

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
Ilya Sutskever
Geoffrey Hinton

University of Toronto
Canada



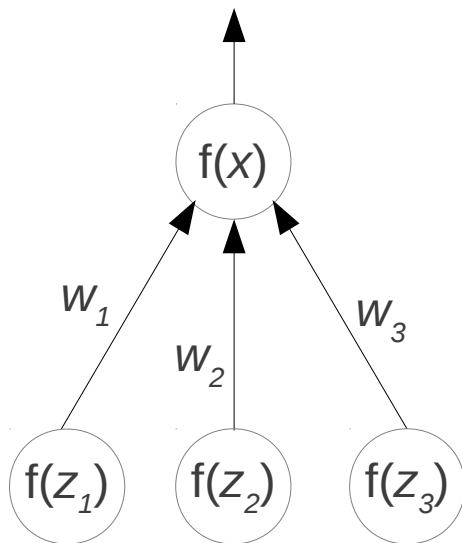
Main idea

Architecture

Technical details

Neural networks

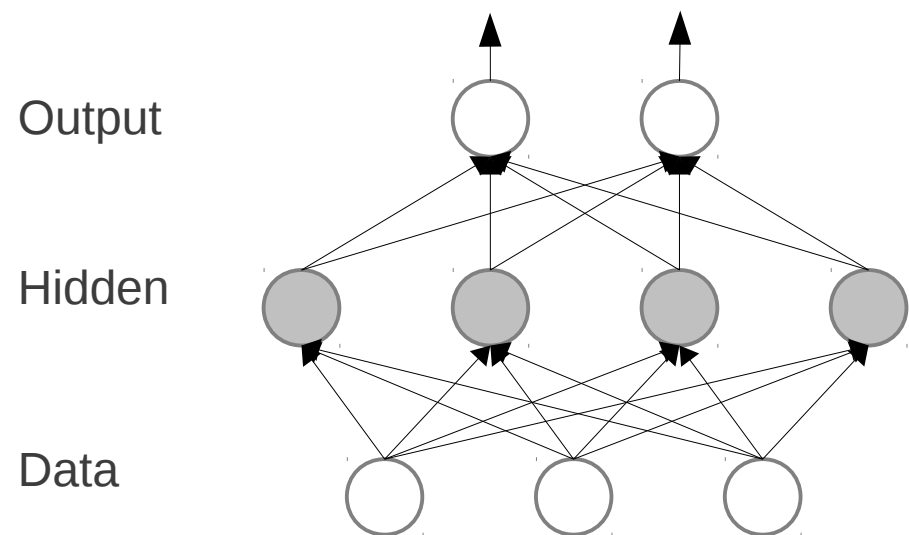
- A neuron



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

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

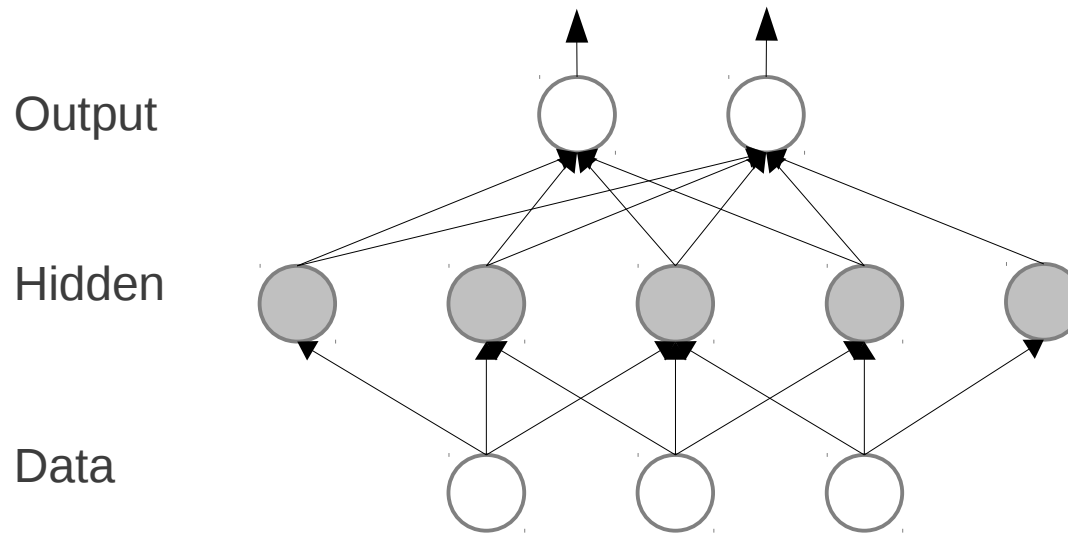
- A neural network



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

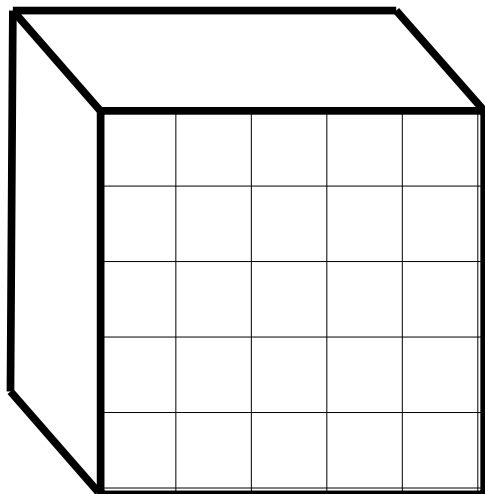
Convolutional neural networks

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

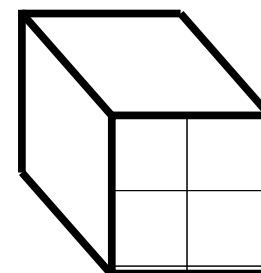
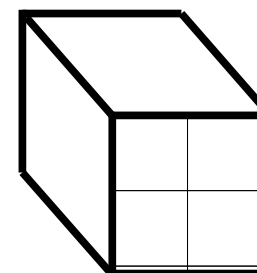
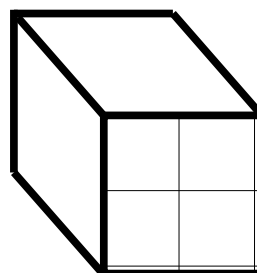
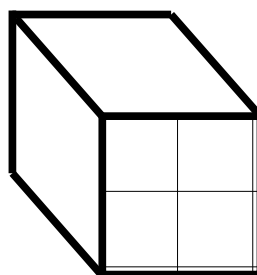


Convolution in 2D

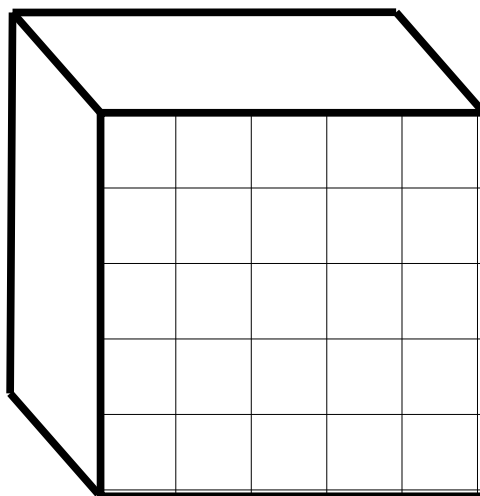
Input “image”



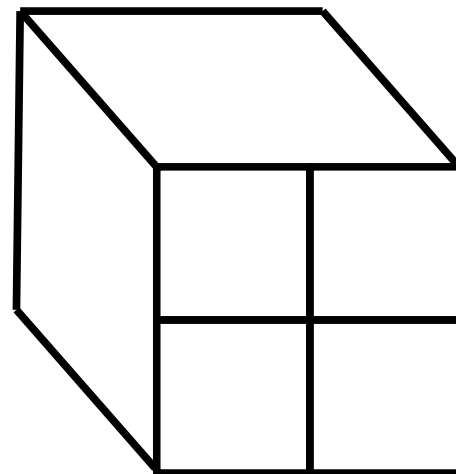
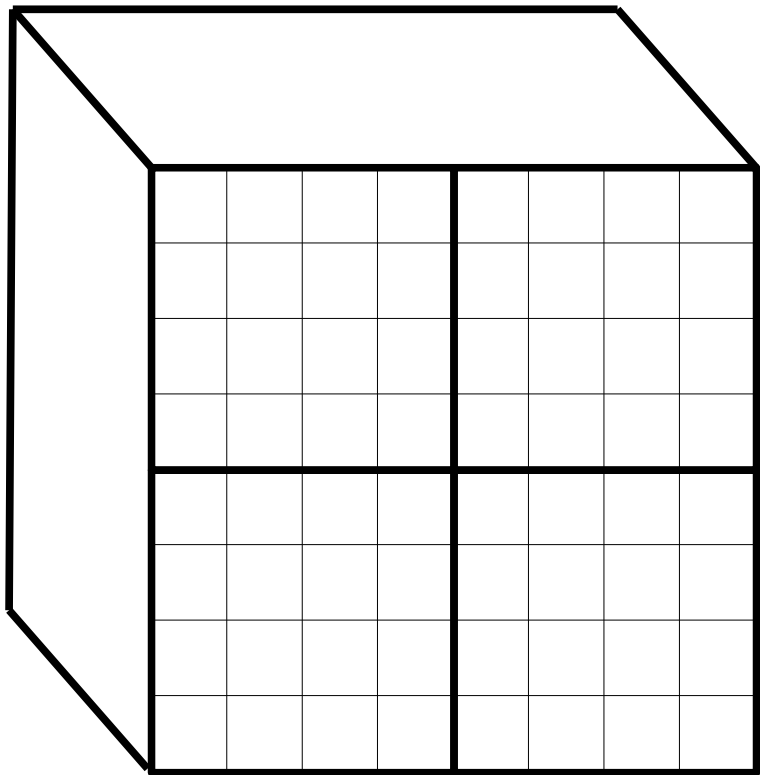
Filter bank



Output map

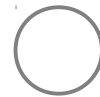
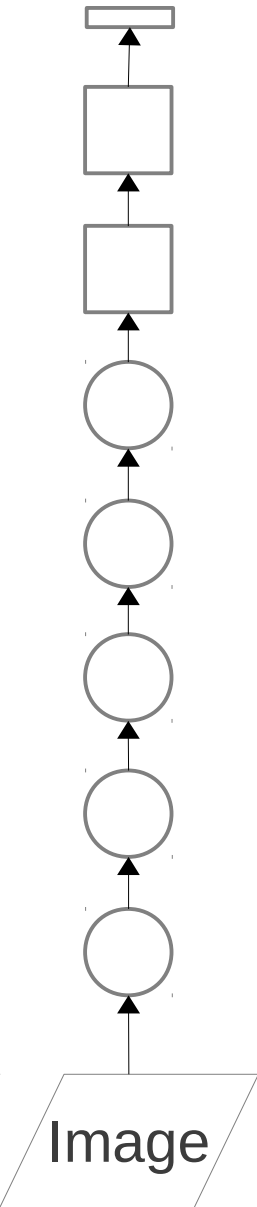


Local pooling



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**



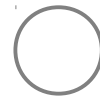
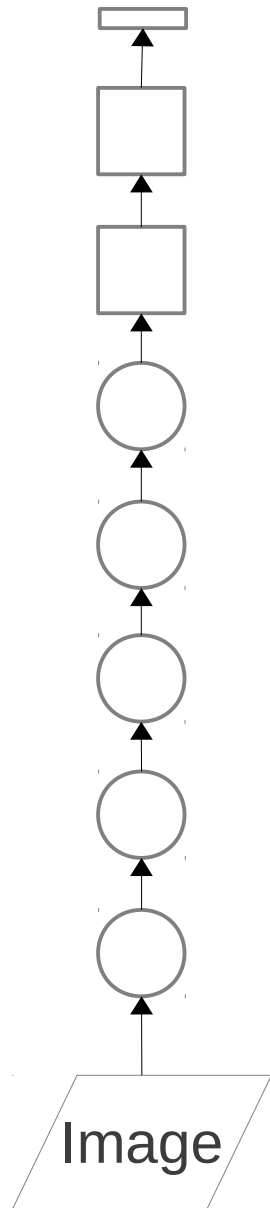
Convolutional layer: convolves its input with a bank of 3D filters, then applies point-wise non-linearity



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



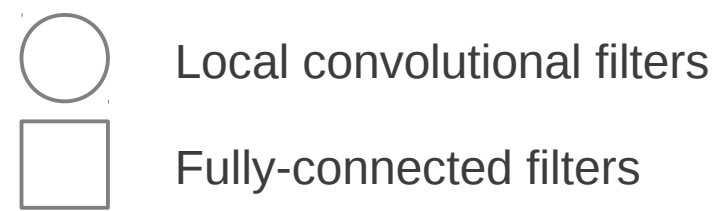
Main idea



Architecture

Technical details

Training



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

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, $\log f(x_t)$

Forward pass

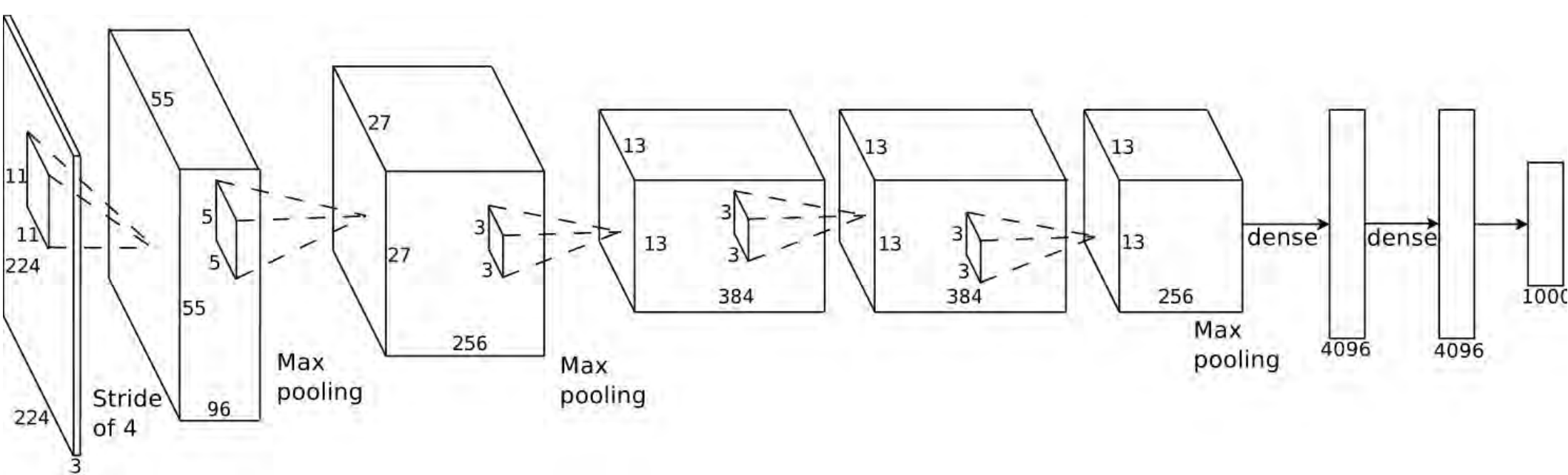
Backward pass

Image

Image

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



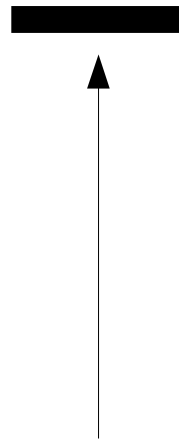
Technical details

Input representation

- Centered (0-mean) RGB values.



An input image (256x256)



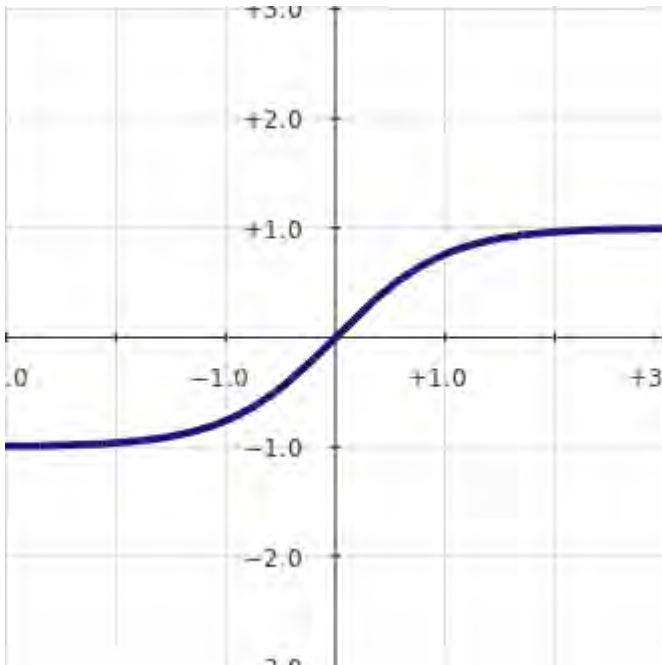
Minus sign



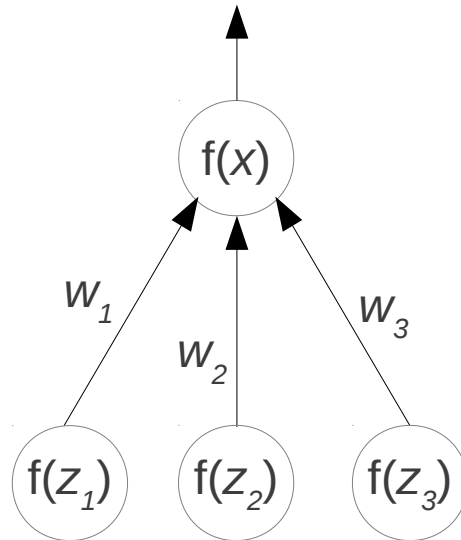
The mean input image

Neurons

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



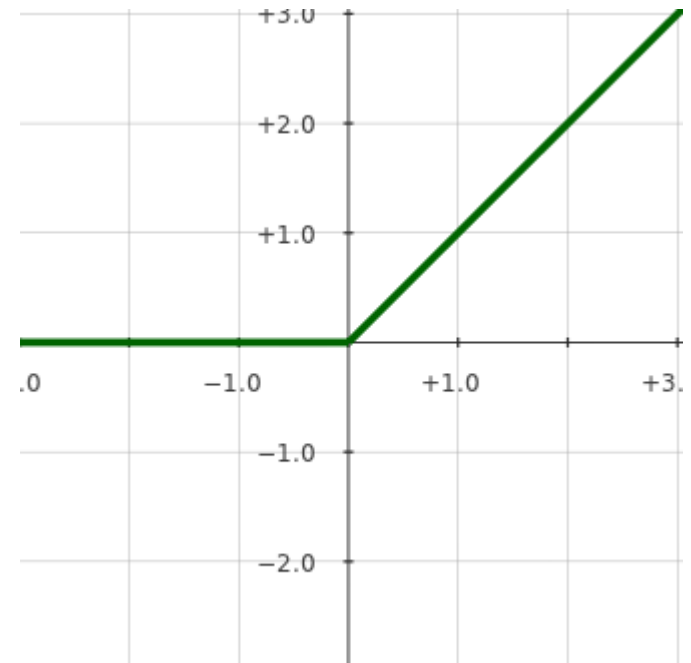
Very bad (slow to train)



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

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

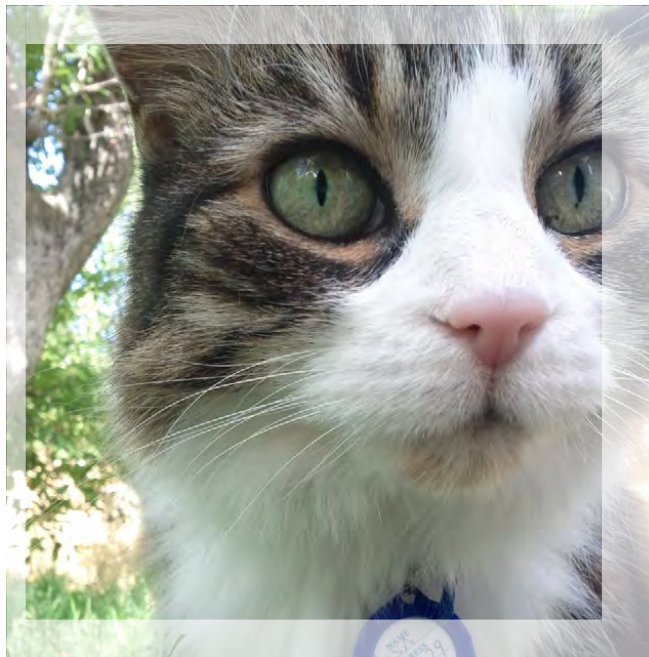
$$f(x) = \max(0, x)$$



Very 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 224×224 patches extracted randomly from 256×256 images, and also their horizontal reflections.

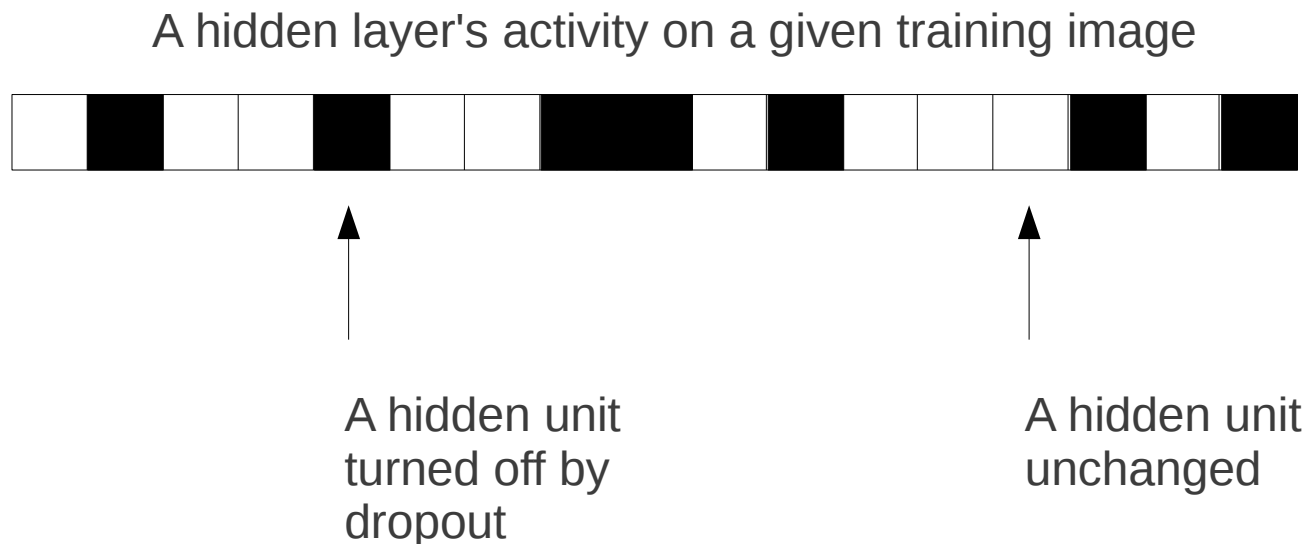


Testing

- 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



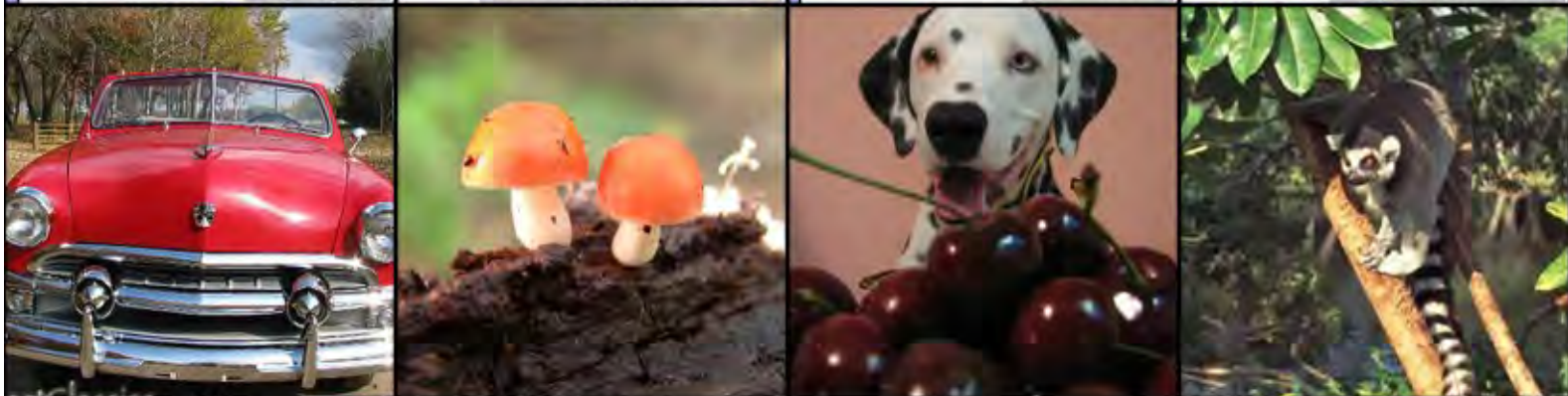
mite

container ship

motor scooter

leopard

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



grille

mushroom

cherry

Madagascar cat

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

Validation classification



lens cap

reflex camera
Polaroid camera
pencil sharpener
switch
combination lock



abacus

abacus
typewriter keyboard
space bar
computer keyboard
accordion



slug

slug
zucchini
ground beetle
common newt
water snake



hen

hen
cock
cocker spaniel
partridge
English setter



tiger

tiger
tiger cat
tabby
boxer
Saint Bernard



chambered nautilus

lampshade
throne
goblet
table lamp
hamper



tape player

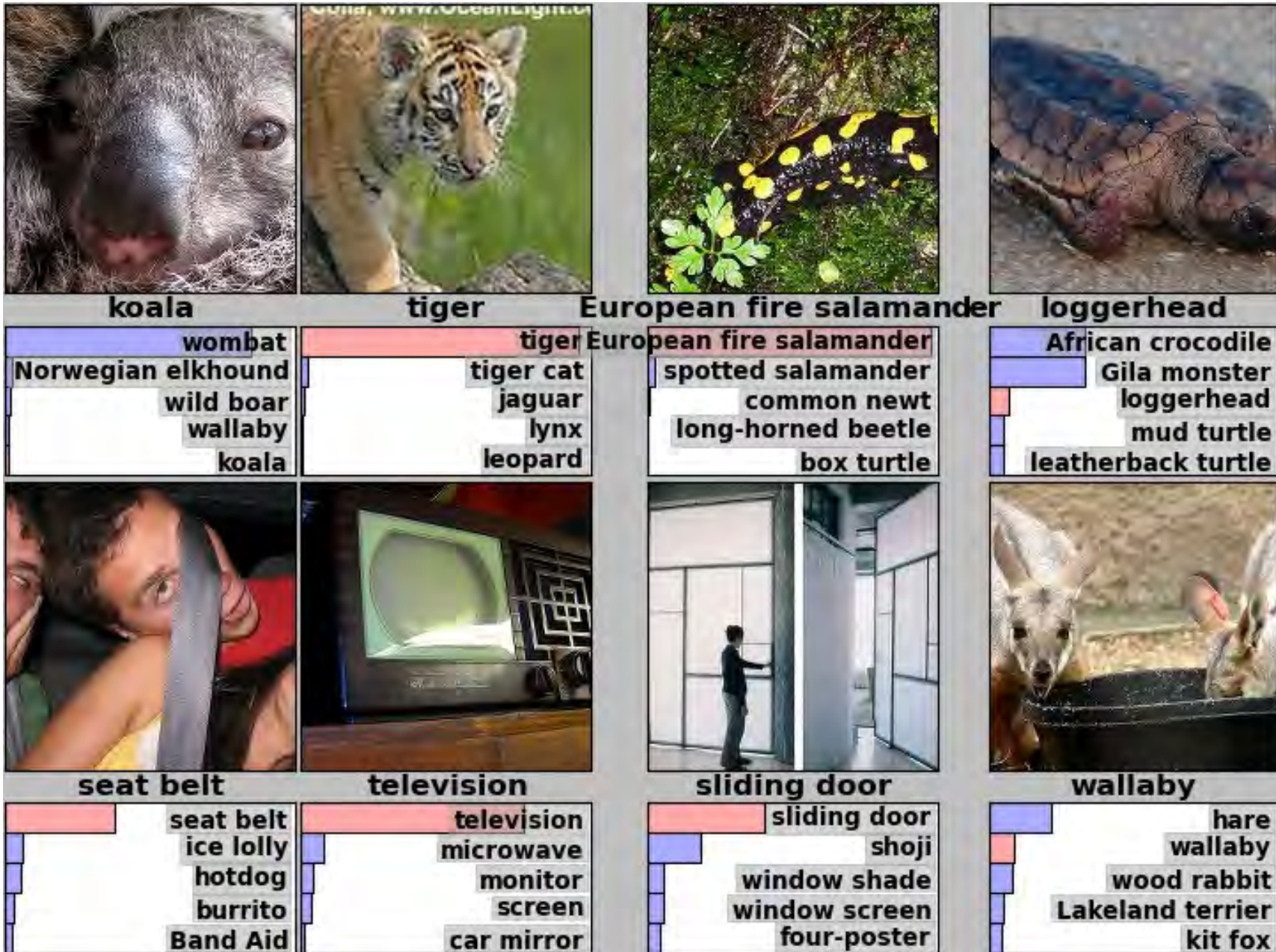
cellular telephone
slot
reflex camera
dial telephone
iPod







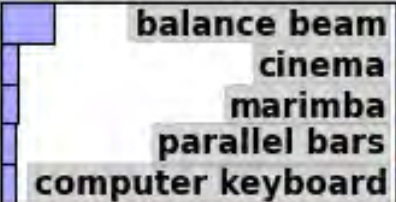
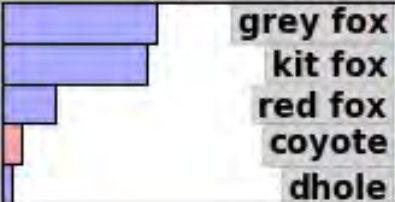
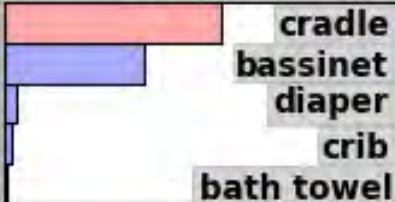
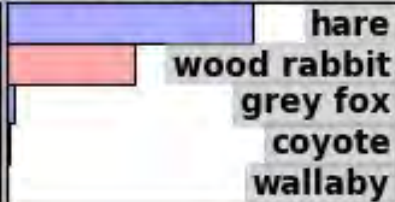




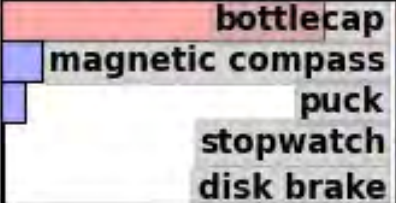

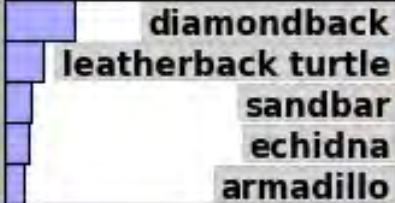
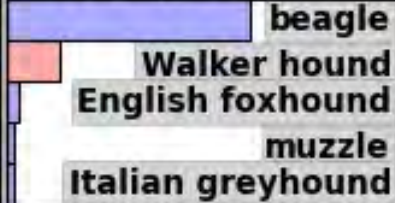
planetarium

planetarium
dome
mosque
radio telescope
steel arch bridge

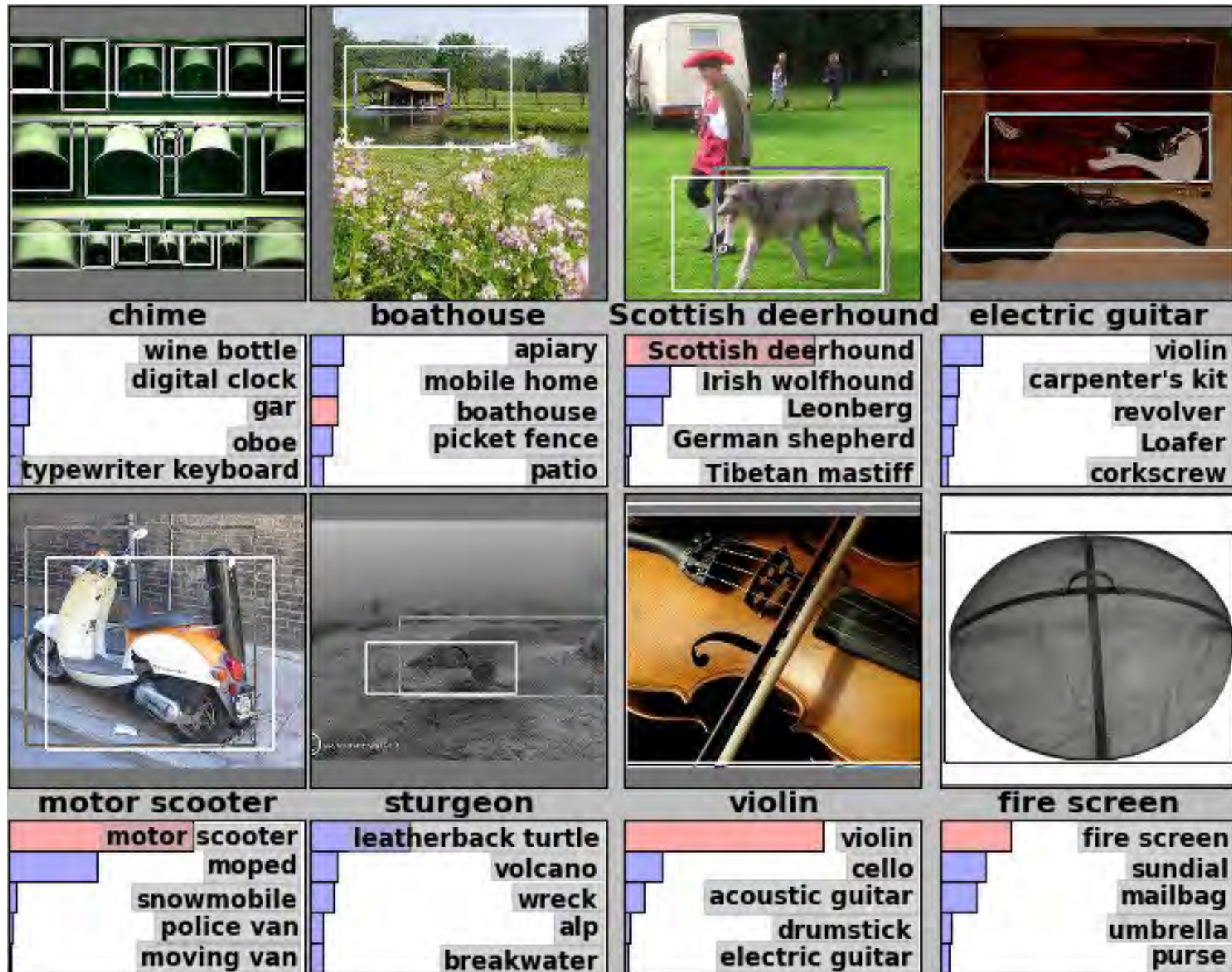
Validation classification



Validation localizations

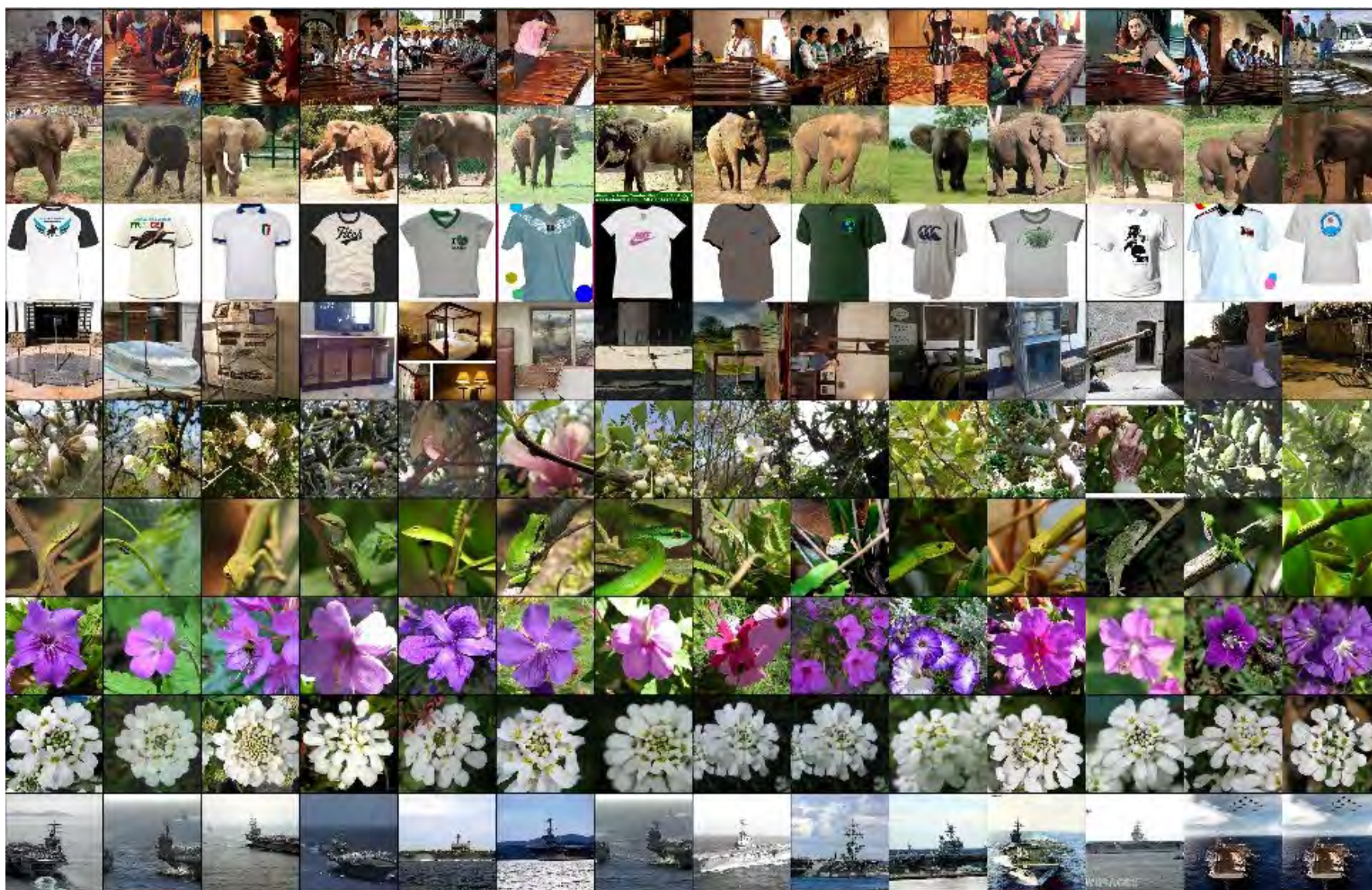
			
bookshop	coyote	cradle	wood rabbit
 <p>balance beam cinema marimba parallel bars computer keyboard</p>	 <p>grey fox kit fox red fox coyote dhole</p>	 <p>cradle bassinet diaper crib bath towel</p>	 <p>hare wood rabbit grey fox coyote wallaby</p>
			
bottlecap	harvester	garter snake	Walker hound
 <p>bottlecap magnetic compass puck stopwatch disk brake</p>	 <p>harvester thresher plow tractor tow truck</p>	 <p>diamondback leatherback turtle sandbar echidna armadillo</p>	 <p>beagle Walker hound English foxhound muzzle Italian greyhound</p>

Validation localizations



Retrieval experiments

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



Retrieval experiments

