# Machine Perception with Neural Networks

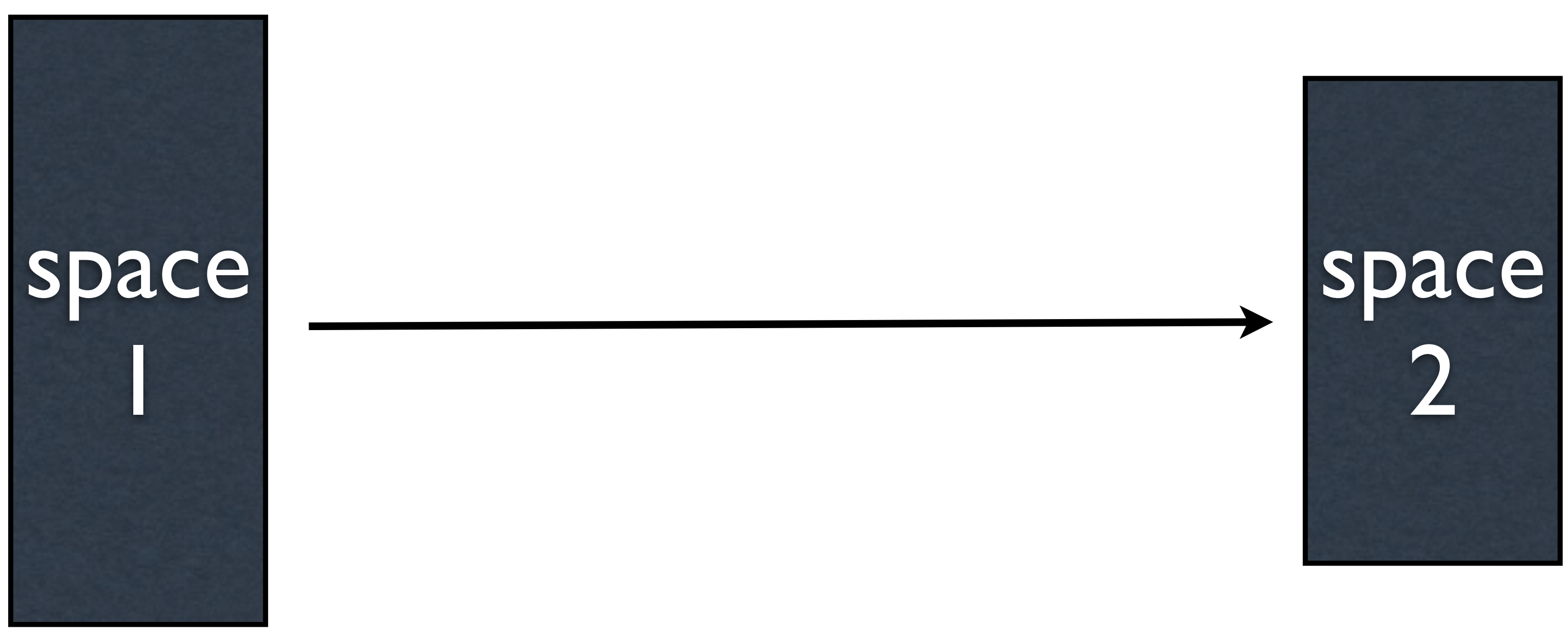
Ilya Sutskever

Google

Joint work with:
Ashish Agarwal, Anelia Angelova, Samy Bengio, Kai Chen, Greg Corrado Jeff Dean, Matthieu Devin, Andrea Frome, Geoffrey Hinton, John Lamping, Alex Krizhevsky, Quoc Le, Josh Levenberg, Mark Mao, Tomas Mikolov, Rajat Monga, Andrew Ng, Marc'Aurelio Ranzato, Andrew Senior, Noam Shazeer, Jon Shlens, Yoram Singer, Benoit Steiner, Thomas Strohmann, Emanuel Taropa, Simon Tong, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Manjunath Venkatakrishna, Ke Yang, and others...

# Neural network

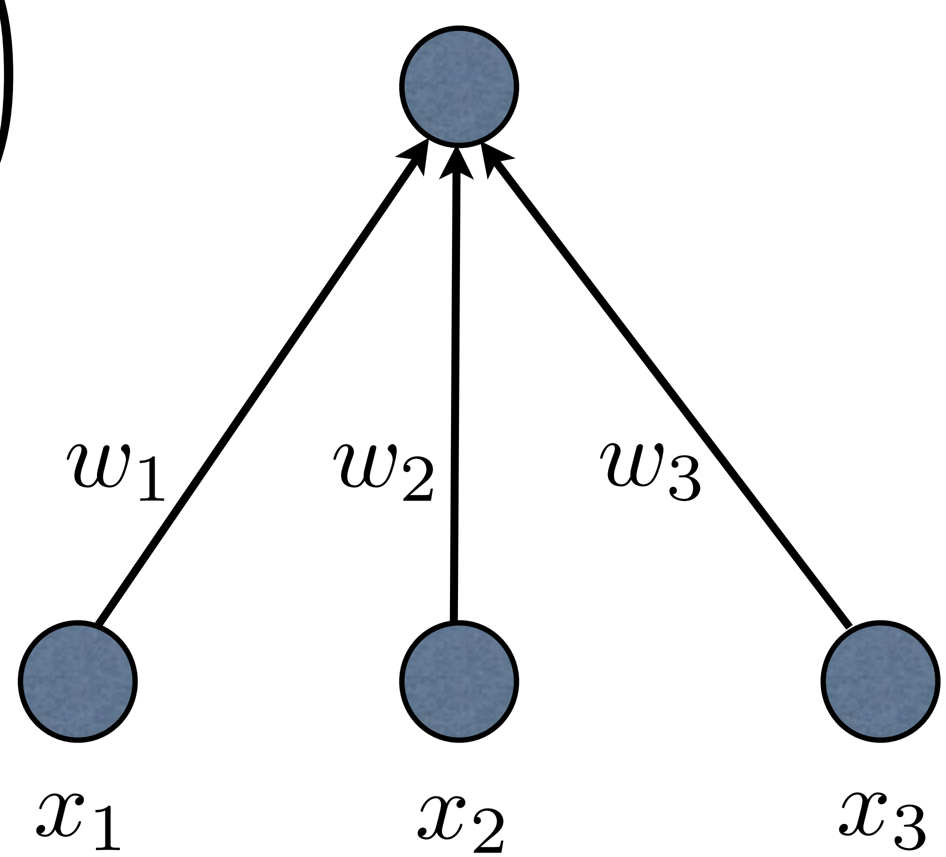
- Learn a complicated function from data

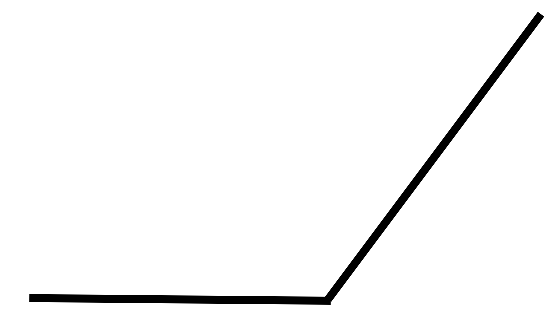


# The neuron

- Different weights compute different functions

$$y_i = F\left(\sum_i w_i x_i\right)$$

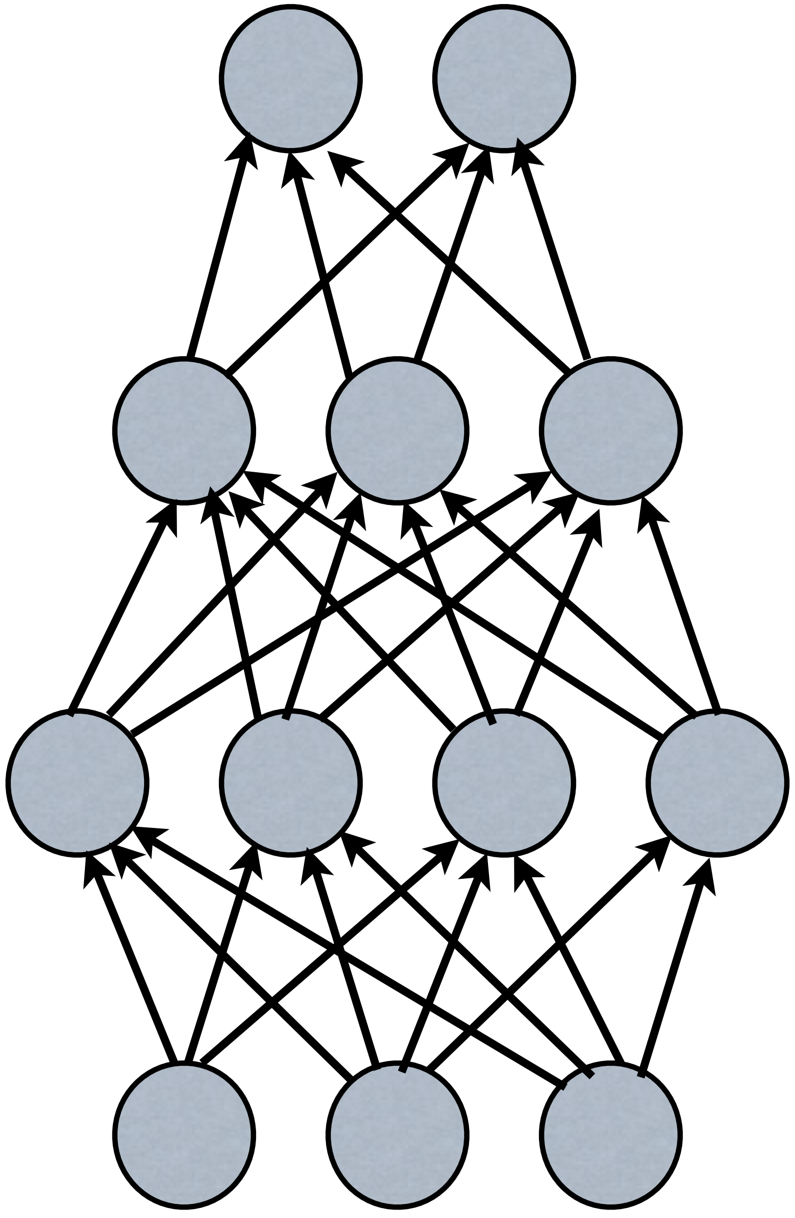


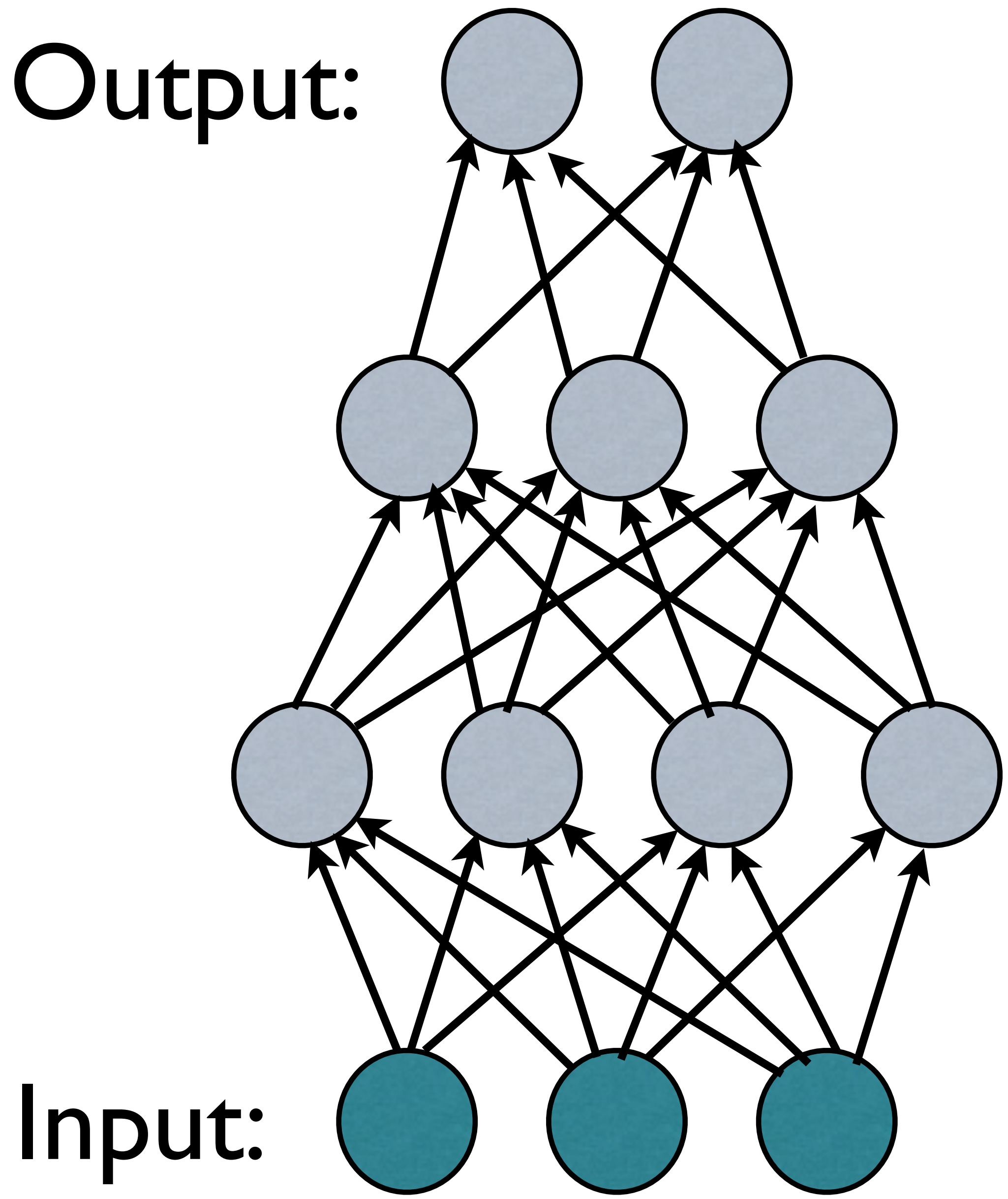
$F(x) = \max(0, x)$

# Neural networks

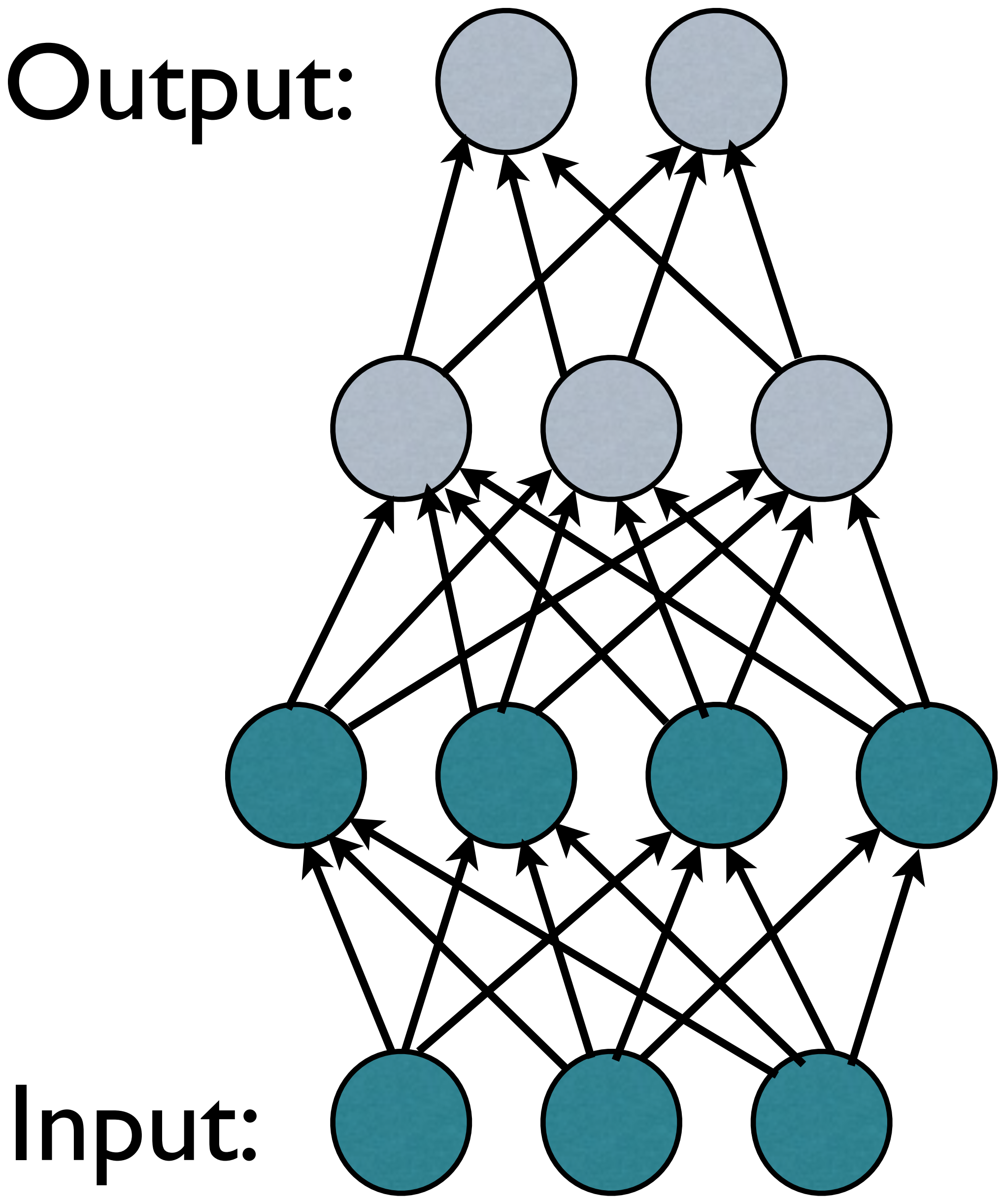
- Different weights compute different functions



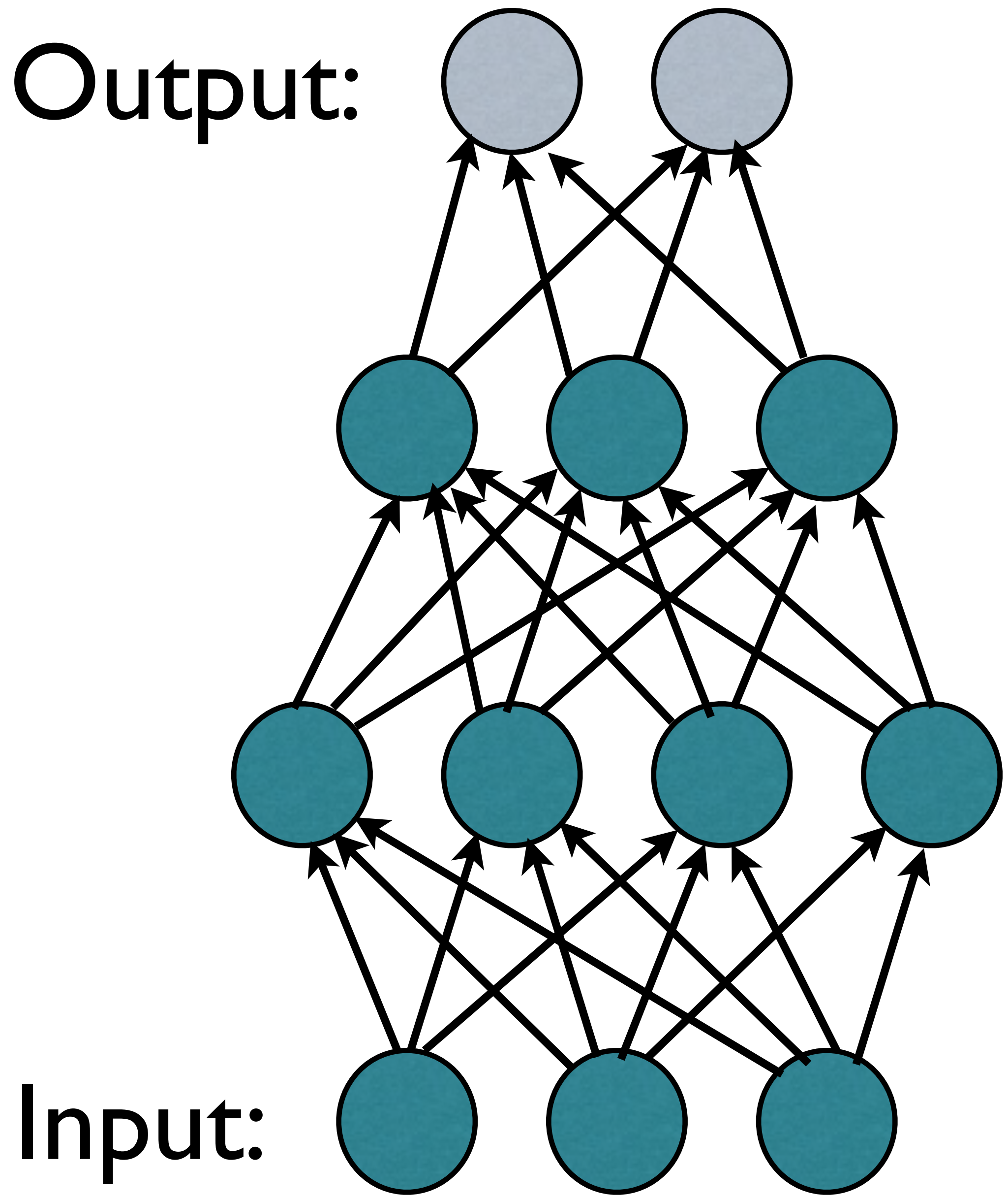
# Neural networks



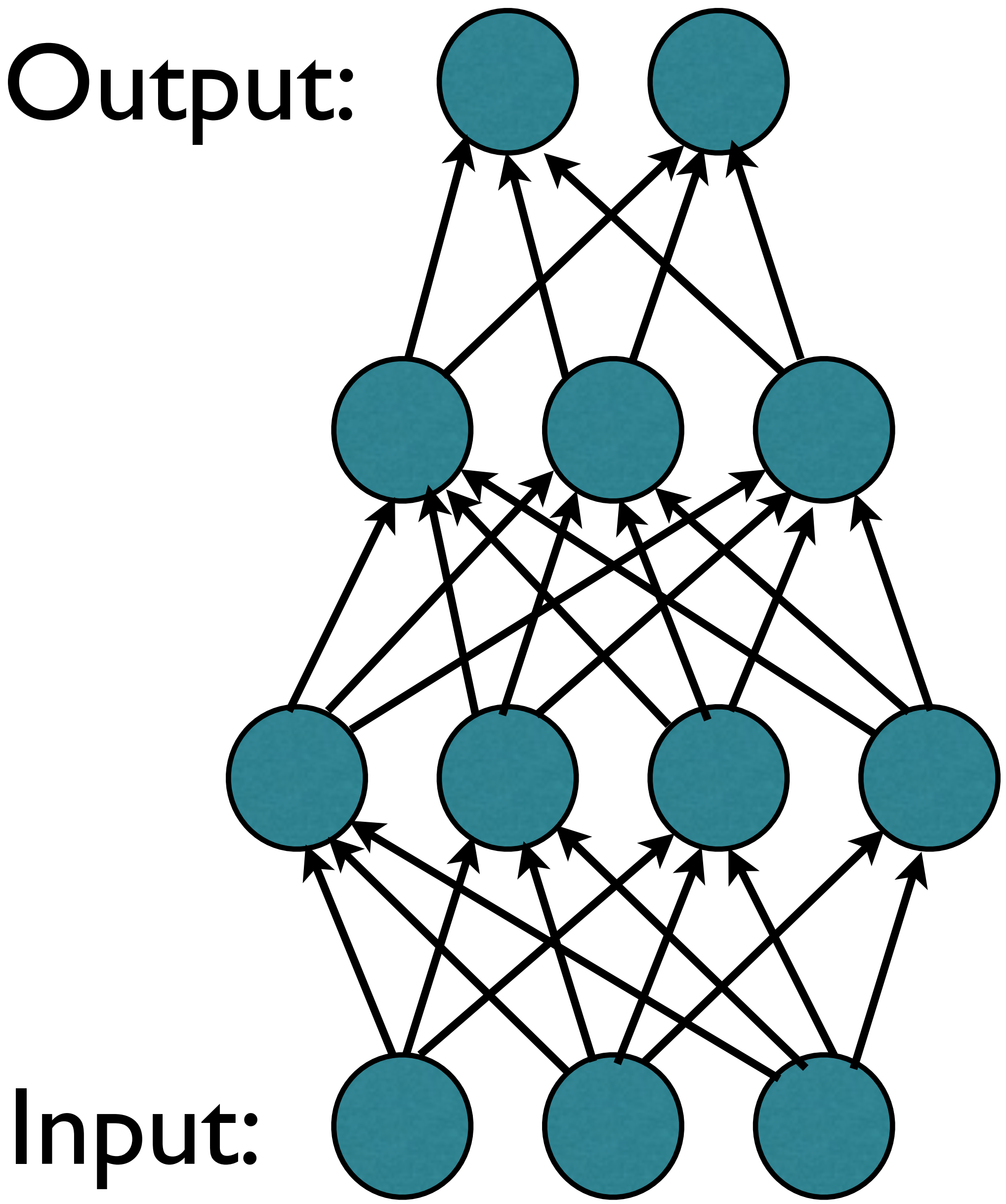
# Neural networks



# Neural networks



# Neural networks



# Learning algorithm

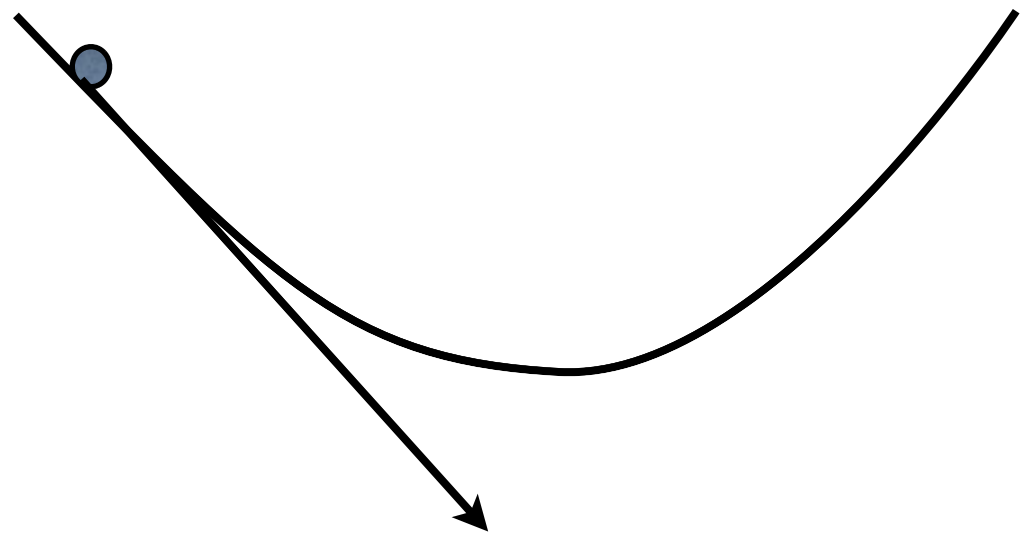
- **while** not done
  - pick a random training case **(x, y)**
  - run neural network on input **x**
  - modify connections to make prediction closer to **y**

# Learning algorithm

- **while** not done
  - pick a random training case **(x, y)**
  - run neural network on input **x**
  - <u>modify connections to make prediction closer to **y**</u>

# How to modify connections?

- Follow the gradient of the error w.r.t. the connections



Gradient tells us how to change the parameters

# We can learn 10-layer networks

- We have a recipe of how to do it
- Not much theory

# What can neural nets compute?

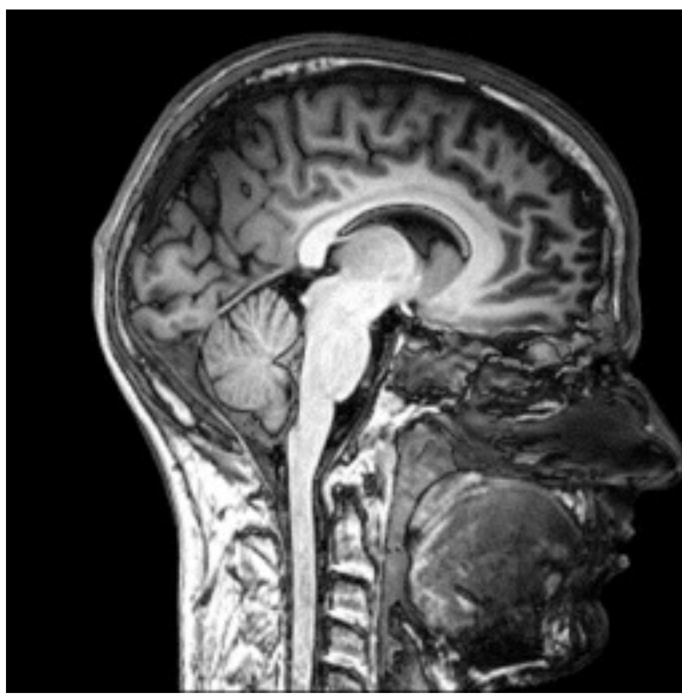
- Theoretical results
- Modest-sized neural networks with 2 hidden layers can:
  - Sort N N-bit numbers
  - Multiply N binary numbers
  - Compute any analytic function to high precision

# What can neural nets compute?

- Human perception is very fast (0.1 second)
  - Recognize objects ("see") 
  - Recognize speech ("hear") 
  - Recognize emotion 
  - Instantly see how to solve some problems
  - And many more!

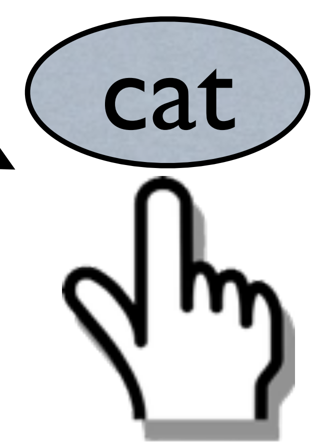
# What can neural nets compute?

0.1 sec: neurons fire only 10 times!



see image

click if cat



# What can neural nets compute?

- **Anything humans can do in 0.1 seconds, a big 10-layer network can do, too**

# How big is big?

- <u>Big != exponentially big</u>
  - universal approximator = exponentially big
- Number of training cases ~ number of trainable parameters
- Feasible if one is motivated

# How to solve it

- Get a very fast neural network implementation
  - Fast implementation = big nets
- Get enough training cases
- Train the network for a long time

# To summarize

- Human perception is very fast
- Neurons have time to fire only 10 times during perception
- Thus perception is solvable by some 10 layer neural networks
- So we just need to train these networks

# It actually works!

- It is a speculative argument
- Nonetheless, neural networks are unquestionably best at:
  - Speech Recognition
  - Visual Object Recognition

# How to get good results?

- Collect a big training set
- Train very big neural networks

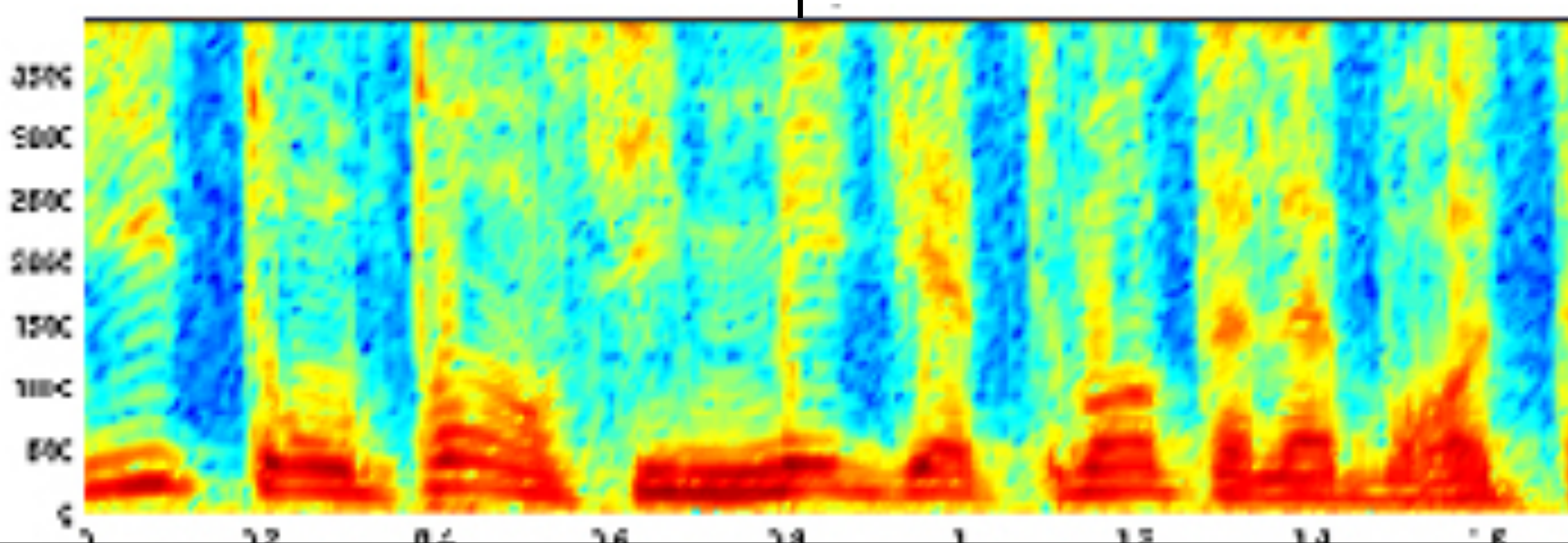
# Case study: speech recognition

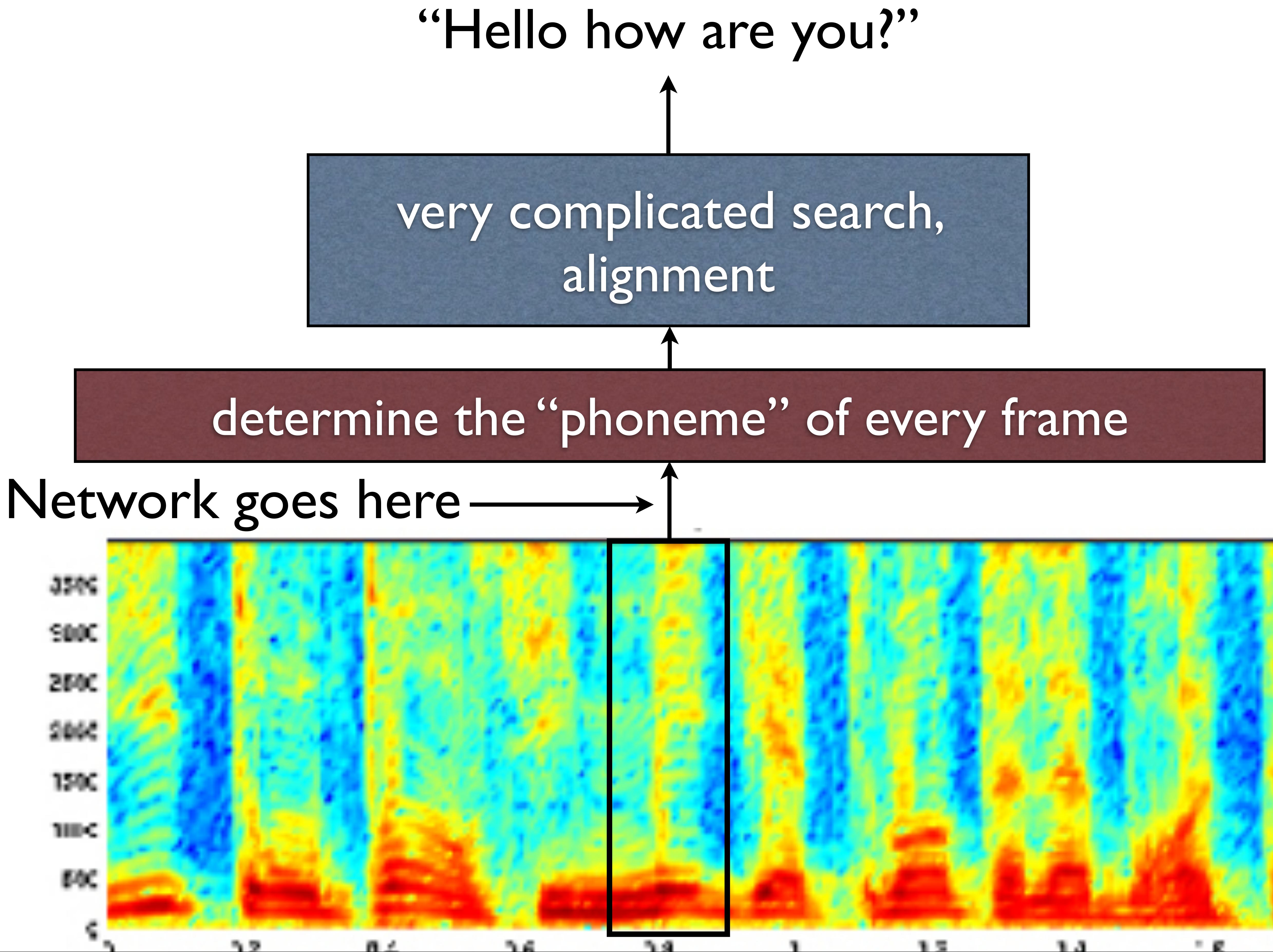
- Pioneered by
  - Abdel-Rahman Mohamed (IBM research)
  - George Dahl (UToronto)
  - Navdeep Jaitly (Google/UToronto)
  - Geoff Hinton (Google/UToronto)
- Developed further by IBM, Microsoft, Google

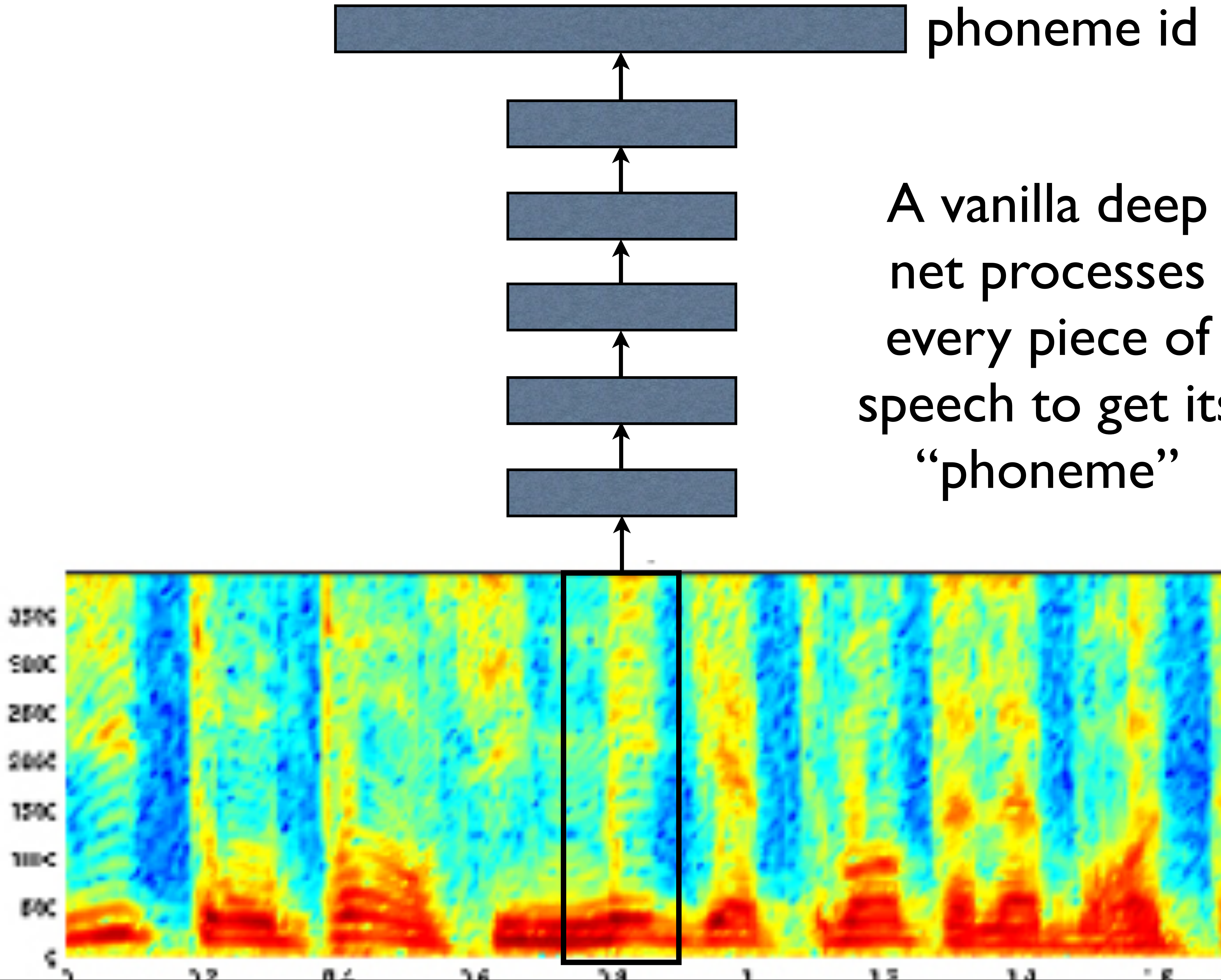
# The problem

- Transcribe speech

"Hello how are you?"




"Hello how are you?"
very complicated search, alignment
determine the "phoneme" of every frame
Network goes here




A vanilla deep net processes every piece of speech to get its "phoneme"

# Completely vanilla neural network

- Every neuron is connected to every other neuron
- Thousands of neurons per layer, except output layer, which is larger
- 8 layers
- Train it on a lot of data

# To conclude

- A neural network in an appropriate location greatly improves speech recognition
- The rest of the speech pipeline does not use neural networks (yet)
- A more developed neural network is used in Google's speech recognition

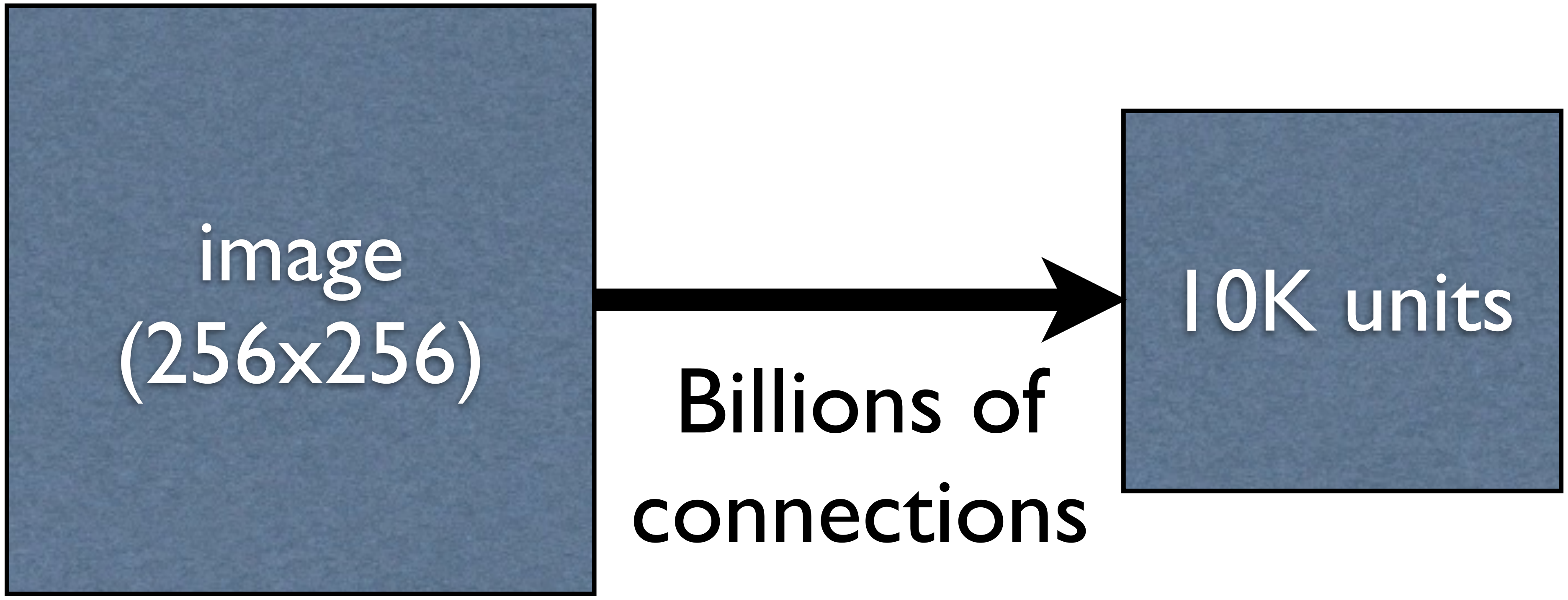
# Case study: visual object recognition

- Task: determine which object is in an image

“dog”

# Just use a neural network?

- Problem: images are very high-dimensional objects
  - Naive neural networks would have too many connections and parameters

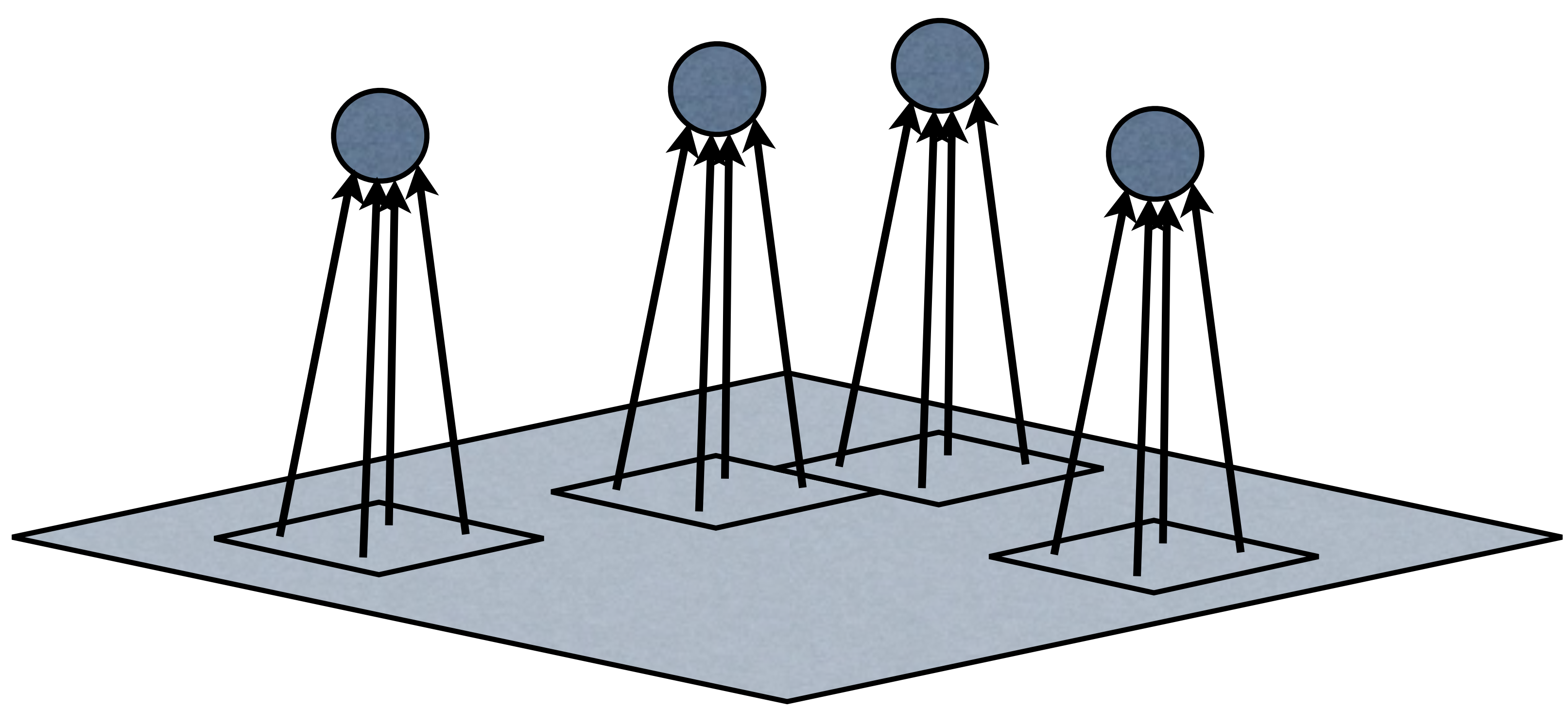


# Convolutional neural networks

- For images, we can **vastly** reduce the number of connections and parameters without hurting expressiveness
- Key idea: the neural network should perform the **same** kind of **local processing** in every image region

# Convolutional neural networks

Each neuron is connected to a local image patch with the same connections
(First introduced by Yann LeCun, NYU/Facebook)

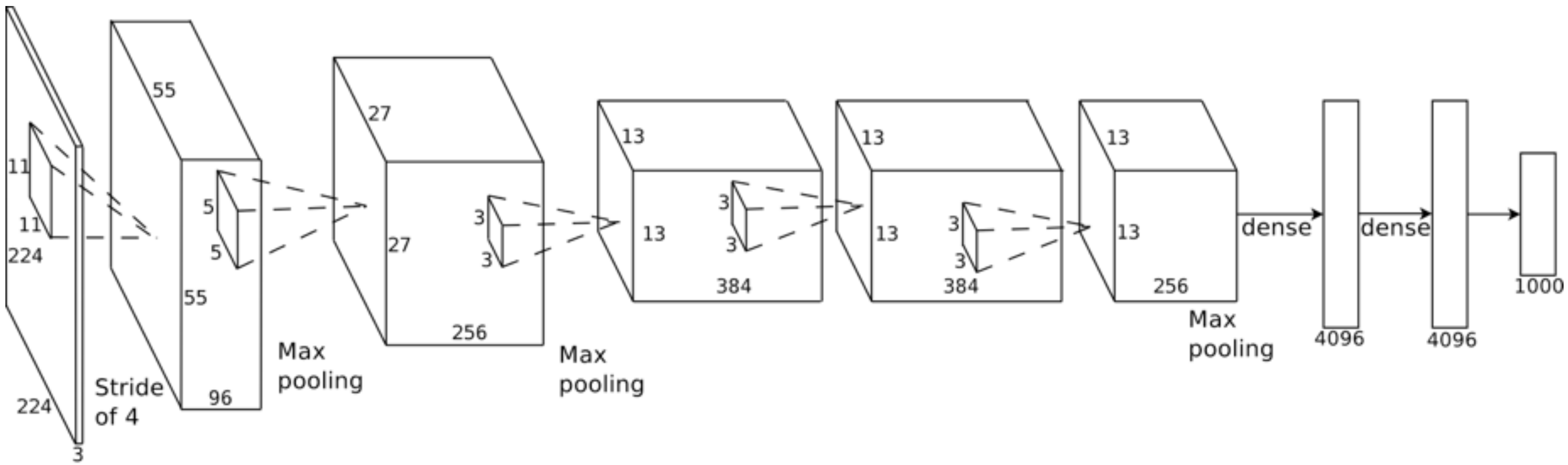


Result: vastly fewer connections and parameters, no loss in expressiveness

# Convolutional neural networks

- Has many fewer connections and parameters, but (probably) similar expressiveness to a much larger fully connected network
  - This is **always** a good thing
- Allows us to use neural networks with **hundreds of thousands of neurons**

# A good architecture

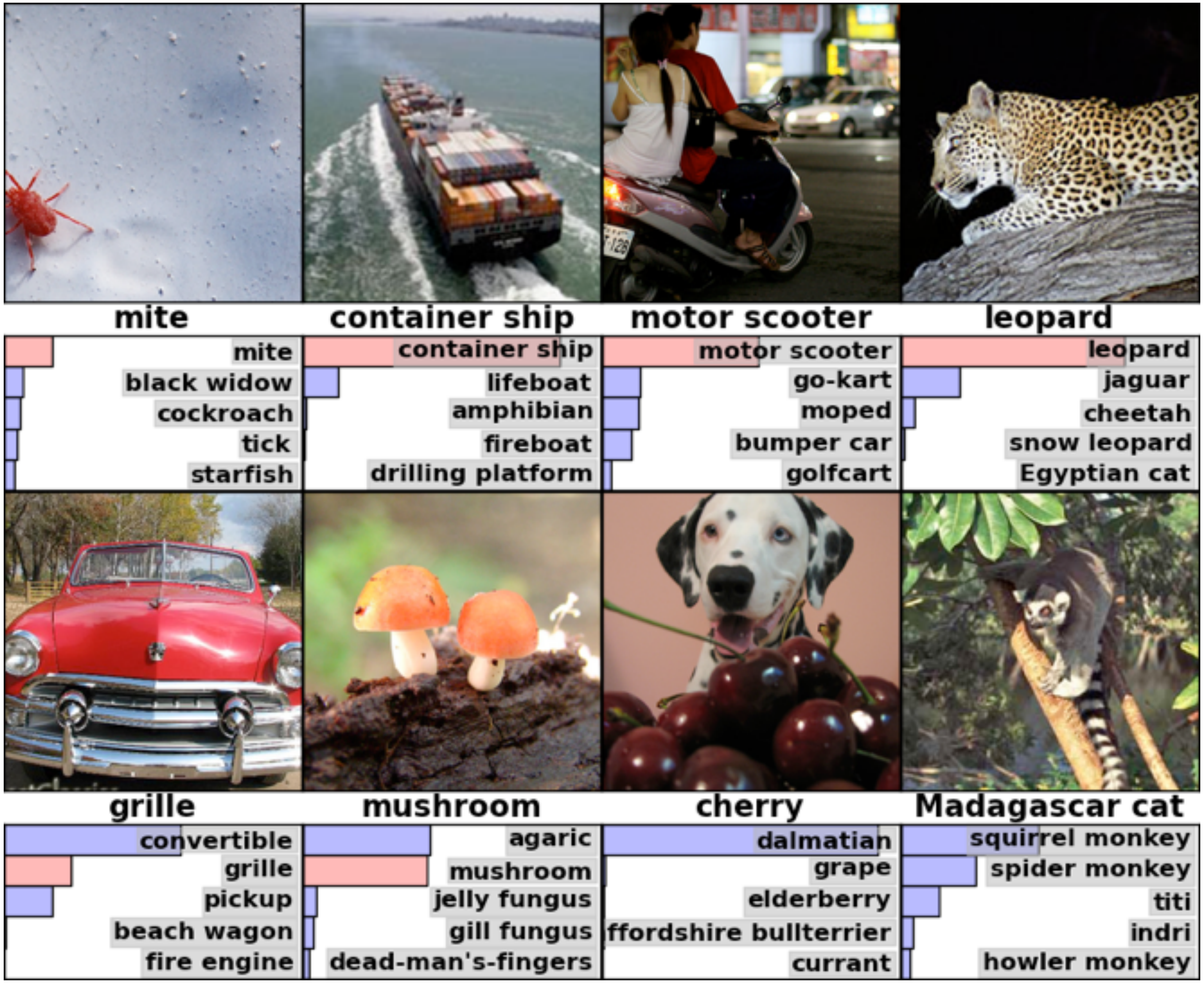



- Originally developed by Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton (Google)

# Deployed in Photo Search

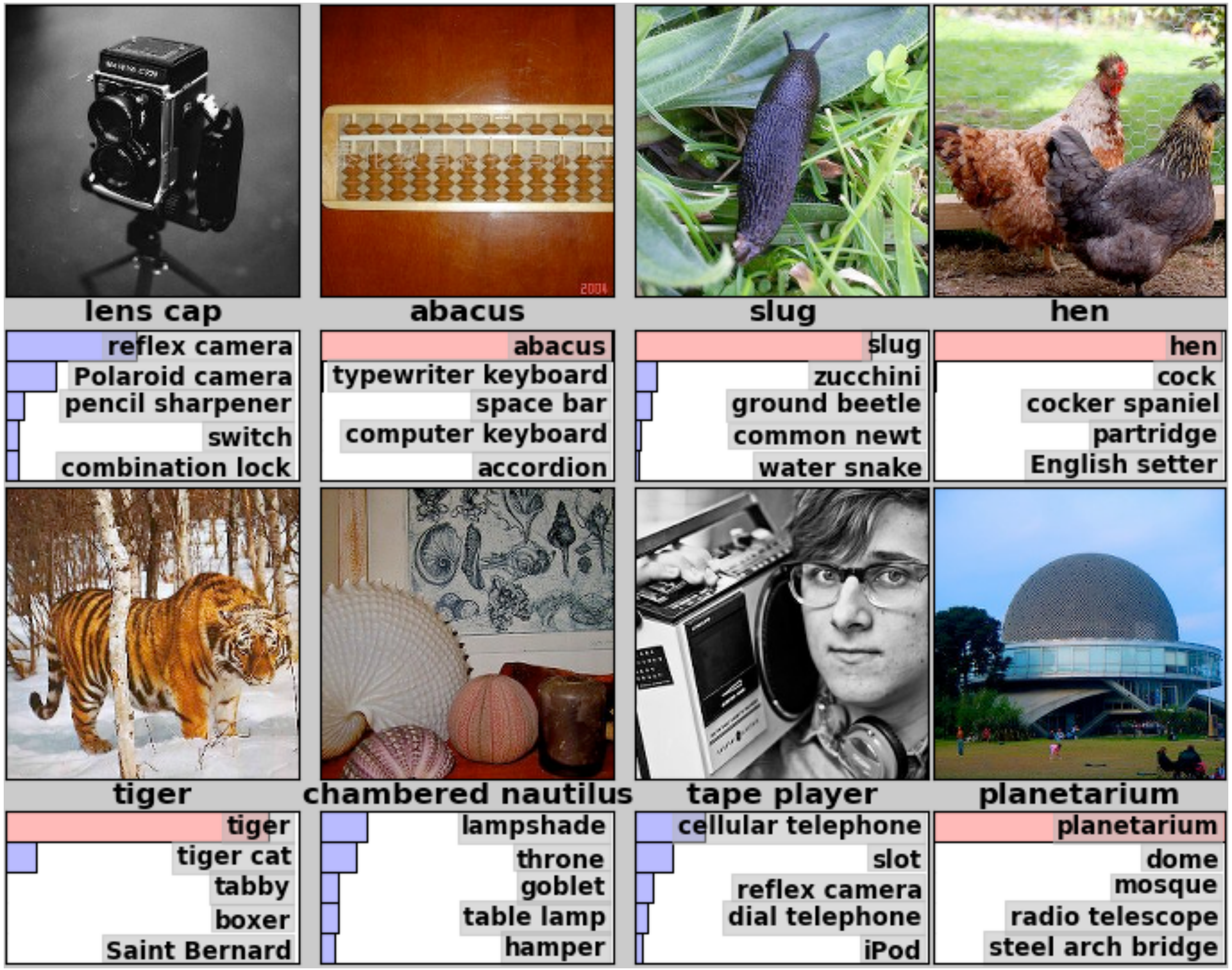
- An advanced version of this neural network is deployed in G+ photo search and elsewhere

# Examples



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

# Examples



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

# Examples


koala
wombat
Norwegian elkhound
wild boar
wallaby
koala
tiger
tiger
tiger cat
jaguar
lynx
leopard
European fire salamander
European fire salamander
spotted salamander
common newt
long-horned beetle
box turtle
loggerhead
African crocodile
Gila monster
loggerhead
mud turtle
leatherback turtle
seat belt
seat belt
ice lolly
hotdog
burrito
Band Aid
television
television
microwave
monitor
screen
car mirror
sliding door
sliding door
shoji
window shade
window screen
four-poster
wallaby
hare
wallaby
wood rabbit
Lakeland terrier
kit fox

# Examples: specific

“hibiscus”

“dahlia”

# Examples: broad

Both recognized as a “meal”

# Examples: errors

“snake”

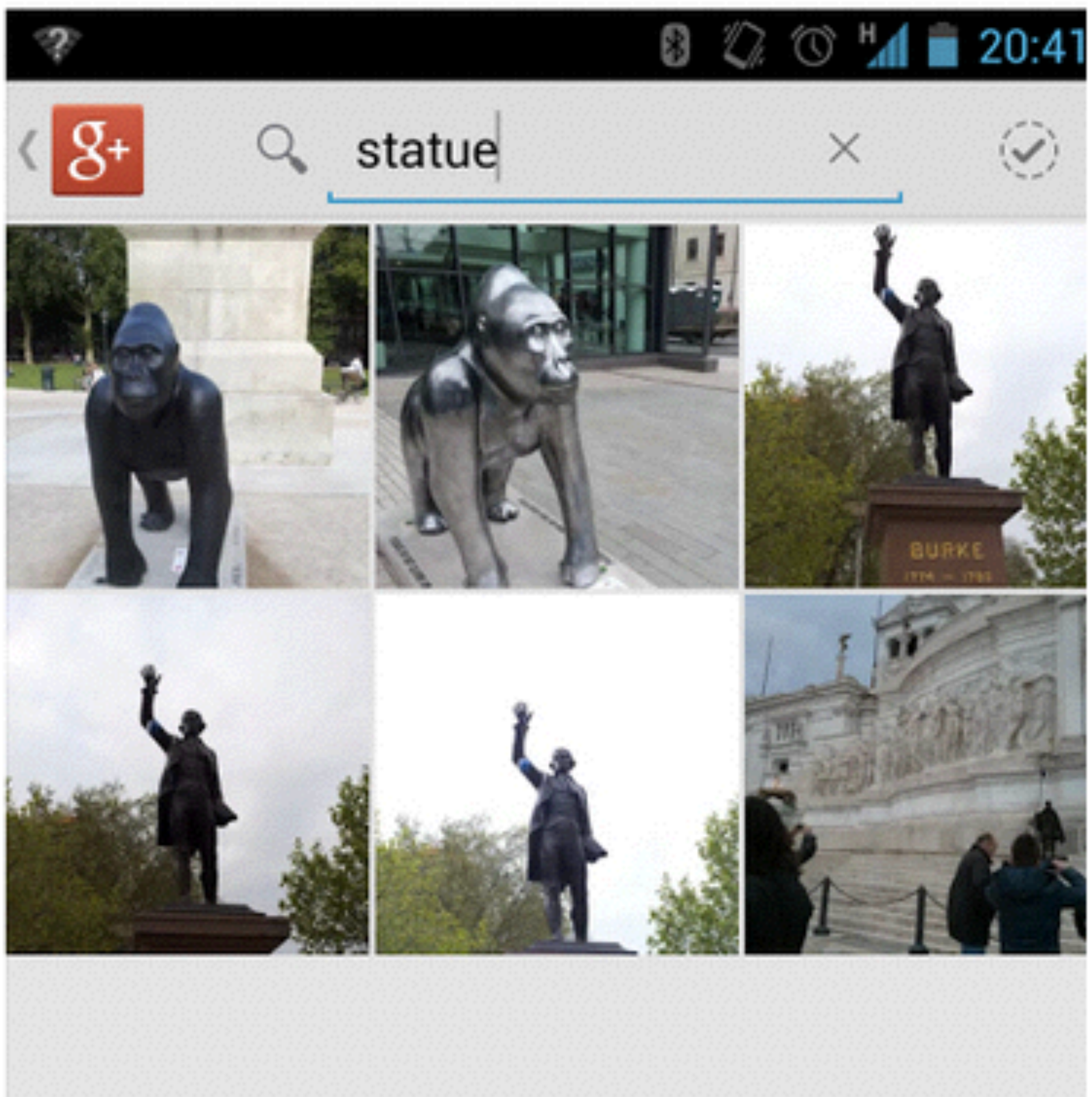
“dog”

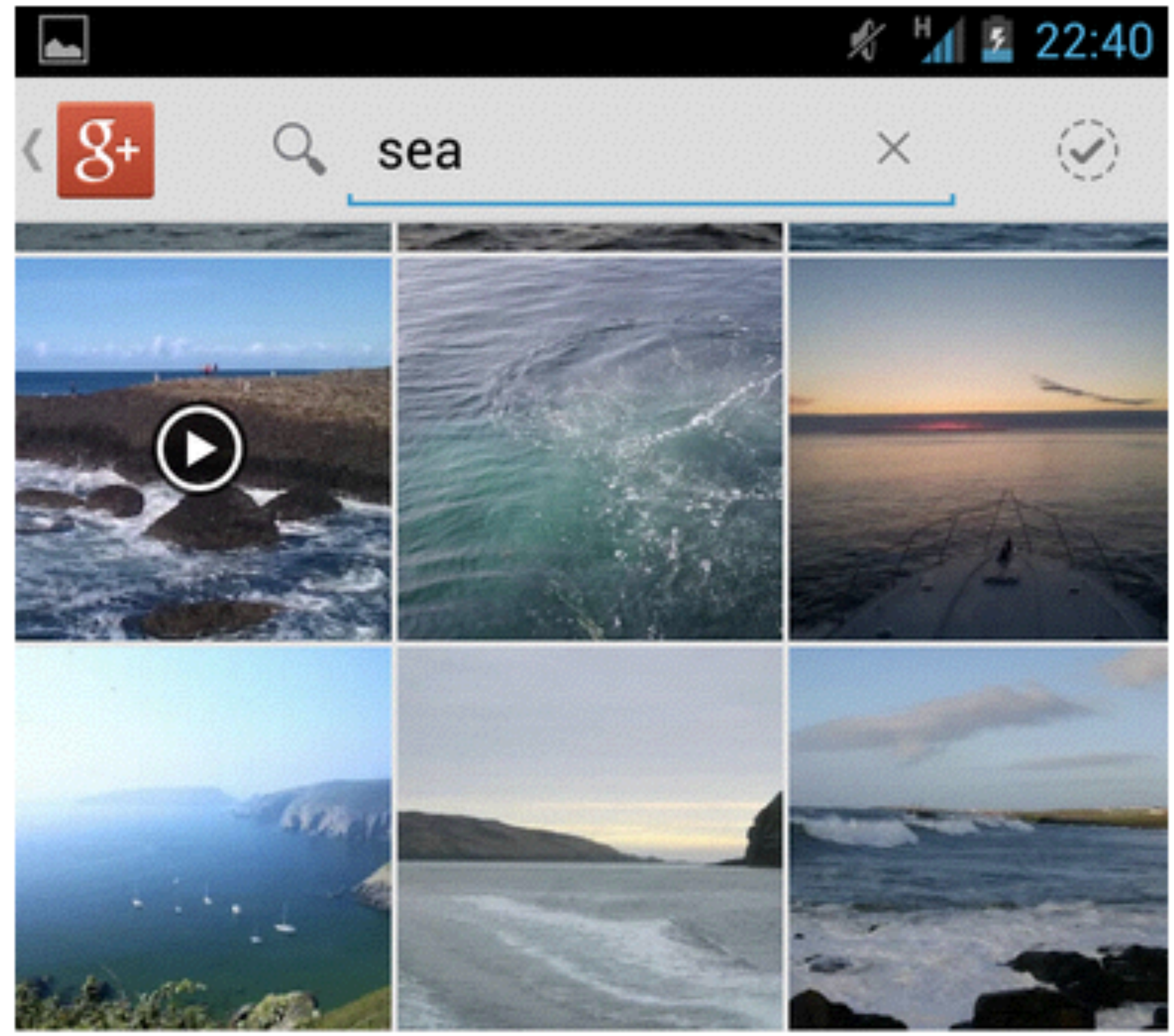
# Examples: Google+

Wow.

The new Google plus photo search is a bit insane.

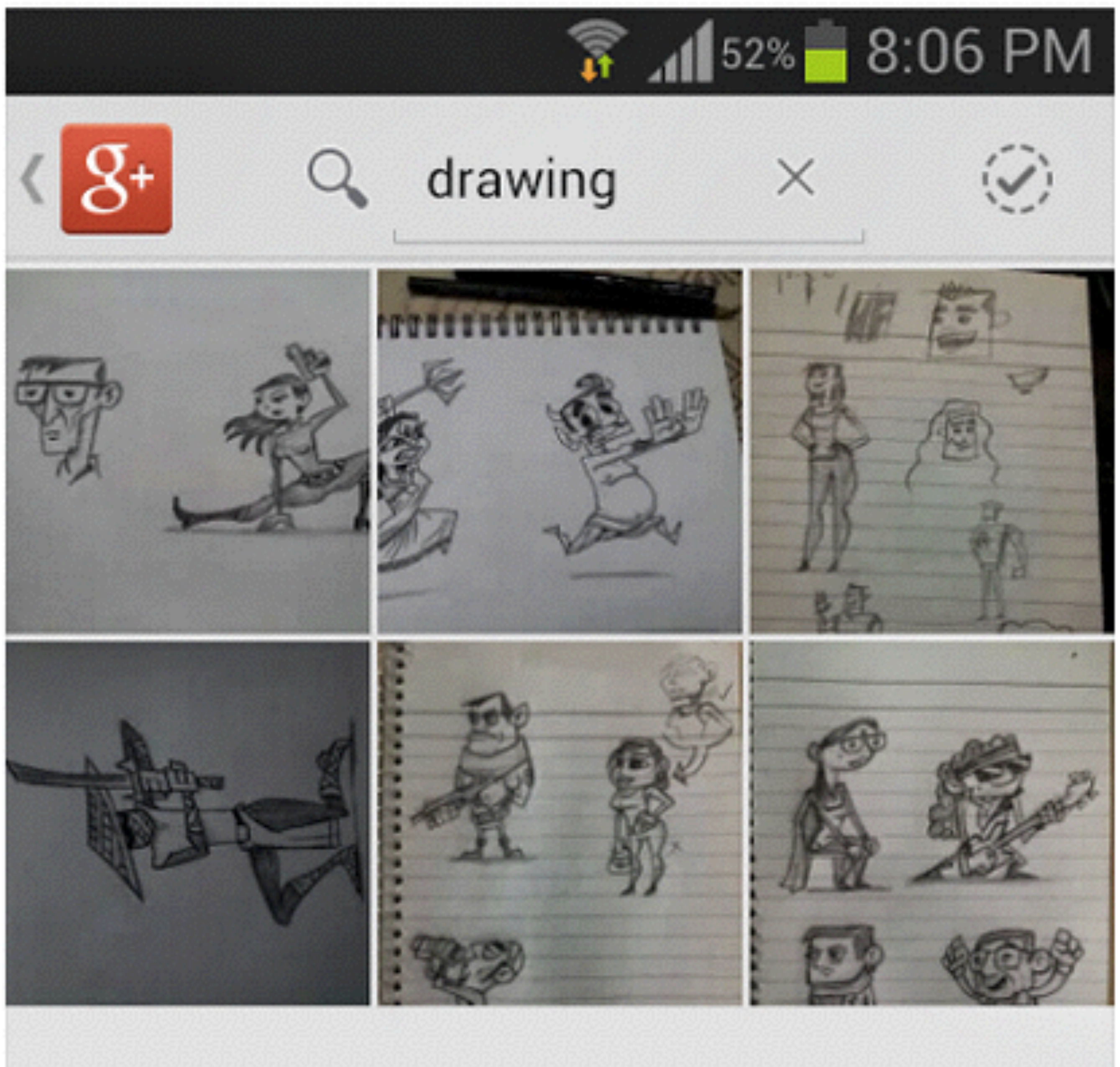
I didn't tag those... :)



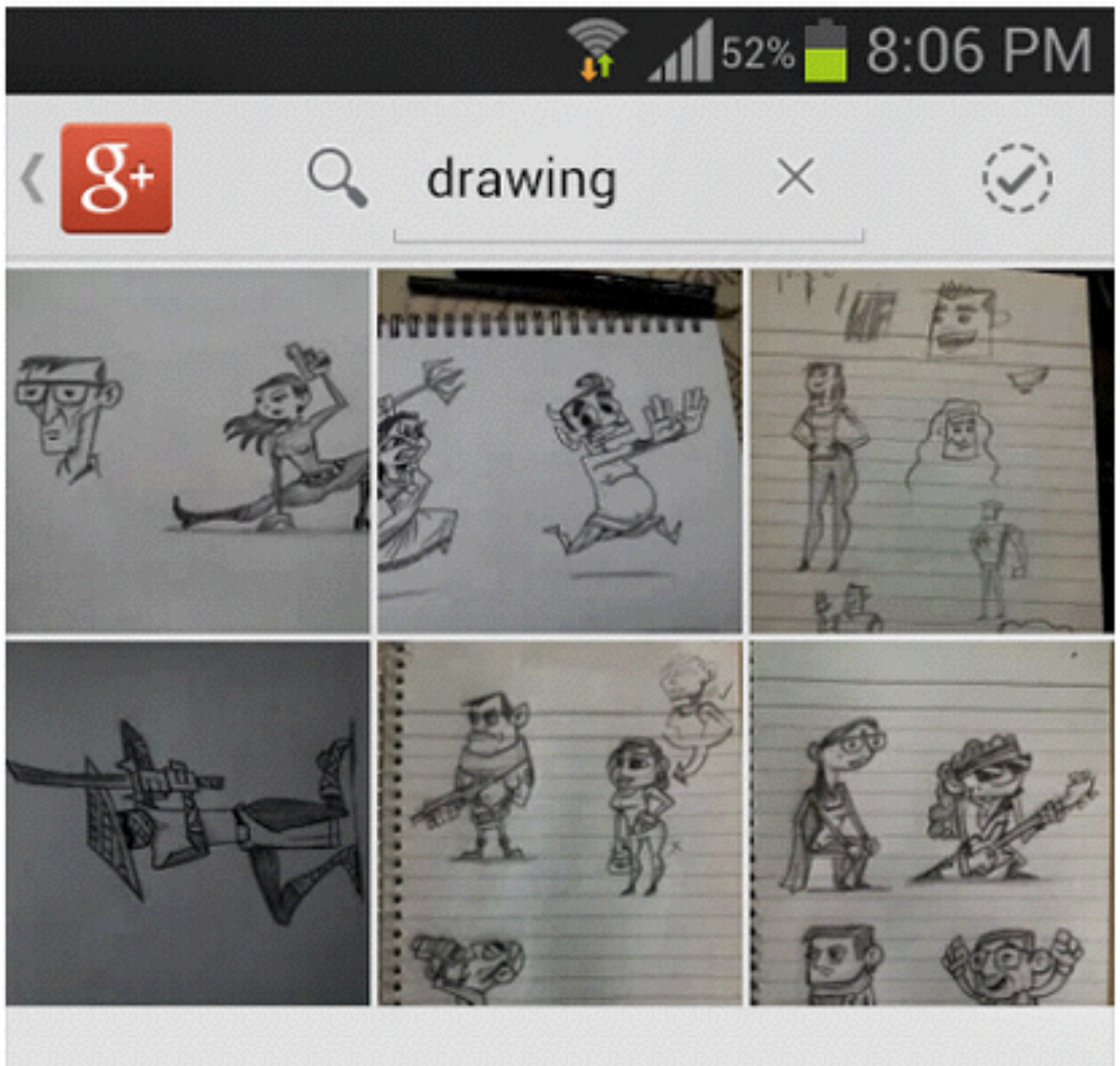
# Examples: Google+

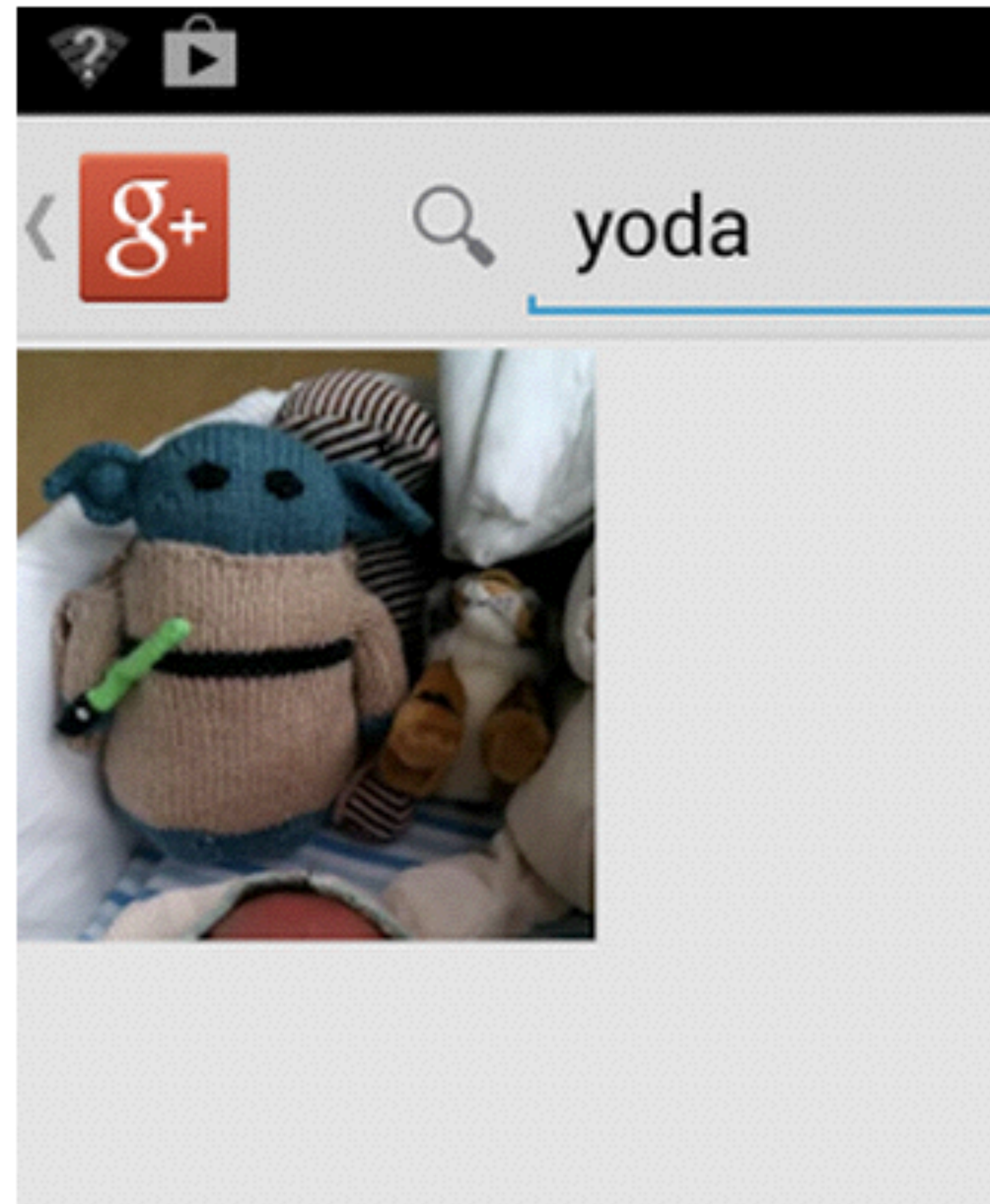
Google Plus photo search is awesome. Searched with keyword 'Drawing' to find all my scribbles at once :D



# Examples: Google+

Google Plus photo search is awesome. Searched with keyword 'Drawing' to find all my scribbles at once :D



# What's new since the 80s?

- These neural networks were invented in the 80s
- What's different now?
  - Much more data
  - Much faster computers
  - That's it!

# Slow computers cannot succeed

- We want to solve really hard problems
- Hard problems need big neural networks
- Slow computers can only train small neural networks
    - Current neural networks use hundreds of millions FLOPs for a single recognition

# Small training sets cannot succeed

- Big neural networks cannot be successfully trained on small datasets
  - Need more examples than parameters
  - Tens of millions of parameters or more!

# Importance of depth

- Until recently, researchers didn't appreciate the connection between neural network depth and computation
- Now we know: deep neural networks are essential for hard perception tasks

# The guarantee

- Given **any** problem that:
  - Humans can solve very quickly
  - Has very many labelled training examples
- Then a big 10-layer neural network is **likely** to get excellent performance when trained on enough examples

# Words as vectors

- Words are discrete objects
- Machine learning algorithms are good with vectors
- So it is useful to represent words with vectors for other applications

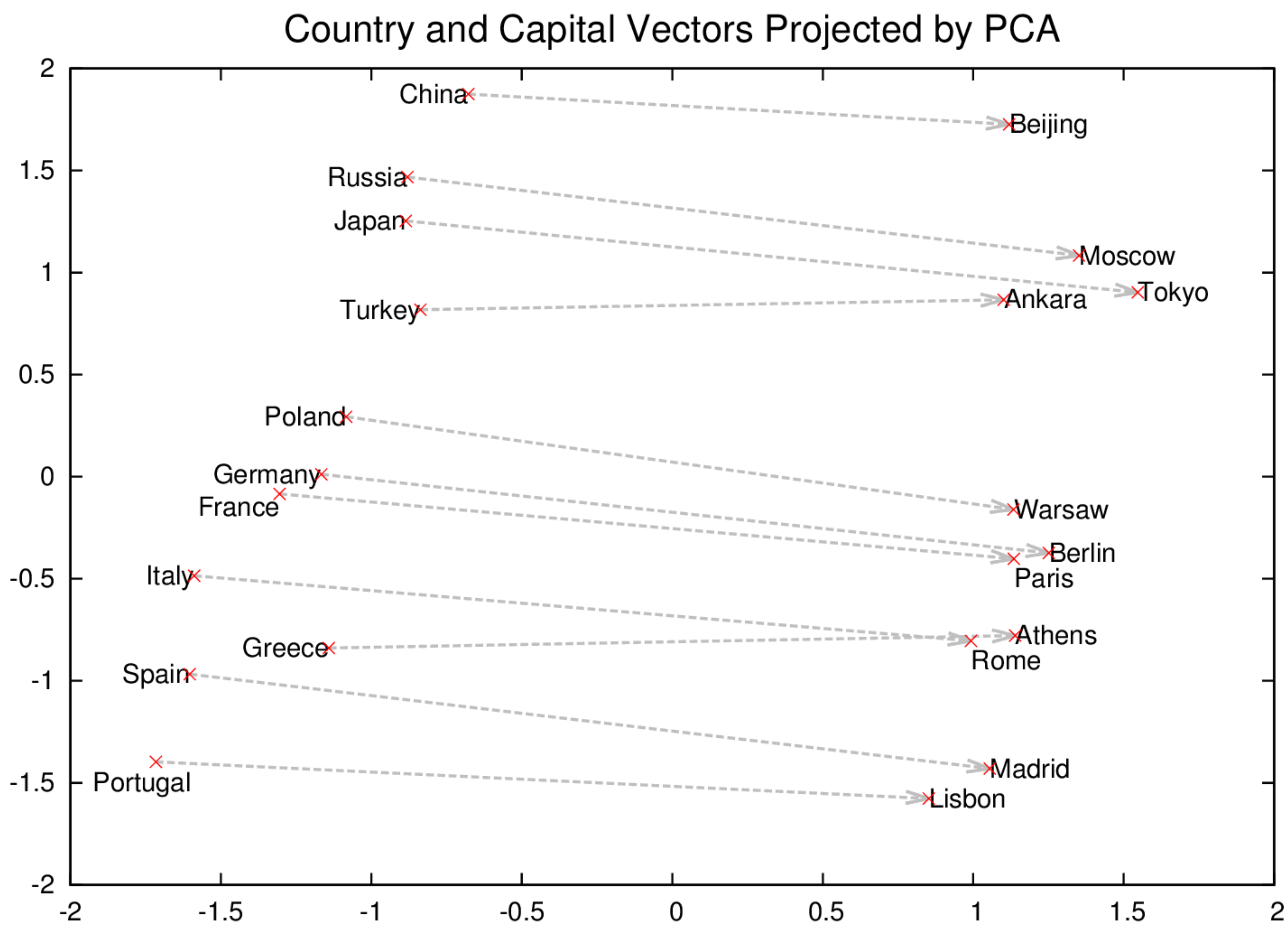
# word2vec

- A neural network that learns useful vector representations of words
- Objective: find vectors so that words that tend to appear together have similar vectors
- Learning is very fast (and mysterious)
- Developed by Tomas Mikolov et al.

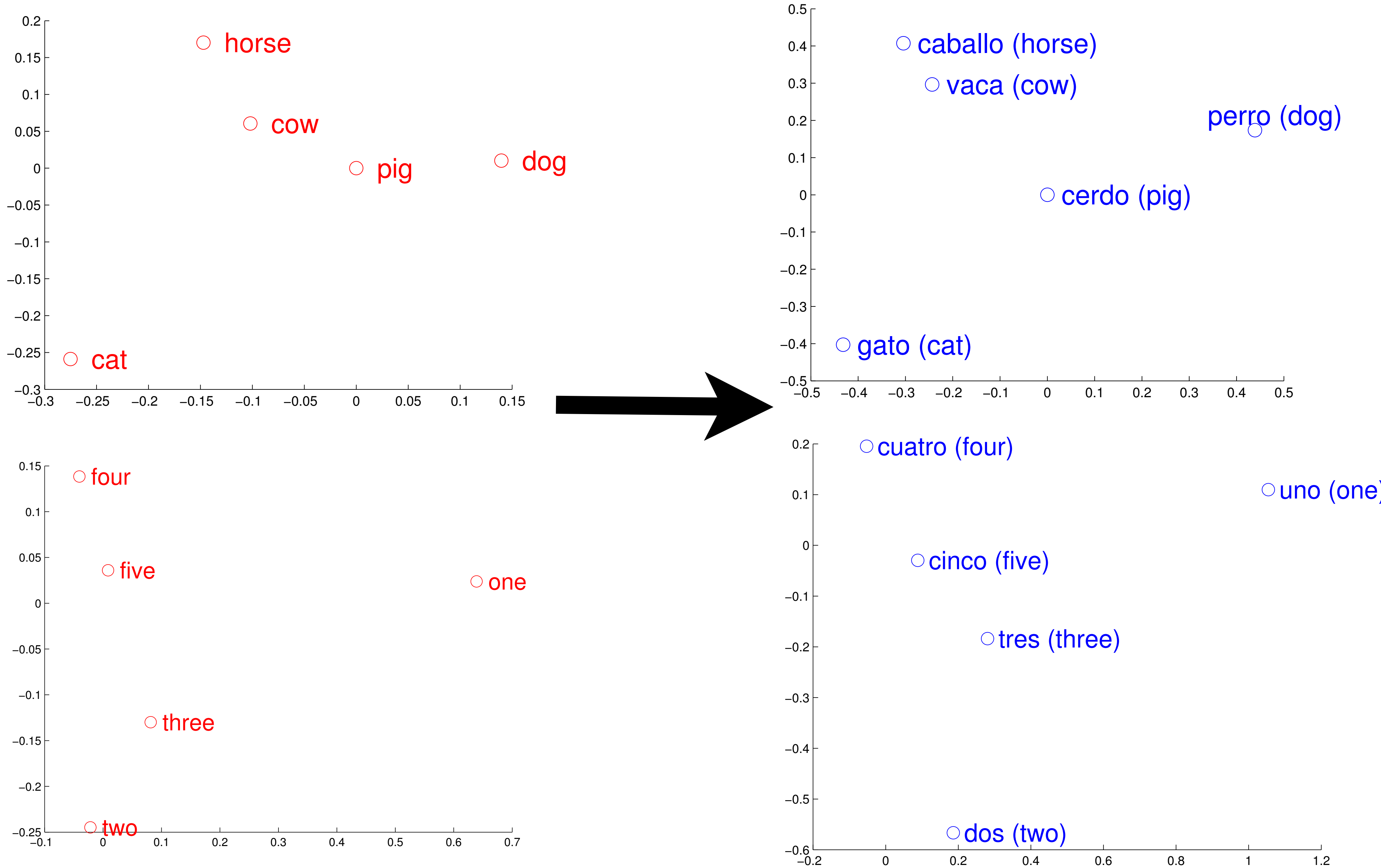
# Words as vectors

- The vectors have interesting properties
  - Similar words have similar vectors
  - vec(King) - vec(Man) + vec(Woman) is close to vec(Queen)
  - vec(China) + vec(Currency) is close to vec(Yuan)

# Words as vectors



# Structure is similar across languages



# Will neural networks help you?

- If you have:
  - a fair bit of **labelled** data?
  - a decent implementation?
- Then you too could **improve** the accuracy of your classifiers with a neural network!
  - (but average it with your old classifier, don't throw it away!)

# They have potential

- Supervised learning has obvious limits
- New ideas are needed to make progress
  - Unsupervised learning
  - Reinforcement learning
  - Much more powerful models

# Thank you!