Deep learning

1.2. Current applications and success

François Fleuret
https://fleuret.org/dlc/

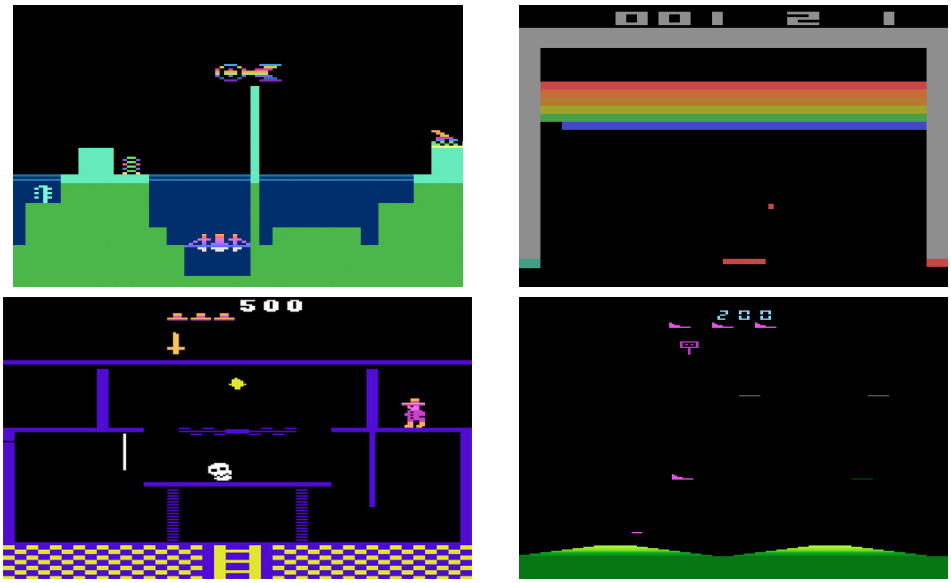
Object detection and segmentation

(Pinheiro et al., 2016)

Notes

Deep learning is now used for virtually any software dealing with complex structured real-world signals.
Semantic segmentation is the task of labeling individual pixels with the class of the object it belongs to, and may also aims at differentiating different instances of the same class (e.g. person, car).

Reinforcement learning



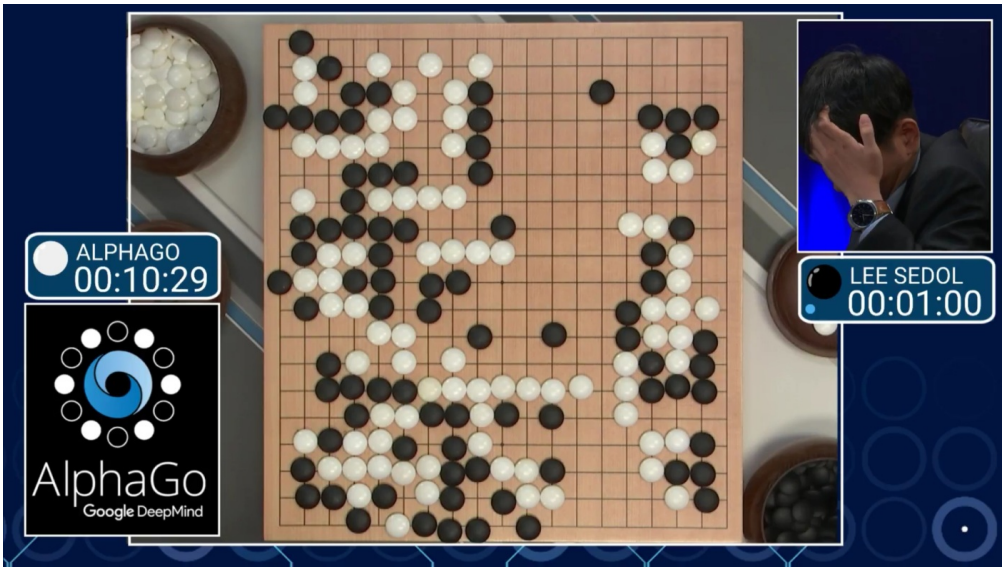
Self-trained, plays 49 games at human level.

(Mnih et al., 2015)

Notes

In reinforcement learning, the goal is to predict the best action to do given the perception of an agent to reach a distant goal.
In the work of Mnih et al. (2015), the machine is trained to play Atari games. It is given the input image, and the model predicts which button to push on the joystick. In this work, the machine is able to play at human level.

Strategy games



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

(Silver et al., 2016)

Notes

In 2016, DeepMind's AlphaGo beat Lee Sedol, one of the best Go players in the world. This was a surprising and shocking result even for experts from the field of Go algorithms, who did not expect it before at least one more decade.
This algorithm combines a stochastic tree search with a neural network to estimate the value of a move. AlphaGo was superseded by AlphaGoZero (Silver et al., 2017), which is trained against itself without data from human games, only from the game's rules, and extended to AlphaZero (Schrittwieser et al., 2019) which can be similarly trained to play chess and shogi (Japanese chess).

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)

Notes

Translation in natural language processing consists in automatically translate a set of sentences from one language to another language.

Question answering

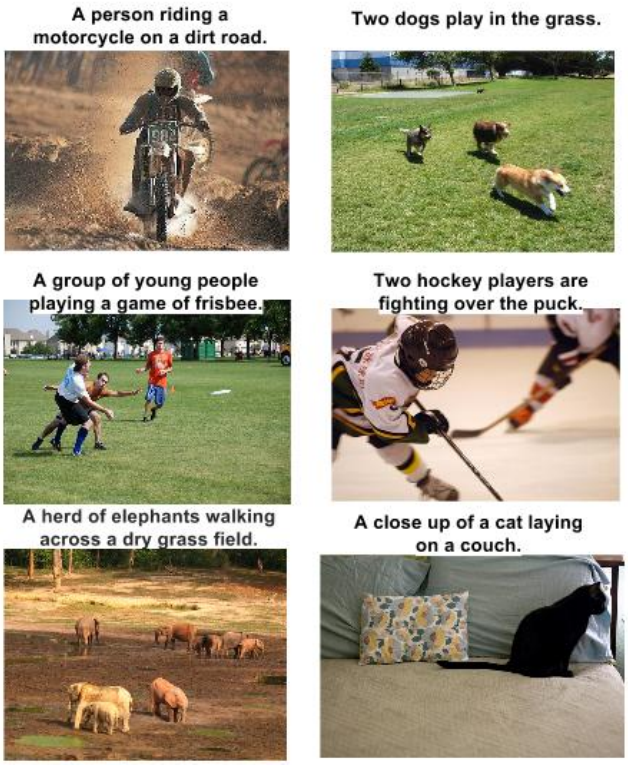
```
I: Jane went to the hallway.
I: Mary walked to the bathroom.
I: Sandra went to the garden.
I: Daniel went back to the garden.
I: Sandra took the milk there.
Q: Where is the milk?
A: garden

I: It started boring, but then it got interesting.
Q: What's the sentiment?
A: positive
```

(Kumar et al., 2015)

Notes

In the task of question answering, the software is given an input text at the end of which it is asked a question about it.
This task requires a form of memory, and the ability to connect nouns, actions, places, etc.
The potential application would be AI-based assistants such as chatbots.

Auto-captioning



(Vinyals et al., 2015)

Notes

Auto-captioning is a mix of image and natural language processing: given an input image, the system outputs a piece of text describing its content.
Auto-captioning has many applications such as retrieving an image from its description, or auto-captioning of images on social medias for blind and visually impaired people.

Image generation

(Brock et al., 2018)

Notes

Deep models are also able to learn distributions of signals such as images, sounds or text and provide means to sample new examples.

Text generation

System Prompt (human-written)

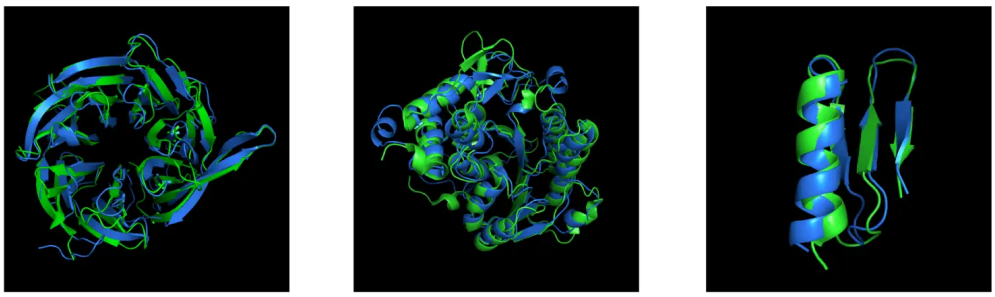
```
In a shocking finding, scientist discovered a herd of unicorns living
in a remote, previously unexplored valley, in the Andes
Mountains. Even more surprising to the researchers was the fact that
the unicorns spoke perfect English.
```

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.
```

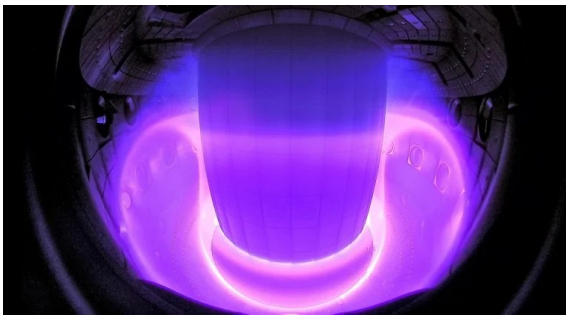
(Radford et al., 2019)

Protein folding prediction



(Jumper et al., 2021)

Plasma confinement



(Degrave et al., 2022)

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.

Notes

The reason of the success of deep learning since 2012 is multi-factorial.
There were five decades of research in machine learning which gave a clear picture of the key theoretical issues in learning theory, and powerful tools in data analysis, optimization, and signal processing.
There were amazing developments of processing hardware (CPUs, GPUs), and storage devices, which were originally made for other applications.
The rise of social media led to the availability of a huge amount of training data. Crowdsourcing platforms, such as Amazon Mechanical Turk, allowed people to perform annotations tasks to train models.
Many tools of machine learning are open source, and are deeply rooted in the culture of collaborative and reproducible science, which allow people to reuse efficiently what has previously been done (git, linux, licenses, distribution, etc.).
There was a change in the world of corporations: we went from a situation where machine learning would have been useful for corporations only if it completely solved a given task, to a situation where there were a financial incentive to improve machine learning even marginally ("click rate").

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.

Notes

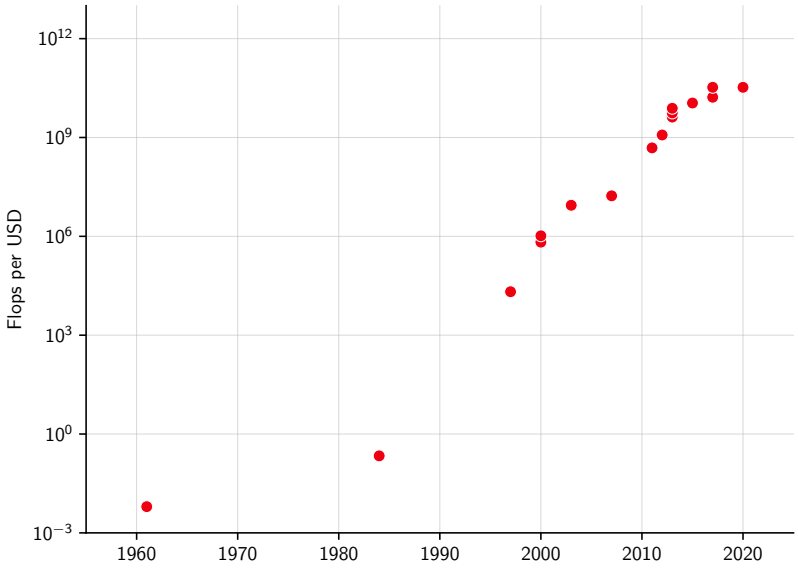
Research in machine learning over the past fifty years brought a taxonomy of principles and methods which predate deep learning.
All these concepts were cast in a sound statistical formalization and help understand of what models work and what generalization capabilities they can have.
The research also brought a clear picture of fundamental issues which arise when training models. In particular, the bias/variance dilemma which states that when the capacity of the model increases, more data is needed to train it.
Concept such as the Vapnik-Chervonenkis dimension helped to characterize the complexity of a model beyond its number of parameters.
Machine learning is closely related to the field of optimization which provides tools to train in very high dimensions spaces, and techniques such as regularization. And it also has strong connection with computer science to make algorithms scale up to very large data sets.

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.

Notes

Deep learning reduces the need for a strong mathematical background: a high level understanding is enough to start using frameworks (PyTorch, Tensorflow, etc.).
An important part of deep learning now is dealing with software issues, that is how to make the code run. The design of models is done at a higher level of combining existing modules, rather than implementing the modules themselves in detail at a low level. These pre-existing implementation and tools also allow to use dedicated hardware and large-scale clusters in a transparent manner.
An important aspect of deep learning is its ability to leverage very large data sets, without plateauing as usual machine learning methods. When one has more data, one can make the models bigger easily (with more parameters), and the performance increases.
A central phenomenon of deep learning is that a trained predictor is an asset in itself. It has become commonplace that people re-use models trained by others to start their task. The frontier between designing algorithms that train networks, and producing trained networks is getting blurry.



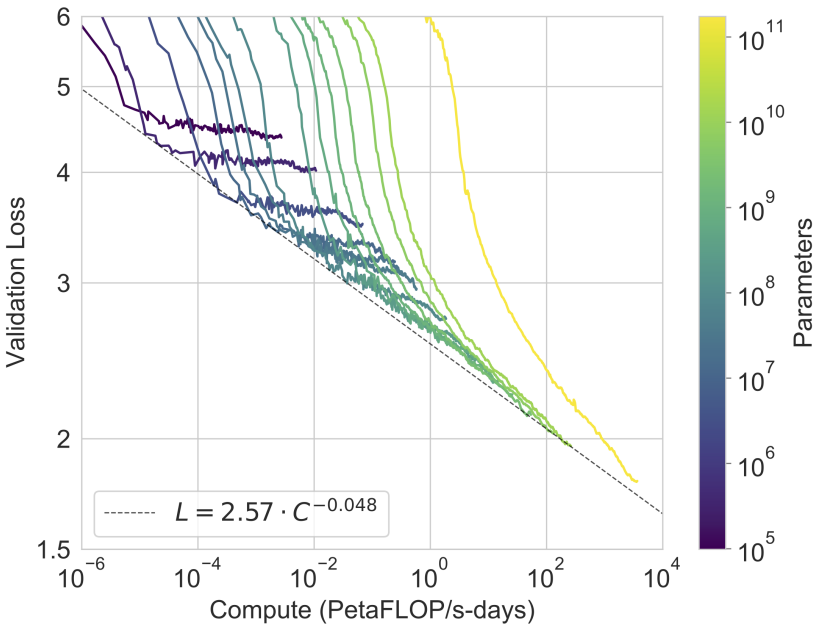
(Wikipedia "FLOPS")

	TFlops (10^{12})	**Price**	**GFlops per $**
Intel Core i7-6700K	0.2	$275	0.7
Intel Core i9-7980XE	0.9	$1'999	0.5
AMD Ryzen 7 PRO 4750G	1.1	$640	1.7
NVIDIA GTX 2080 Ti	14.2	$999	14.2
NVIDIA GTX 3090	35.5	$1'500	23.7
AMD Radeon RX 6900 XT	23.0	$999	23.0

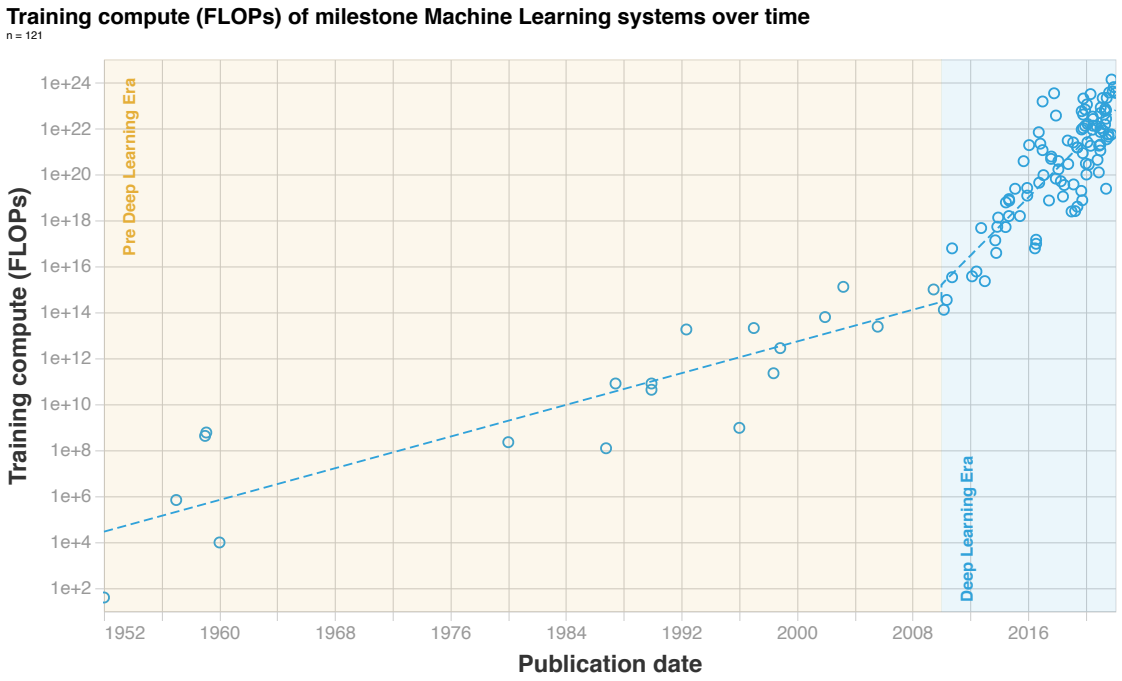
Notes

The graph shows the number of flops (floating point operations per second, log scale) per USD as a function of decades. The increase is exponential and contributed to the rise of deep learning. A standard gaming GPU can now perform $\simeq 20$ trillions (10^{12}) operations per second.
Note that comparing raw numbers can be misleading, since actual performance for deep-learning strongly depends on the drivers and libraries.

Validation loss for language models vs. training compute.



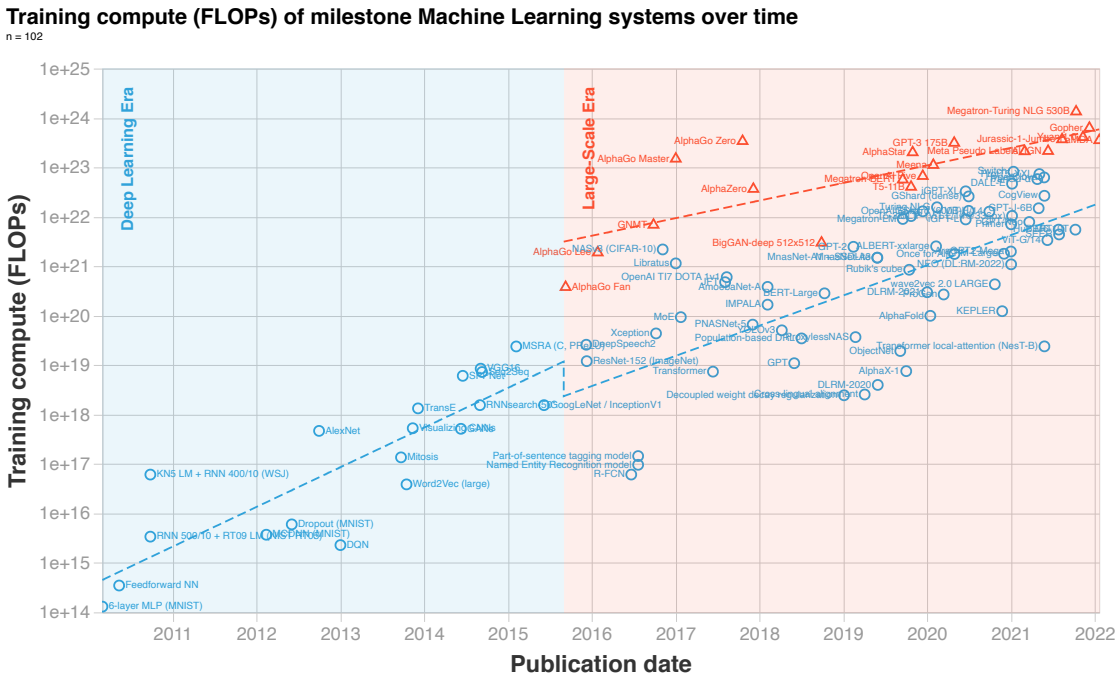
(Brown et al., 2020)



(Sevilla et al., 2022)

Notes

The graph above represents the computation required to train several states-of-the-art models vs. the year.
We observe an exponential increase in the computation required to train models, mostly because researchers solve more complex tasks, but also because the computation power is available.



(Sevilla et al., 2022)

Notes

The graph above represents the computation required to train several states-of-the-art deep learning models vs. the year. These models are not specific to one domain and come from computer vision (AlexNet, VGG), natural language understanding (Seq2Seq, Neural Machine Translation), and strategy games (AlphaZero).

Computer vision

Data-set		Year	Nb. images	Size
MNIST	(classification)	1998	60K	12Mb
Caltech 101	(classification)	2003	9.1K	130Mb
Caltech 256	(classification)	2007	30K	1.2Gb
CIFAR10	(classification)	2009	60K	160Mb
ImageNet	(classification)	2012	1.2M	150Gb
MS-COCO	(segmentation)	2015	200K	32Gb
Cityscape	(segmentation)	2016	25K	60Gb
LAION-5B	(multi-modal)	2022	5.85B	240Tb

Natural Language Processing

Data-set		Year	Size
SST2	(sentiment analysis)	2013	20Mb
WMT-18	(translation)	2018	7Gb
OSCAR	(language model)	2020	6Tb

Notes

In parallel to the growth of the models and computation means, data sets also increased enormously.

The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin.

(Richard Sutton, 2019)

Quantity has a Quality All Its Own.

(Thomas A. Callaghan Jr., 1979)

Notes

One core reason of deep learning's success is purely quantitative: models have very large number of parameters, they require lots of computation, with very large training sets.
It is probably wrong to translate an interpretation of small networks to big ones. Models have become so complicated that it is very hard to have a grasp at what is really happening in detail inside a network. It is likely that processing in very large models qualitatively differs from the processing in small ones.

Implementing a deep network, PyTorch

Deep-learning development is usually done in a framework:

	Language(s)	**License**	**Main backer**
PyTorch	**Python**, C++	BSD	Facebook
TensorFlow	Python, C++	Apache	Google
JAX	Python	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. Python has an incredible ecosystem and is used across fields.

Notes

Most deep learning frameworks consist of

- a backend in a low level language (C/C++/CUDA) which directly interacts with the hardware and the libraries which control the hardware;
- a frontend in a high level language. Python got very popular thanks to his extremely rich ecosystem of libraries for plotting, loading data formats, machine learning, signal processing, etc.

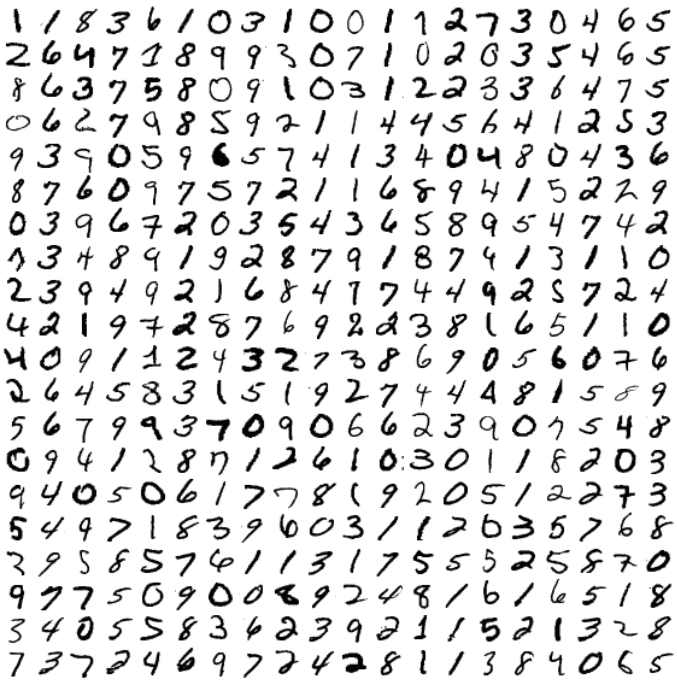
We will use the PyTorch framework for our experiments (Paszke et al., 2019).

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"*

MNIST data-set



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

(LeCun et al., 1998)

Notes

MNIST is one of the most standard computer vision data set published in 1998, and can be seen as a minimal real-world image problem. A good practice when designing a new method is to benchmark it on this corpus to have a grasp of how it behaves.

① ② ③ ④ ⑤

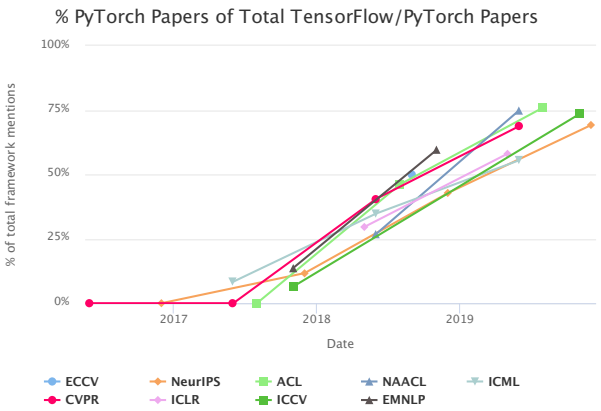
```
model = nn.Sequential(
    nn.Conv2d( 1, 32, 5), nn.MaxPool2d(3), nn.ReLU(),
    nn.Conv2d(32, 64, 5), nn.MaxPool2d(2), nn.ReLU(),
    nn.Flatten(),
    nn.Linear(256, 200), nn.ReLU(),
    nn.Linear(200, 10)
)

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_targets = train_input.to(device), train_targets.to(device)

mu, std = train_input.mean(), train_input.std()
train_input.sub_(mu).div_(std)

for e in range(nb_epochs):
    for input, targets in zip(train_input.split(batch_size),
                              train_targets.split(batch_size)):
        output = model(input)
        loss = criterion(output, targets)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

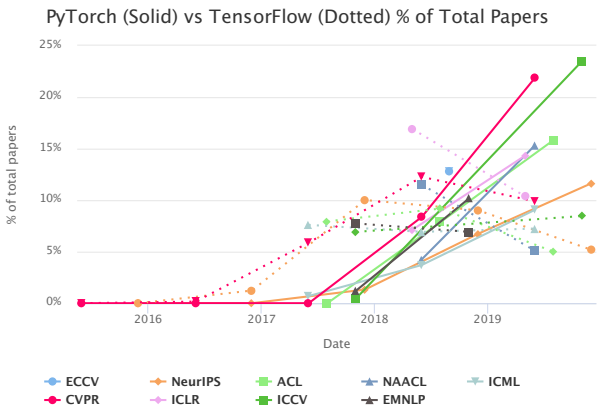
$\simeq$8s on a GTX1080, $\simeq$1% test error

Notes

Here is a minimal piece of code to train a network for classifying MNIST images. This snippet of code is shown here to give a sense of the amount and complexity of code required, we will re-visit in details all the aspects in the course:

1. The model is a sequence of pre-defined PyTorch modules.
2. We define the parameters of the optimization, including the choice of the criterion that specified the quantity to minimize, and the optimizer itself which is the method we will use.
3. The data and the model are moved to the device we want to use, usually a GPU.
4. The data set is normalized. We will see that normalizing the data set is important for faster and better training.
5. The training itself consist of several loops, each going through all the data by "mini batches" of samples.


% PyTorch Papers of Total TensorFlow/PyTorch Papers
% of total framework mentions
100%
75%
50%
25%
0%
2017
2018
2019
Date
ECCV
CVPR
NeurIPS
ICLR
ACL
ICCV
NAACL
EMNLP
ICML


PyTorch (Solid) vs TensorFlow (Dotted) % of Total Papers
% of total papers
25%
20%
15%
10%
5%
0%
2016
2017
2018
2019
Date
ECCV
CVPR
NeurIPS
ICLR
ACL
ICCV
NAACL
EMNLP
ICML

(He, 2019)

References

A. Brock, J. Donahue, and K. Simonyan. **Large scale GAN training for high fidelity natural image synthesis**. CoRR, abs/1809.11096, 2018.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. **Language models are few-shot learners**. CoRR, abs/2005.14165, 2020.

J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de las Casas, C. Donner, L. Fritz, C. Galperti, A. Huber, J. Keeling, M. Tsimpoukelli, J. Kay, A. Merle, J.-M. Moret, S. Noury, F. Pesamosca, D. Pfau, O. Sauter, C. Sommariva, S. Coda, B. Duval, A. Fasoli, P. Kohli, K. Kavukcuoglu, D. Hassabis, and M. Riedmiller. **Magnetic control of tokamak plasmas through deep reinforcement learning**. Nature, 602(7897):414–419, 2022.

H. He. **The state of machine learning frameworks in 2019**. web, Oct 2019. https://thegradient.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry/.

J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis. **Highly accurate protein structure prediction with alphafold**. Nature, 596(7873):583–589, Aug 2021.

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.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. **Pytorch: An imperative style, high-performance deep learning library**. In Neural Information Processing Systems (NeurIPS), pages 8024–8035, 2019.

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/.

J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver. **Mastering atari, go, chess and shogi by planning with a learned model**. CoRR, abs/1911.08265, 2019.

J. Sevilla, L. Heim, A. Ho, T. Besiroglu, M. Hobbhahn, and P. Villalobos. **Compute trends across three eras of machine learning**. CoRR, abs/2202.05924, 2022.

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.

D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. **Mastering the game of go without human knowledge**. Nature, 550:354–358, Oct. 2017.

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.

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.