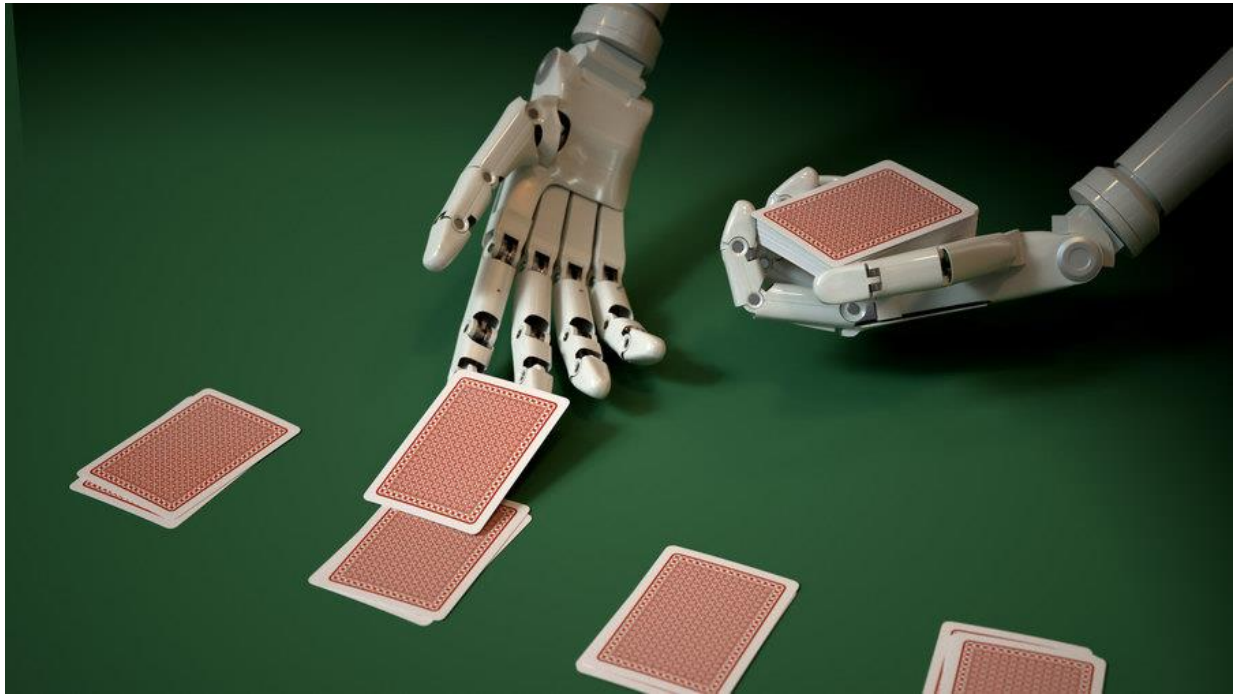
# practical deep learning
## introduction to machine learning

Alex Honchar
University of Verona

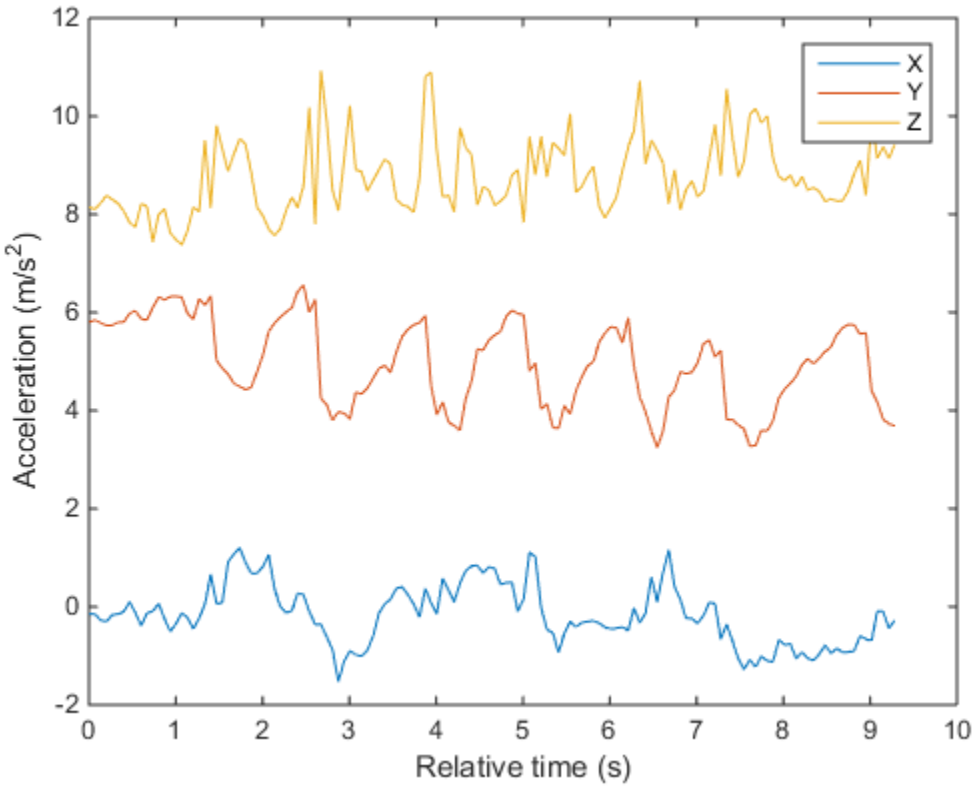
# my experience

# Playing bot for online game (UA, RU 2014)



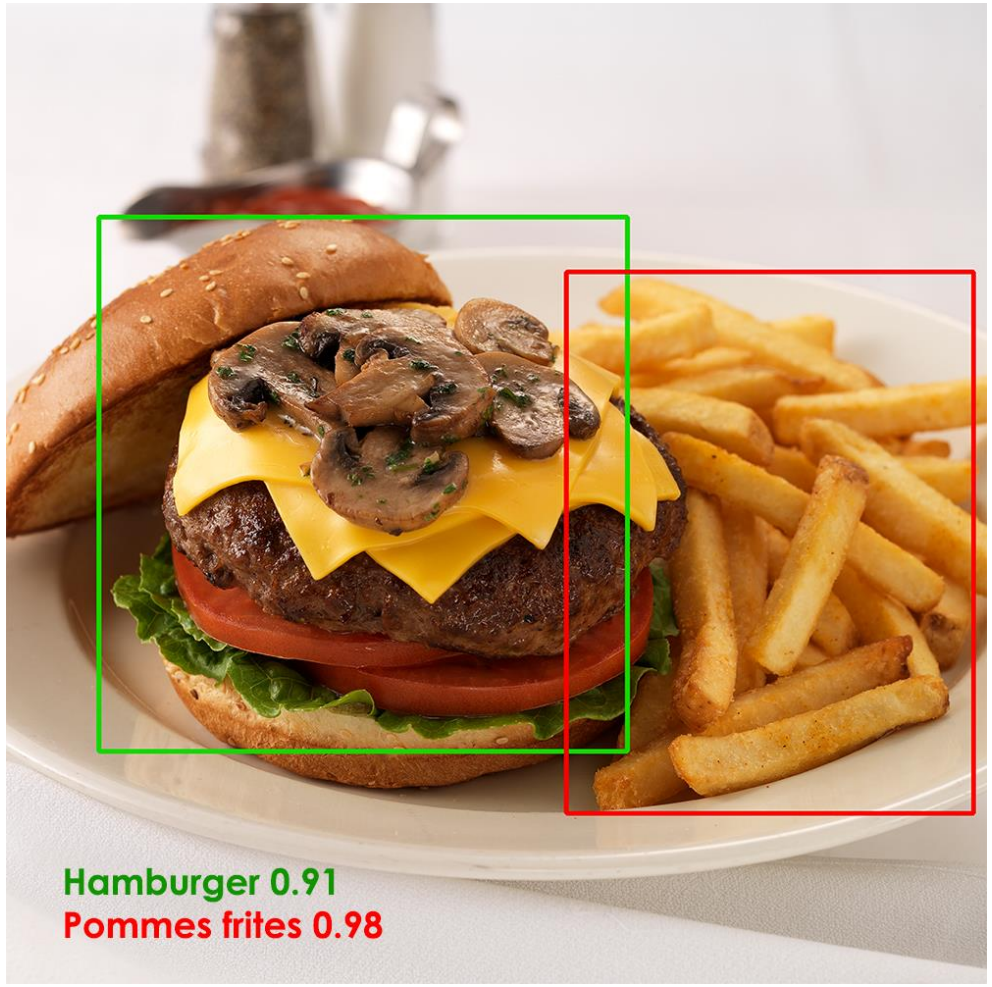
- Game modelling
- Opponent action prediction

# Mobile sensor data analysis (USA 2015)




- Activity recognition
- Blood pulse calculation

# Food image recognition (USA 2015)




- Recognition of several meals
- Recognition of ingredients inside

# Advertisement classification (RU 2016)

**Selling used smartphone**
**Meizu M1 Note for $100**

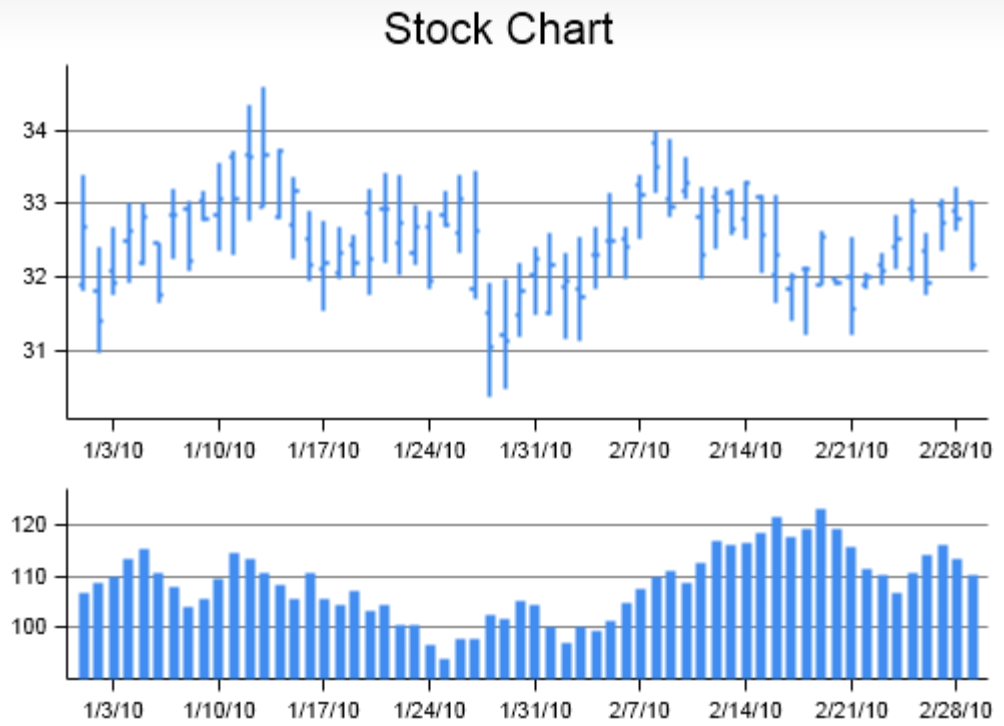
**Selling used smartphone**
**Meizu M1 Note for $100**

Black - not defined entity
Green - brand
Red - category
Blue - model

- Named entity recognition (NER) problem
- Over 100’000’000 advertisements

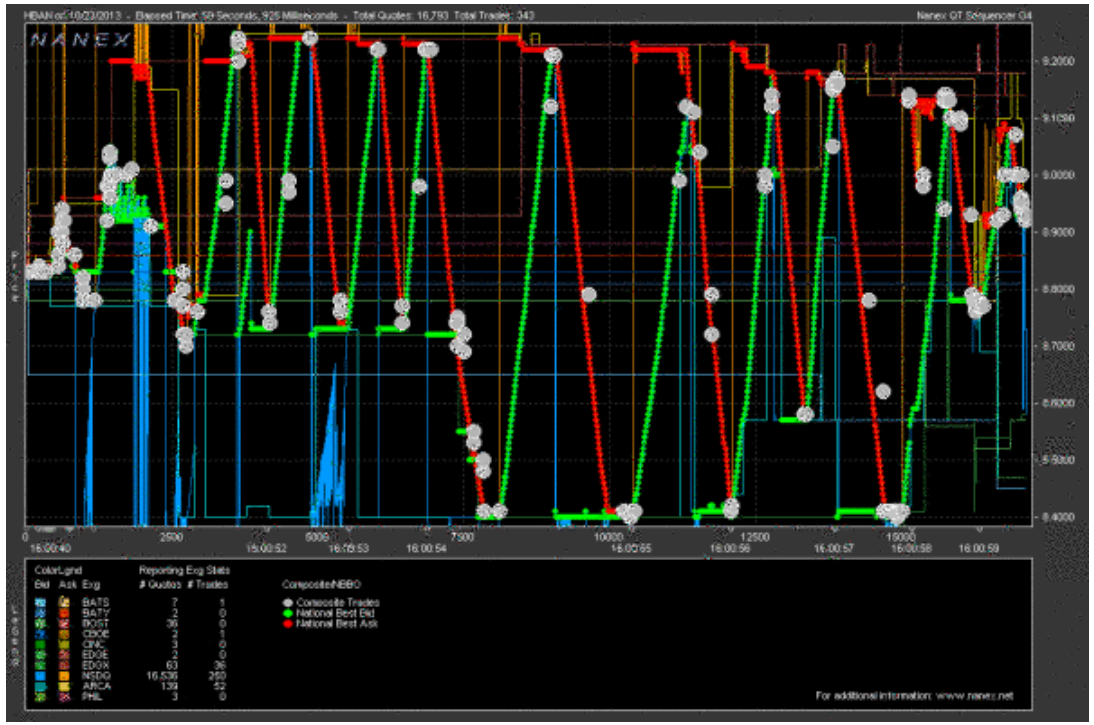
# Image style transfer (mlvch.com 2016)

# Financial forecasting challenge (2016)




- Numer.ai – forecasting from encrypted financial data
- In September achieved top 10% in public leaderboard

# Current research




High frequency trading


Conversational bots

# my mission

machines **work**, people create

# my mission

machines **work**, people create

machines **learn**, people discover

# my mission

machines **work**, people create

machines **learn**, people discover

machines **like**, people love

# your experience

# seminar goals

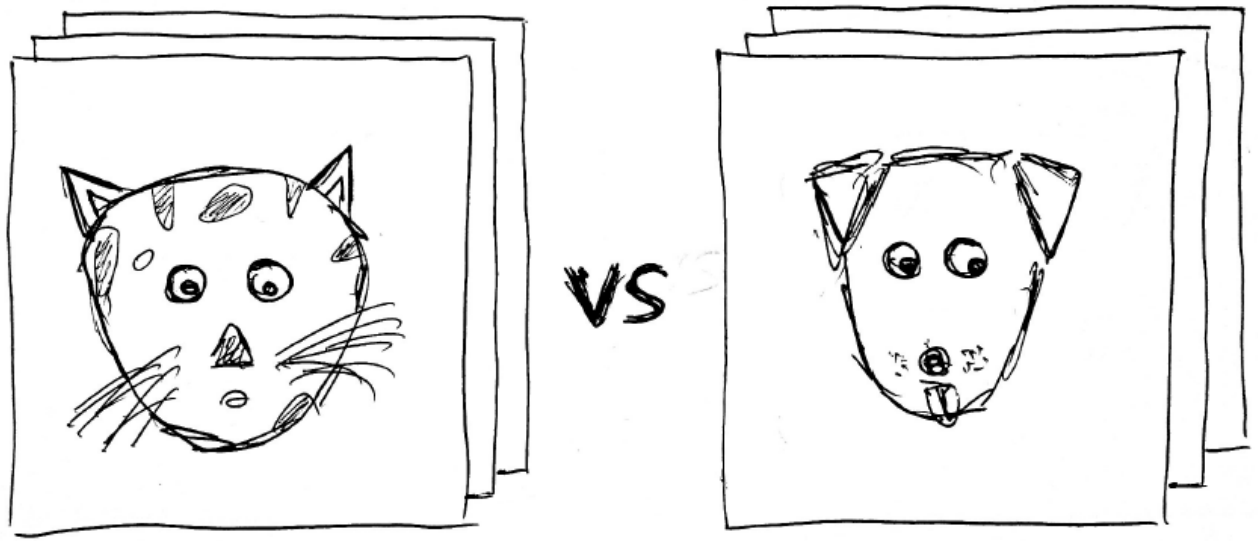
1. You **understand** how deep learning works
2. You create a **prototype** with **hands-on tools**
3. You know the path to **get the job** in machine learning

# Day 1 goals

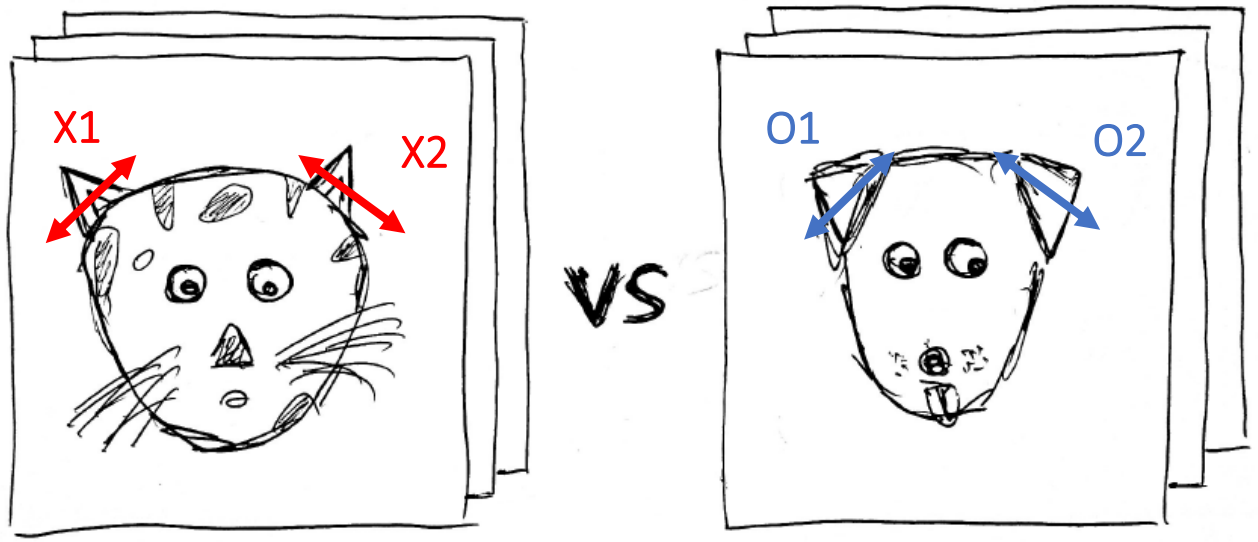
- You understand basics of machine learning
- You understand difference between machine learning and deep learning (most engineers don’t!)
- You can define machine learning problem with ERTP blocks

# machine learning in nutshell

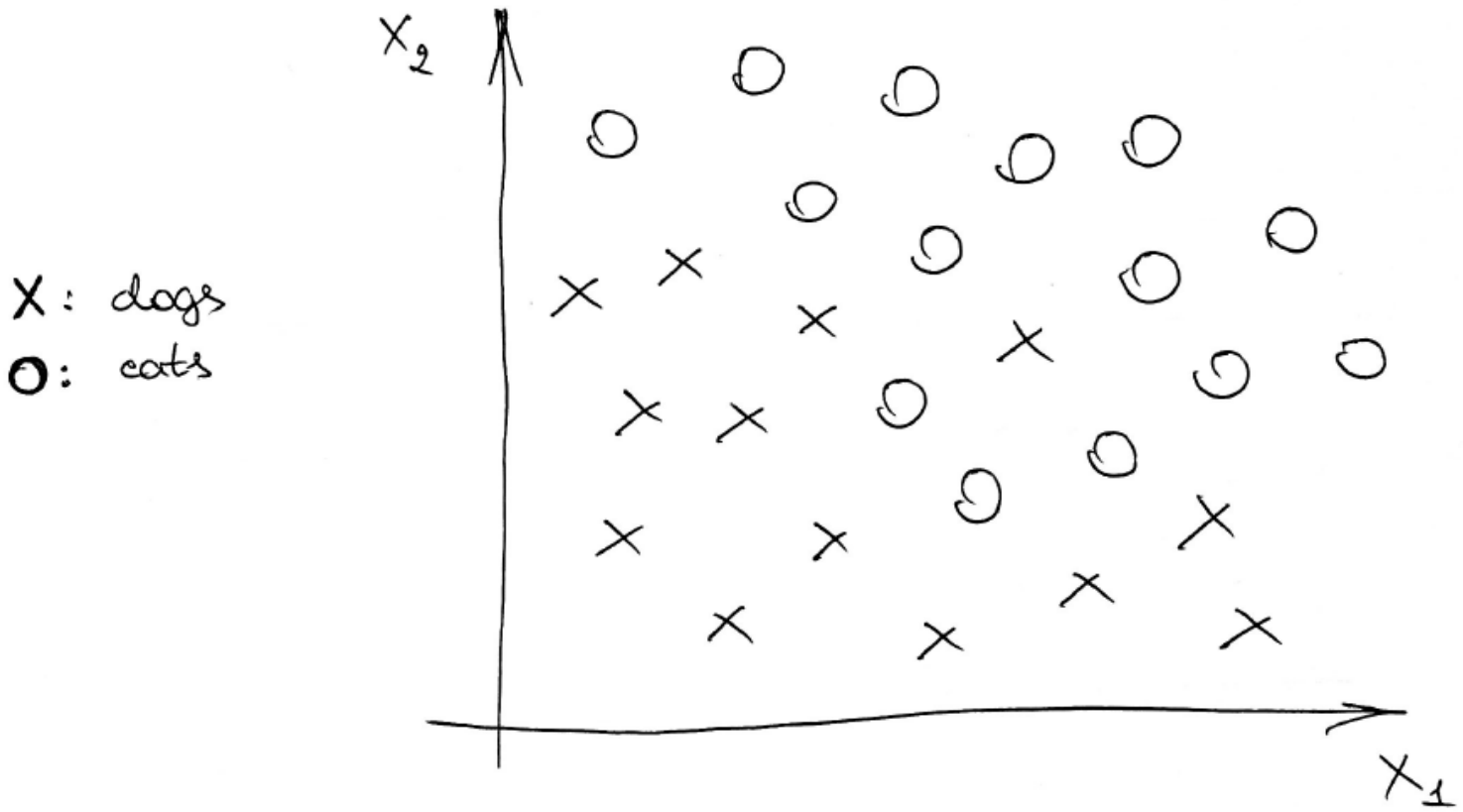
# Cats vs Dogs



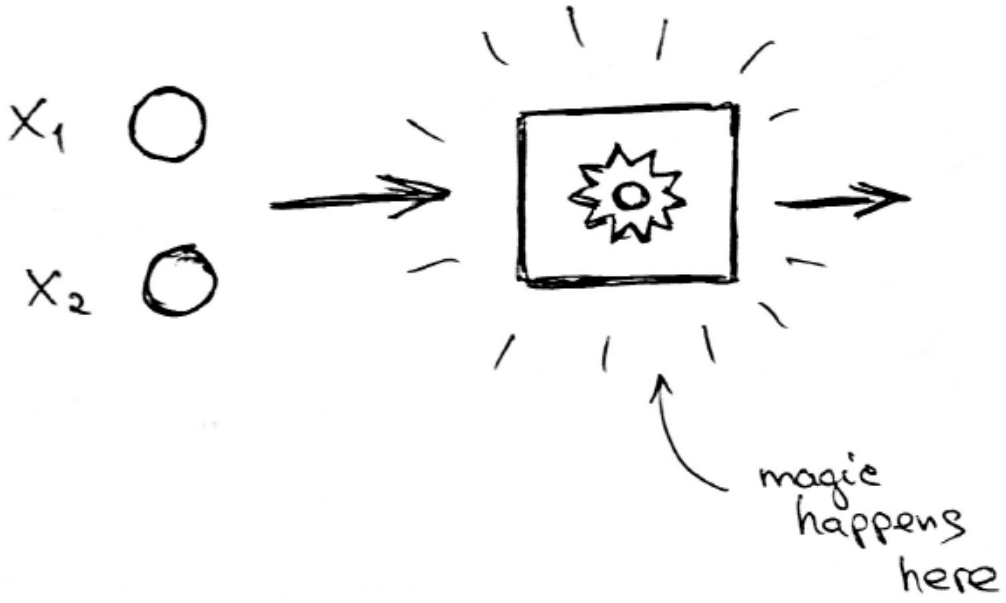
# Cats vs Dogs

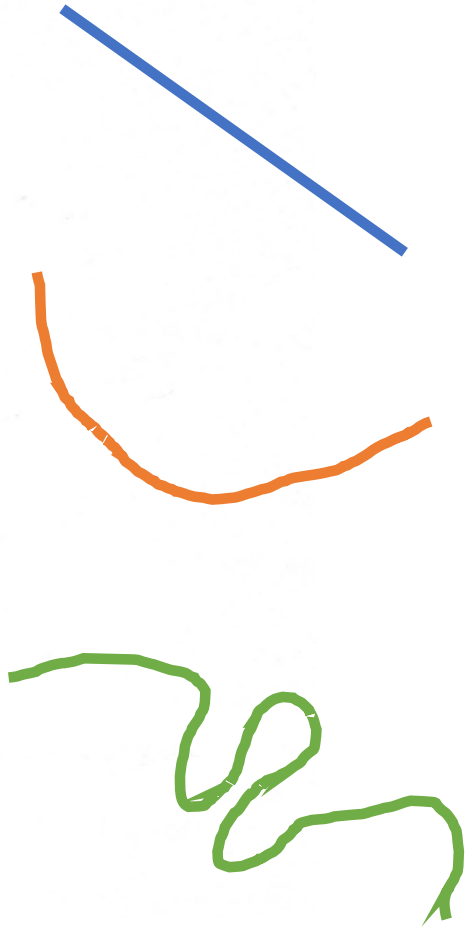


# Visual machine learning

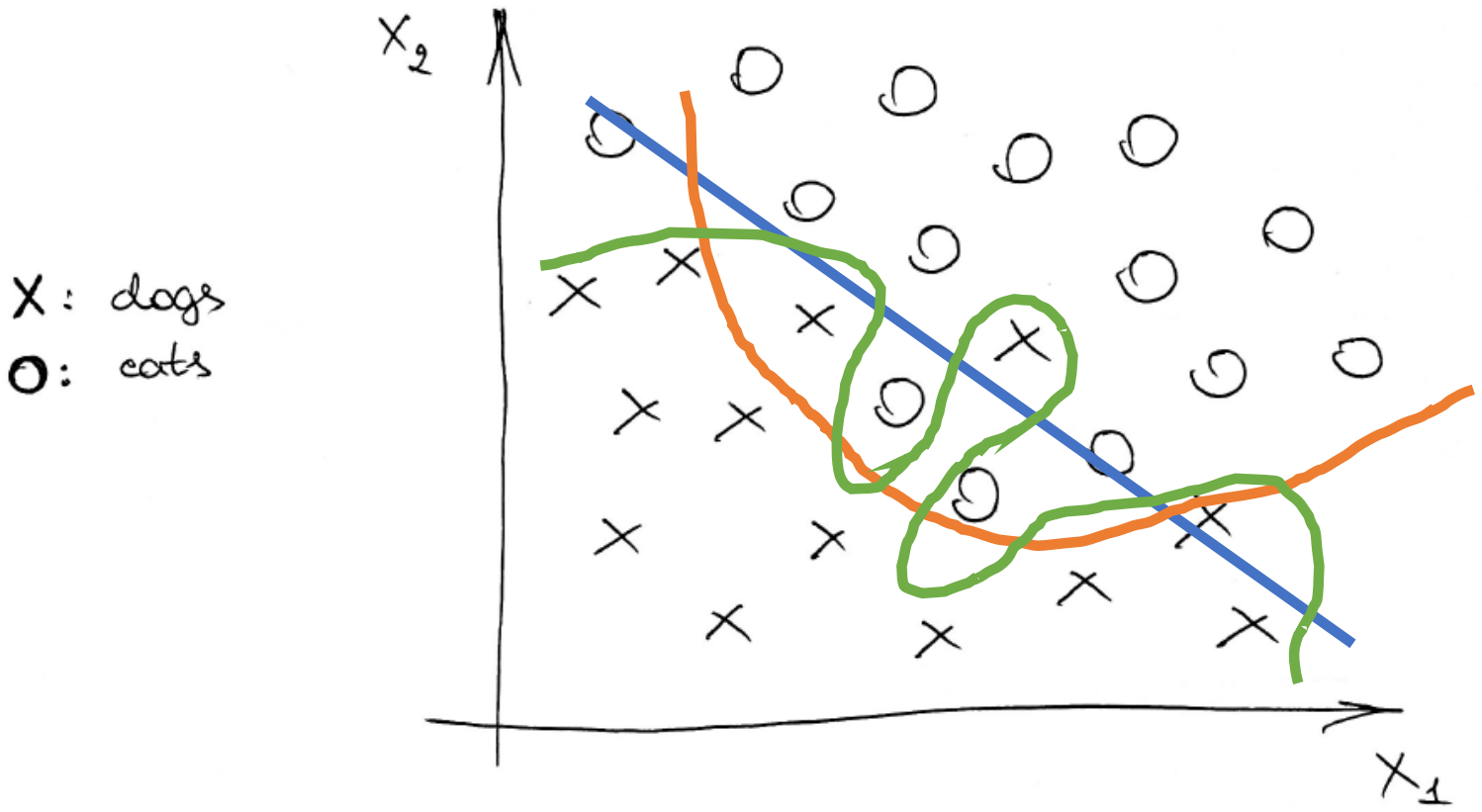


# Visual machine learning

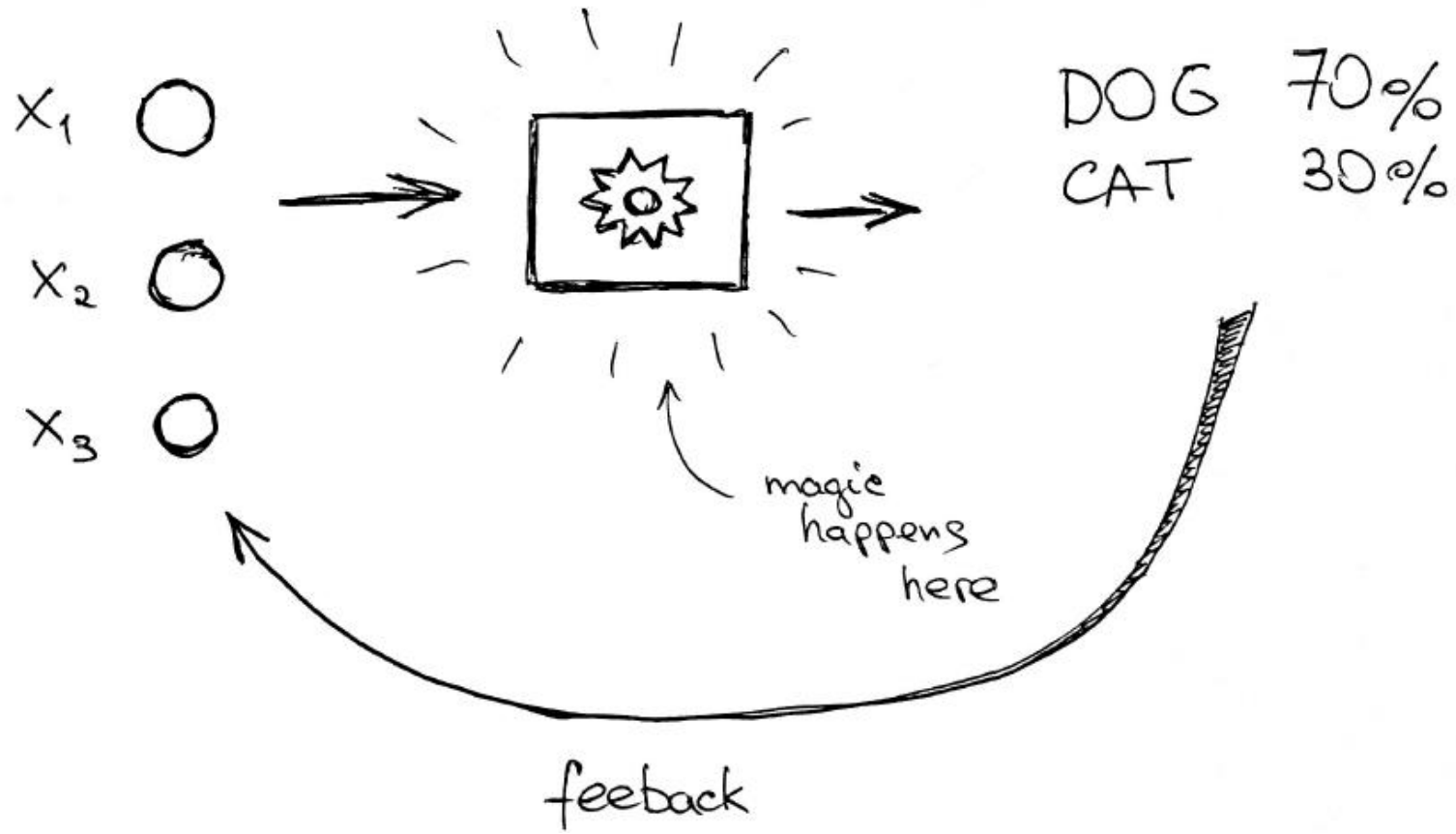


# Cats vs Dogs



# How it learns?



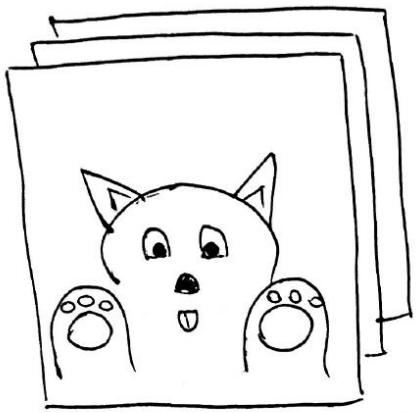
# résumé

Learning algorithm is just a **baby** – we show him examples of different objects and he tries to **generalize** them to correctly recognize new objects (separate one from another)

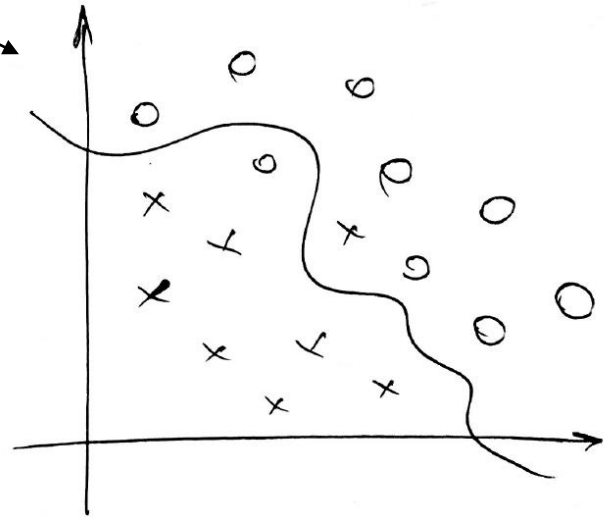
# Mathematical framework

- Program is said to learn from experience E with respect to some class of **tasks T** and performance **measure P** if performance improves

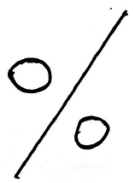
Mitchell, 1997



$\{ \mathbb{R}^{250 \times 250} \}$



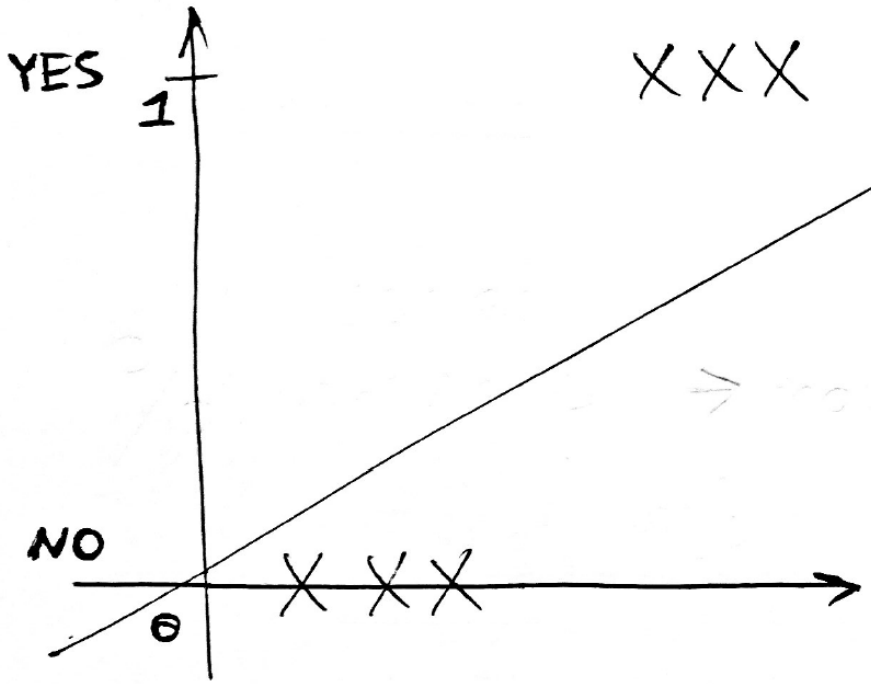
$f: \mathbb{R}^{250 \times 250} \rightarrow \{cat, dog\}$



correct answers → max

{cat, dog}

# Logistic regression




$$f = ax + b$$

# Logistic regression

## Hypothesis

$$h_w(x) = W^T x$$

$$W_0 + W_1 x$$

$$W_0 + W_1 x_1 + W_2 x_2$$

$$W_0 + W_1 x_1 + W_2 x_2^2 + W_3 x_2^2$$

$$W_0 + W_1 x_1 + W_2 x_2 + W_3 x_2^2 + W_4 x_2^2$$

## Threshold classifier

if $h_w(x) \geq 0.5$:

- "y = YES"

if $h_w(x) < 0.5$:

- "y = NO"

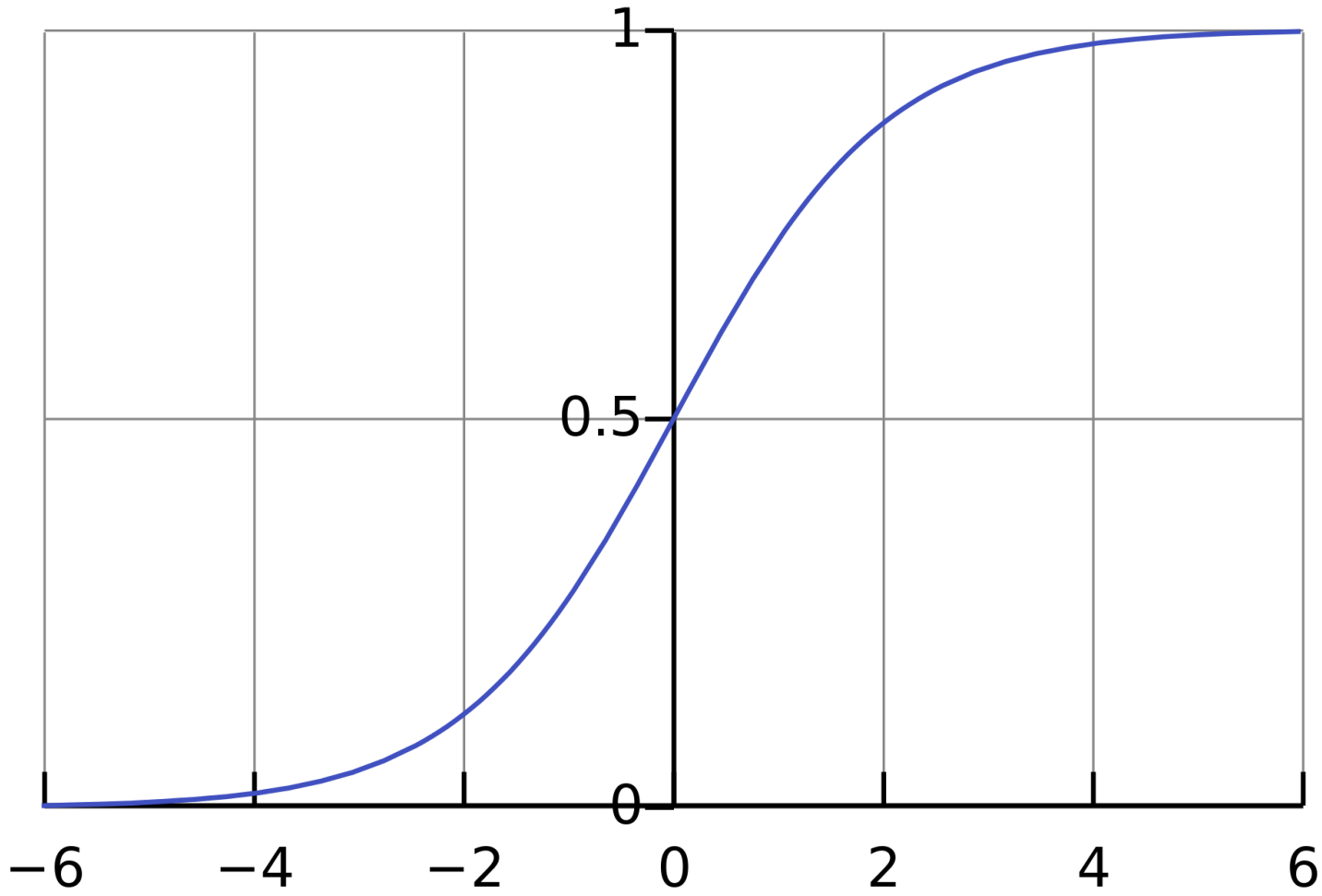
# Logistic regression

$$h_w(x) \in [0, 1]$$

$$h_w(x) = g(W^T x)$$

$$g(z) = \frac{1}{e^{-z} + 1} \Longrightarrow h_w(x) = \frac{1}{1 + e^{-W^T x}}$$

# Logistic function



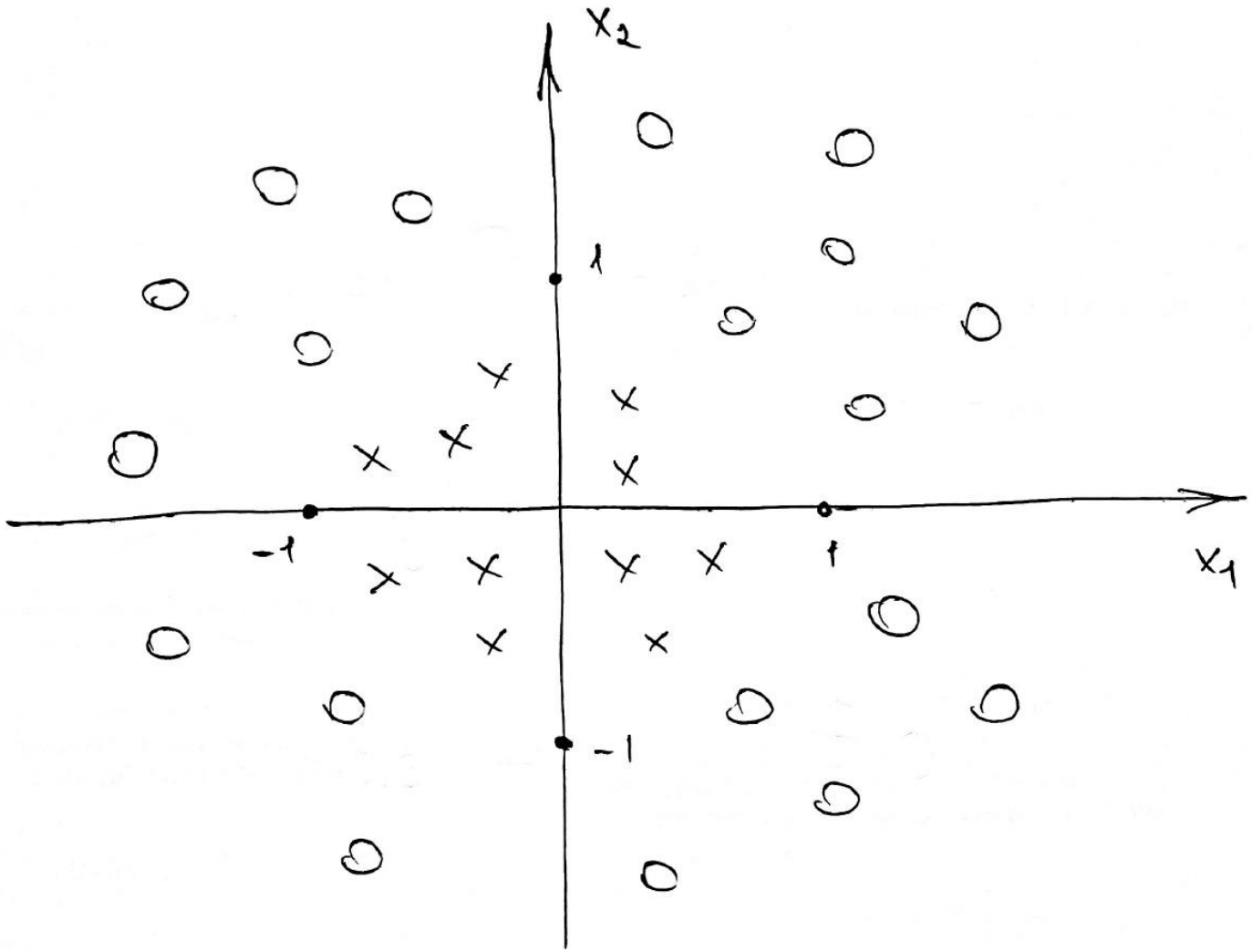
# Logistic regression

$$h_w(x) = P(y = 1 \mid x; W)$$

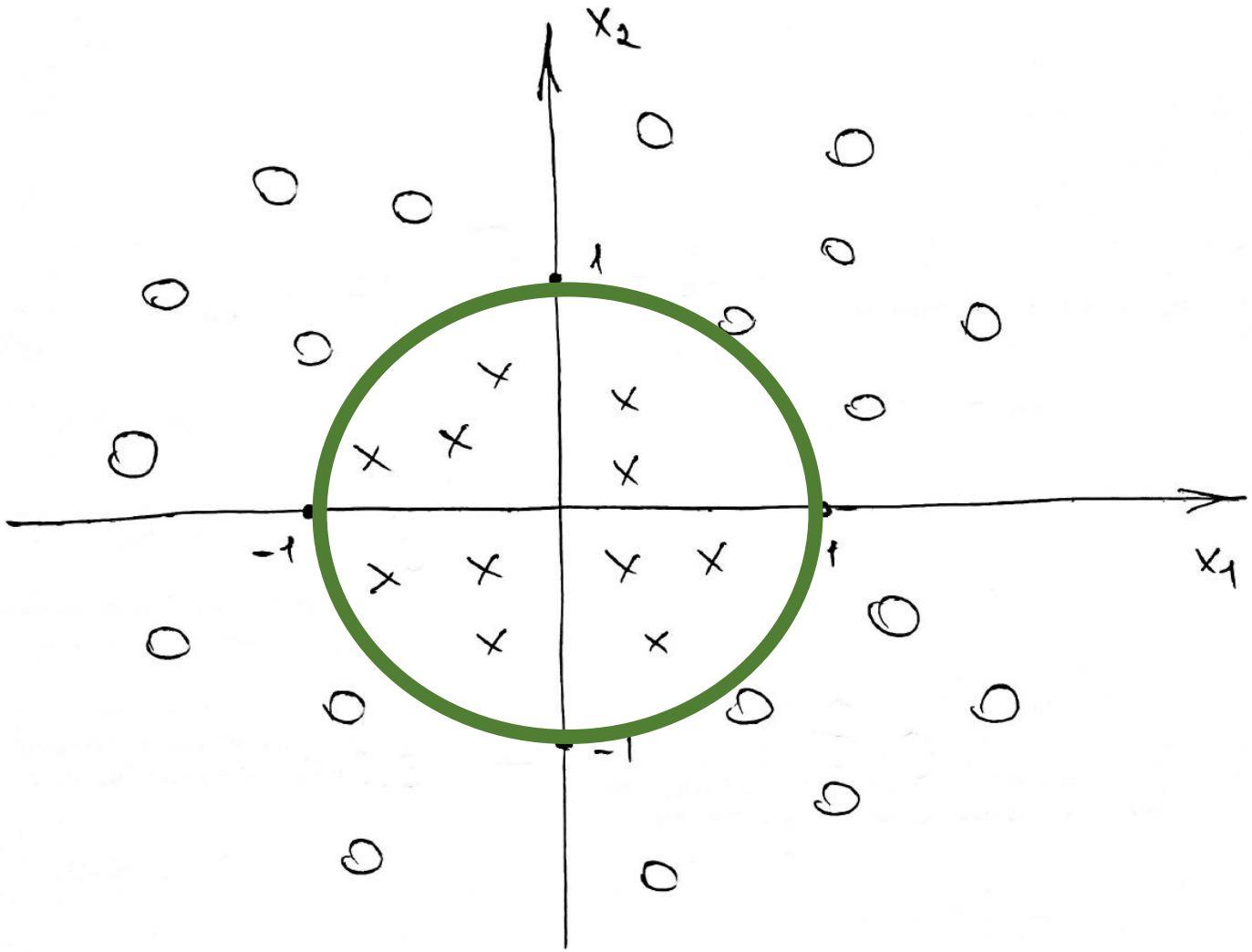
$$g(W^T x)$$

$$h_w(x) \geq 0{,}5 \Rightarrow W^T x \geq 0$$

# Logistic regression



# Logistic regression



# Logistic regression

$$h_w(x) = g\left(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2\right)$$

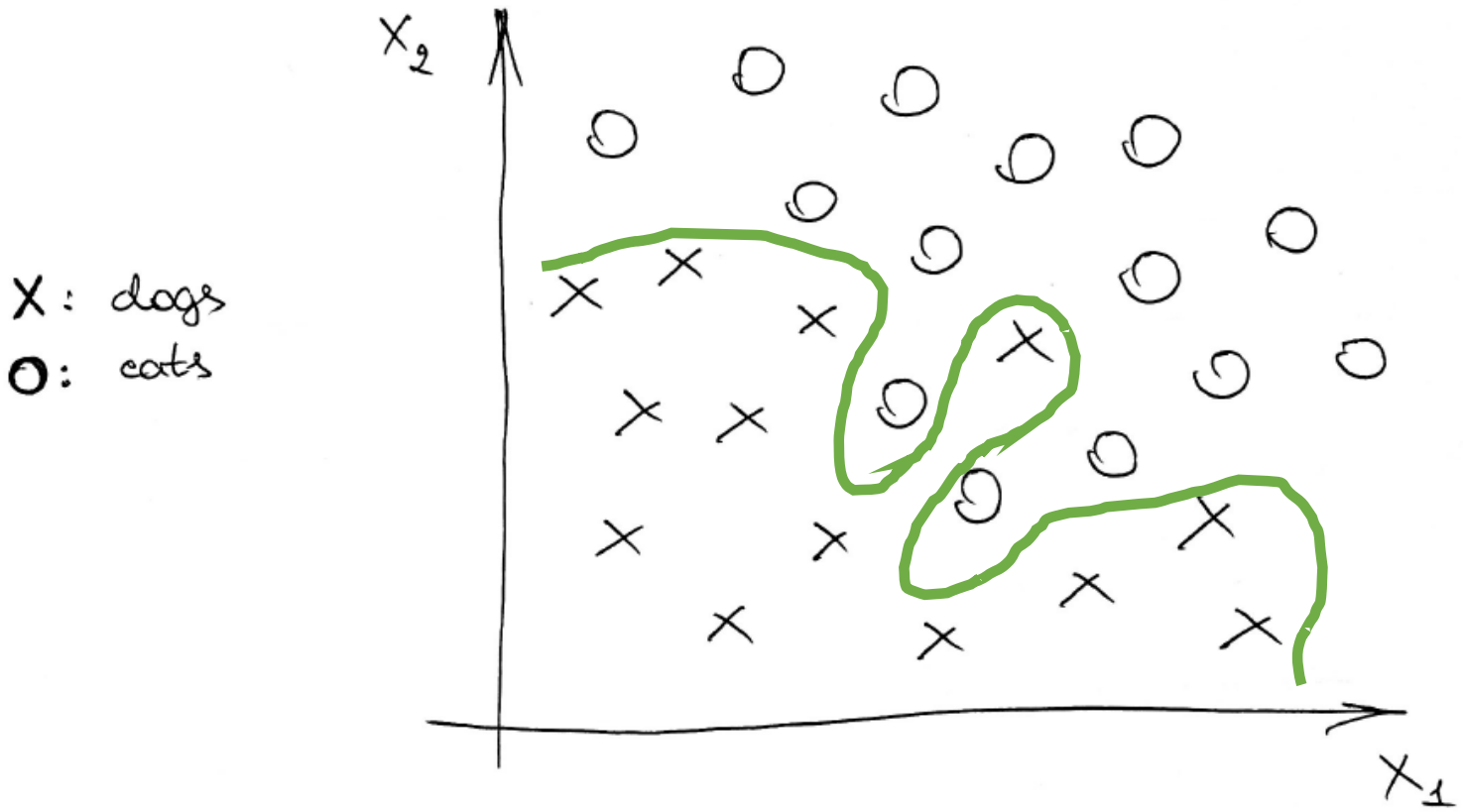
$$W = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

If $-1 + x_1^2 + x_2^2 \geq 0 \Rightarrow$ "$y = 0$"

# Overfitting

$$h_w(x) = g\left( W_0 + W_1 x_1 + W_2 x_2 + \dots + W_{100000} x_1^{100000} + \dots \right)$$

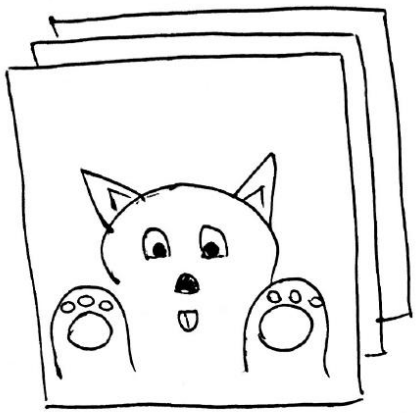
# Overfitting



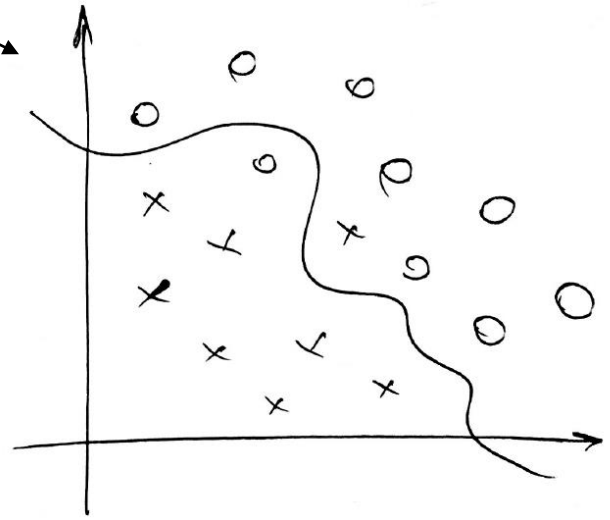
# Mathematical framework

- Program is said to learn from experience E with respect to some class of tasks T and performance measure P if performance improves

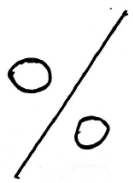
Mitchell, 1997



$\{ \mathbb{R}^{250 \times 250} \}$



$f: \mathbb{R}^{250 \times 250} \rightarrow \{cat, dog\}$



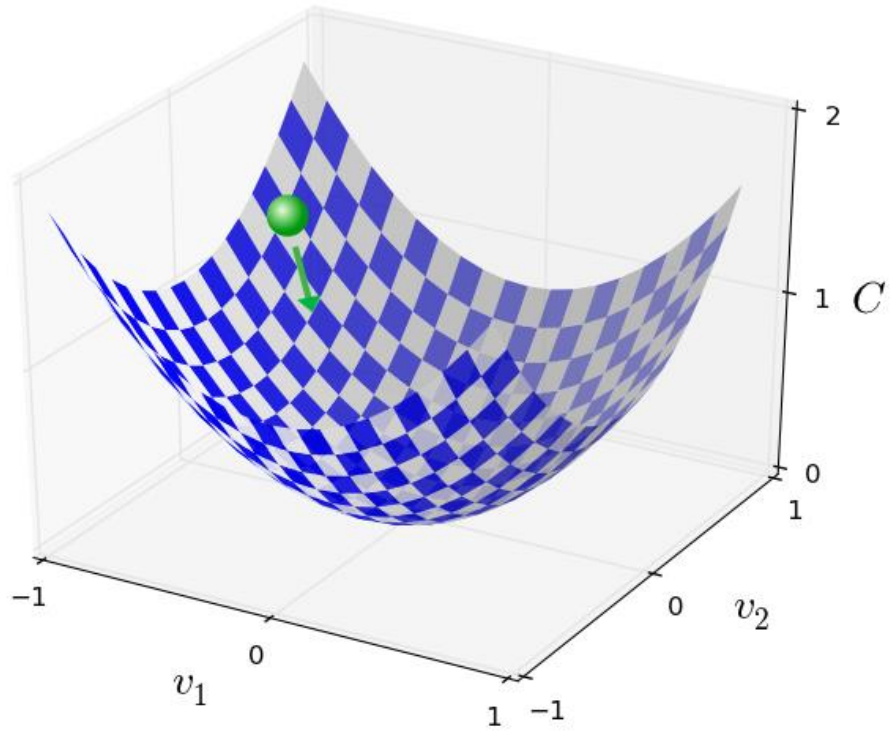
correct answers → max

{cat, dog}

# Measure

$$\frac{1}{2m} \sum_{i=1}^{m} \left( h_w(x^{(i)}) - y^{(i)} \right)^2$$

# How it learns?




- We want to minimize some $J(x1, x2)$
- **Gradient** $(\nabla J)$ – vector, showing the direction of the fastest **increasing** of J.
- To minimize, we have to move in the direction of **antigradient** $-\nabla C$ with some **step** $\lambda$.

# Gradient descent

$$\mathcal{J}(x_1, x_2) = x_1^2 + x_2^2$$

$$x^{(0)} = (5,5)^T$$

$$\lambda = 0.1$$

UPDATE RULE

$$x^{(k+1)} = x^{(k)} - \lambda \cdot \nabla \mathcal{J}(x^{(k)})$$

# Gradient descent

$$\nabla J(x^{(0)}) = \begin{pmatrix} 2 \cdot x_1^{(0)} \\ 2 \cdot x_2^{(0)} \end{pmatrix} = \begin{pmatrix} 10 \\ 10 \end{pmatrix}$$

$$x^{(1)} = \begin{pmatrix} 5 \\ 5 \end{pmatrix} - \lambda \cdot \begin{pmatrix} 10 \\ 10 \end{pmatrix} = \begin{pmatrix} 5 \\ 5 \end{pmatrix} - 0.1 \cdot \begin{pmatrix} 10 \\ 10 \end{pmatrix} =$$

$$= \begin{pmatrix} 5 \\ 5 \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 4 \\ 4 \end{pmatrix}$$

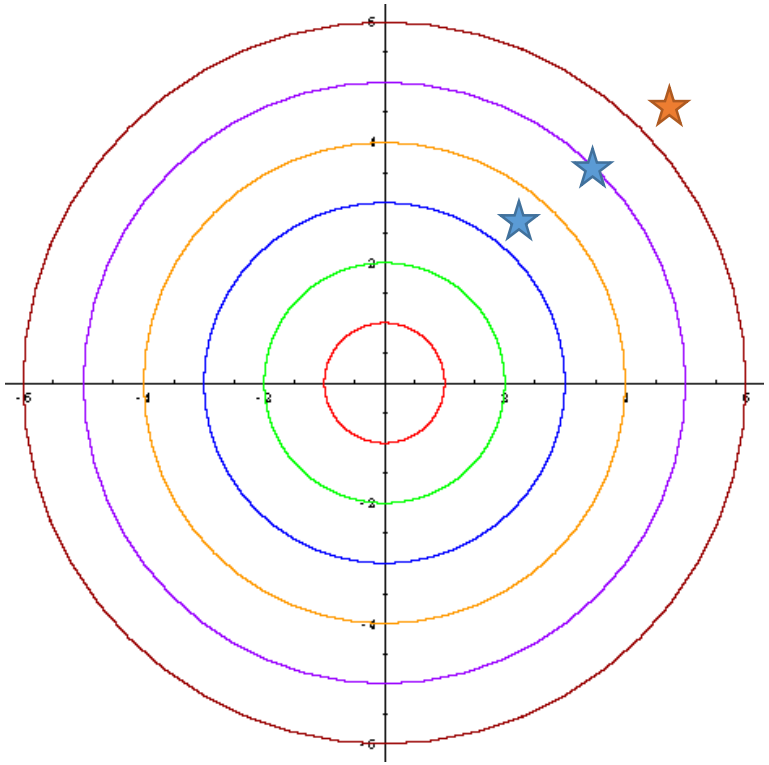
# Gradient descent

$$\nabla J(x^{(1)}) = \begin{pmatrix} 2 \cdot x_1^{(1)} \\ 2 \cdot x_2^{(1)} \end{pmatrix} = \begin{pmatrix} 8 \\ 8 \end{pmatrix}$$

$$x^{(2)} = \begin{pmatrix} 4 \\ 4 \end{pmatrix} - \lambda \cdot \begin{pmatrix} 8 \\ 8 \end{pmatrix} = \begin{pmatrix} 4 \\ 4 \end{pmatrix} - 0.1 \cdot \begin{pmatrix} 8 \\ 8 \end{pmatrix} =$$

$$= \begin{pmatrix} 4 \\ 4 \end{pmatrix} - \begin{pmatrix} 0.8 \\ 0.8 \end{pmatrix} = \begin{pmatrix} 3.2 \\ 3.2 \end{pmatrix}$$

# Gradient descent



# Gradient descent

$$h_w(x_1, x_2) = W_0 + W_1 x_1 + W_2 x_2$$

$$J(W_0, W_1, W_2) = \frac{1}{2m} \sum_{i=1}^{m} \left( h_w(x_1^{(i)}, x_2^{(i)}) - y^{(i)} \right)^2$$

$$\frac{\partial}{\partial W_0} J(W_0, W_1, W_2) = \frac{\partial}{\partial W_0} \frac{1}{2m} \sum_{i=1}^{m} \left( h_w(\bar{x}^{(i)}) - y^{(i)} \right)^2 =$$

$$= \frac{\partial}{\partial W_0} \frac{1}{2m} \sum_{i=1}^{m} \left( (W_0 + W_1 x_1^{(i)} + W_2 x_2^{(i)})^2 - 2y^{(i)}(W_0 + W_1 x_1^{(i)} + W_2 x_2^{(i)}) + y^{(i)2} \right) =$$

$$= \frac{1}{2m} \sum_{i=1}^{m} \left( 2(W_0 + W_1 x_1^{(i)} + W_2 x_2^{(i)}) - 2y^{(i)} \right) =$$

$$= \frac{1}{m} \sum_{i=1}^{m} \left( W_0 + W_1 x_1^{(i)} + W_2 x_2^{(i)} - y^{(i)} \right) =$$

$$= \frac{1}{m} \sum_{i=1}^{m} \left( h_w(x_1^{(i)}, x_2^{(i)}) - y^{(i)} \right)$$

# Gradient descent

$$\frac{\partial}{\partial w_0} J(\bar{w}) = \frac{1}{m} \sum_{i=1}^{m} \left( h_w(\bar{x}^{(i)}) - y^{(i)} \right)$$

$$\frac{\partial}{\partial w_1} J(\bar{w}) = \frac{1}{m} \sum_{i=1}^{m} \left( h_w(\bar{x}^{(i)}) - y^{(i)} \right) x_1^{(i)}$$

$$\frac{\partial}{\partial w_2} J(\bar{w}) = \frac{1}{m} \sum_{i=1}^{m} \left( h_w(\bar{x}^{(i)}) - y^{(i)} \right) x_2^{(i)}$$

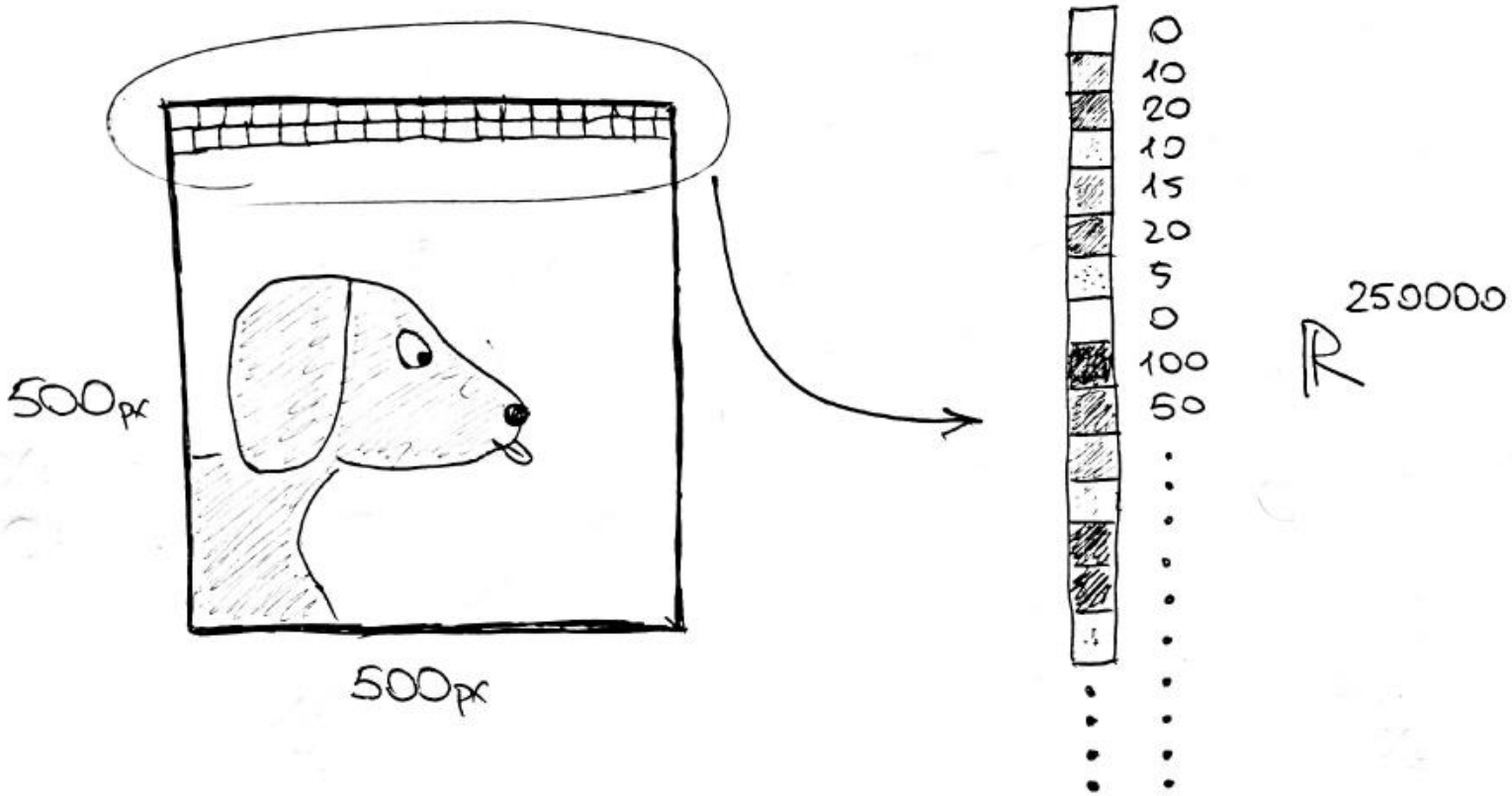
# How it learns?

Program is said to learn from **experience E** (data set with pictures) with respect to some class of **tasks T** (logistic regression) and performance **measure P** (mean squared error) if performance **improves** (with gradient descent iterations)

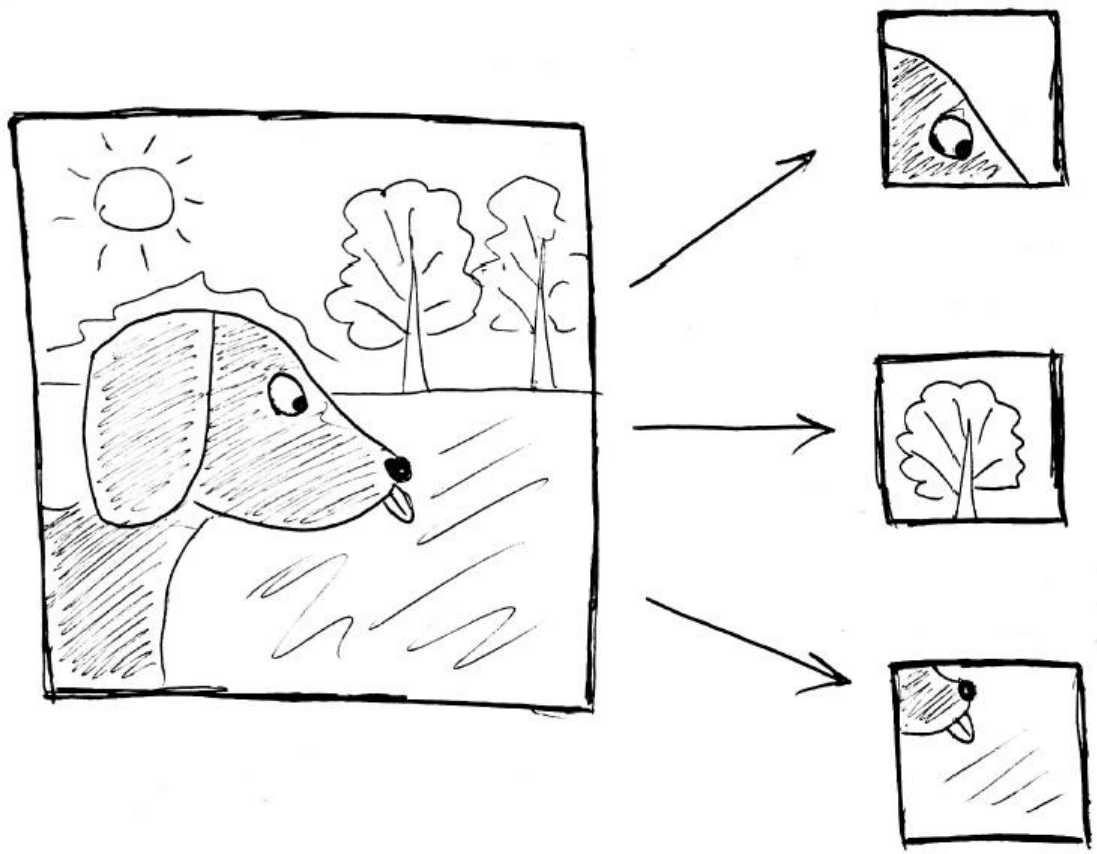
Mitchell, 1997

# deep learning in nutshell

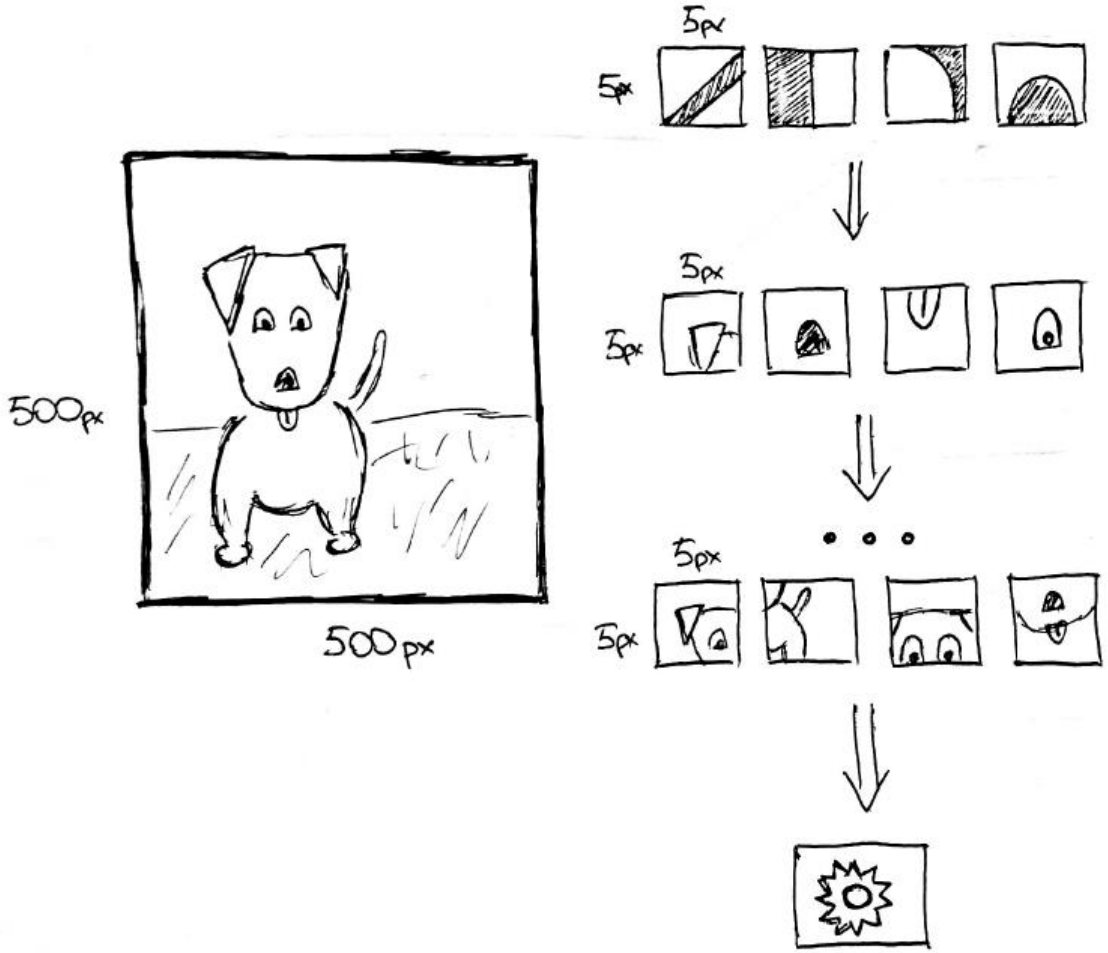
# Image representation



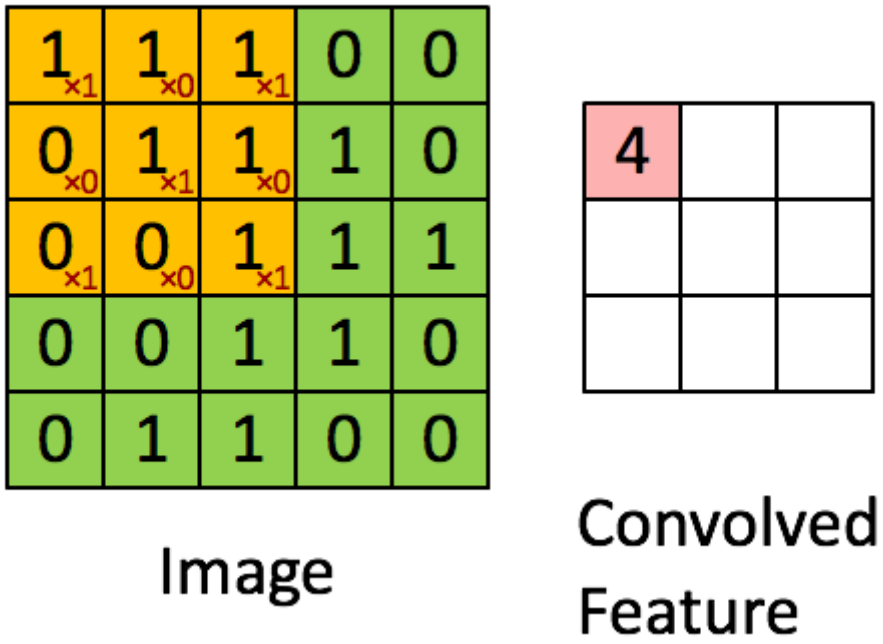
# Image representation



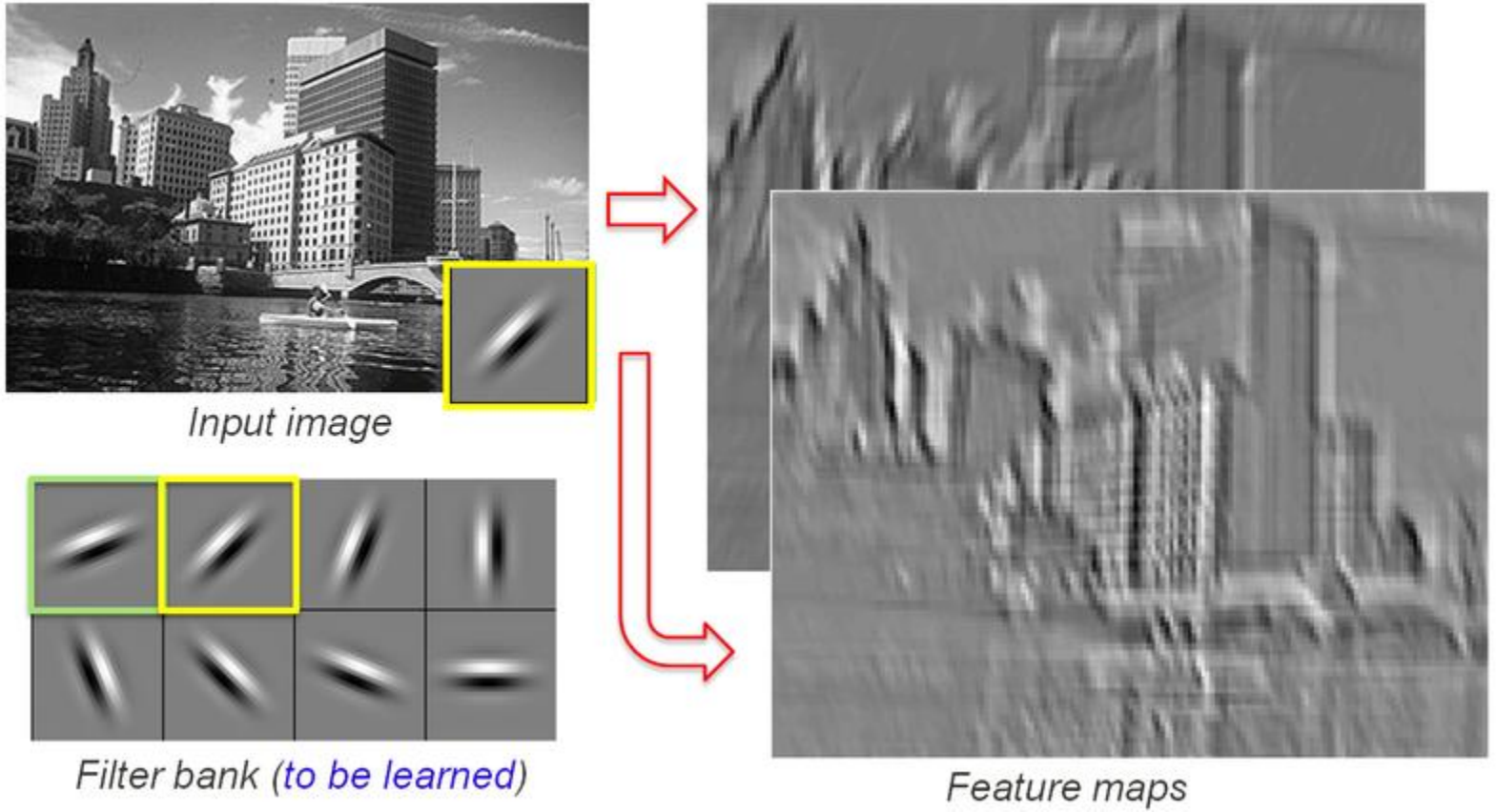
# Image representation



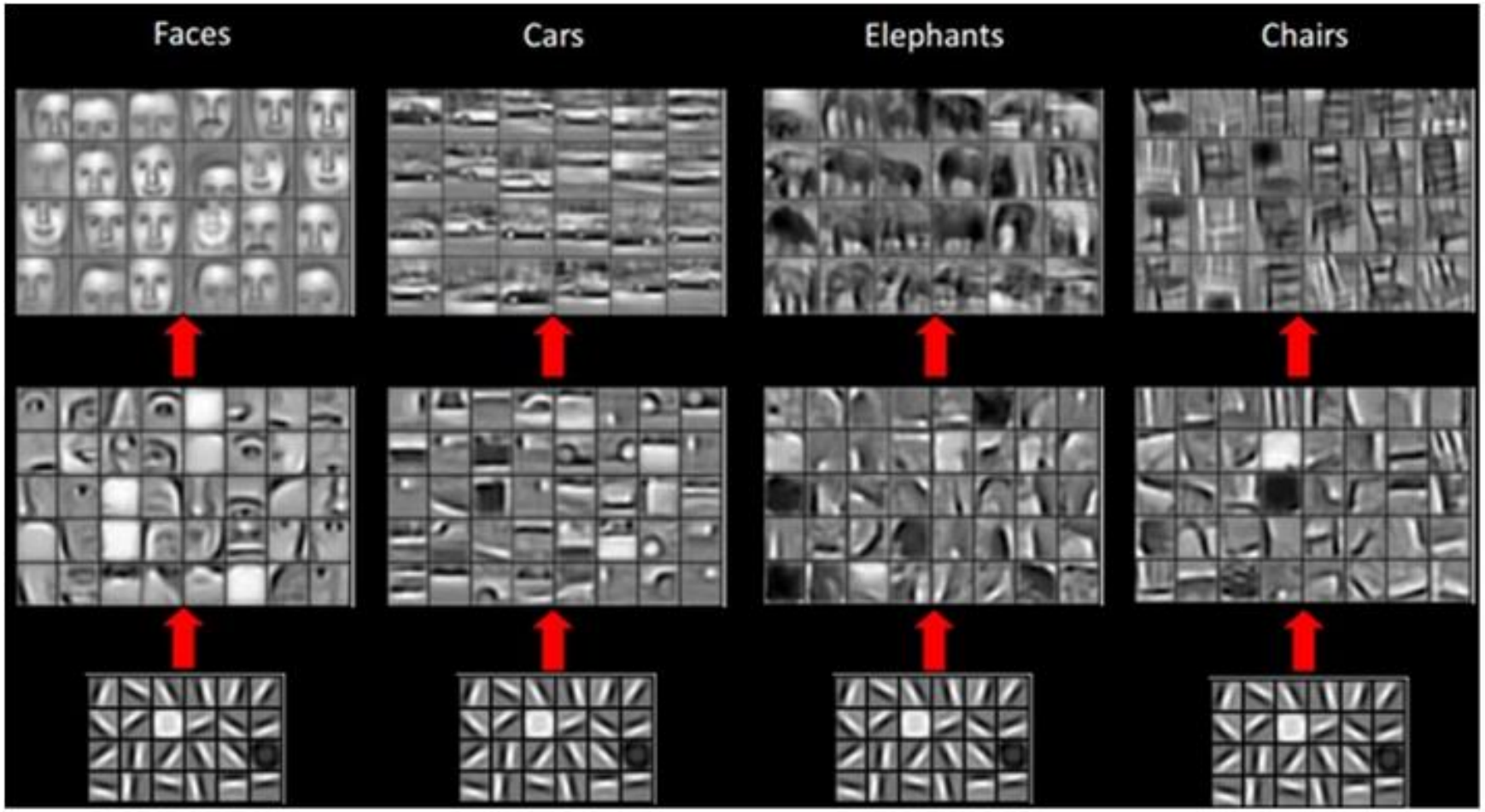
# Image representation



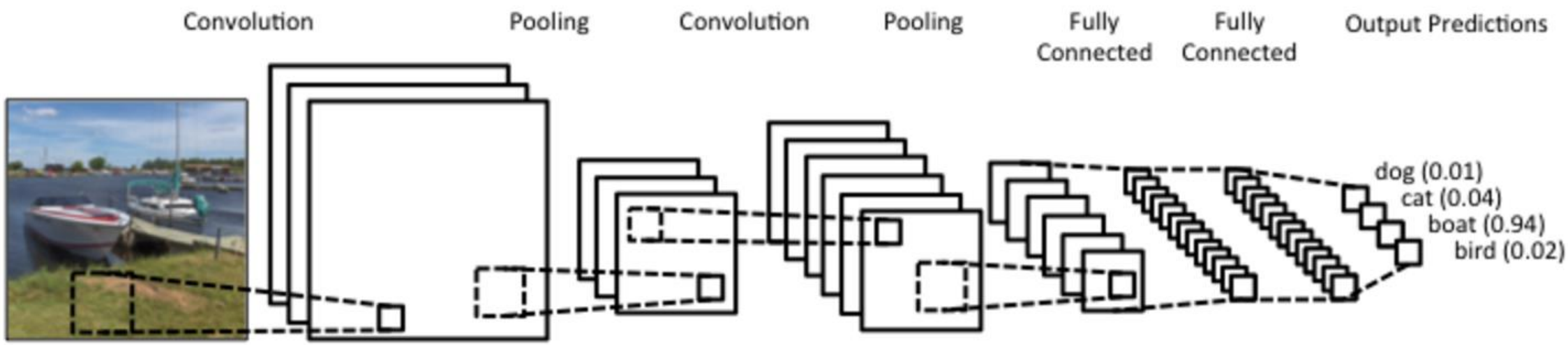
# Image representation



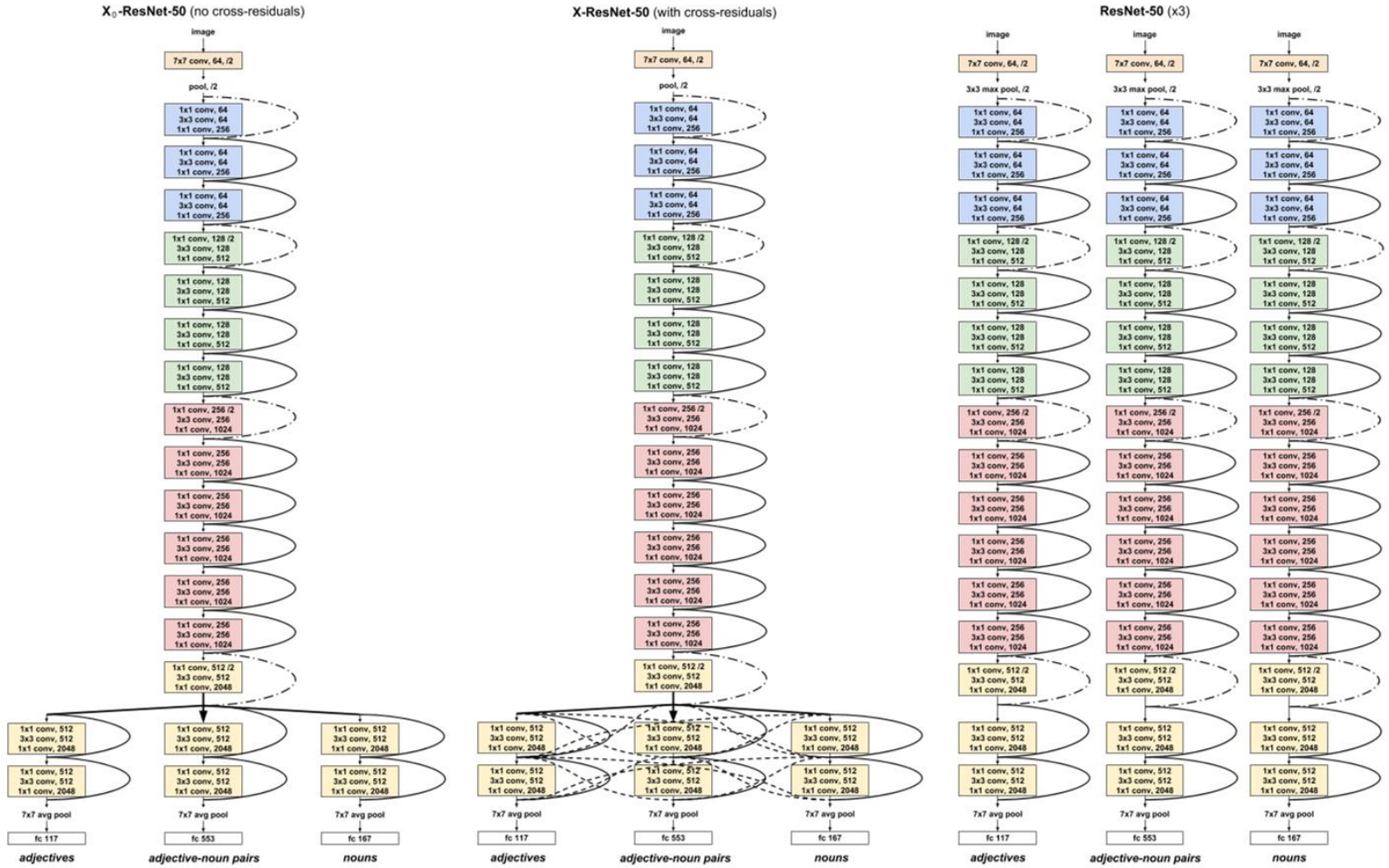
# Image representation



# Convolutional neural network



# Convolutional neural network



# résumé

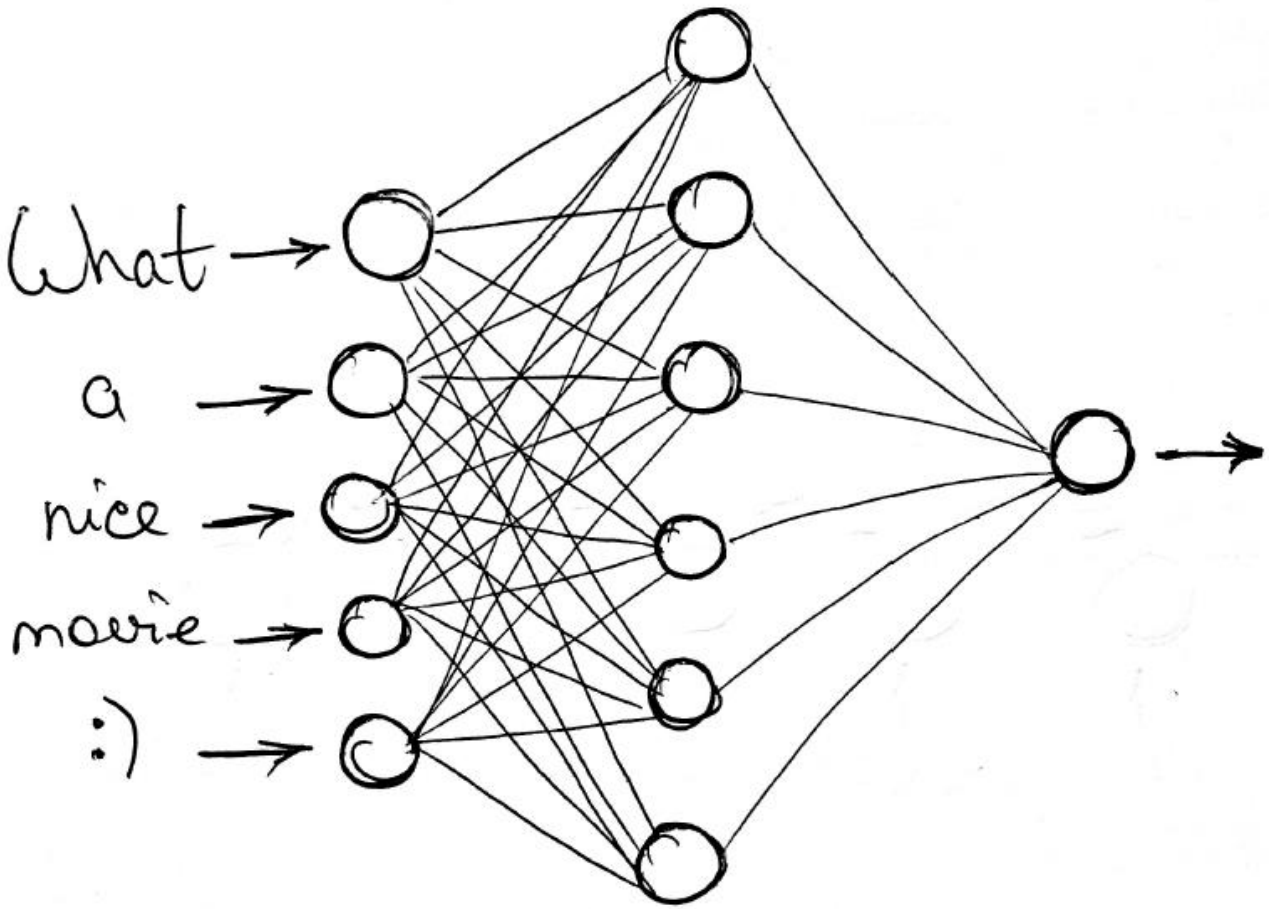
We want to extract the most **meaningful** information from the images to pass them to some algorithm that can correctly split these representations into different categories

# Positive vs Negative comment

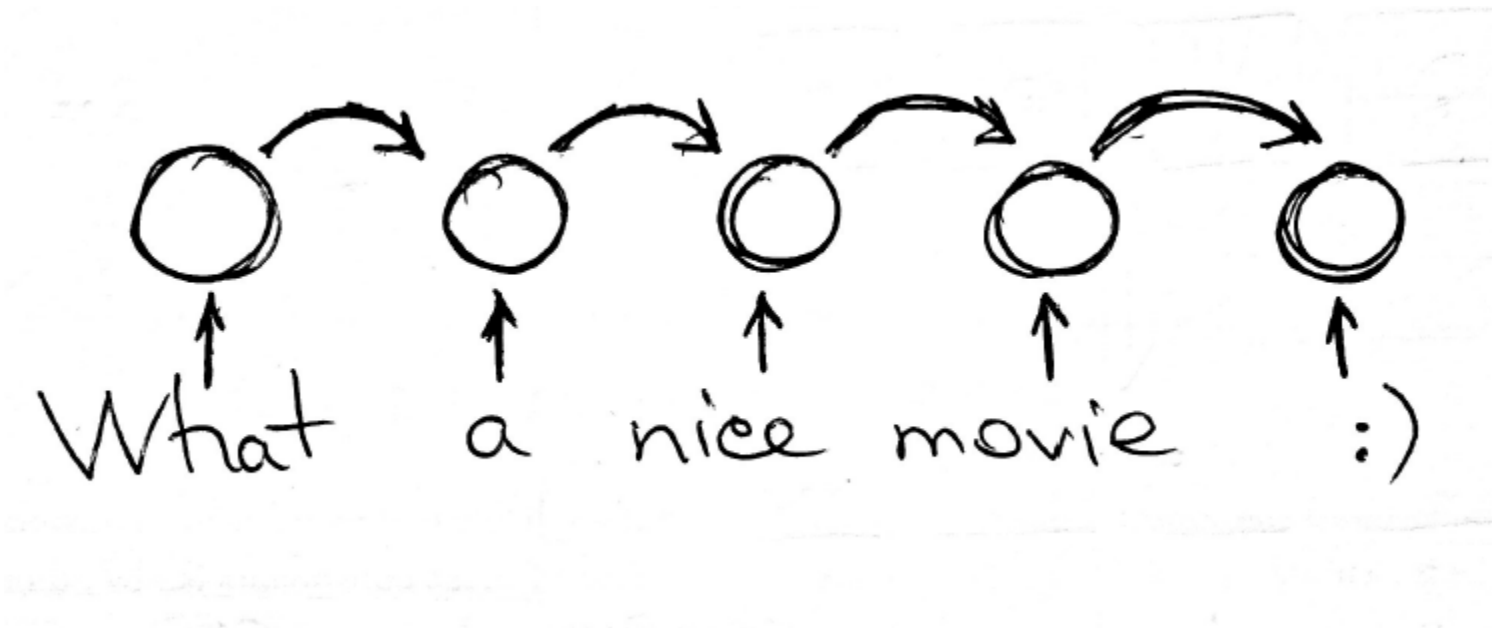

- "Amazing movie, really loved it, j'adore!!! :)"
- "Scheiss schrecklich! I wish Jason Statham played Bond, Connery is so bad :("

Positive or Negative?

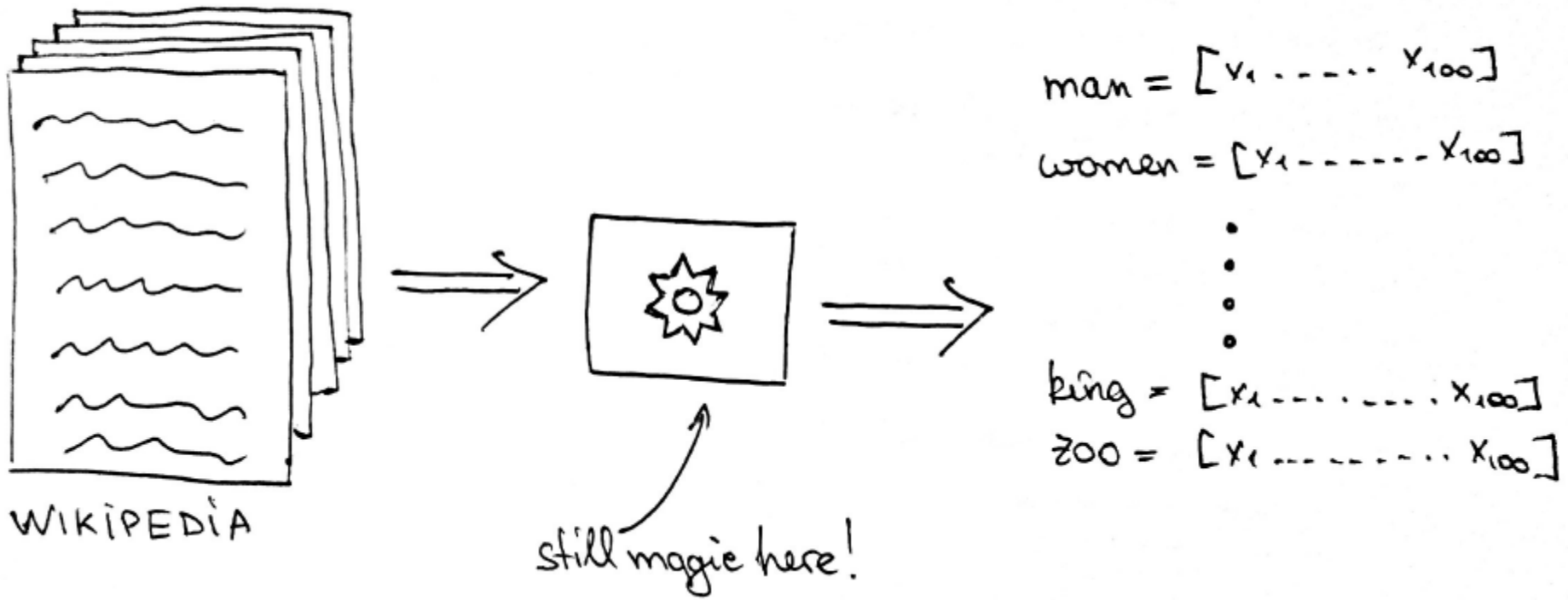
# Text to neural network



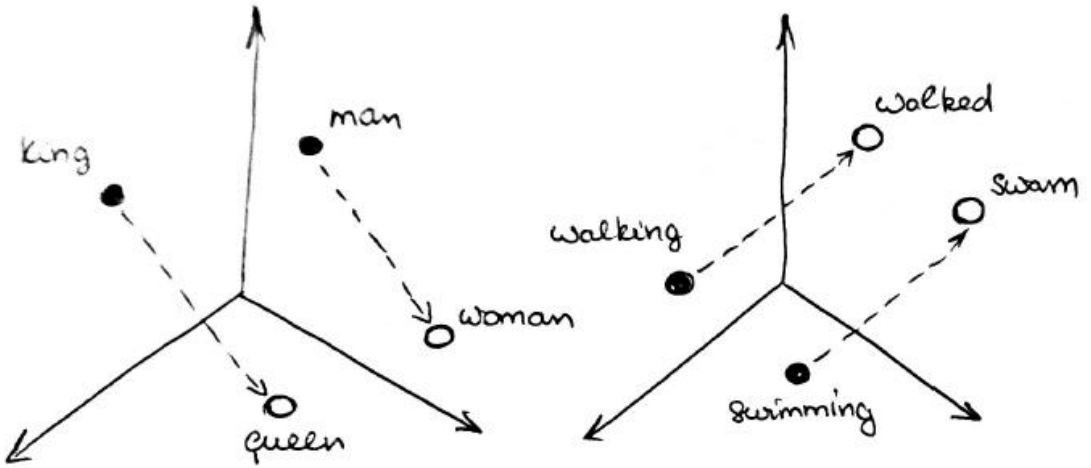
# Text to **recurrent** neural network



# How to represent words?



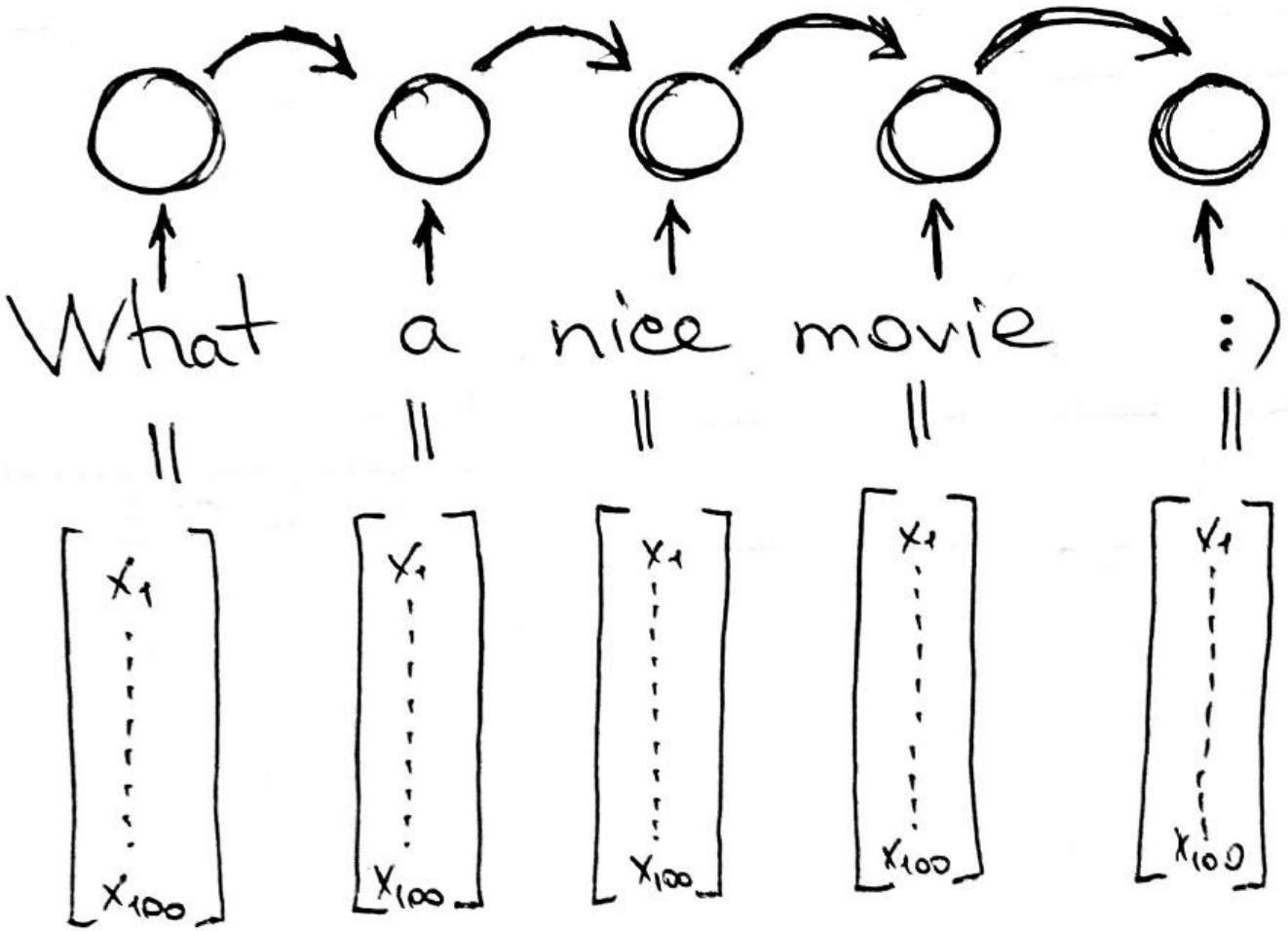
# How to represent words?



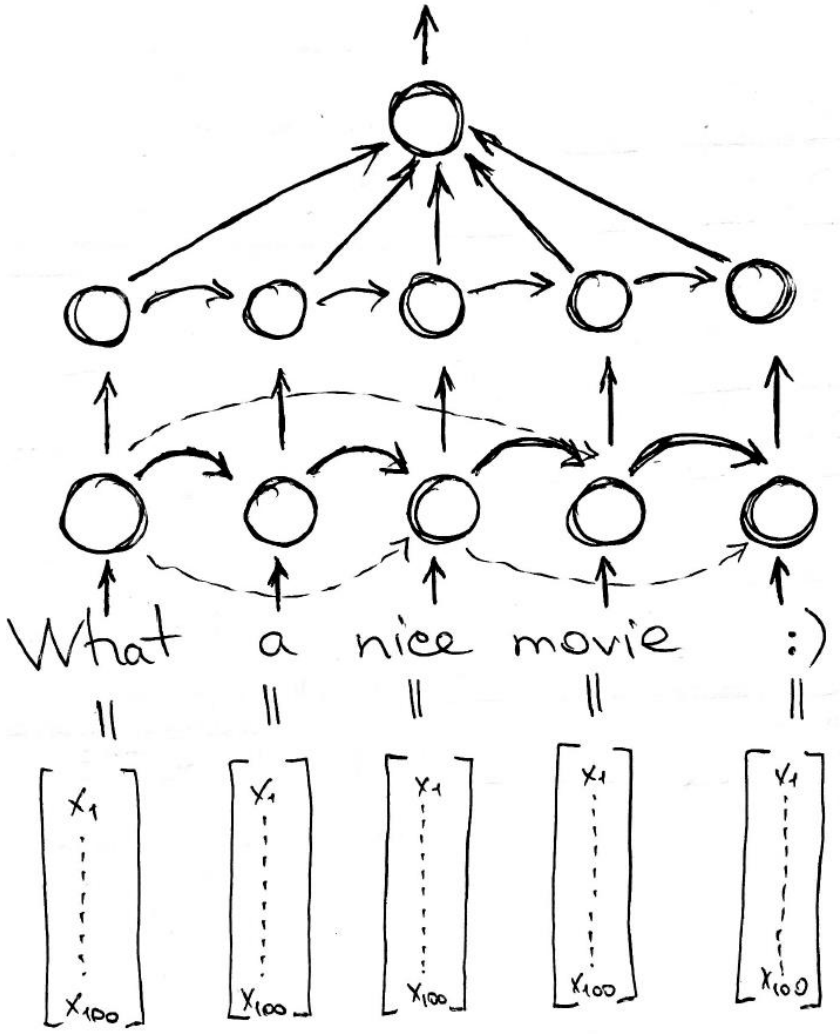

Spain — Madrid
Italy — Rome
Germany — Berlin
Ukraine — Kiev
Japan — Tokyo
Russia — Moscow
Turkey — Ankara

Iraq – Violence = Jordan
Human – Animal = Ethics
President – Power = Prime Minister
Library – Books = Hall

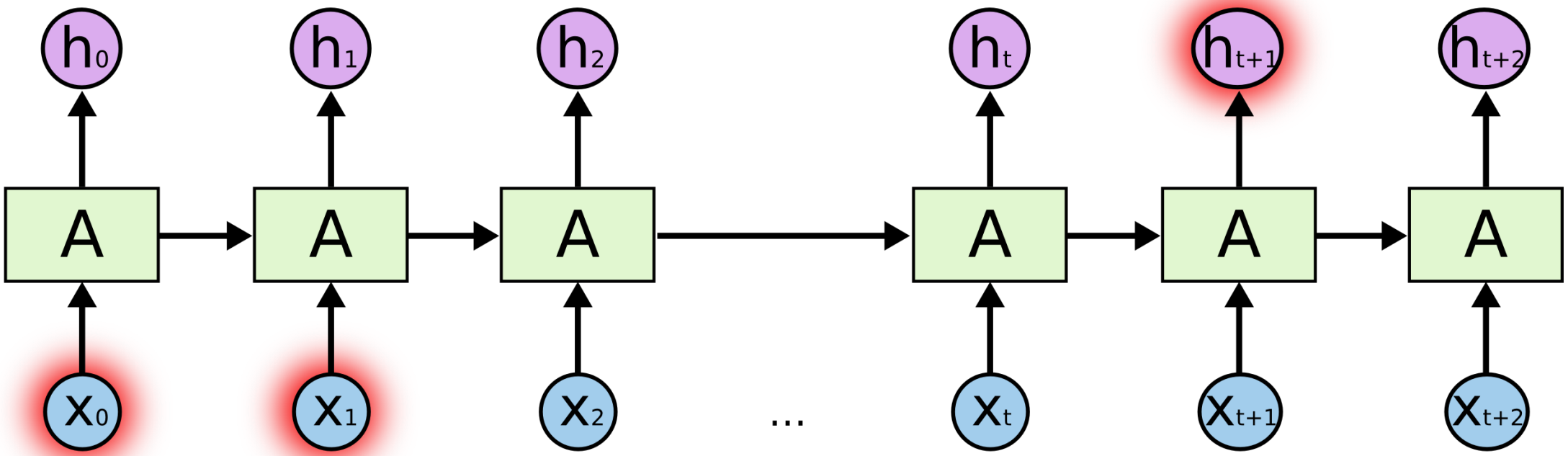
# Text to **recurrent** neural network



# Text to **recurrent** neural network



# Recurrent neural network

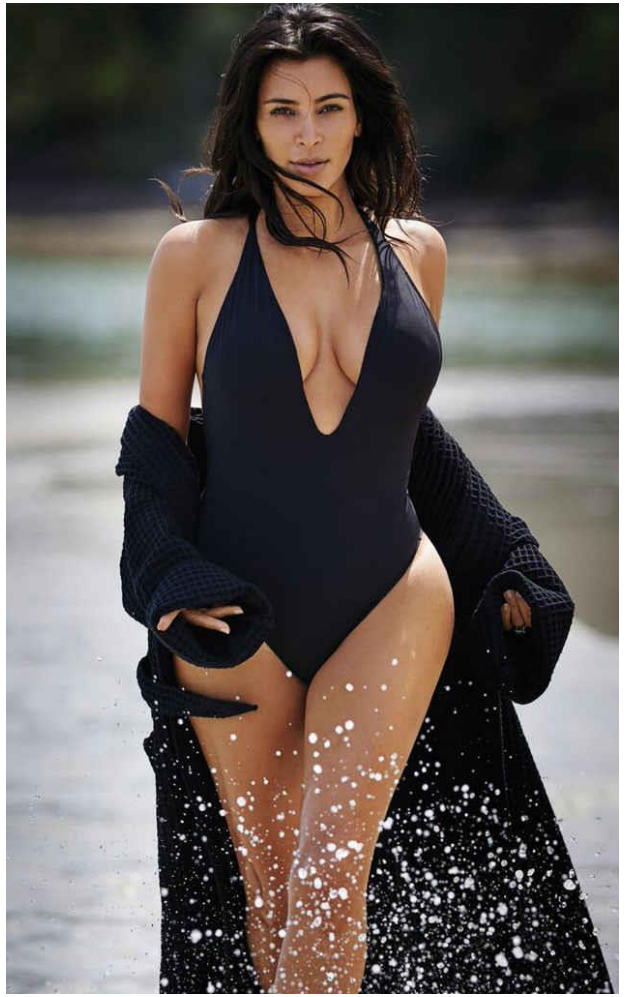


# résumé

We use **recurrent** neural networks to work with **sequential** data; we also can represent all words with **vectors**, that show their sense and similarity

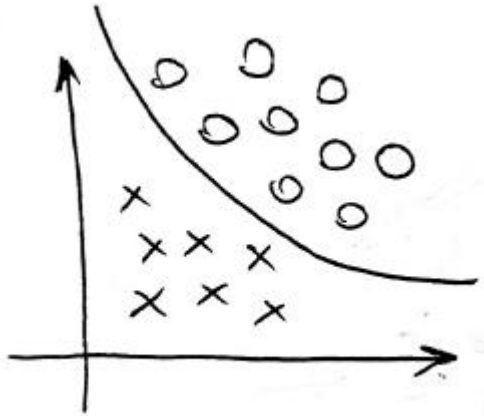
# deep building blocks

# Data



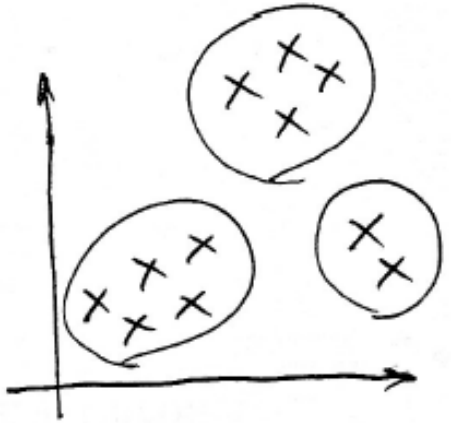
Dies ist ein Blindtext. An ihm lässt sich vieles über die Schrift ablesen, in der er gesetzt ist. Auf den ersten Blick wird der Grauwert der Schriftfläche sichtbar. Dann kann man prüfen, wie gut die Schrift zu lesen ist und wie sie auf den Leser wirkt. Dies ist ein Blindtext. An ihm lässt sich vieles über die Schrift ablesen, in der er gesetzt ist. Auf den ersten Blick wird der Grauwert der Schriftfläche sichtbar. Dann kann man prüfen, wie gut die Schrift zu lesen ist und wie sie auf den Leser wirkt.

# Task

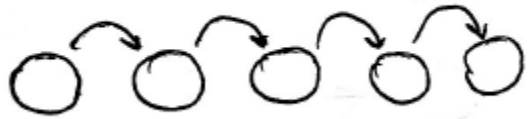


Classification

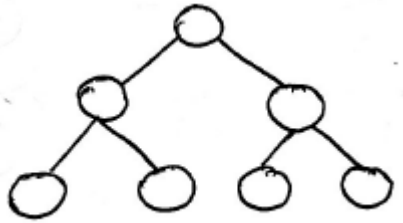
Regression (real value R)



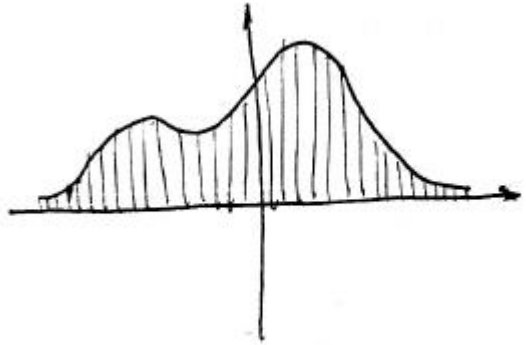
Clustering



Sequence output



Structure output (graph)

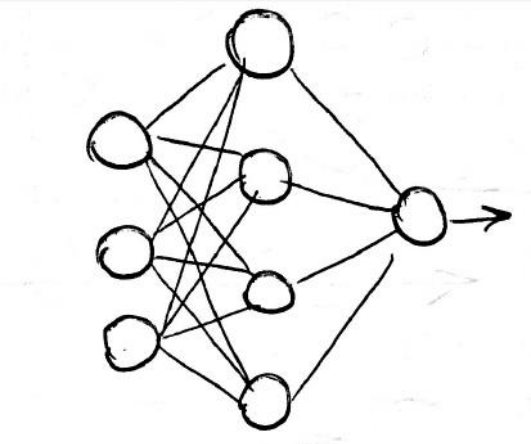


Synthesis and sampling

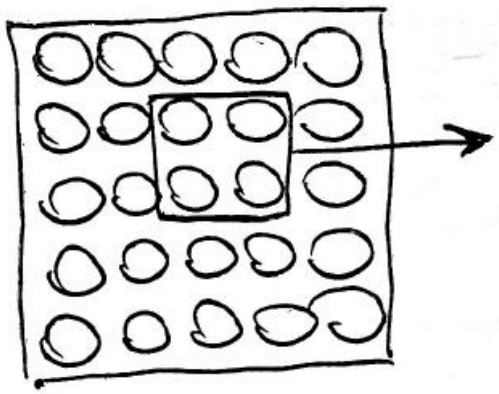
# Measure (loss functions)

- Mean squared error
- Mean absolute error
- Hinge loss
- Cross-entropy loss
- Kullback-Leibler divergence
- Cosine proximity
- …

# Feature extractor



Fully-connected layer

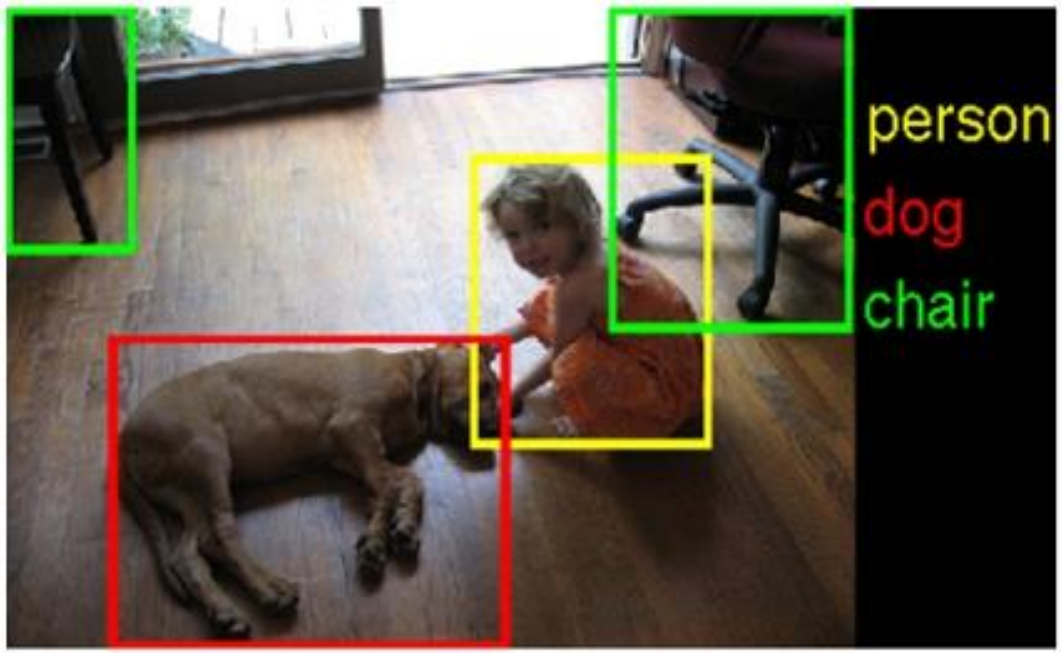


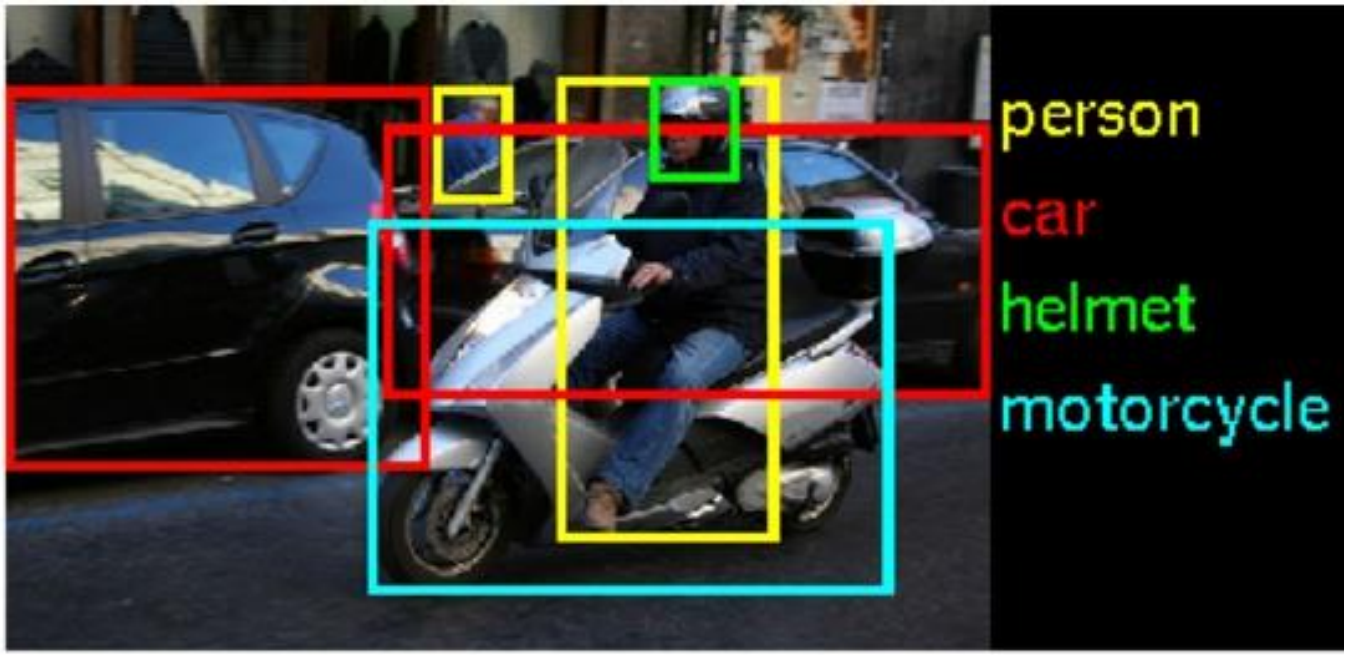
Convolutional layer

Recurrent layer

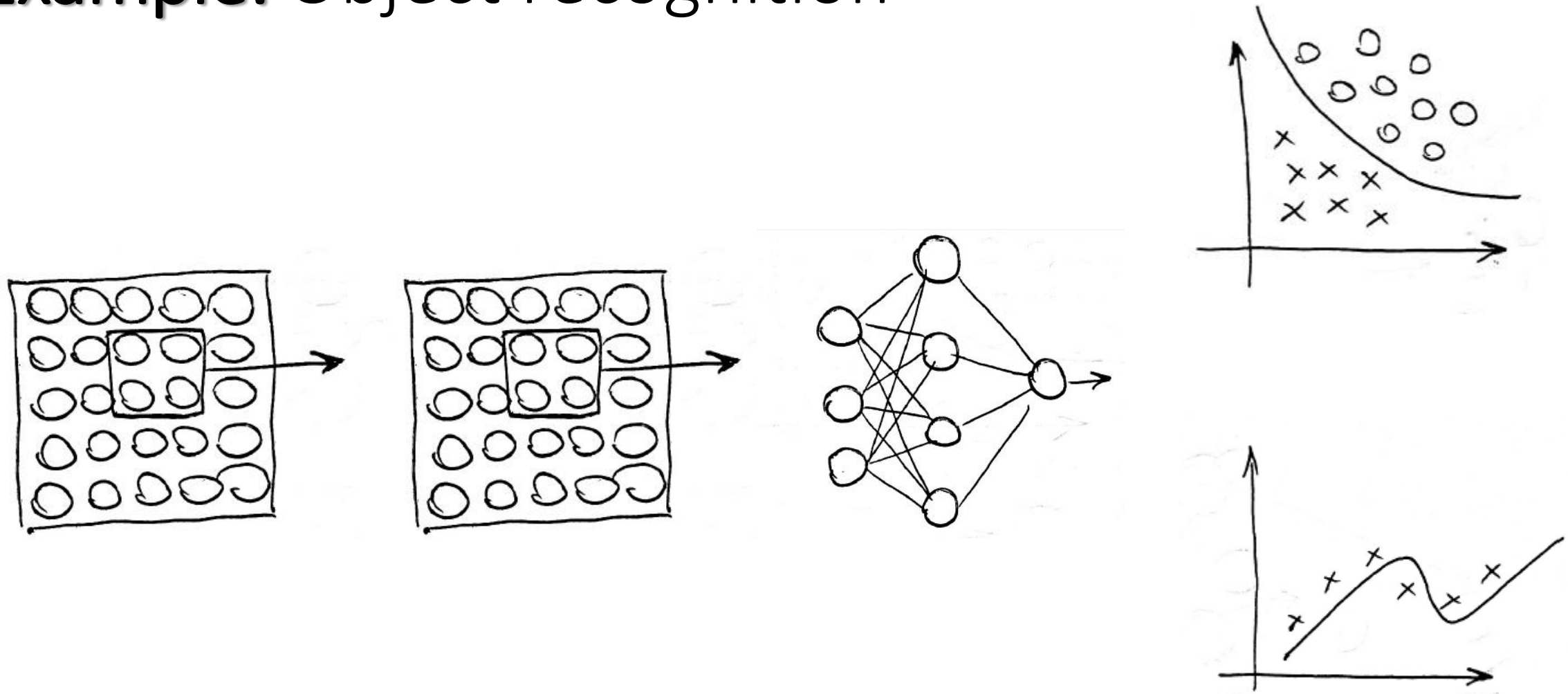
# examples

# **Example:** Object recognition

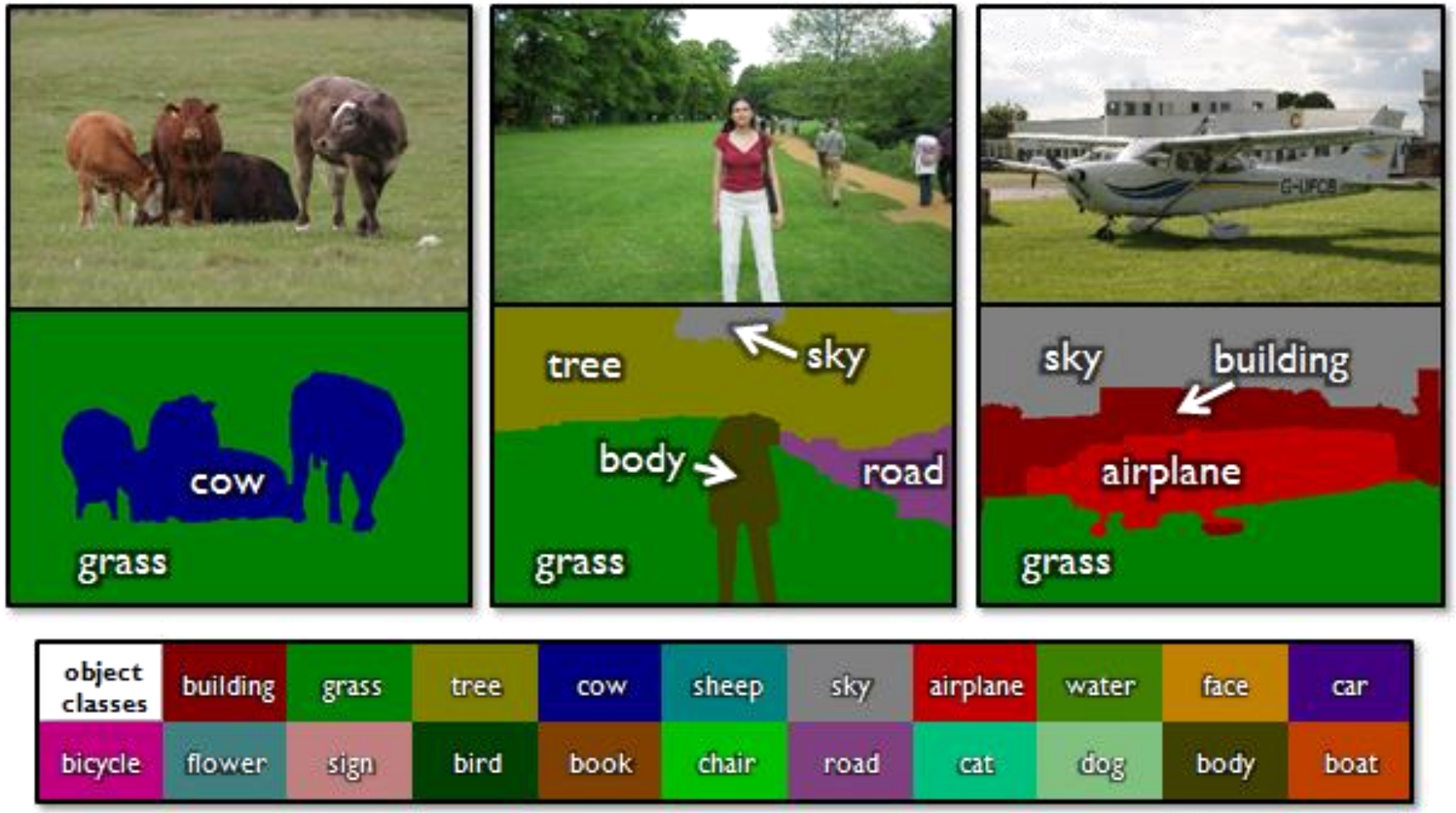


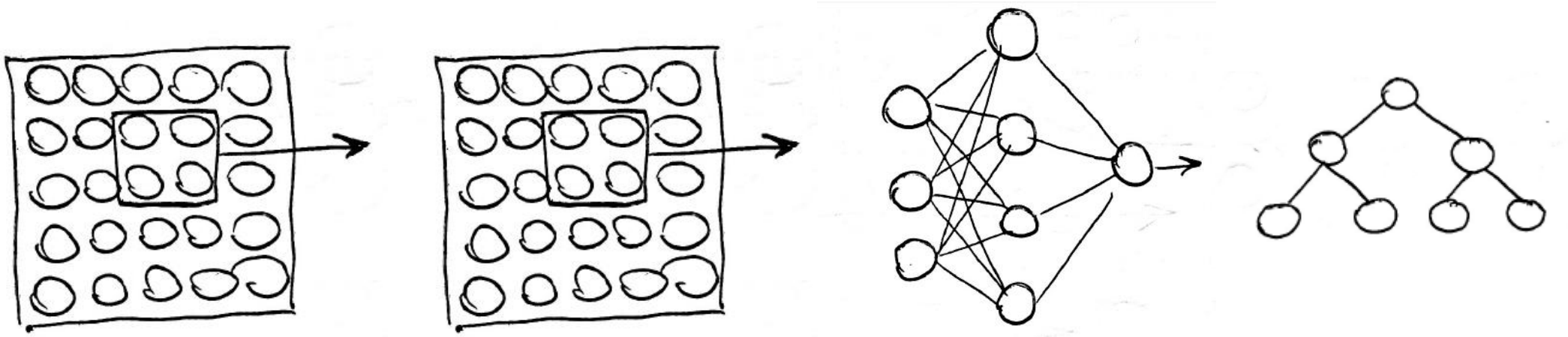
# **Example:** Object recognition



# **Example:** Object segmentation

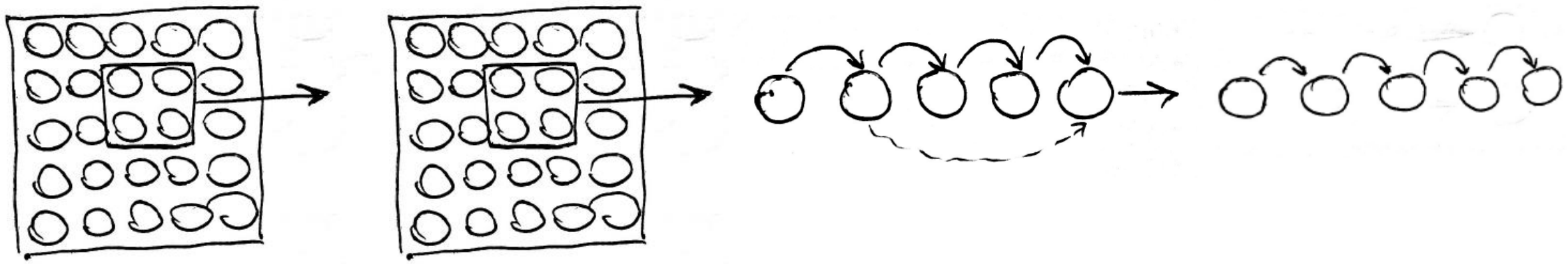


# **Example:** Object segmentation

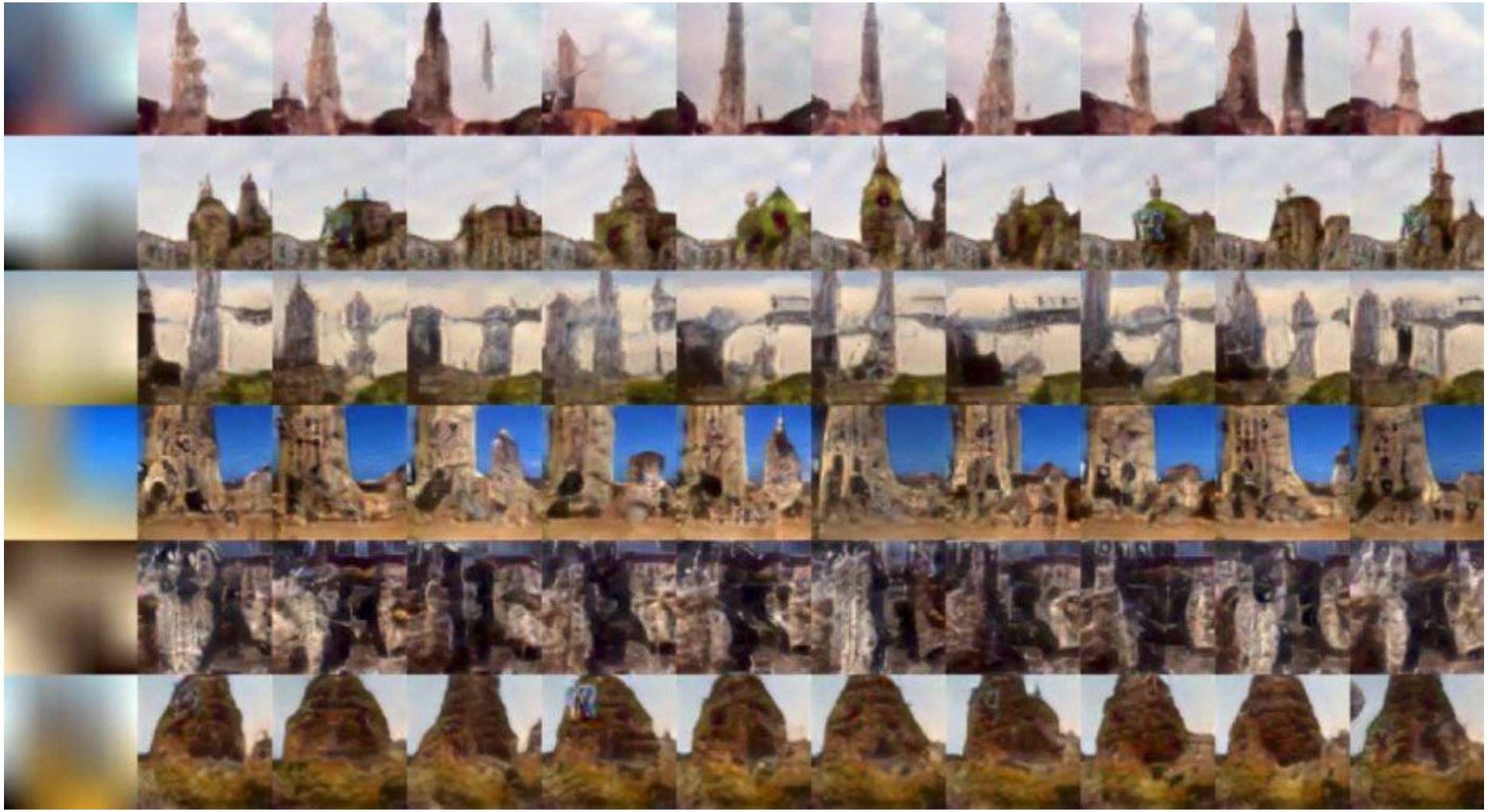


# **Example:** Image captioning

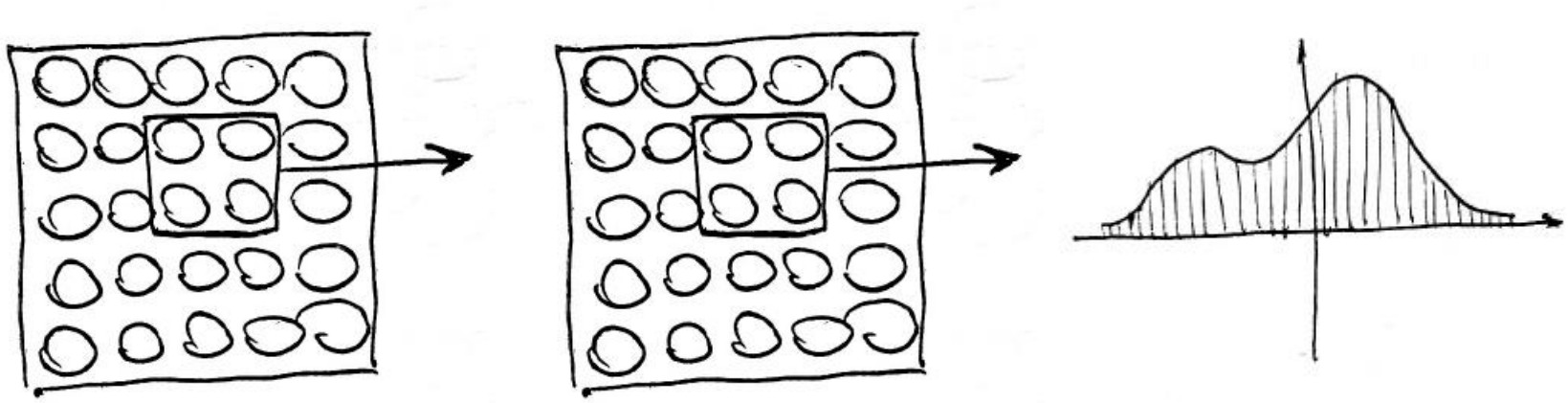
# **Example:** Image captioning



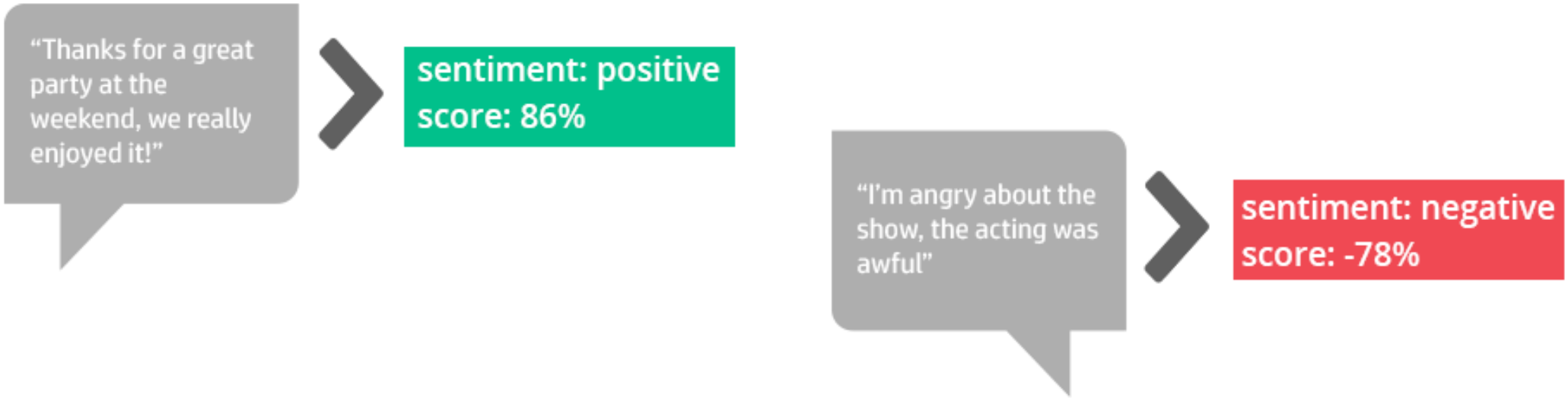
# **Example:** Image generation



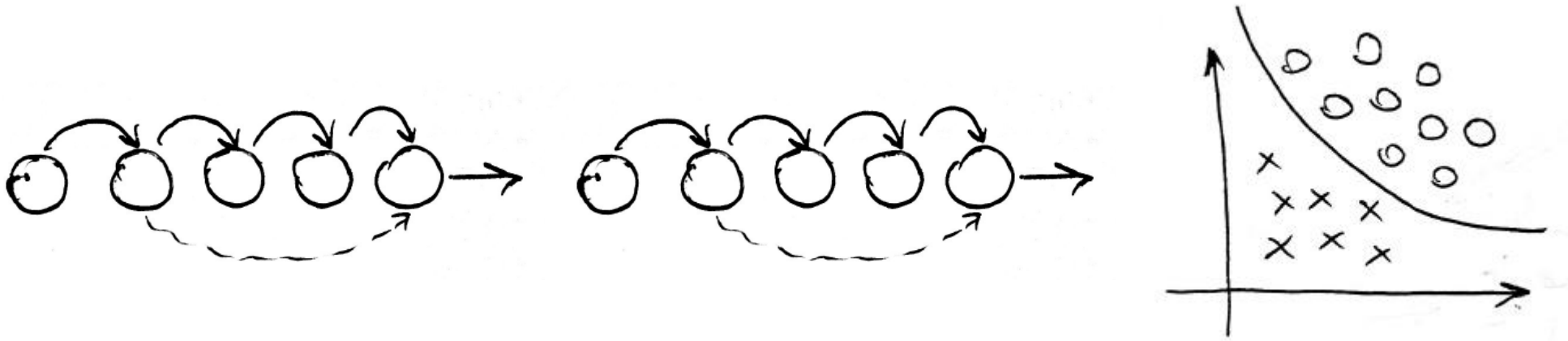
# **Example:** Image generation



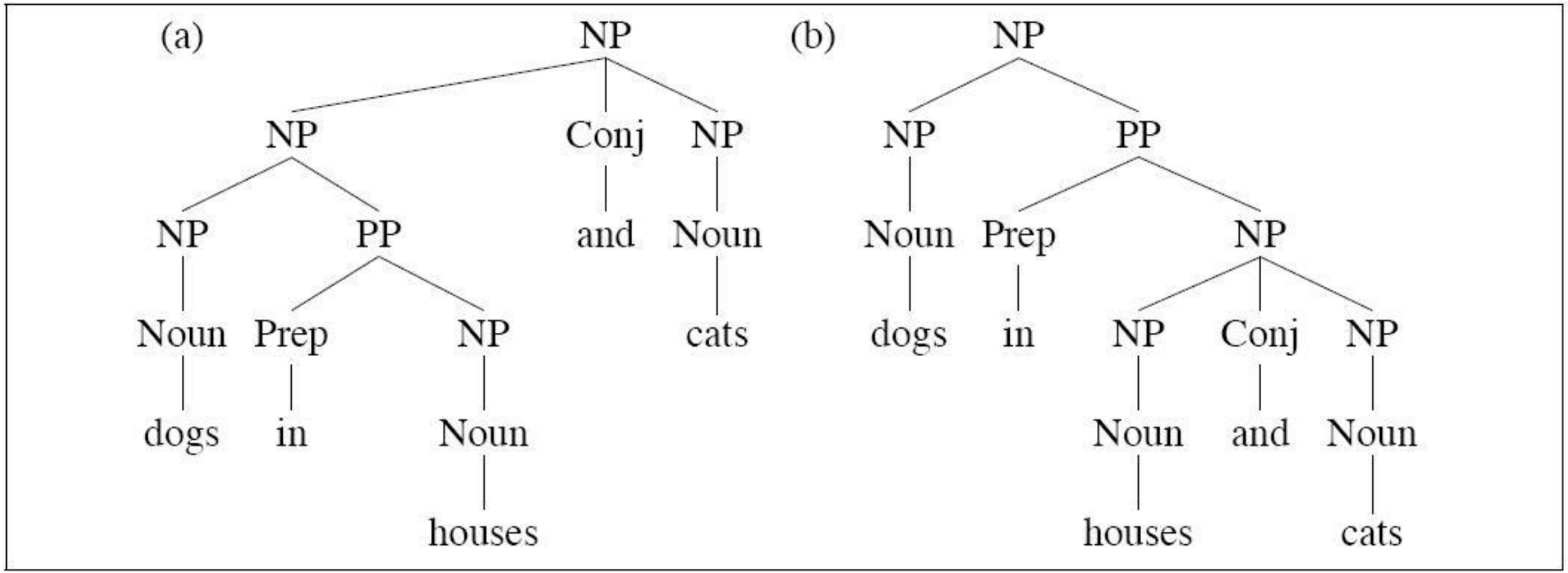
# **Example:** Sentiment analysis



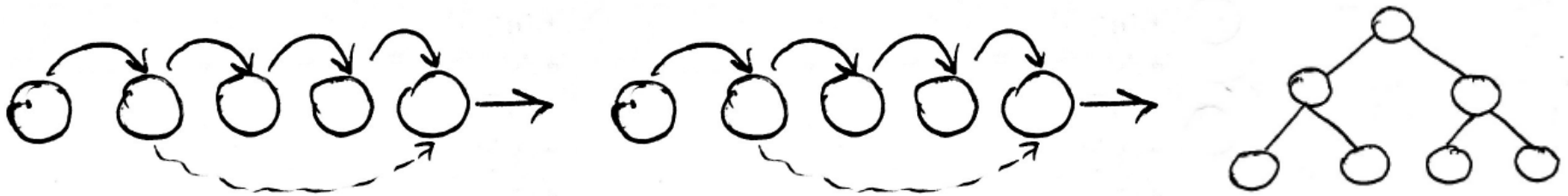
# **Example:** Sentiment analysis



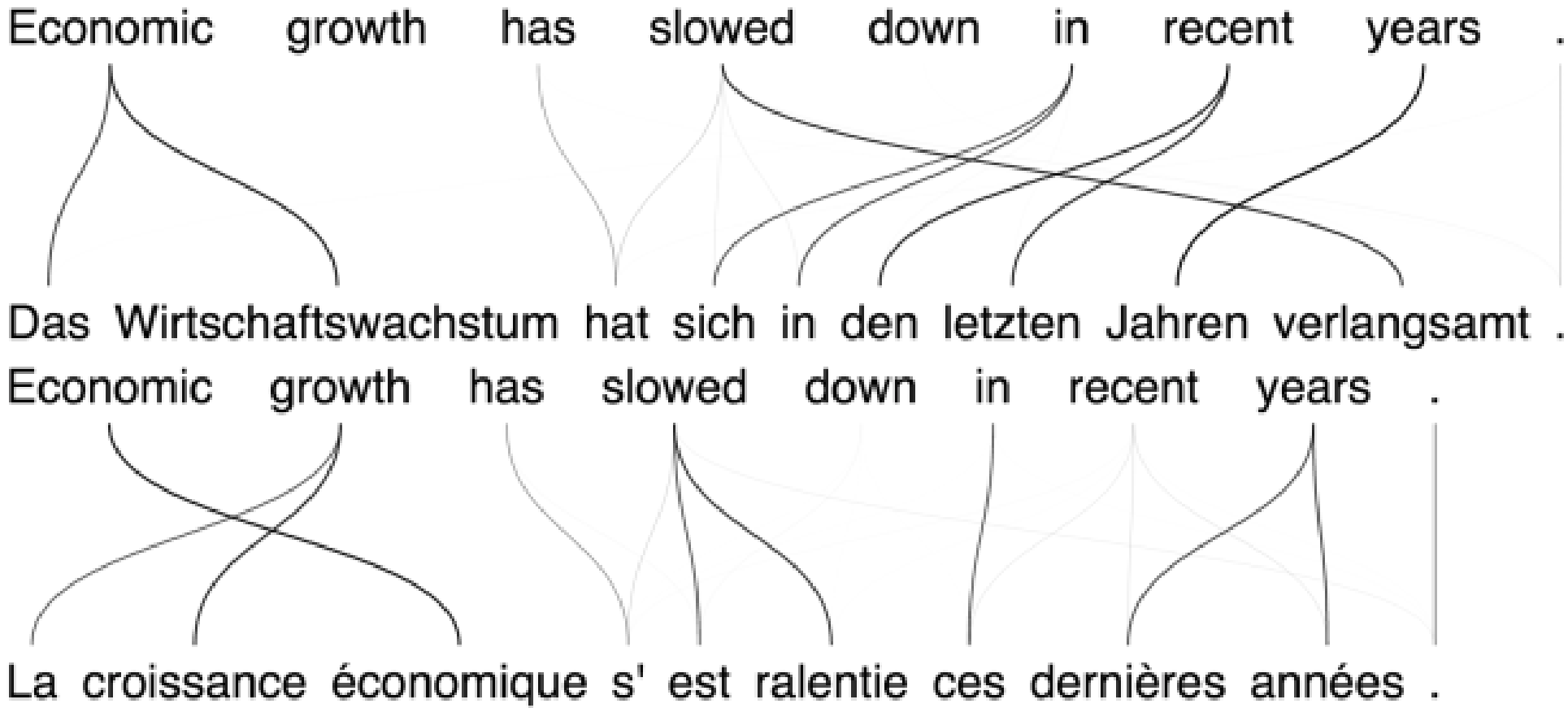
# **Example:** Language understanding



# **Example:** Language understanding



# **Example:** Machine translation



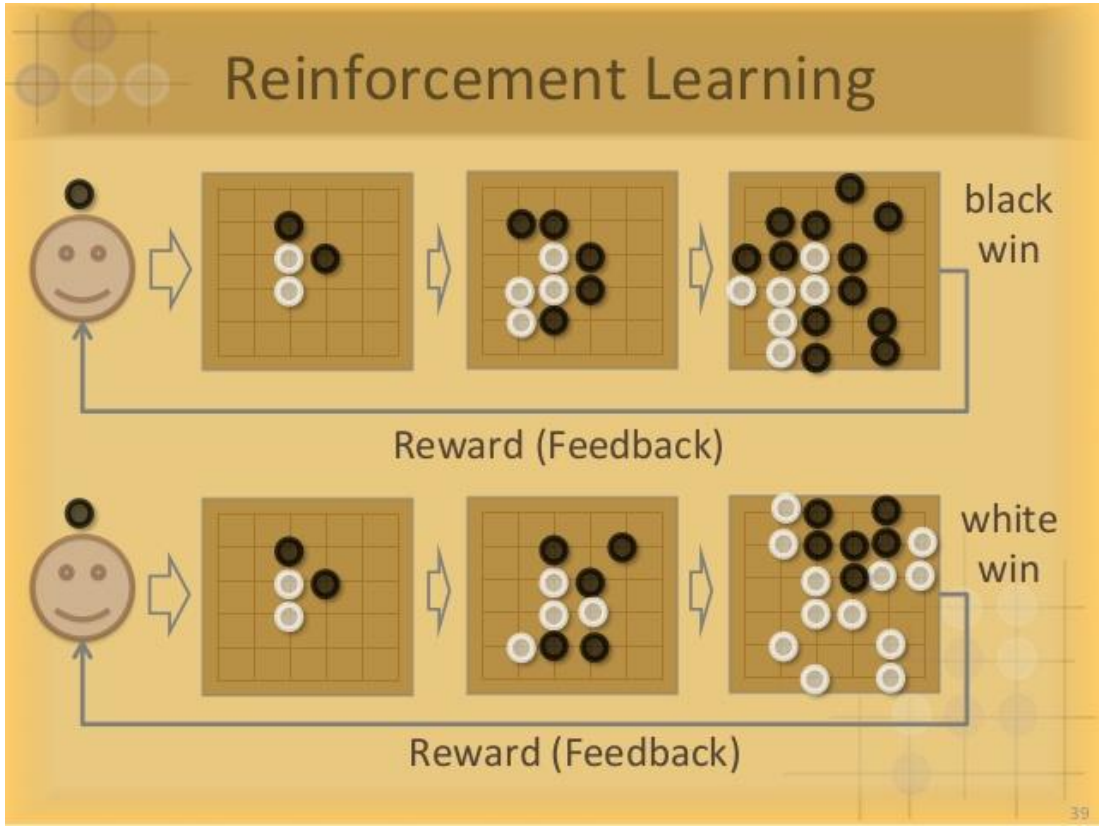
# **Example:** Machine translation

# **Example:** Neural conversational model

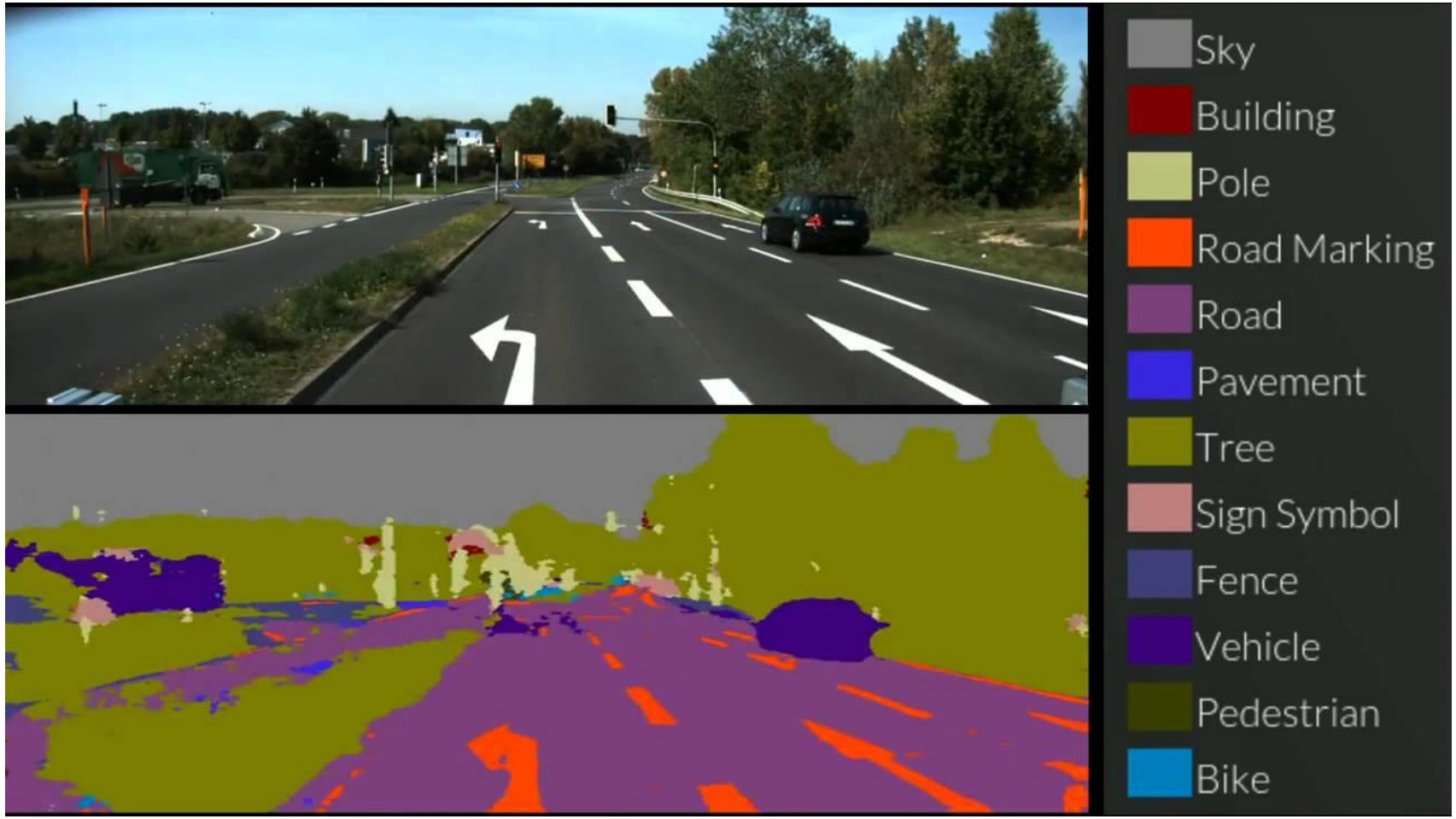
| | |
|---|---|
| *message* | Where do you live now? |
| *response* | I live in Los Angeles. |
| *message* | In which city do you live now? |
| *response* | I live in Madrid. |
| *message* | In which country do you live now? |
| *response* | England, you? |
| *message* | Where were you born? |
| *response* | I was born in Canada. |
| *message* | Where are you from? |
| *response* | England, you? |
| *message* | Where did you grow up? |
| *response* | I grew up in Texas. |
| *message* | How old are you? |
| *response* | 16 and you? |
| *message* | What's your age? |
| *response* | 18. |
| *message* | What is your major? |
| *response* | I'm majoring in psychology |
| *message* | What did you study in college? |
| *response* | English lit. |

# **Example:** Neural conversational model

# **Example:** Playing games (reinforcement learning)




# **Example:** Driving cars (reinforcement learning)



# And a bit more…

- Video prediction (what will happen in the future!)
- Music generation
- Face recognition and search
- Emotion recognition
- Diagnostics in medicine
- Speech recognition and generation
- Artistic style transfer
- Autopilot car driving
- High frequency trading
- Neural photo editor
- Recommender systems
- Data noise reduction
- Visual question answering
- Robotics
- Chemistry, Physics…

# **Example:** Object recognition

```
model = Sequential()

model.add(Convolution2D(nb_filters, kernel_size[0], kernel_size[1],
                        border_mode='valid',
                        input_shape=input_shape))
model.add(Activation('relu'))
model.add(Convolution2D(nb_filters, kernel_size[0], kernel_size[1]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=pool_size))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(nb_classes))
model.add(Activation('softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer='adadelta',
              metrics=['accuracy'])

model.fit(X_train, Y_train, batch_size=batch_size, nb_epoch=nb_epoch,
          verbose=1, validation_data=(X_test, Y_test))
```

# **Example:** Sentiment analysis

```
model = Sequential()
model.add(Embedding(max_features, 128, dropout=0.2))
model.add(LSTM(128, dropout_W=0.2, dropout_U=0.2))  # try
using a GRU instead, for fun
model.add(Dense(1))
model.add(Activation('sigmoid'))

# try using different optimizers and different optimizer configs
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

print('Train...')
model.fit(X_train, y_train, batch_size=batch_size,
nb_epoch=15,
          validation_data=(X_test, y_test))
```

# How it learns? (machine learning)

Program is said to learn from **experience E** (data set with pictures) with respect to some class of **tasks T** (logistic regression) and performance **measure P** (mean squared error) if performance **improves** (with gradient descent iterations)

Mitchell, 1997

# How it learns? (deep learning)

Program is said to learn from **learned representation** R (convolutional layers) of **experience E** (data set with pictures) with respect to some class of **tasks T** (logistic regression) and performance **measure P** (mean squared error) if performance **improves** (with gradient descent iterations)

# résumé

Deep learning networks are built from **blocks** – we combine and optimize these blocks to extract correct **representation** of complex data (images, text, speech) and give correct **output**

# roadmap

# Day 1 goals

- You understand basics of machine learning
- You understand difference between machine learning and deep learning (most engineers don’t!)
- You can define machine learning problem with ERTP blocks

# Day 2 goals

- You understand artificial neural networks structure
- You understand backpropagation algorithm (the most important thing)
- You can train your own neural network and continuously improve performance
- You can use Python **Keras** framework

# Day 3 goals

- You understand how convolutional neural network (CNN) works
- You understand modern approaches to CNNs
- You can train your own CNN for different computer vision problems
- You can use **Caffe** framework

# Day 4 goals

- You understand how recurrent neural network (RNN) works
- You can train your own RNN for text analysis
- You can use word2vec for text analysis
- You can train your own RNN for time series analysis
- You can use **TensorFlow** and **gensim** frameworks

# Day 5 goals

- You understand modern research trends in deep learning
- You understand how to get a job in machine learning area
- You can perform the full deep learning pipeline after making a final project

# Project proposals

- Titanic survivors prediction
- House price prediction
- Fashion image classification
- Food image classification
- Arrhythmia ECG detection
- Plants images classification
- Adult content recognition
- Stock price prediction
- Face identification
- Motivational quotes generation
- Chat bots
- Speech recognition
- Machine translation
- Stealing ML models from API
- Your ideas

# thank you for attention!

rachnogstyle@gmail.com
facebook |rachnogstyle