

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
Ilya Sutskever
Geoffrey Hinton

University of Toronto
Canada

Paper with same name to appear in NIPS 2012



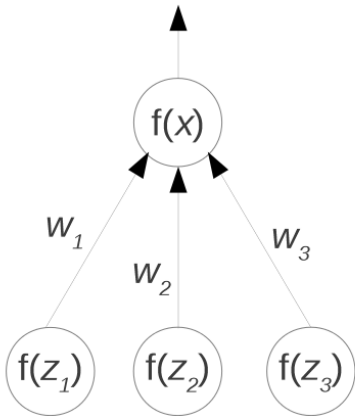
Main idea

Architecture

Technical details

Neural networks

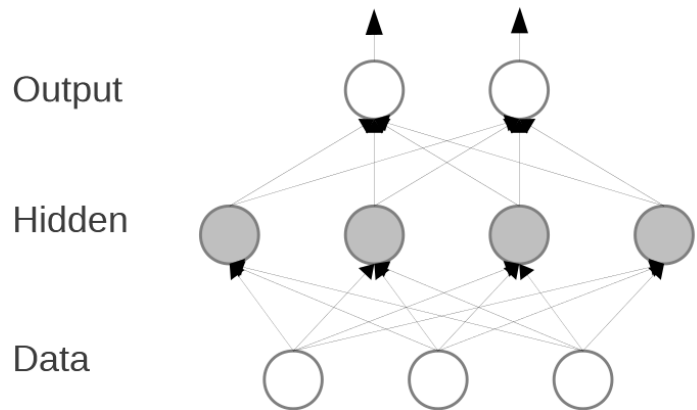
- A neuron



$$x = w_1 f(z_1) + w_2 f(z_2) + w_3 f(z_3)$$

x is called the total input to the neuron, and $f(x)$ is its output

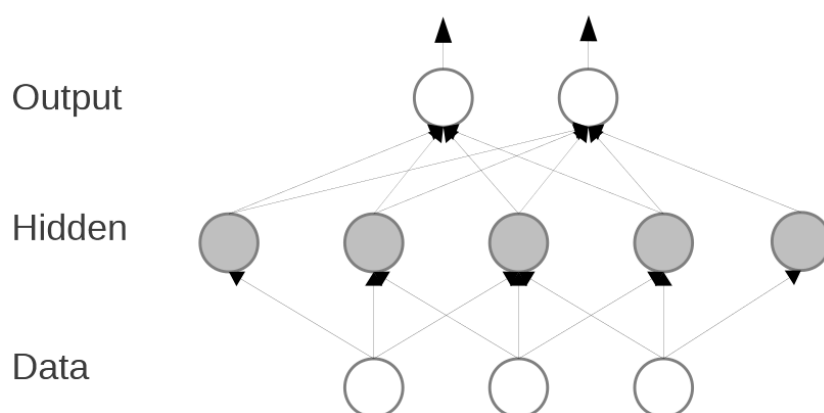
- A neural network



A neural network computes a differentiable function of its input. For example, ours computes: $p(\text{label} \mid \text{an input image})$

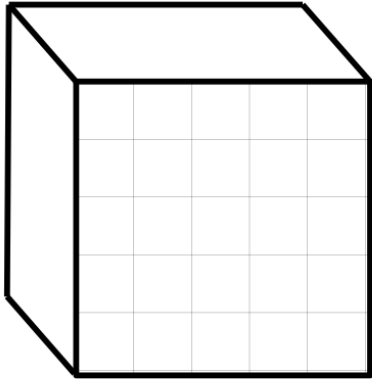
Convolutional neural networks

- Here's a one-dimensional convolutional neural network
- Each hidden neuron applies **the same localized, linear filter** to the input

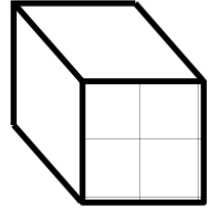
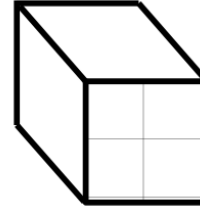
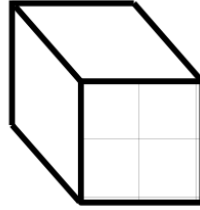
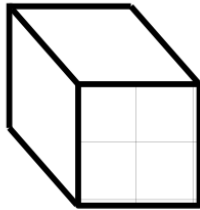


Convolution in 2D

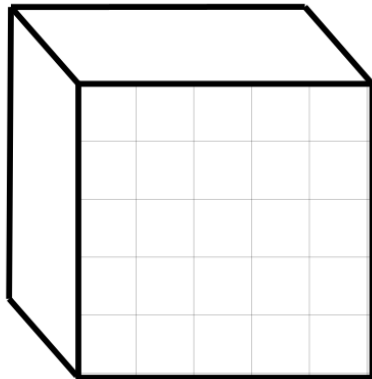
Input "image"



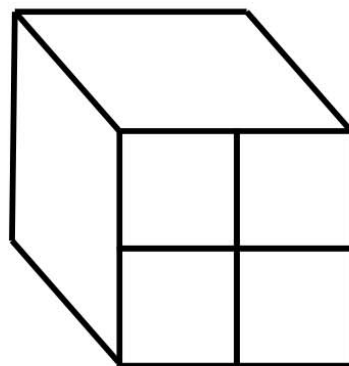
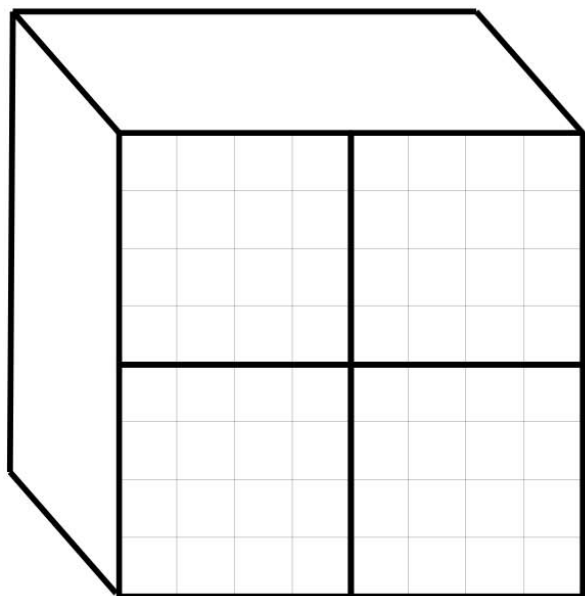
Filter bank



Output map

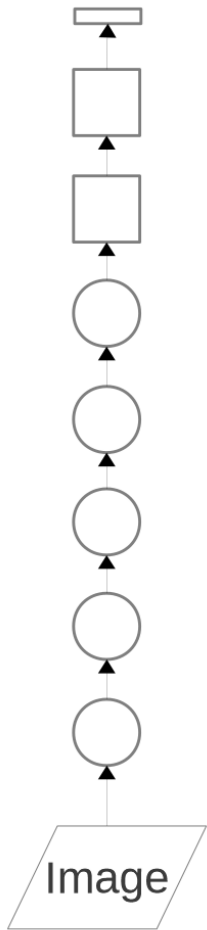


Local pooling



Overview of our model

- **Deep:** 7 hidden “weight” layers
- **Learned:** all feature extractors initialized at white Gaussian noise and learned from the data
- Entirely supervised
- **More data = good**



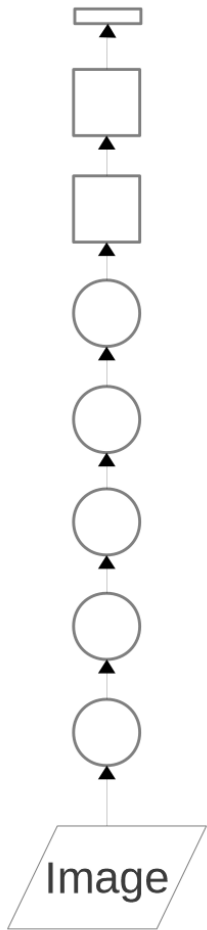
Convolutional layer: convolves its input with a bank of 3D filters, then applies point-wise non-linearity



Fully-connected layer: applies linear filters to its input, then applies point-wise non-linearity

Overview of our model

- Trained with stochastic gradient descent on two NVIDIA GPUs for about a week
- 650,000 neurons
- 60,000,000 parameters
- 630,000,000 connections
- **Final feature layer: 4096-dimensional**

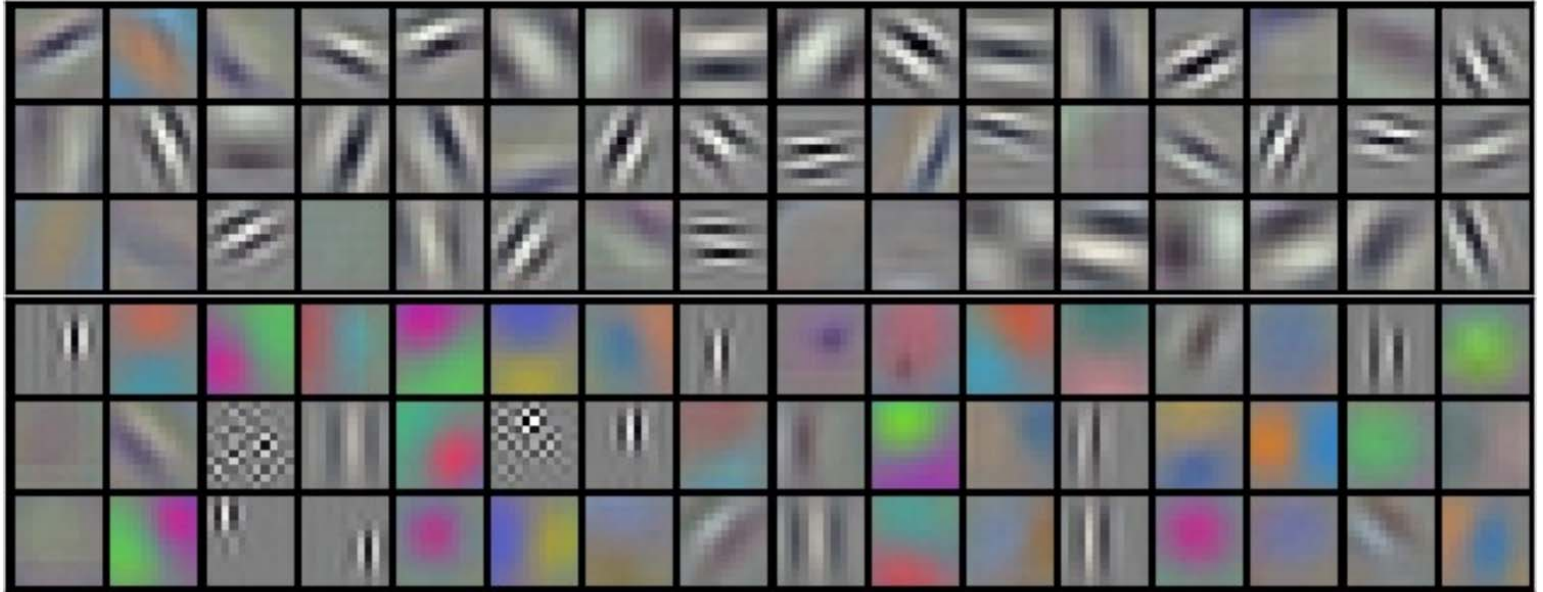


Convolutional layer: convolves its input with a bank of 3D filters, then applies point-wise non-linearity



Fully-connected layer: applies linear filters to its input, then applies point-wise non-linearity

96 learned low-level filters



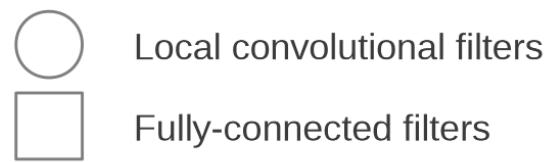
Main idea



Architecture

Technical details

Training



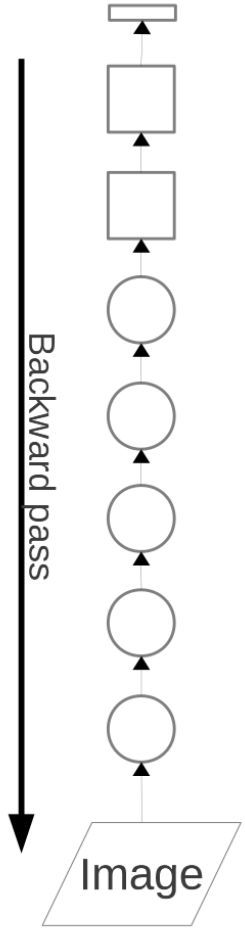
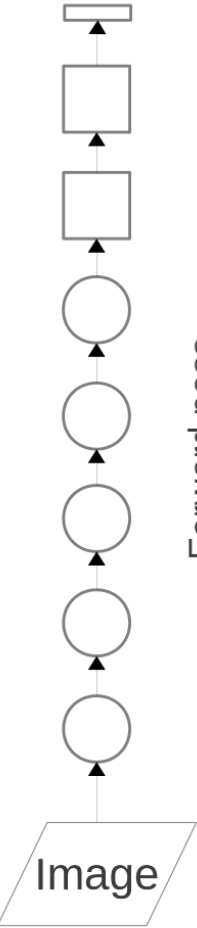
Using stochastic gradient descent and the *backpropagation algorithm* (just repeated application of the chain rule)

Forward pass

One output unit per class
 $x_i =$ total input to output unit i
$$f(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{1000} \exp(x_j)}$$

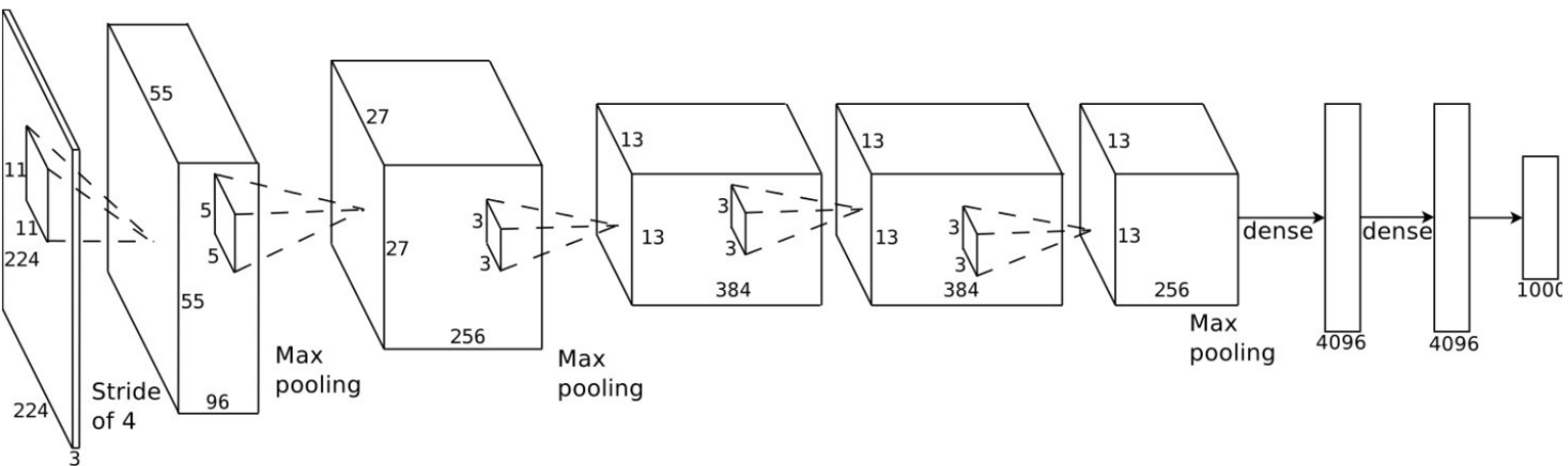
We maximize the log-probability of the correct label, $\log f(x_t)$

Backward pass



Our model

- Max-pooling layers follow first, second, and fifth convolutional layers
- The number of neurons in each layer is given by 253440, 186624, 64896, 64896, 43264, 4096, 4096, 1000



Main idea
Architecture



Technical details

Input representation

- Centered (0-mean) RGB values.



An input image (256x256)



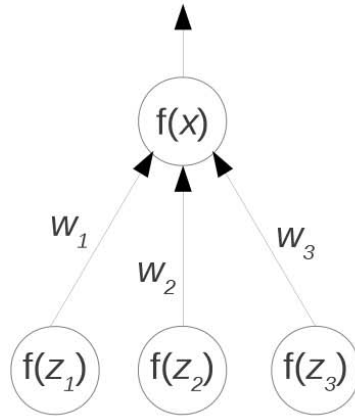
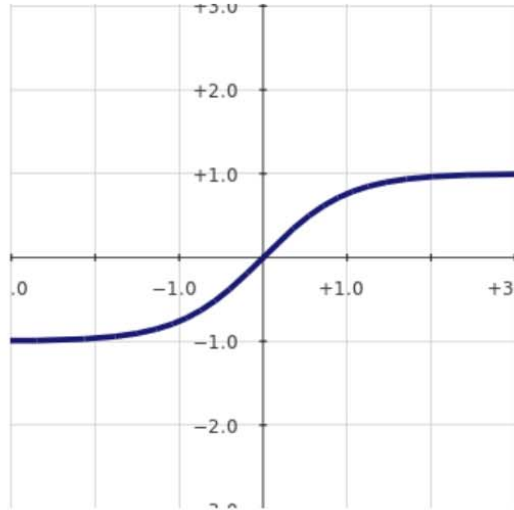
Minus sign



The mean input image

Neurons

$$f(x) = \tanh(x)$$



$$x = w_1 f(z_1) + w_2 f(z_2) + w_3 f(z_3)$$

x is called the total input to the neuron, and $f(x)$ is its output

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



Very bad (slow to train)

Very good (quick to train)

Data augmentation

- Our neural net has 60M real-valued parameters and 650,000 neurons
- It overfits a lot. Therefore we train on 224x224 patches extracted randomly from 256x256 images, and also their horizontal reflections.

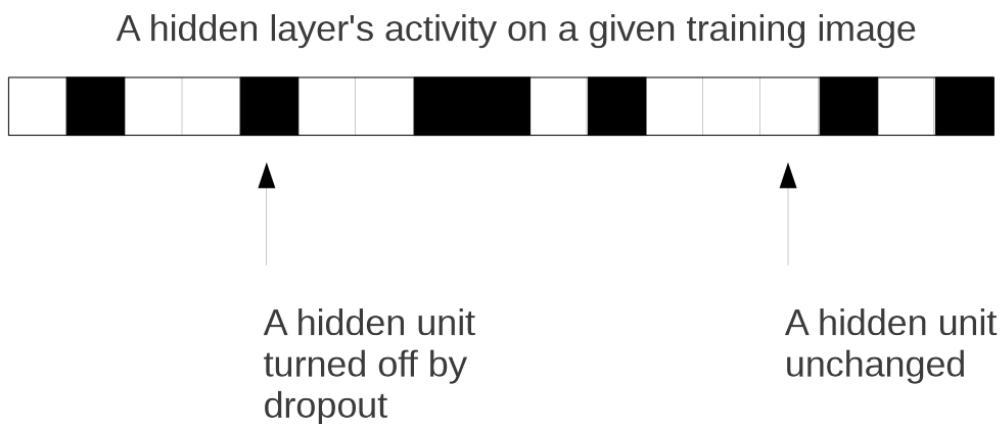


Testing

- Average predictions made at five 224x224 patches and their horizontal reflections (four corner patches and center patch)
- Logistic regression has the nice property that it outputs a probability distribution over the class labels
- Therefore no score normalization or calibration is necessary to combine the predictions of different models (or the same model on different patches), as would be necessary with an SVM.

Dropout

- Independently set each hidden unit activity to zero with 0.5 probability
- We do this in the two globally-connected hidden layers at the net's output



Implementation

- The only thing that needs to be stored on disk is the raw image data
- We stored it in JPEG format. It can be loaded and decoded entirely in parallel with training.
- Therefore only 27GB of disk storage is needed to train this system.
- Uses about 2GB of RAM on each GPU, and around 5GB of system memory during training.

Implementation

- Written in Python/C++/CUDA
- Sort of like an instruction pipeline, with the following 4 instructions happening in parallel:
 - Train on batch n (on GPUs)
 - Copy batch $n+1$ to GPU memory
 - Transform batch $n+2$ (on CPU)
 - Load batch $n+3$ from disk (on CPU)

Validation classification



mite

container ship

motor scooter

leopard

	mite black widow cockroach tick starfish		container ship lifeboat amphibian fireboat drilling platform		motor scooter go-kart moped bumper car golfcart		leopard jaguar cheetah snow leopard Egyptian cat
--	---	--	---	--	--	--	---



grille









mushroom

cherry

Madagascar cat

	convertible grille pickup beach wagon fire engine		agaric mushroom jelly fungus gill fungus dead-man's-fingers		dalmatian grape elderberry ffordshire bullterrier currant		squirrel monkey spider monkey titi indri howler monkey
--	--	--	--	--	--	--	---




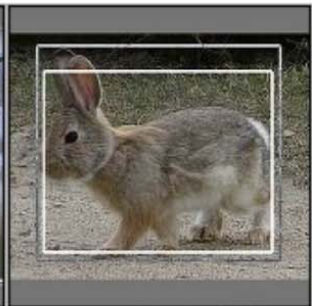




Validation classification

																							
lens cap	abacus	slug	hen																				
<table border="1"> <tr><td>reflex camera</td></tr> <tr><td>Polaroid camera</td></tr> <tr><td>pencil sharpener</td></tr> <tr><td>switch</td></tr> <tr><td>combination lock</td></tr> </table>	reflex camera	Polaroid camera	pencil sharpener	switch	combination lock	<table border="1"> <tr><td>abacus</td></tr> <tr><td>typewriter keyboard</td></tr> <tr><td>space bar</td></tr> <tr><td>computer keyboard</td></tr> <tr><td>accordion</td></tr> </table>	abacus	typewriter keyboard	space bar	computer keyboard	accordion	<table border="1"> <tr><td>slug</td></tr> <tr><td>zucchini</td></tr> <tr><td>ground beetle</td></tr> <tr><td>common newt</td></tr> <tr><td>water snake</td></tr> </table>	slug	zucchini	ground beetle	common newt	water snake	<table border="1"> <tr><td>hen</td></tr> <tr><td>cock</td></tr> <tr><td>cocker spaniel</td></tr> <tr><td>partridge</td></tr> <tr><td>English setter</td></tr> </table>	hen	cock	cocker spaniel	partridge	English setter
reflex camera																							
Polaroid camera																							
pencil sharpener																							
switch																							
combination lock																							
abacus																							
typewriter keyboard																							
space bar																							
computer keyboard																							
accordion																							
slug																							
zucchini																							
ground beetle																							
common newt																							
water snake																							
hen																							
cock																							
cocker spaniel																							
partridge																							
English setter																							
																							
tiger	chambered nautilus	tape player	planetarium																				
<table border="1"> <tr><td>tiger</td></tr> <tr><td>tiger cat</td></tr> <tr><td>tabby</td></tr> <tr><td>boxer</td></tr> <tr><td>Saint Bernard</td></tr> </table>	tiger	tiger cat	tabby	boxer	Saint Bernard	<table border="1"> <tr><td>lampshade</td></tr> <tr><td>throne</td></tr> <tr><td>goblet</td></tr> <tr><td>table lamp</td></tr> <tr><td>hamper</td></tr> </table>	lampshade	throne	goblet	table lamp	hamper	<table border="1"> <tr><td>cellular telephone</td></tr> <tr><td>slot</td></tr> <tr><td>reflex camera</td></tr> <tr><td>dial telephone</td></tr> <tr><td>iPod</td></tr> </table>	cellular telephone	slot	reflex camera	dial telephone	iPod	<table border="1"> <tr><td>planetarium</td></tr> <tr><td>dome</td></tr> <tr><td>mosque</td></tr> <tr><td>radio telescope</td></tr> <tr><td>steel arch bridge</td></tr> </table>	planetarium	dome	mosque	radio telescope	steel arch bridge
tiger																							
tiger cat																							
tabby																							
boxer																							
Saint Bernard																							
lampshade																							
throne																							
goblet																							
table lamp																							
hamper																							
cellular telephone																							
slot																							
reflex camera																							
dial telephone																							
iPod																							
planetarium																							
dome																							
mosque																							
radio telescope																							
steel arch bridge																							

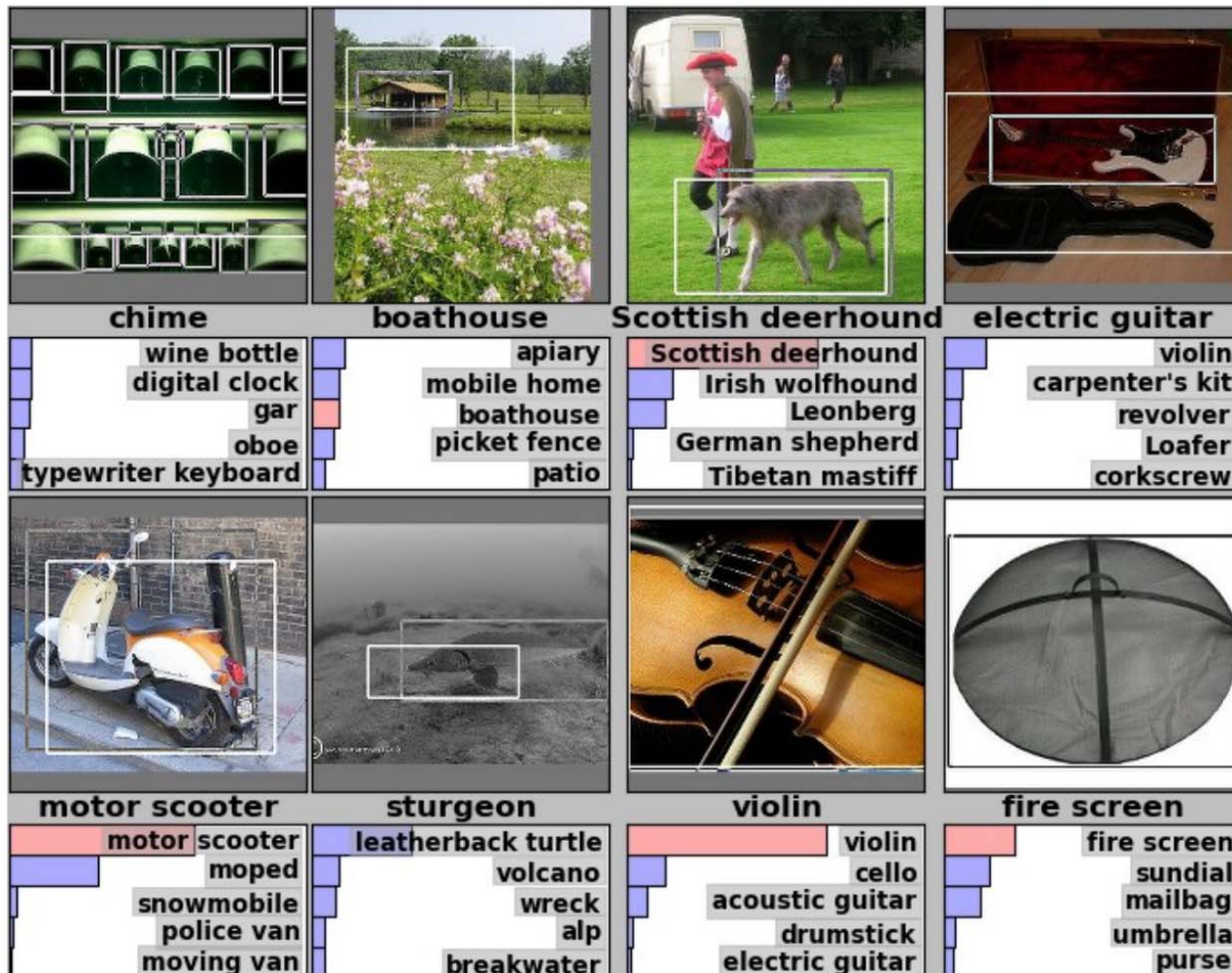
Validation classification



Validation localizations

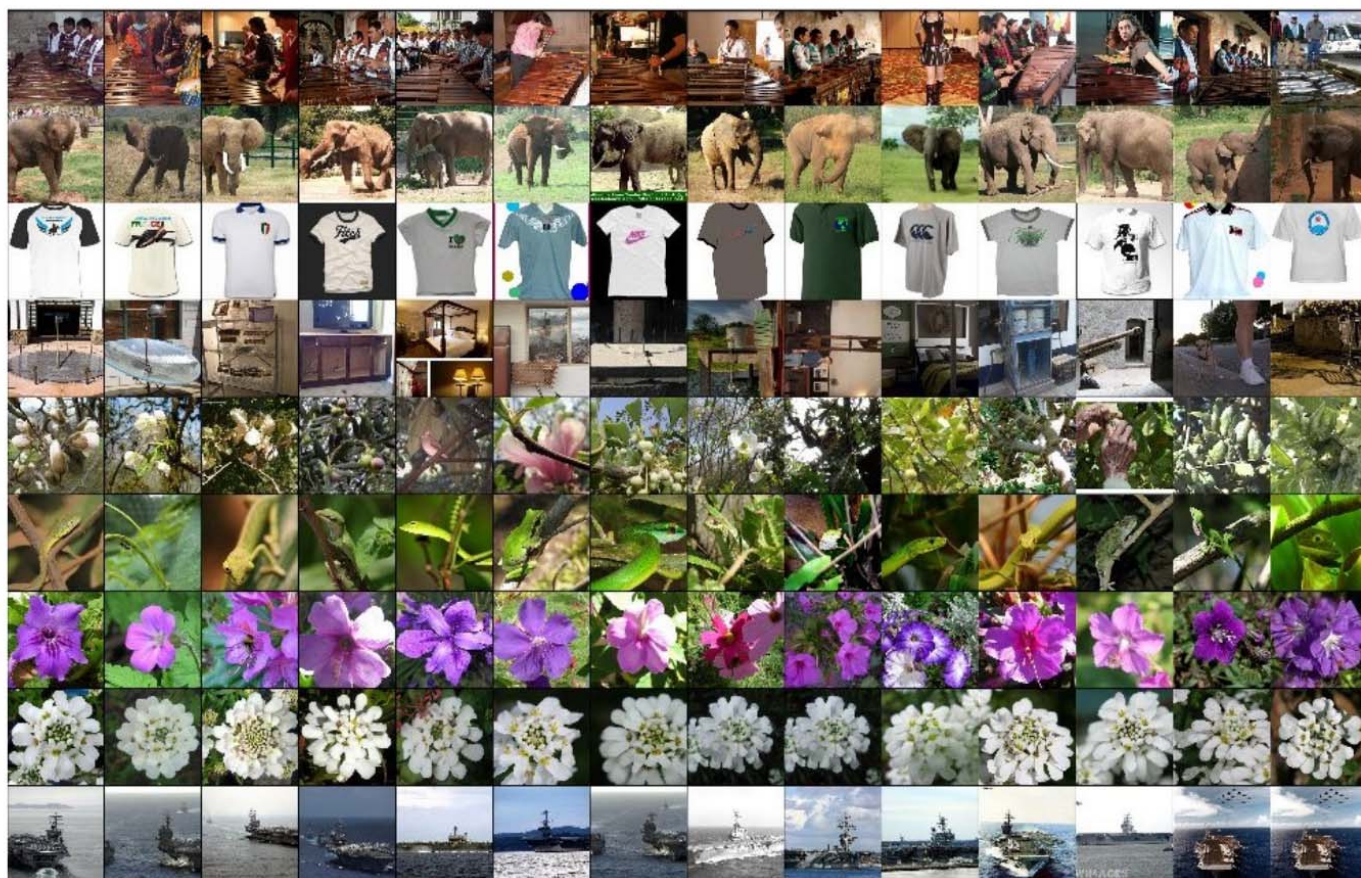
			
bookshop	coyote	cradle	wood rabbit
<ul style="list-style-type: none"> balance beam cinema marimba parallel bars computer keyboard 	<ul style="list-style-type: none"> grey fox kit fox red fox coyote dhole 	<ul style="list-style-type: none"> cradle bassinet diaper crib bath towel 	<ul style="list-style-type: none"> hare wood rabbit grey fox coyote wallaby
			
bottlecap	harvester	garter snake	Walker hound
<ul style="list-style-type: none"> bottlecap magnetic compass puck stopwatch disk brake 	<ul style="list-style-type: none"> harvester thresher plow tractor tow truck 	<ul style="list-style-type: none"> diamondback leatherback turtle sandbar echidna armadillo 	<ul style="list-style-type: none"> beagle Walker hound English foxhound muzzle Italian greyhound

Validation localizations



Retrieval experiments

First column contains query images from ILSVRC-2010 test set, remaining columns contain retrieved images from training set.



Retrieval experiments

