

EE-559 – Deep learning

1.2. Current applications and success

François Fleuret

<https://fleuret.org/ee559/>

Tue Feb 19 14:18:27 UTC 2019

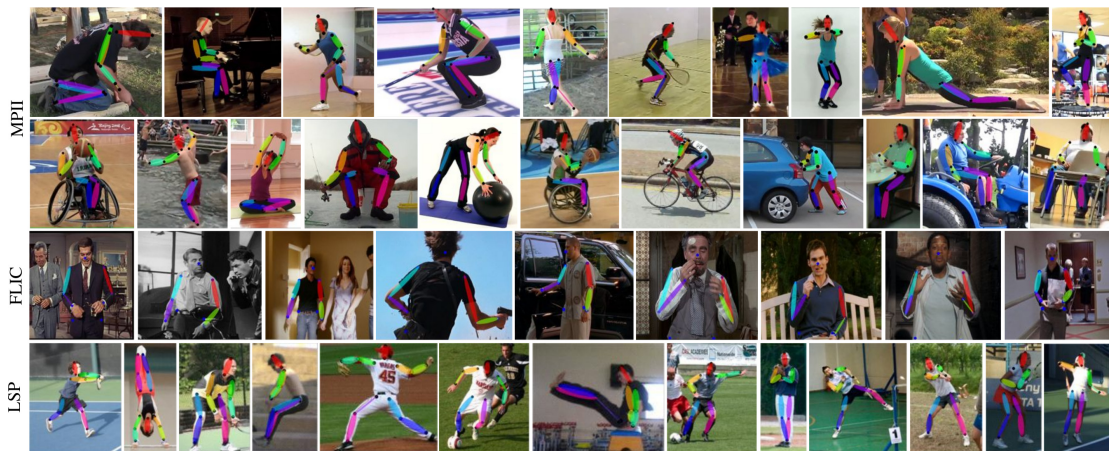


Object detection and segmentation



(Pinheiro et al., 2016)

Human pose estimation



(Wei et al., 2016)

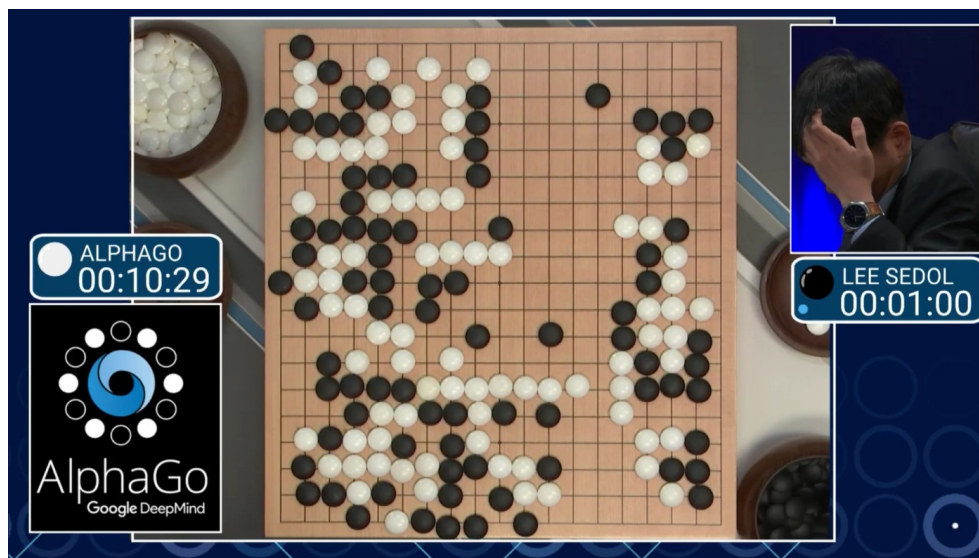
Reinforcement learning



Self-trained, plays 49 games at human level.

(Mnih et al., 2015)

Strategy games



March 2016, 4-1 against a 9-dan professional without handicap.

(Silver et al., 2016)

Translation

“The reason Boeing are doing this is to cram more seats in to make their plane more competitive with our products,” said Kevin Keniston, head of passenger comfort at Europe’s Airbus.

- “La raison pour laquelle Boeing fait cela est de créer plus de sièges pour rendre son avion plus compétitif avec nos produits”, a déclaré Kevin Keniston, chef du confort des passagers chez Airbus.

When asked about this, an official of the American administration replied: “The United States is not conducting electronic surveillance aimed at offices of the World Bank and IMF in Washington.”

- Interrogé à ce sujet, un fonctionnaire de l’administration américaine a répondu: “Les États-Unis n’effectuent pas de surveillance électronique à l’intention des bureaux de la Banque mondiale et du FMI à Washington”

(Wu et al., 2016)

Auto-captioning

A person riding a motorcycle on a dirt road.



Two dogs play in the grass.



A group of young people playing a game of frisbee.



Two hockey players are fighting over the puck.



A herd of elephants walking across a dry grass field.



A close up of a cat laying on a couch.



(Vinyals et al., 2015)

Question answering

I: Jane went to the hallway.

I: Mary walked to the bathroom.

I: Sandra went to the garden.

I: Daniel went back to the garden.

I: Sandra took the milk there.

Q: Where is the milk?

A: garden

I: It started boring, but then it got interesting.

Q: What's the sentiment?

A: positive

(Kumar et al., 2015)

Image generation



(Brock et al., 2018)

Text generation

System Prompt (human-written)

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Model Completion (machine-written, 10 tries)

The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

(Radford et al., 2019)

Why does it work now?

The success of deep learning is multi-factorial:

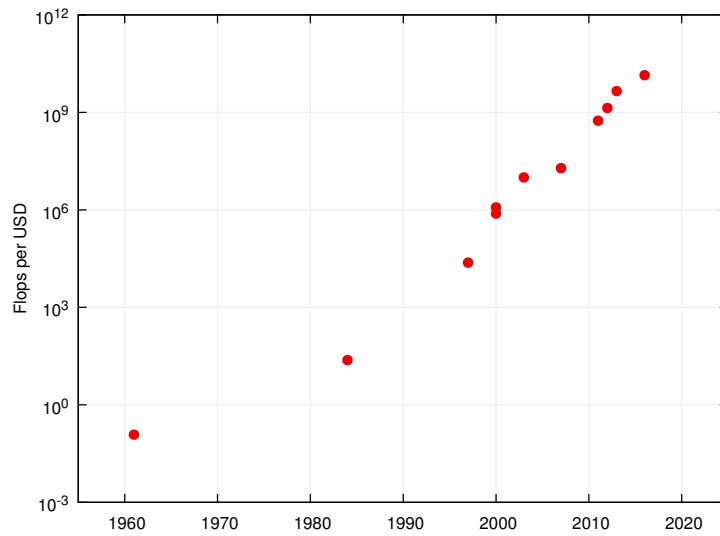
- Five decades of research in machine learning,
- CPUs/GPUs/storage developed for other purposes,
- lots of data from “the internet”,
- tools and culture of collaborative and reproducible science,
- resources and efforts from large corporations.

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.

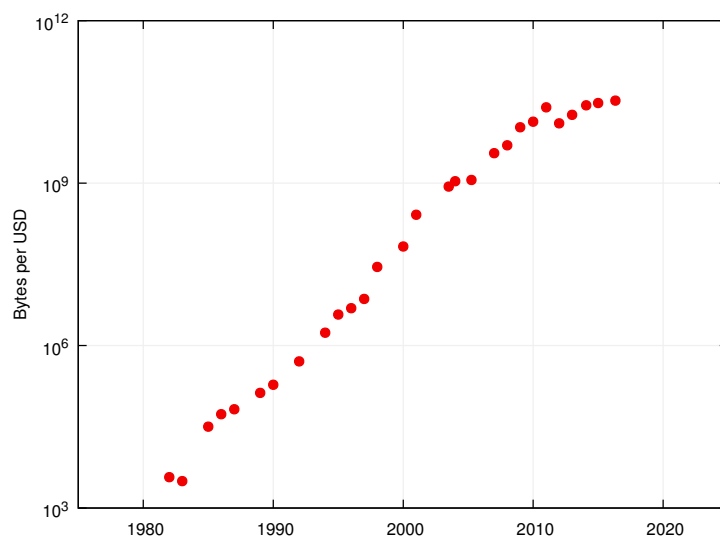
From a practical perspective, deep learning

- 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.



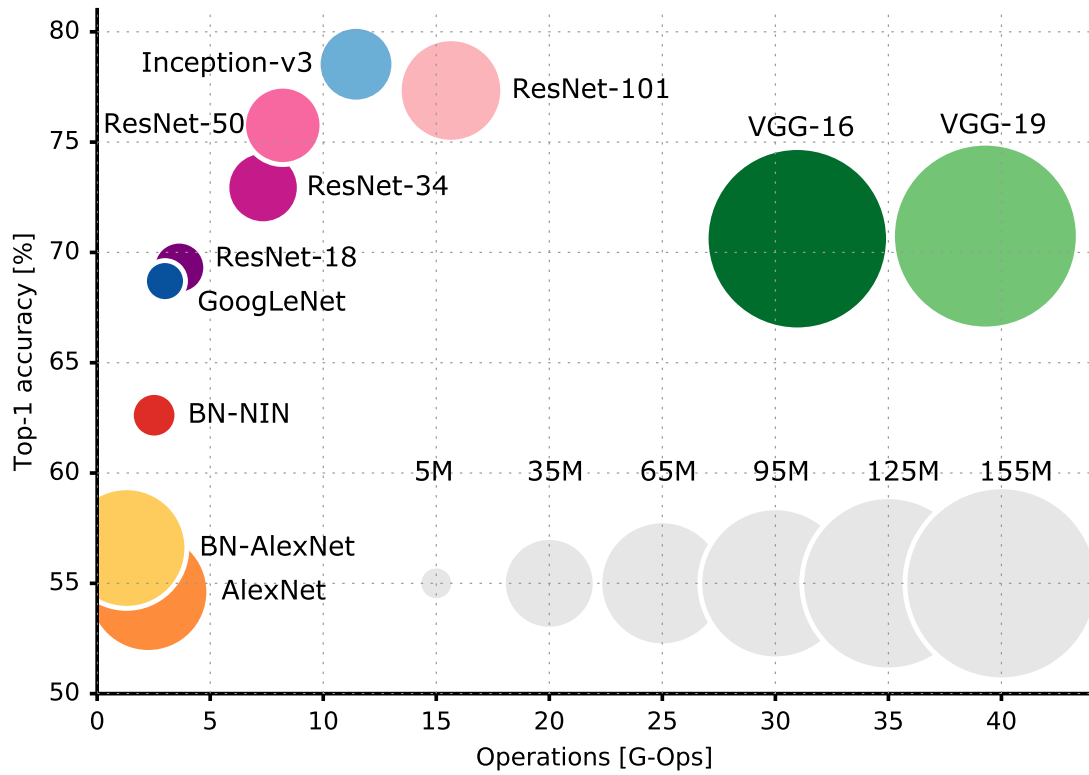
(Wikipedia "FLOPS")

	TFlops (10^{12})	Price	GFlops per \$
Intel i7-6700K	0.2	\$344	0.6
AMD Radeon R-7 240	0.5	\$55	9.1
NVIDIA GTX 750 Ti	1.3	\$105	12.3
AMD RX 480	5.2	\$239	21.6
NVIDIA GTX 1080	8.9	\$699	12.7



(John C. McCallum)

The typical cost of a 4Tb hard disk is \$120 (Dec 2016).



(Canziani et al., 2016)

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

“Quantity has a Quality All Its Own.”

(Thomas A. Callaghan Jr.)

Implementing a deep network, PyTorch

Deep-learning development is usually done in a framework:

	Language(s)	License	Main backer
PyTorch	Python	BSD	Facebook
Caffe2	C++, Python	Apache	Facebook
TensorFlow	Python, C++	Apache	Google
MXNet	Python, C++, R, Scala	Apache	Amazon
CNTK	Python, C++	MIT	Microsoft
Torch	Lua	BSD	Facebook
Theano	Python	BSD	U. of Montreal
Caffe	C++	BSD 2 clauses	U. of CA, Berkeley

A fast, low-level, compiled backend to access computation devices, combined with a slow, high-level, interpreted language.

We will use the PyTorch framework for our experiments.



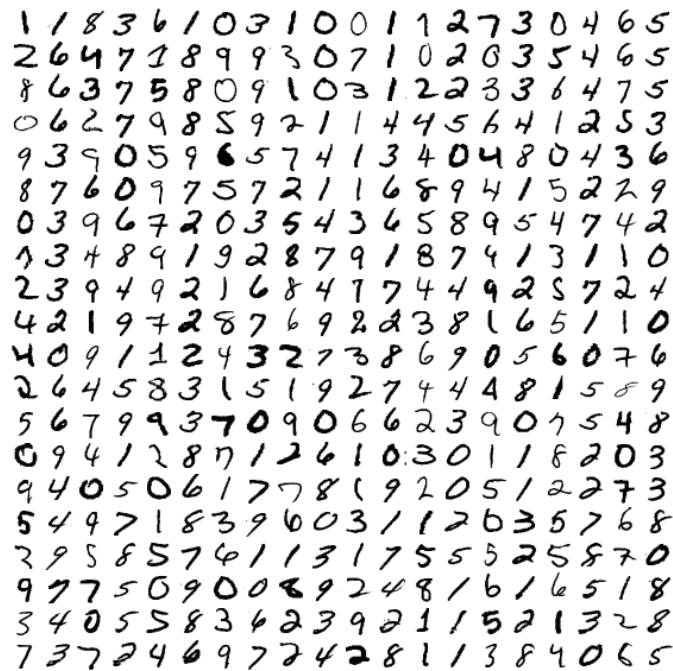
<http://pytorch.org>

"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."

MNIST data-set



1 1 8 3 6 1 0 3 1 0 0 1 1 2 7 3 0 4 6 5
2 6 4 7 1 8 9 9 3 0 7 1 0 2 0 3 5 4 6 5
8 6 3 7 5 8 0 9 1 0 3 1 2 2 3 3 6 4 7 5
0 6 2 7 9 8 5 9 2 1 1 4 4 5 6 4 1 2 5 3
9 3 9 0 5 9 6 5 7 4 1 3 4 0 4 8 0 4 3 6
8 7 6 0 9 7 5 7 2 1 1 6 8 9 4 1 5 2 2 9
0 3 9 6 7 2 0 3 5 4 3 4 5 8 9 5 4 7 4 2
1 3 4 8 9 1 9 2 8 7 9 1 8 7 4 1 3 1 1 0
2 3 9 4 9 2 1 6 8 4 7 7 4 4 9 2 8 7 2 4
4 2 1 9 7 2 8 7 6 9 2 2 3 8 1 6 5 1 1 0
4 0 9 1 1 2 4 3 2 7 3 8 6 9 0 5 6 0 7 6
2 6 4 5 8 3 1 5 1 9 2 7 4 4 4 8 1 5 8 9
5 6 7 9 9 3 7 0 9 0 6 6 2 3 9 0 7 5 4 8
0 9 4 1 2 8 7 1 2 6 1 0 3 0 1 1 8 2 0 3
9 4 0 5 0 6 1 7 7 8 1 9 2 0 5 1 2 2 7 3
5 4 4 7 1 8 3 9 6 0 3 1 1 2 0 3 5 7 6 8
2 9 5 8 5 7 6 1 1 3 1 7 5 5 5 2 5 8 7 0
9 7 7 5 0 9 0 0 8 9 2 4 8 1 6 1 6 5 1 8
3 4 0 5 5 8 3 4 2 3 9 2 1 1 5 2 1 3 2 8
7 3 7 2 4 6 9 7 2 4 2 8 1 1 3 8 4 0 6 5

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

(LeCun et al., 1998)

```
model = nn.Sequential(  
    nn.Conv2d(1, 32, 5), nn.MaxPool2d(3), nn.ReLU(),  
    nn.Conv2d(32, 64, 5), nn.MaxPool2d(2), nn.ReLU(),  
    Flatten(),  
    nn.Linear(256, 200), nn.ReLU(),  
    nn.Linear(200, 10)  
)  
  
nb_epochs, batch_size = 10, 100  
criterion = nn.CrossEntropyLoss()  
optimizer = torch.optim.SGD(model.parameters(), lr = 0.1)  
  
model.to(device)  
criterion.to(device)  
train_input, train_target = train_input.to(device), train_target.to(device)  
  
mu, std = train_input.mean(), train_input.std()  
train_input.sub_(mu).div_(std)  
  
for e in range(nb_epochs):  
    for input, target in zip(train_input.split(batch_size),  
                             train_target.split(batch_size)):  
        output = model(input)  
        loss = criterion(output, target)  
        optimizer.zero_grad()  
        loss.backward()  
        optimizer.step()
```

≈7s on a GTX1080, ≈1% test error

References

- A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis. *CoRR*, abs/1809.11096, 2018.
- A. Canziani, A. Paszke, and E. Culurciello. An analysis of deep neural network models for practical applications. *CoRR*, abs/1605.07678, 2016.
- A. Kumar, O. Irsoy, J. Su, J. Bradbury, R. English, B. Pierce, P. Ondruska, I. Gulrajani, and R. Socher. Ask me anything: Dynamic memory networks for natural language processing. *CoRR*, abs/1506.07285, 2015.
- Y. leCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015.
- P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *European Conference on Computer Vision (ECCV)*, pages 75–91, 2016.
- A. Radford, J. Wu, D. Amodei, D. Amodei, J. Clark, M. Brundage, and I. Sutskever. Better language models and their implications. web, February 2019. <https://blog.openai.com/better-language-models/>.
- D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.
- O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- S. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. *CoRR*, abs/1602.00134, 2016.
- Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google’s neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.