Deep Learning

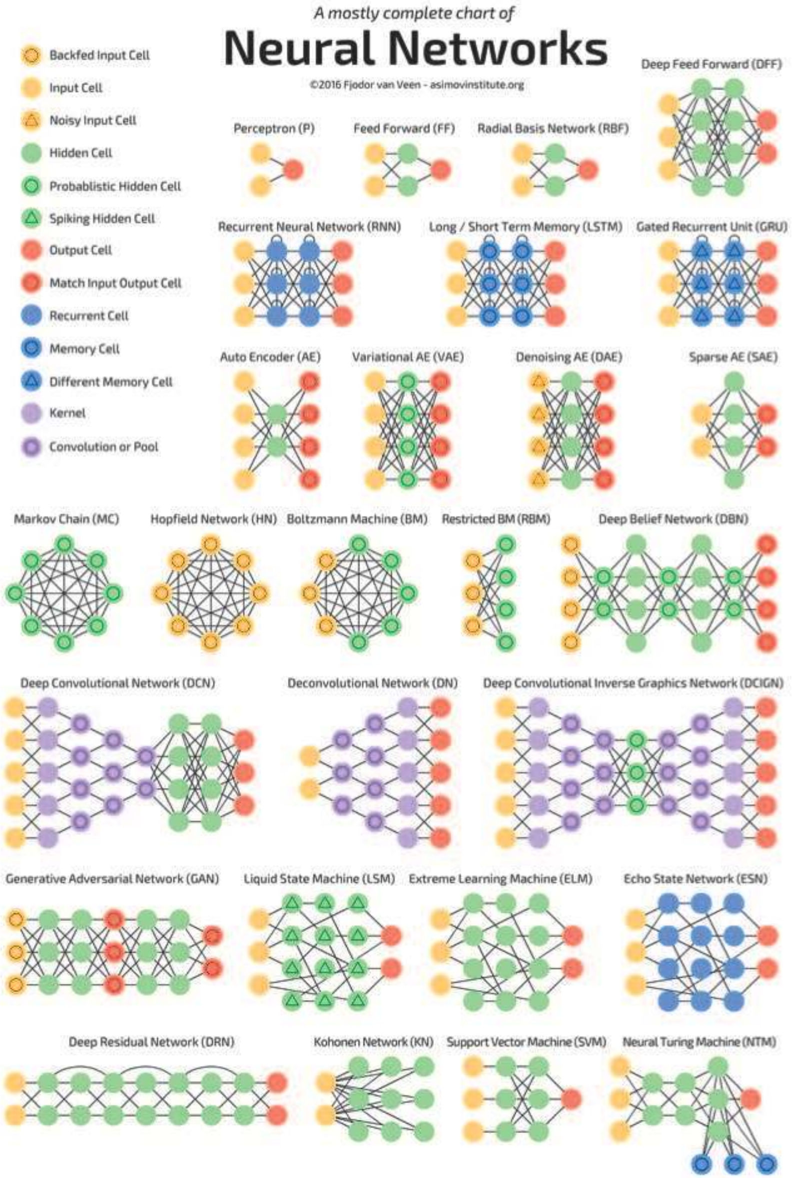
A journey from feature extraction and engineering to end-to-end pipelines

Part 1: Introduction, Computer Vision

Andrei Bursuc

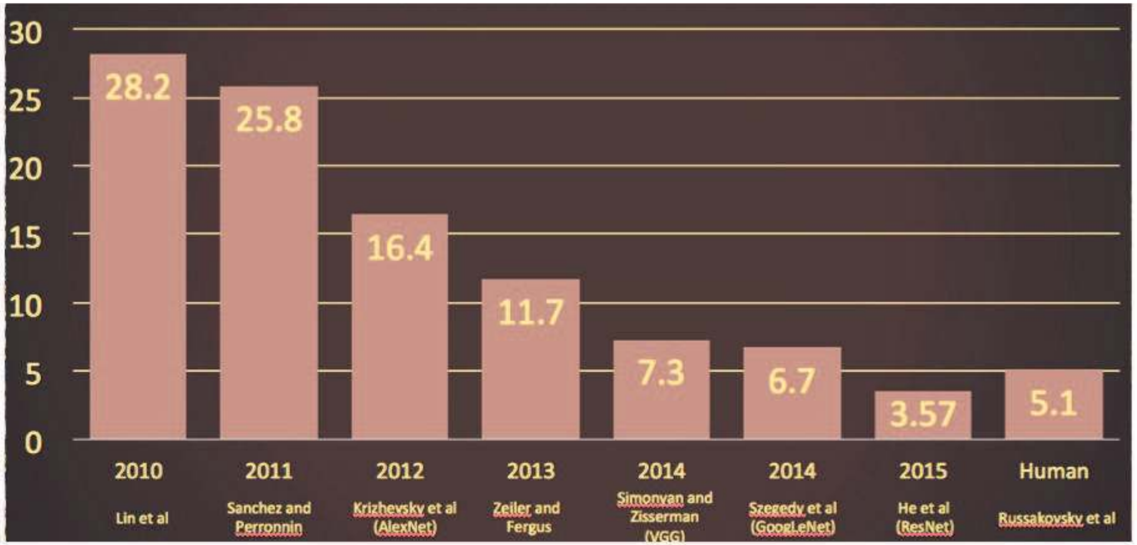
With slides from A. Karpathy, F. Fleuret, J. Johnson, S. Yeung, G. Louppe, Y. Avrithis ...

Deep Learning - the hype?



Deep Learning - the hype?

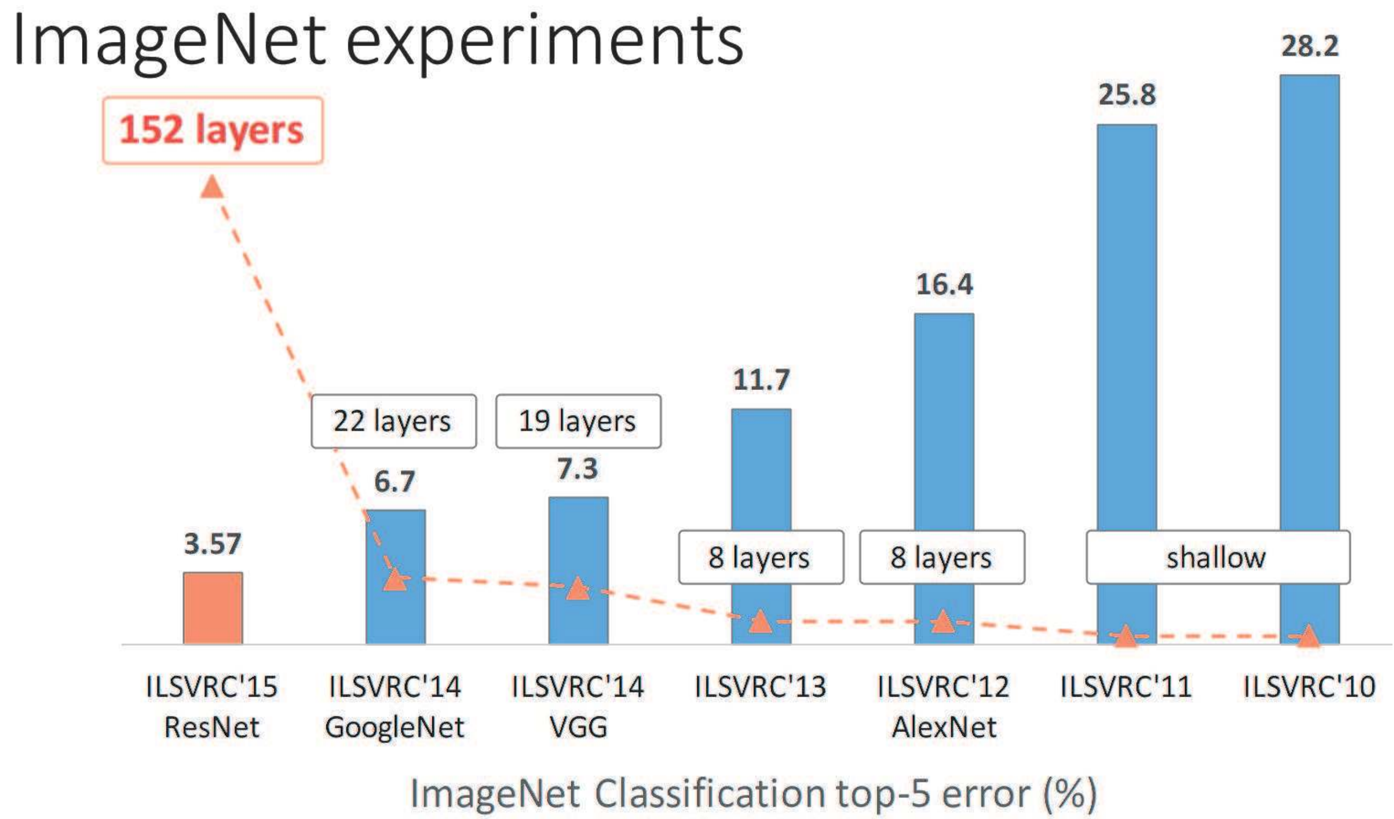
- Evolution of ImageNet large scale visual recognition challenge
- 1.2 M training images with 1K object categories



ImageNet top-5 error (%)

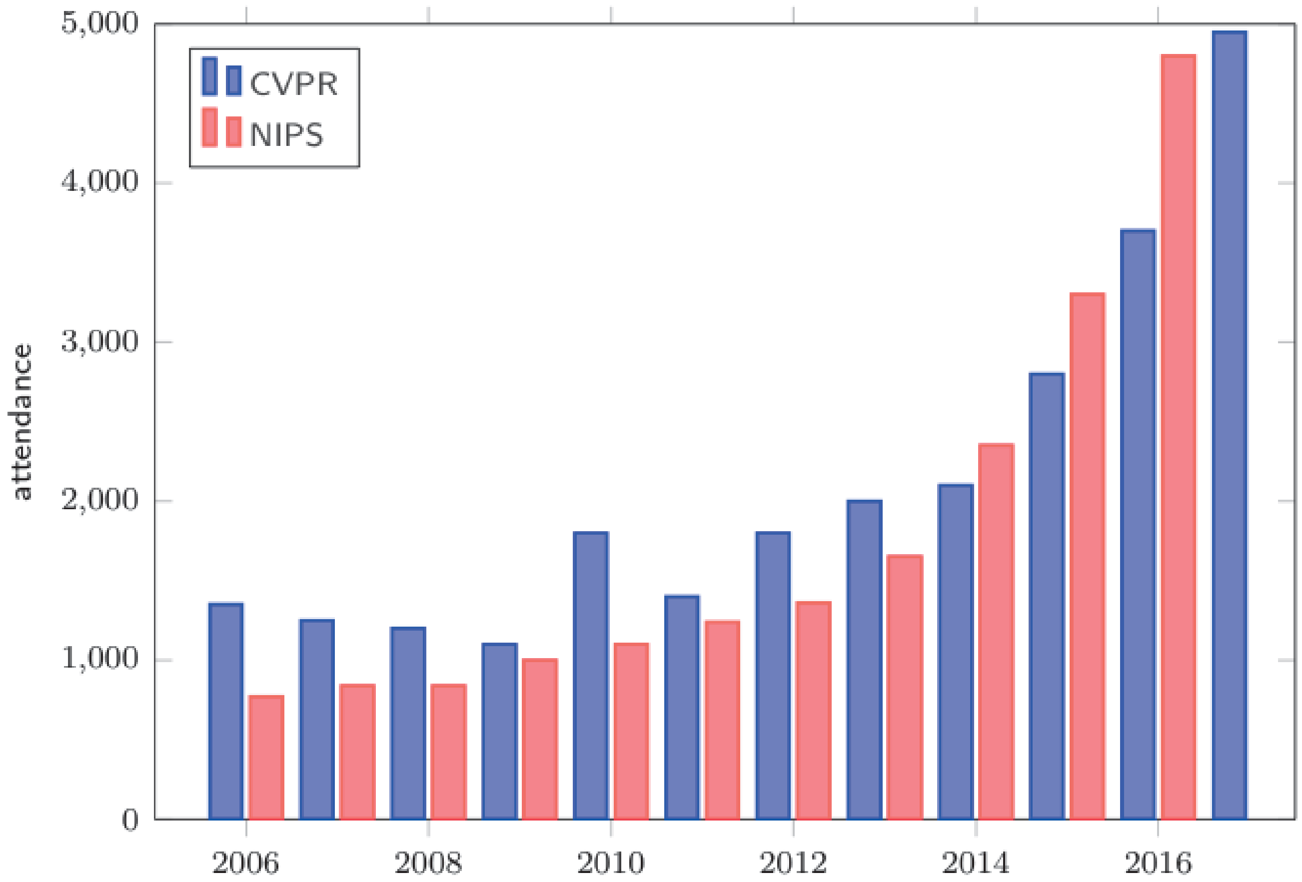
Deep Learning - the hype?

- Evolution of ImageNet large scale visual recognition challenge
- 1.2 M training images with 1K object categories



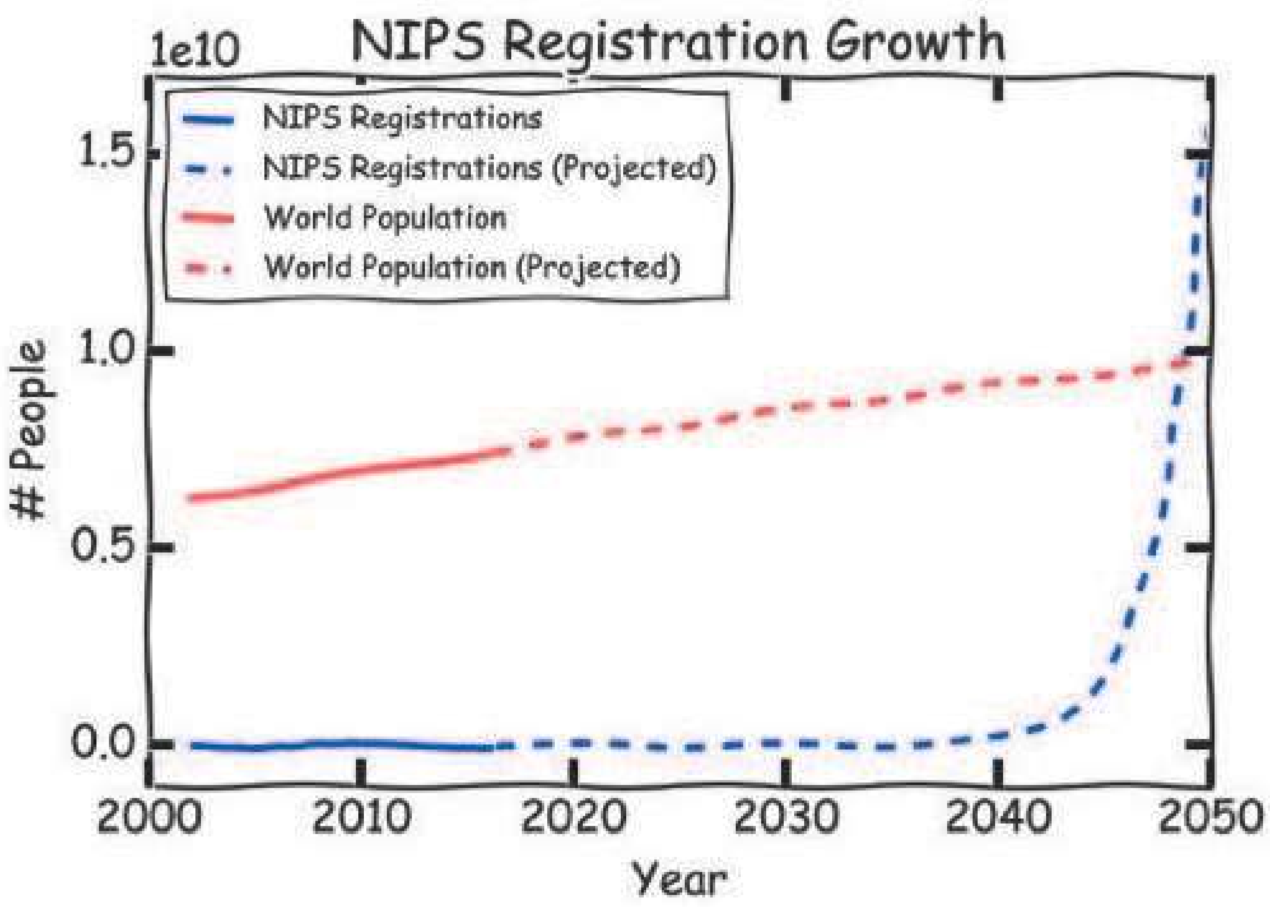
Deep Learning - the hype?

Conference attendance growth



Deep Learning - the hype?

Conference attendance growth

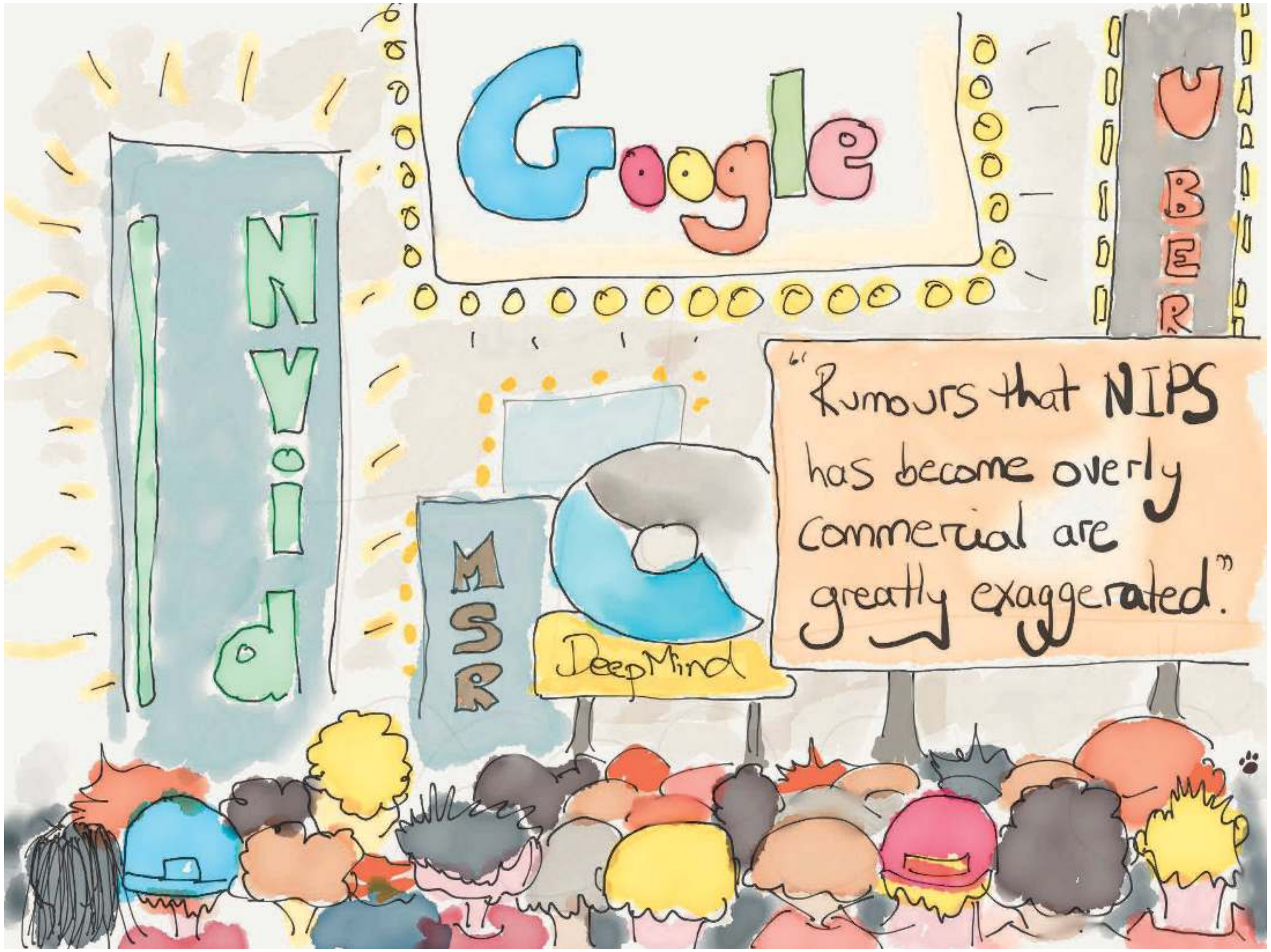


Deep Learning - the hype?

CVPR 2017 sponsors

Deep Learning - the hype?

Industry participation

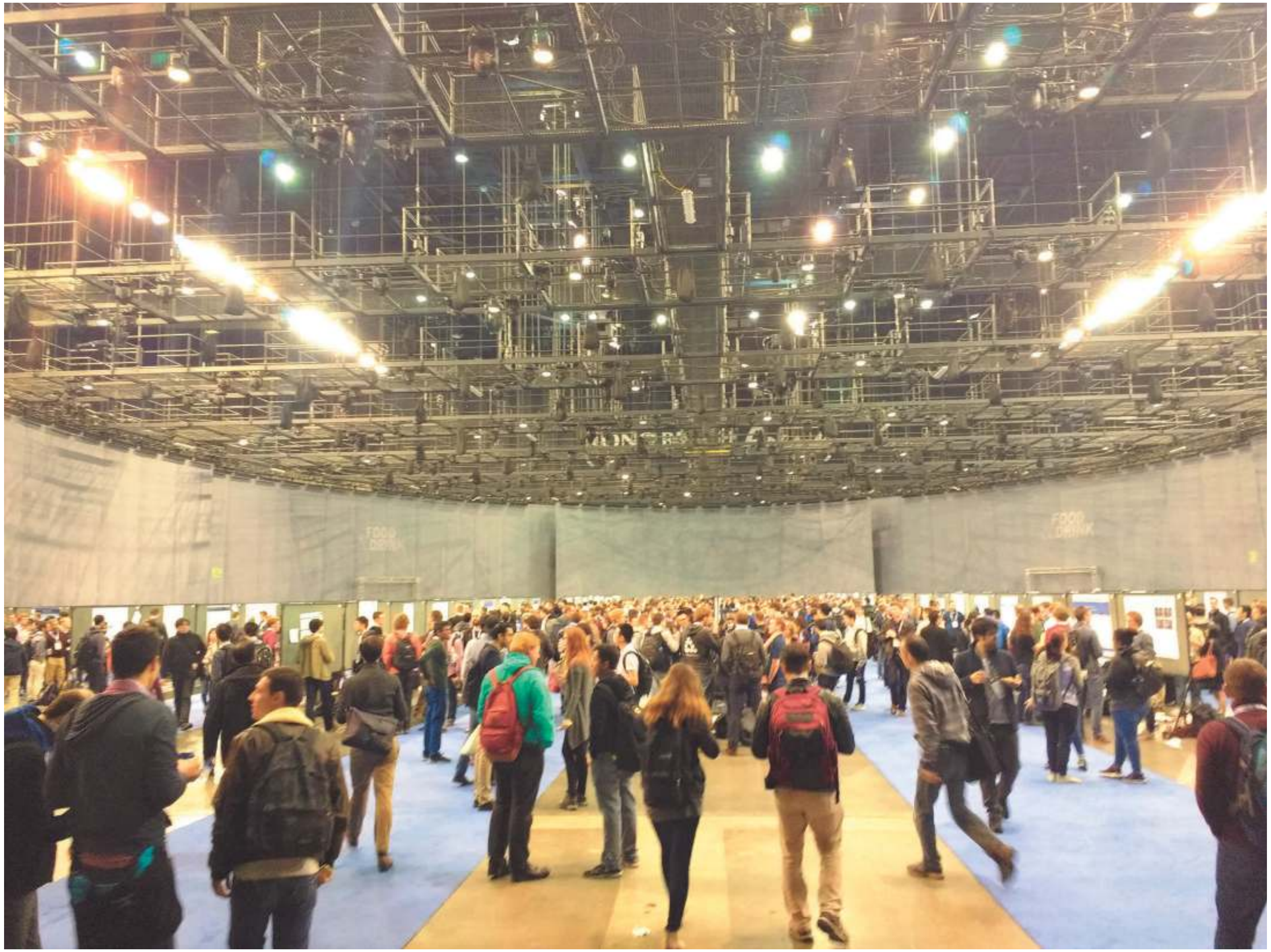


Deep Learning - the hype?

1/2 parallel session at NIPS 2017

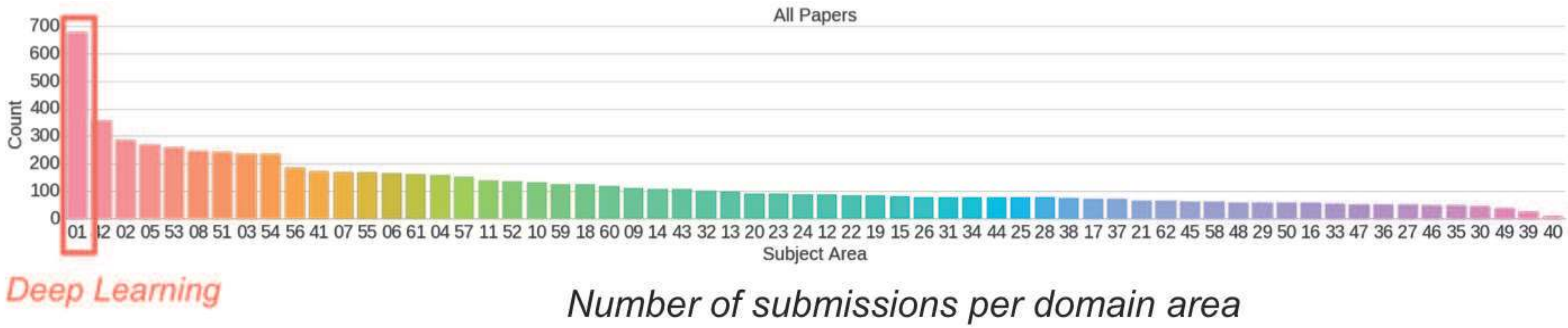
Deep Learning - the hype?

Poster session at NIPS 2017



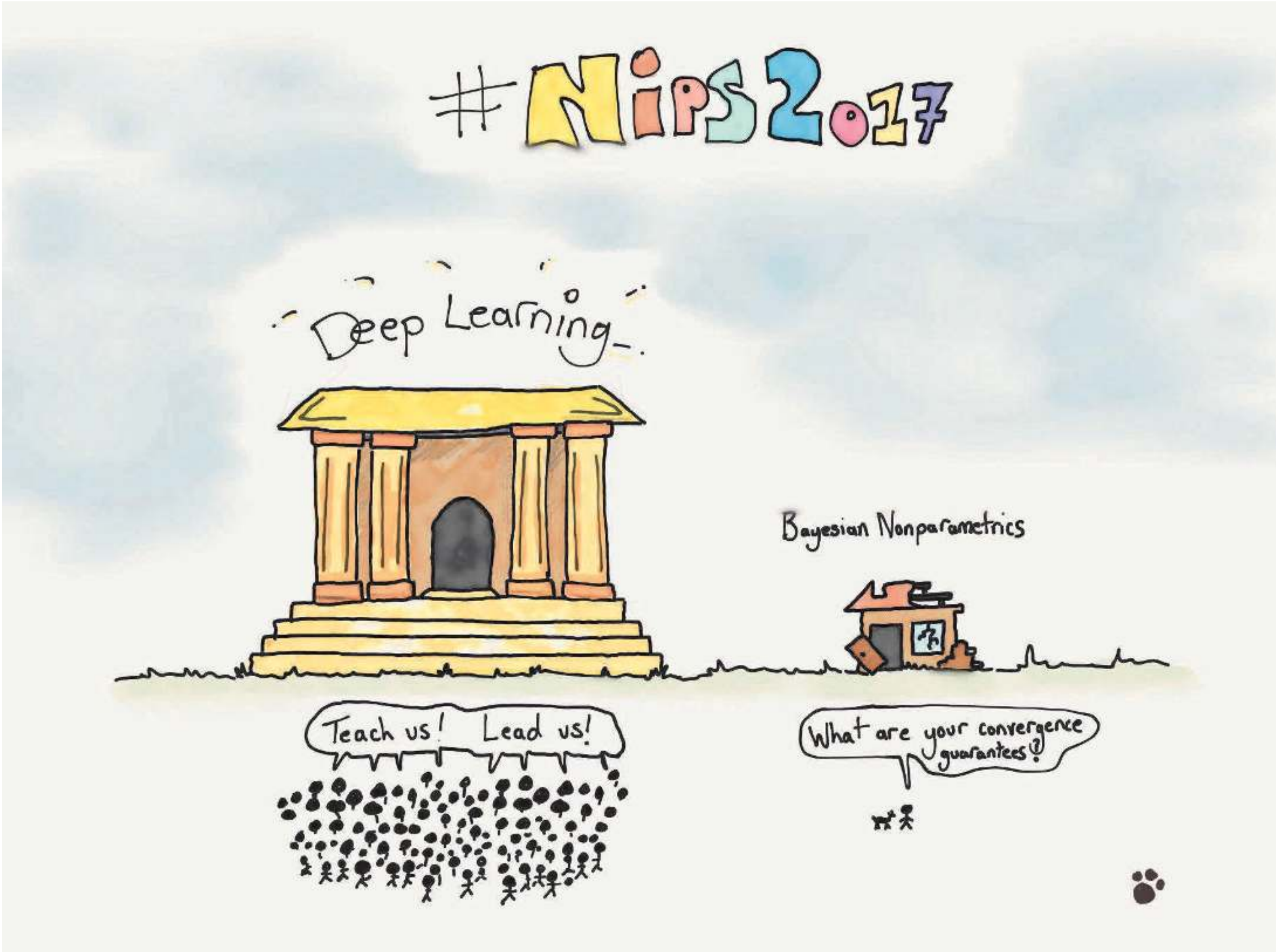
Deep Learning - the hype?

Primary topic in submissions at NIPS 2017



Deep Learning - the hype?

Primary topic in submissions at NIPS 2017



Deep Learning - the hype?

Primary topic in submissions at NIPS 2017

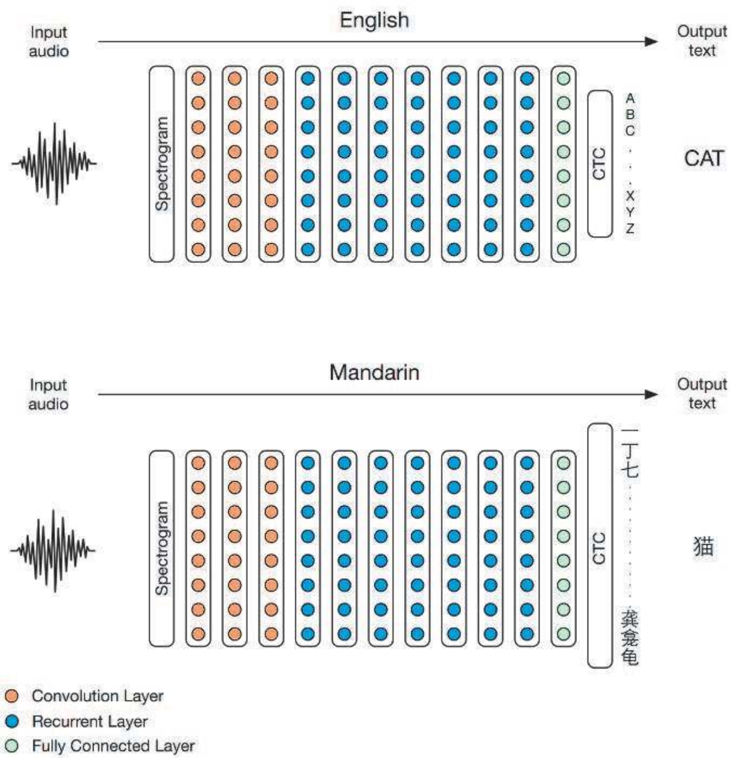
Deep Learning - the hype?

Other remarkable changes

- Paper publishing is more intense: papers are released on arXiv right after submission deadline
- Results of papers can be already outperformed by the time of the conference
- Code and/or trained networks are released with paper most of the times
- High number of published datasets
- Contributions arrive also from non computer vision / machine learning classic domains: genomics, mechanics.

Domain applications of Deep Learning?

Speech-to-Text



[Baidu 2014]

Domain applications of Deep Learning?

Computer Vision

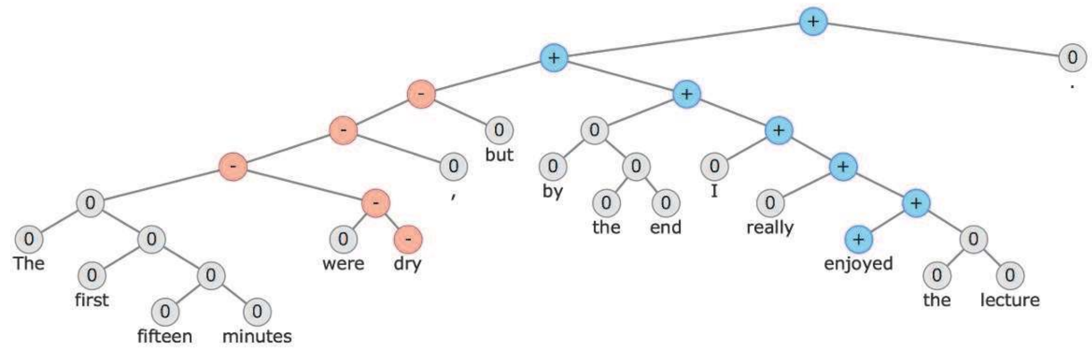
Domain applications of Deep Learning?

Computer Vision

Domain applications of Deep Learning?

NLP

[Google Translate System - 2016]



[Socher 2015]

Domain applications of Deep Learning?

NLP

[Google Inbox Smart Reply]

[Amazon Echo / Alexa]

Domain applications of Deep Learning?

Vision + NLP

Domain applications of Deep Learning?

Generative models

Sampled celebrities [Nvidia 2017]

Domain applications of Deep Learning?

Generative models

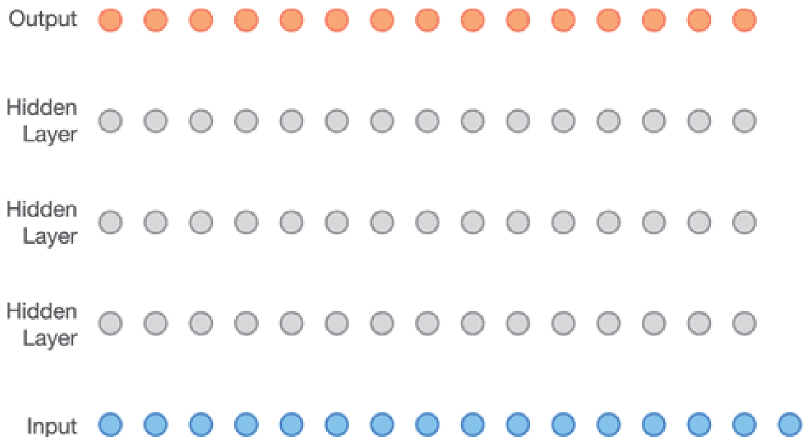
StackGAN v2 [Zhang 2017]

Domain applications of Deep Learning?

Image translation

Domain applications of Deep Learning?

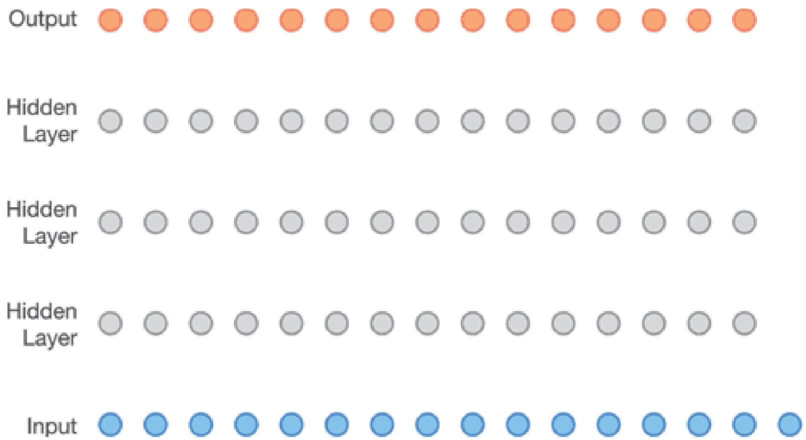
Generative models



Sound generation with WaveNet [DeepMind 2017]

Domain applications of Deep Learning?

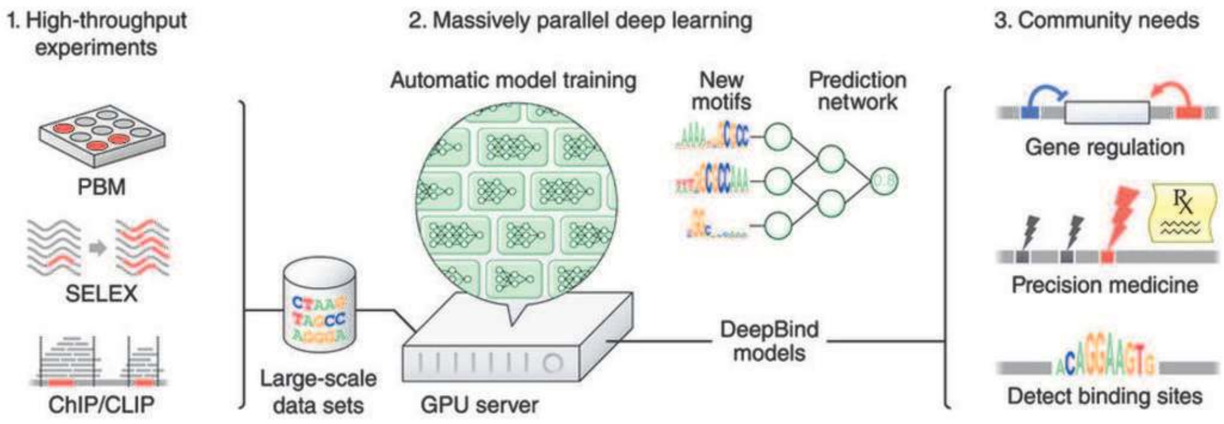
Generative models



Sound generation with WaveNet [DeepMind 2017]

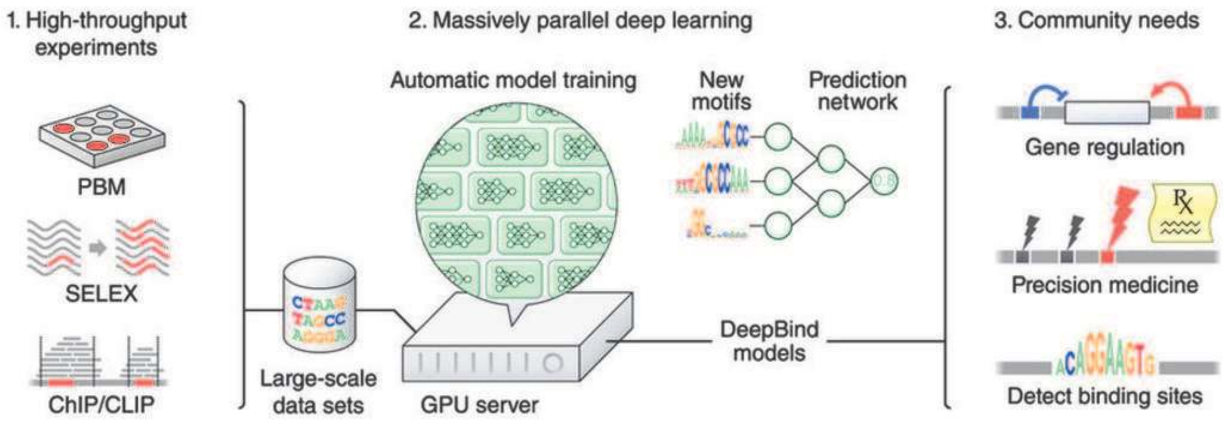
Guess which one is generated?

DL in other sciences



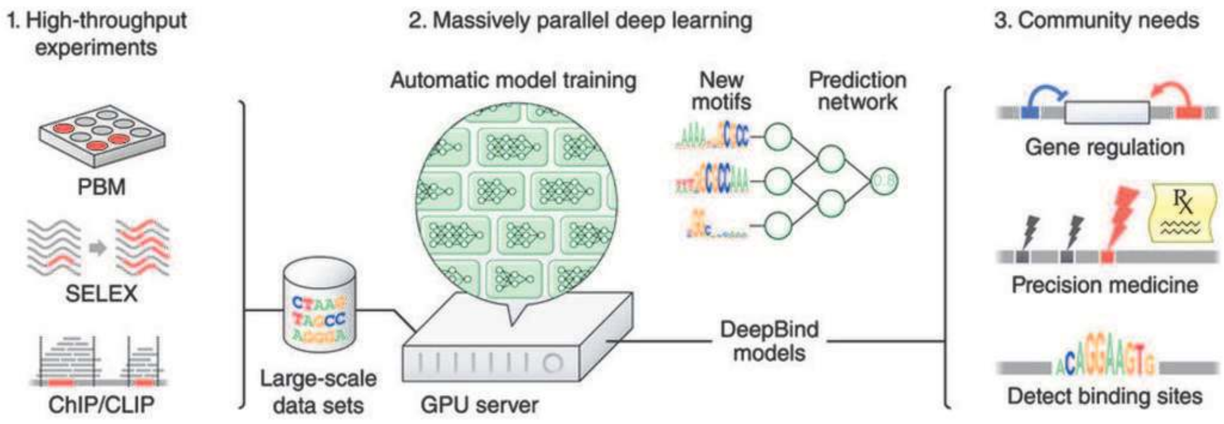
[Deep Genomics 2017]

DL in other sciences

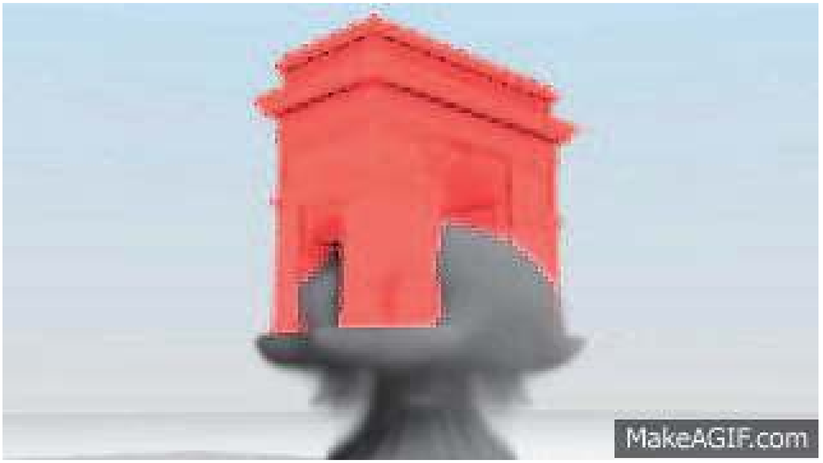


[Deep Genomics 2017]

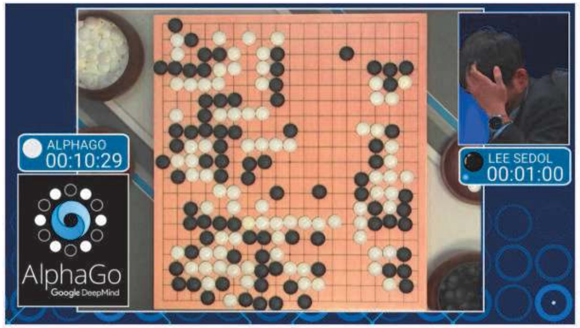
DL in other sciences

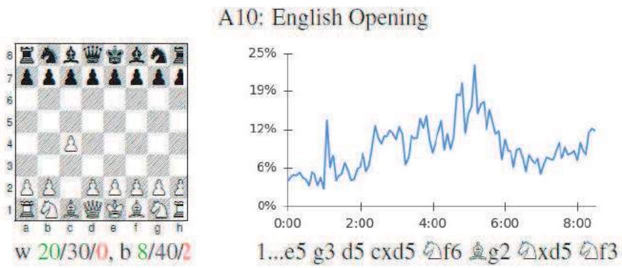


[Deep Genomics 2017]

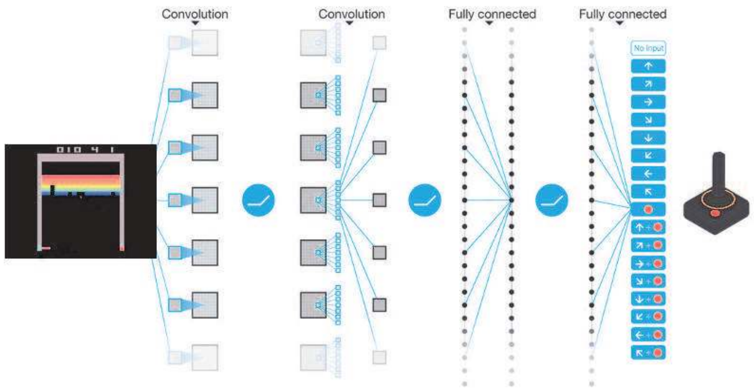


DL for AI in games





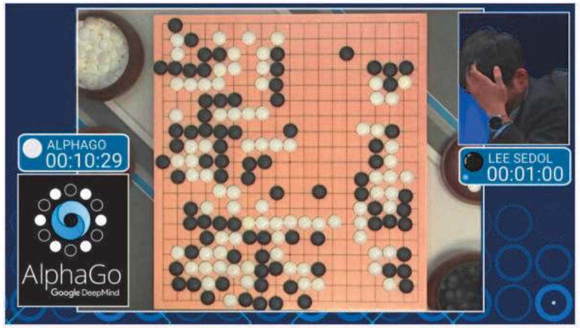
[Deepmind AlphaGo / Zero 2017]

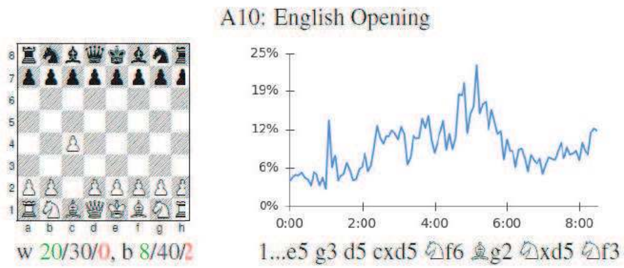


[Atari Games - DeepMind 2016]

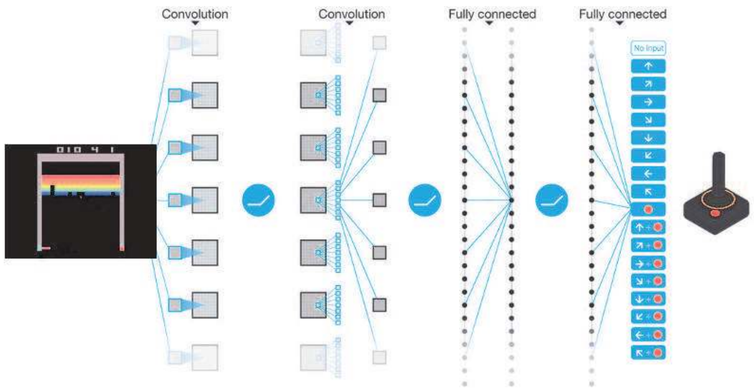
[Starcraft 2 for AI research]

DL for AI in games





[Deepmind AlphaGo / Zero 2017]



[Atari Games - DeepMind 2016]

[Starcraft 2 for AI research]

AlphaGo/Zero: Monte Carlo Tree Search, Deep Reinforcement Learning, self-play

What is Deep Learning?

- Neural Networks with more layers/modules
- Non-linear, hierarchical, abstract representations of data
- Flexible models with any input/output size
- Differentiable functional programming

What is Deep Learning?

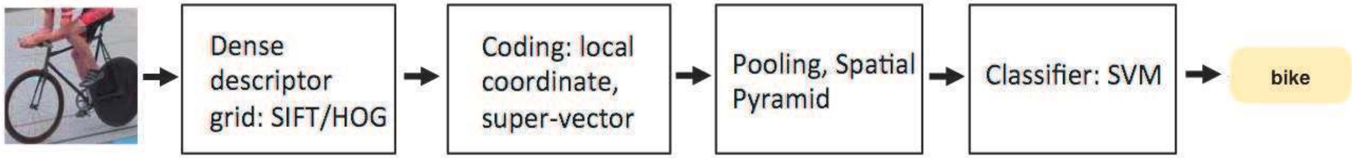
In other words: a graph of tensor operators taking advantage of:

- the chain rule (back-propagation),
- stochastic gradient descent,
- convolutions,
- parallel operations on GPU

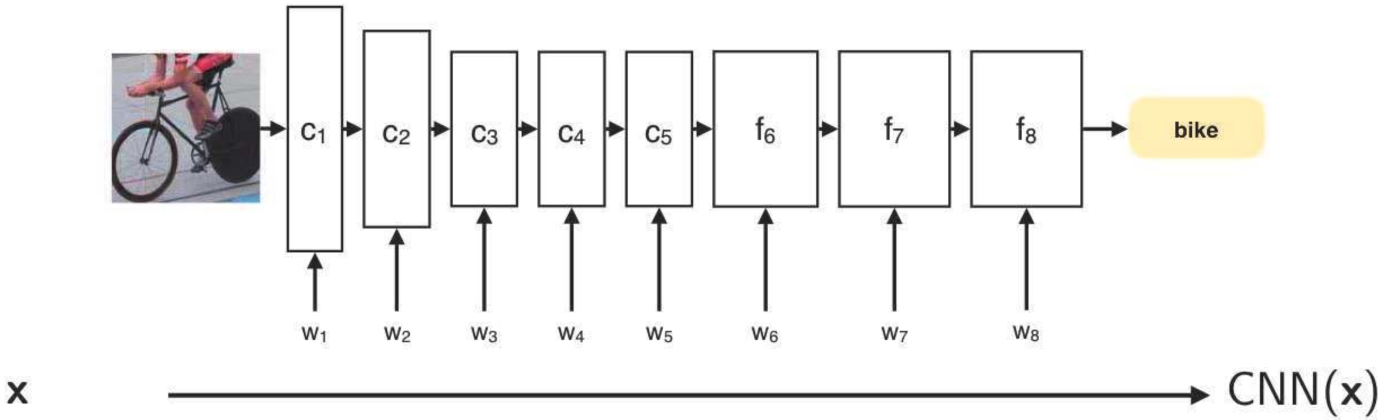
We kind of had most of it in the networks from long ago

Why going deep?

- Traditional recognition: "shallow" architecture
  - Each block is designed and implemented individually



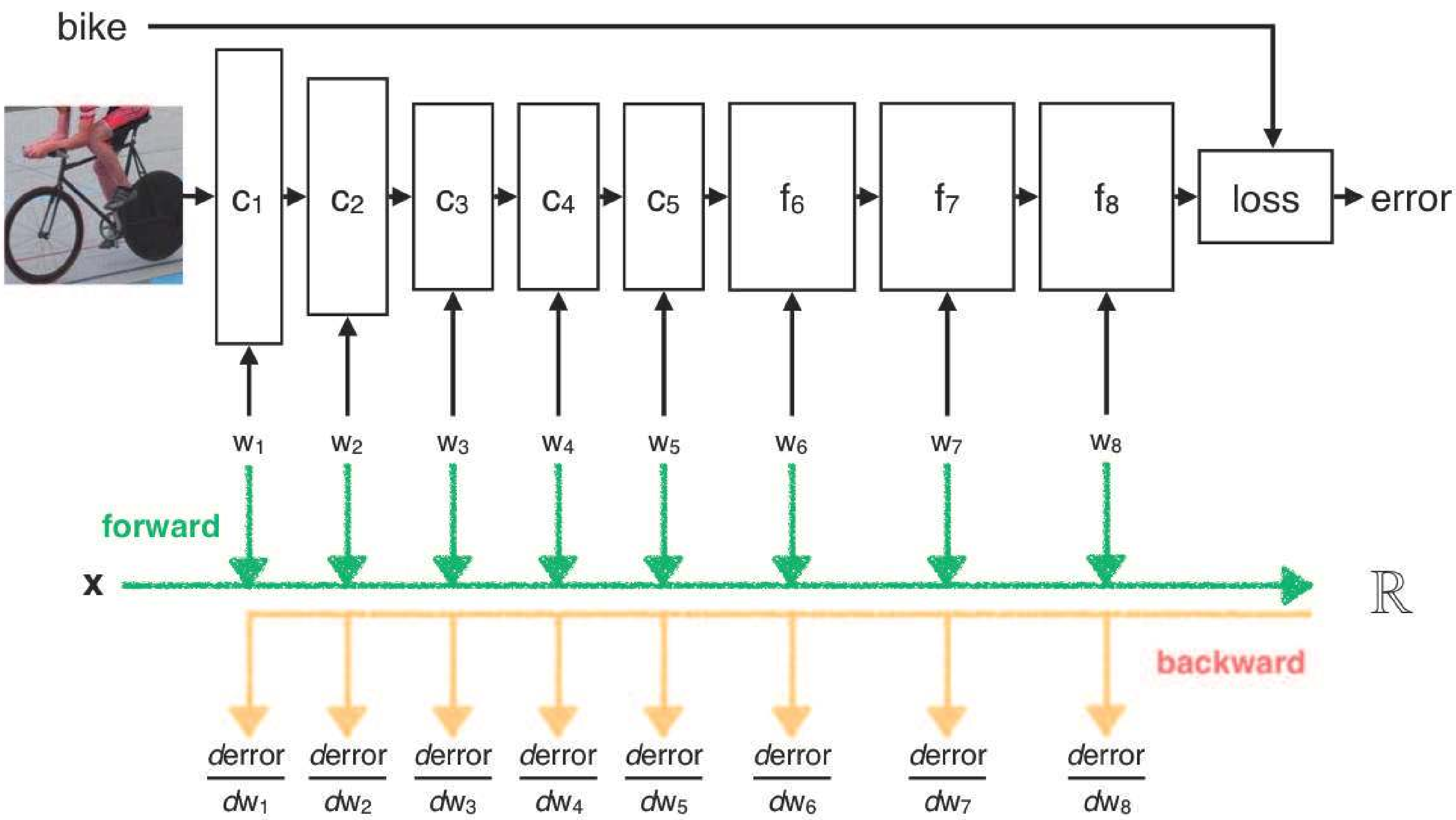
- "Deep" architecture (Convolutional Neural Network)



Why going deep?

Graph of tensors where blocks are trained and optimized jointly

- 1 - 140M trainable parameters



Why Deep Learning works now?

- Five decades of research in machine learning
- Computing and storage power
- Lots of (labelled) data from the internet
- Tools and culture of collaborative and reproducible science
- Resources and efforts from large companies

Why Deep Learning works now?

Five decades of research in ML provided:

- a taxonomy of ML concepts (classification, generative models, clustering, kernels, linear embeddings, etc.),
- a sound statistical formalization (Bayesian estimation, PAC),
- a clear picture of fundamental issues (bias/variance dilemma, VC dimension, generalization bounds, etc.),
- a good understanding of optimization issues,
- efficient large-scale algorithms.

Why Deep Learning works now?

From a practical perspective, deep learning:

- lessens the need for a deep mathematical grasp,
- makes the design of large learning architectures a system/software development task,
- allows to leverage modern hardware (clusters of GPUs),
- does not plateau when using more data,
- makes large trained networks a commodity.

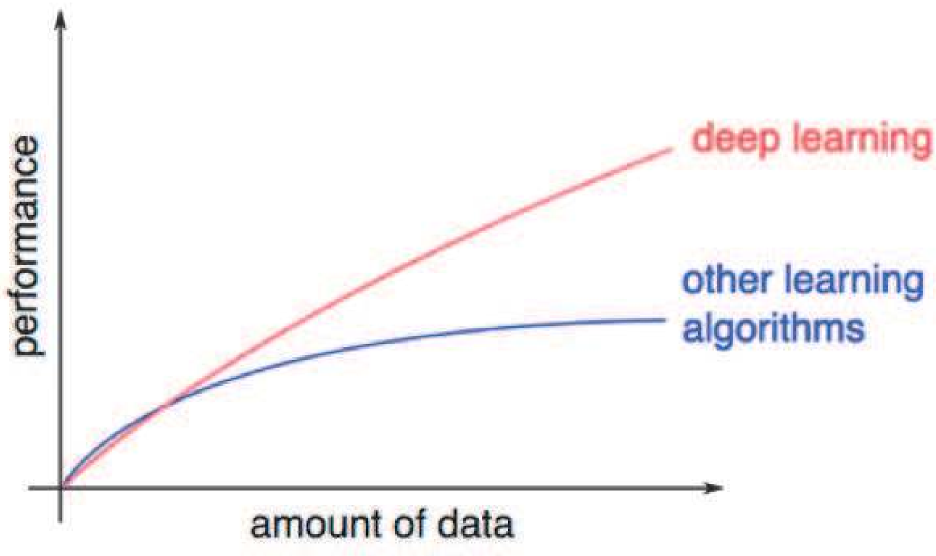
Why Deep Learning works now?

Evolution in computer vision datasets

Data-set	Year	Nb. images	Resolution	Nb. classes
MNIST	1998	6.0×10^4	28×28	10
NORB	2004	4.8×10^4	96×96	5
Caltech 101	2003	9.1×10^3	$\simeq 300 \times 200$	101
Caltech 256	2007	3.0×10^4	$\simeq 640 \times 480$	256
LFW	2007	1.3×10^4	250×250	–
CIFAR10	2009	6.0×10^4	32×32	10
PASCAL VOC	2012	2.1×10^4	$\simeq 500 \times 400$	20
MS-COCO	2015	2.0×10^5	$\simeq 640 \times 480$	91
ImageNet	2016	14.2×10^6	$\simeq 500 \times 400$	$21,841$
Cityscape	2016	25×10^3	$2,000 \times 1000$	30

Why Deep Learning works now?

When more data is available



Why Deep Learning works now?

- Many deep learning frameworks freely available as open source
- Frequent changes and updates (every few weeks)
- Most frameworks supported by a large company

Deep Learning - the hype?

- Many deep learning frameworks freely available as open source
- Frequent changes and updates (every few weeks)
- Most frameworks supported by a GAFA company

PyTorch

PyTorch

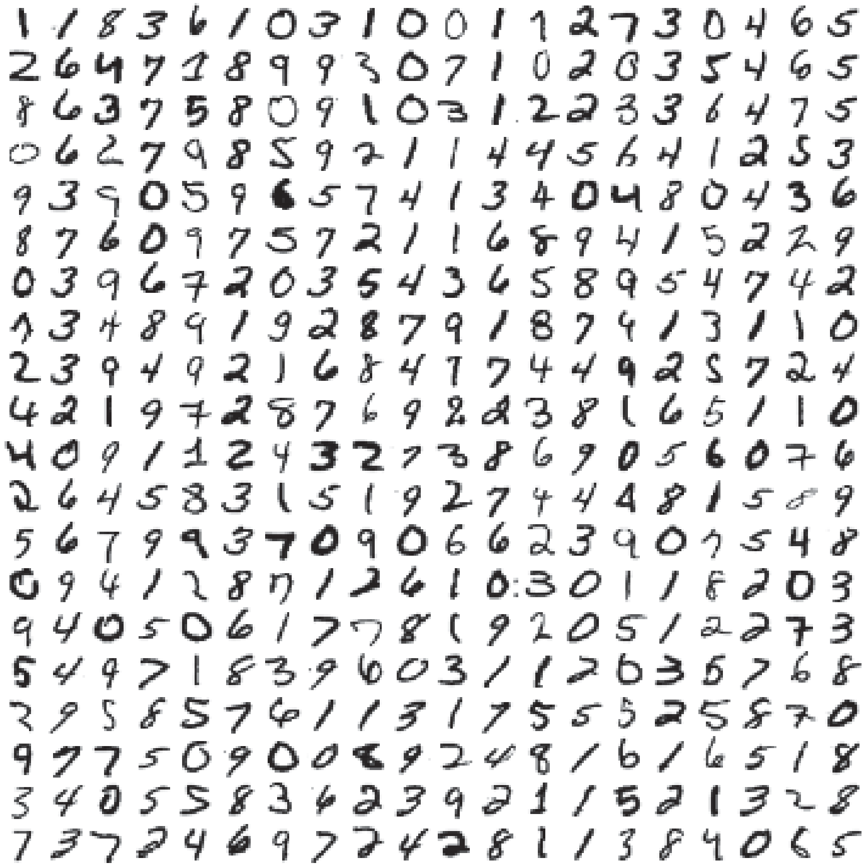
"PyTorch is a python package that provides two high-level features:

- Tensor computation (like numpy) with strong GPU acceleration
- Deep Neural Networks built on a tape-based autograd system

You can reuse your favorite python packages such as numpy, scipy and Cython to extend PyTorch when needed."

PyTorch

MNIST dataset



28×28 grayscale images, $60k$ train samples, $10k$ test samples

```
class Net(nn.Module):
  def __init__(self):
    super(Net, self).__init__()
    self.conv1 = nn.Conv2d(1, 32, kernel_size=5)
    self.conv2 = nn.Conv2d(32, 64, kernel_size=5)
    self.fc1 = nn.Linear(256, 200)
    self.fc2 = nn.Linear(200, 10)

  def forward(self, x):
    x = F.relu(F.max_pool2d(self.conv1(x), kernel_si
    x = F.relu(F.max_pool2d(self.conv2(x), kernel_si
    x = x.view(-1, 256)
    x = F.relu(self.fc1(x))
    x = self.fc2(x)
    return x

model = Net()

mu, std = train_input.data.mean(), train_input.data.
train_input.data.sub_(mu).div_(std)
optimizer = optim.SGD(model.parameters, lr=1e-1)
criterion, batch_size = nn.CrossEntropyLoss(), 100

model.cuda()
criterion.cuda()
train_input, train_target = train_input.cuda(), trai

for e in range(10):
  for b in range(0, nb_train_samples , bs):
    output = model(train_input.narrow(0, b, bs))
    loss = criterion(output , train_target.narrow(0,
    model.zero_grad()
    loss.backward()
    optimizer.step()
```

a few seconds on a low-end GPU, 1% test error

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=5)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=5)
        self.fc1 = nn.Linear(256, 200)
        self.fc2 = nn.Linear(200, 10)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), kernel_size
        x = F.relu(F.max_pool2d(self.conv2(x), kernel_size
        x = x.view(-1, 256)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x

model = Net()

mu, std = train_input.data.mean(), train_input.data.std
train_input.data.sub_(mu).div_(std)
optimizer = optim.SGD(model.parameters, lr=1e-1)
criterion, batch_size = nn.CrossEntropyLoss(), 100

model.cuda()
criterion.cuda()
train_input, train_target = train_input.cuda(), train_t

for e in range(10):
    for b in range(0, nb_train_samples , bs):
        output = model(train_input.narrow(0, b, bs))
        loss = criterion(output , train_target.narrow(0, b,
        model.zero_grad()
        loss.backward()
        optimizer.step()
```

```python
class Net(nn.Module):
  def __init__(self):
    super(Net, self).__init__()
    self.conv1 = nn.Conv2d(1, 32, kernel_size=5)
    self.conv2 = nn.Conv2d(32, 64, kernel_size=5)
    self.fc1 = nn.Linear(256, 200)
    self.fc2 = nn.Linear(200, 10)

  def forward(self, x):
    x = F.relu(F.max_pool2d(self.conv1(x), kernel_size=
    x = F.relu(F.max_pool2d(self.conv2(x), kernel_size=
    x = x.view(-1, 256)
    x = F.relu(self.fc1(x))
    x = self.fc2(x)
    return x

model = Net()

mu, std = train_input.data.mean(), train_input.data.st
train_input.data.sub_(mu).div_(std)
optimizer = optim.SGD(model.parameters, lr=1e-1)
criterion, batch_size = nn.CrossEntropyLoss(), 100

model.cuda()
criterion.cuda()
train_input, train_target = train_input.cuda(), train_

for e in range(10):
  for b in range(0, nb_train_samples , bs):
    output = model(train_input.narrow(0, b, bs))
    loss = criterion(output , train_target.narrow(0, b,
    model.zero_grad()
    loss.backward()
    optimizer.step()
```

```
class Net(nn.Module):
  def __init__(self):
    super(Net, self).__init__()
    self.conv1 = nn.Conv2d(1, 32, kernel_size=5)
    self.conv2 = nn.Conv2d(32, 64, kernel_size=5)
    self.fc1 = nn.Linear(256, 200)
    self.fc2 = nn.Linear(200, 10)

  def forward(self, x):
    x = F.relu(F.max_pool2d(self.conv1(x), kernel_size=
    x = F.relu(F.max_pool2d(self.conv2(x), kernel_size=
    x = x.view(-1, 256)
    x = F.relu(self.fc1(x))
    x = self.fc2(x)
    return x

model = Net()

mu, std = train_input.data.mean(), train_input.data.st
train_input.data.sub_(mu).div_(std)
optimizer = optim.SGD(model.parameters, lr=1e-1)
criterion, batch_size = nn.CrossEntropyLoss(), 100

model.cuda()
criterion.cuda()
train_input, train_target = train_input.cuda(), train_

for e in range(10):
  for b in range(0, nb_train_samples , bs):
    output = model(train_input.narrow(0, b, bs))
    loss = criterion(output , train_target.narrow(0, b,
    model.zero_grad()
    loss.backward()
    optimizer.step()
```

```
class Net(nn.Module):
  def __init__(self):
    super(Net, self).__init__()
    self.conv1 = nn.Conv2d(1, 32, kernel_size=5)
    self.conv2 = nn.Conv2d(32, 64, kernel_size=5)
    self.fc1 = nn.Linear(256, 200)
    self.fc2 = nn.Linear(200, 10)

  def forward(self, x):
    x = F.relu(F.max_pool2d(self.conv1(x), kernel_size=
    x = F.relu(F.max_pool2d(self.conv2(x), kernel_size=
    x = x.view(-1, 256)
    x = F.relu(self.fc1(x))
    x = self.fc2(x)
    return x

model = Net()

mu, std = train_input.data.mean(), train_input.data.st
train_input.data.sub_(mu).div_(std)
optimizer = optim.SGD(model.parameters, lr=1e-1)
criterion, batch_size = nn.CrossEntropyLoss(), 100

model.cuda()
criterion.cuda()
train_input, train_target = train_input.cuda(), train_

for e in range(10):
  for b in range(0, nb_train_samples , bs):
    output = model(train_input.narrow(0, b, bs))
    loss = criterion(output , train_target.narrow(0, b,
    model.zero_grad()
    loss.backward()
    optimizer.step()
```

```
class Net(nn.Module):
  def __init__(self):
    super(Net, self).__init__()
    self.conv1 = nn.Conv2d(1, 32, kernel_size=5)
    self.conv2 = nn.Conv2d(32, 64, kernel_size=5)
    self.fc1 = nn.Linear(256, 200)
    self.fc2 = nn.Linear(200, 10)

  def forward(self, x):
    x = F.relu(F.max_pool2d(self.conv1(x), kernel_size=
    x = F.relu(F.max_pool2d(self.conv2(x), kernel_size=
    x = x.view(-1, 256)
    x = F.relu(self.fc1(x))
    x = self.fc2(x)
    return x

model = Net()

mu, std = train_input.data.mean(), train_input.data.st
train_input.data.sub_(mu).div_(std)
optimizer = optim.SGD(model.parameters, lr=1e-1)
criterion, batch_size = nn.CrossEntropyLoss(), 100

model.cuda()
criterion.cuda()
train_input, train_target = train_input.cuda(), train_

for e in range(10):
   for b in range(0, nb_train_samples , bs):
     output = model(train_input.narrow(0, b, bs))
     loss = criterion(output , train_target.narrow(0,
     model.zero_grad()
     loss.backward()
     optimizer.step()
```

Computer Vision

The old-school way

The problem with Computer Vision

The (human) brain is so good at interpreting visual information that the gap between raw data and its semantic interpretation is difficult to assess intuitively:

This is a mushroom.

This is a mushroom.

```
In [1]: from matplotlib.pyplot import imread
        imread("mushroom-small.png")

Out[1]: array([[[0.03921569, 0.03529412, 0.02352941, 1.        ],
                [0.2509804 , 0.1882353 , 0.20392157, 1.        ],
                [0.4117647 , 0.34117648, 0.37254903, 1.        ],
                ...,
                [0.20392157, 0.23529412, 0.17254902, 1.        ],
                [0.16470589, 0.18039216, 0.12156863, 1.        ],
                [0.18039216, 0.18039216, 0.14117648, 1.        ]],

               [[0.1254902 , 0.11372549, 0.09411765, 1.        ],
                [0.2901961 , 0.2509804 , 0.24705882, 1.        ],
                [0.21176471, 0.2       , 0.20392157, 1.        ],
                ...,
                [0.1764706 , 0.24705882, 0.12156863, 1.        ],
                [0.10980392, 0.15686275, 0.07843138, 1.        ],
                [0.16470589, 0.20784314, 0.11764706, 1.        ]],

               [[0.14117648, 0.12941177, 0.10980392, 1.        ],
                [0.21176471, 0.1882353 , 0.16862746, 1.        ],
                [0.14117648, 0.13725491, 0.12941177, 1.        ],
                ...,
                [0.10980392, 0.15686275, 0.08627451, 1.        ],
                [0.0627451 , 0.08235294, 0.05098039, 1.        ],
                [0.14117648, 0.2       , 0.09803922, 1.        ]],

               ...,
```

This is a mushroom.

This is known as the semantic gap. Extracting semantic information requires models of high complexity.