are really indicating the presence of “eyes” and “cat faces”, etc.



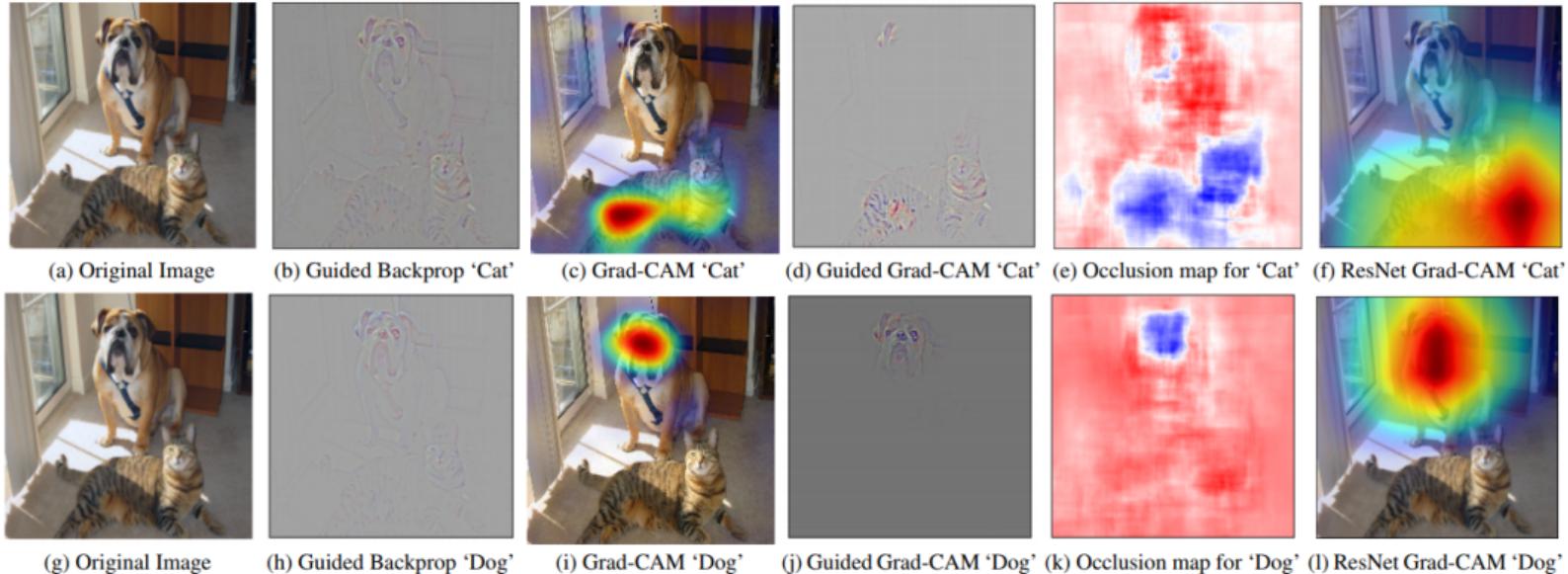
CNNs: What's Going On

- The **Grad-CAM** algorithm is an intuitive way to visualize, well, **what makes a cow, a cow** [Selvaraju et al., 2017]:



CNNs: What's Going On

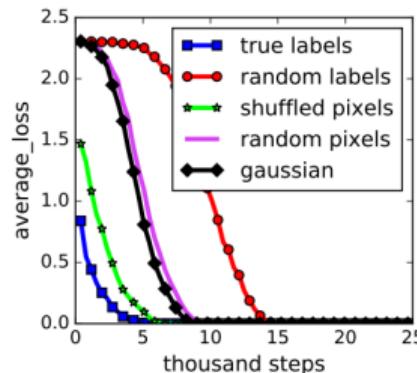
- The **Grad-CAM** algorithm is an intuitive way to visualize, well, **what makes a cow, a cow** [Selvaraju et al., 2017]:



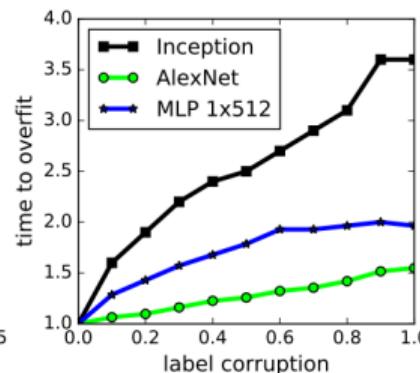
Discussion

Discussion: The burden of supervision

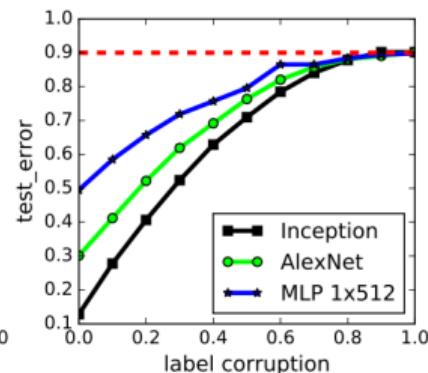
- What is 1.5M annotations really worth [Zhang et al., 2016]?
- If labels are equally probable, a randomly shuffled set of ImageNet labels contains about $1.5M * \log_2(1000) \approx 15\text{Mbits}$.



(a) learning curves



(b) convergence slowdown



(c) generalization error growth

Understanding deep learning requires rethinking generalization

Discussion: The state-of-the-art

- We have come a **long way** in a few years.

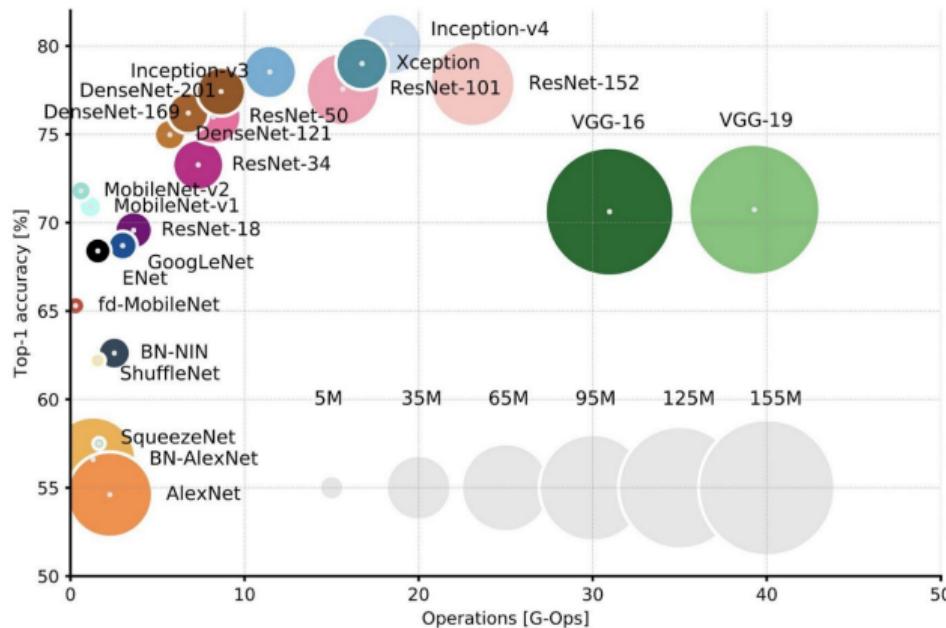


Figure from: [Canziani et al., 2016]

Discussion: CNNs and the way forward



References

- A. Canziani, A. Paszke, and E. Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016.
- M. A. Fischler and R. A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on computers*, (1):67–92, 1973.
- K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Bibliography ii

- D. Marr. Vision: A computational investigation into the human representation and processing of visual information, 1982.
- L. G. Roberts. *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology, 1963.
- R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 618–626, 2017.
- K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- A. W. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (12):1349–1380, 2000.
- N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL <http://jmlr.org/papers/v15/srivastava14a.html>.

Bibliography iii

- M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.
- C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.