# CS 194-129 Deep Neural Networks
# Spring 2018 Canny Lecture 1

## 1 Reading

The true challenge to artificial intelligence arises from solving problems that can be easily performed [intuitively] by humans but cannot be easily described [formally]. For example, speech recognition is an "intuitive problem." Currently, the solution to such problems lies in allowing computers to learn from experience and to understand the world in a hierarchy of concepts, with each concept defined through its relation to simpler ones. In this way a computer can learn complicated ideas by building them out of simpler elements. Typically this process occurs over many layers, which can be represented as a deep graph. Hence this approach to AI is called **deep learning**.

Instead of relying on hardcoded knowledge, AI systems should be able to acquire their own knowledge via **machine learning** (which here means "extracting patterns from raw data"). It is important to note that the performance of machine learning approaches depends heavily on the data representation they are given. Relevant features must be present in the representation, whether implicitly or explicitly, or the ML algorithm will not be able to make a meaningful prediction.

However, it is often difficult to know exactly which features should be extracted. Therefore we often use ML to discover not only the mapping from representation to output but also the representation itself (**representation learning**). The quintessential example of a representation learning algorithm is the autoencoder, which combines an encoder (which converts the input data into a different representation) and a decoder (which converts the new representation into the original format). The goal is to create a new representation with various nice properties while preserving information present in the original form.

Deep learning learns representations in an incremental way, by building representations out of other, simpler representations. The quintessential example of a deep learning model is the feedforward deep network, or **multilayer perceptron**. This is just a mathematical function mapping a set of input values to output values, and is formed by composing many simpler functions. Each application of a different mathematical function will provide a new representation of the input.

In summary, deep learning is an approach to AI, specifically a type of machine learning (a technique that enables computer systems to improve with experience and data) which represents the world as a nested hierarchy of concepts. The book contends that machine learning is the *only* viable approach for building AI systems capable of operating in complex real-world environments.

### The Increasing Accuracy, Complexity, and Impact of Deep Learning

In recent years, deep learning has seen tremendous growth in popularity and usefulness, primarily as a result of more powerful computers (enabling the training of more complex models), larger datasets, and a plethora of techniques for training deeper networks.

- **Increasing model sizes.** Biological neurons are not very densely connected, so our models have had a *number of connections per neuron* similar to that of mammalian brains for decades. On the other hand, neural networks are not projected to have the same number of neurons as the human brain [in total] until at least the 2050s. As it stands, total model size has been doubling roughly every 2.4 years.
- **Increasing dataset sizes.** As of 2016, a rough rule of thumb is that a supervised deep learning algorithm achieves acceptable performance with about 5000 labeled examples per category and matches or exceeds human performance when trained with a dataset of at least 10 million labeled examples.

Deep learning has had a dramatic impact on tasks such as speech recognition and image segmentation. After all, the complexity of solvable tasks increases alongside the scale and accuracy of deep networks.

- LSTMs are able to output an entire sequence of characters transcribed from an image, rather than just identify a single object.
- RNNs such as the aforementioned LSTM sequence model can be used to model relationships between *sequences* and other *sequences* rather than just fixed inputs. Such sequence-to-sequence learning is on the cusp of revolutionizing machine translation.
- Neural Turing machines (i.e. self-programming technology), though in their infancy, learn to read from memory cells and write arbitrary content to memory cells. Such neural networks can learn programs from only examples of desired behavior (e.g. learn to sort lists of numbers given examples of scrambled and sorted sequences).
- In the context of reinforcement learning, an autonomous agent must learn to perform a task by trial and error, without any guidance from the human operator. RL systems are capable of playing Atari games and performing complex manipulation tasks, among a vast expanse of other applications.

## 2 Lecture

There is no doubt that AI has seen tremendous success in recent times. For starters, the application scope of AI is much higher than it used to be. We no longer need to customize a representation for every new task; instead we can apply the same methods to several different problems and achieve good results on all. Additionally, many problems previously considered "AI-hard" no longer hold that distinction. Despite our previous belief that Jeopardy, Go, and face recognition would require general intelligence, machines are already able to accomplish these tasks at a human level or better – *and they're still improving.* If competition results for [TIMIT] speech recognition and image classification are any indicator, deep learning error floors are drawing ever further away from the classical AI floors of the past.

Nevertheless, results are not as optimistic as news articles often make them out to be (in one sense because the toy problems we're solving are not the same as the real problems we aim to solve). The press likes to report on things that are wrong, things that are eye-catching and fantastical. One truth that the media has touched on, however, is that there *are* many risks associated with AI:

- **Economic risks:** AI *will* displace jobs. Machines perform very well at specific tasks, and human societies are stratified by specialization when it comes to employment. Maybe machines don't have general intelligence, but it doesn't matter... one need not be good at jeopardy to excel at a service job.
- **Existential risks:**
  - **Security:** As a result of AI we might have mutating viruses, which have never been dealt with before. Furthermore, with all of our interconnectedness a security breach could cause any number of systems to run amok. Considering our infrastructure, this could be absolutely devastating in an infinite number of ways.
  - **The internet:** Unlike physical systems with checks and balances, there is no limit to the speed at which virtual entities (e.g. entities on the internet) can move around and reproduce. This is a real risk, and demands addressing ASAP. Especially when it comes to AI, technology is not showing the same kinds of limits as previous generations!
  - When Google acquired DeepMind (a company building state-of-the-art AI systems), one of the conditions DeepMind imposed was that Google create an AI ethics/evaluation board to study new technologies. This was motivated in part by the thought that DeepMind, once a cottage company, might now find itself in everyone's homes where its AI could do some real damage.

According to Yann LeCun, DNNs require intuitive insights, theoretical modeling, practical implementation, and empirical/scientific analysis in order to be understood. There is no single framework or set of principles that can be used to explain everything. On that note, this class is structured around effective design patterns and analysis for deep networks. We will use visualization to understand such networks, e.g. as the animation of optimization strategies for a DNN.

The first third of the class will focus on computer vision and general DNN principles. The second third will involve natural language. The final third will involve imitation/reinforcement learning.

## Rationale for Deep Neural Networks

### Phases of Neural Network Research

- **1940s-1960s:** Cybernetics, developing electronic signals to emulate the brain
- **1960s-1980s:** Computer systems went digital
- **1980s-1990s:** Connectionism, backpropagation, performing complex calculations in aggregate
- **1990s-2000s:** Computational learning theory grew up, yet learning still computationally hard
- **2006- :** Deep learning, end-to-end training, large datasets, many applications

Why has deep learning been so successful?

Well, first of all, it turns out the problems we thought were hard were actually not so hard after all. Often when we called a problem hard, we were really saying that *an instance* of the problem was hard. But in practice we have an entire distribution of input instances, and we care more about the [more reasonable] typical case than we do about the worst case. One might argue that even if this is the case, it is an NP-hard task to identify the optimal solution when training a deep network. *But we don't need the optimal solution.* Instead of finding the optimal solution, we can use gradient-based or greedy algorithms in order to find a *greedy solution* – and as it happens, this isn't half as difficult. In this sense, if a problem has a greedy solution it can be considered "easy." Therefore, the problems we want to solve are actually easy! Many learning problems do indeed have reliable steepest descent paths to a good model.

Also, hierarchical representations (manifested as a multi-layer structure) have helped a lot. Before DNNs, people had hand-designed hierarchical models for computer vision (input image $\rightarrow$ edge image $\rightarrow$ 2.5D sketch $\rightarrow$ 3D model). Now, even without hand-engineering, CNNs are able to do something similar by going from pixels to edges to contours to blobs... simply by virtue of their architecture and training process.

Why are hierarchical representations so useful? We can uncover an answer to this question by observing humans. In general, humans learn higher-level skills via an iterative process. We develop one skill (e.g. hitting a tennis ball), then advance to the next skill (e.g. serving or volleying). This matches well with deep architectures for learning. Human learning is hierarchical and we know it works, so it makes a lot of sense that hierarchical models would also work for deep networks.

## A Brief History of Computer Vision

- **1970:** David Marr articulated that vision is a challenge of layered representation, arguing that we should go from *raw pixels* to *an edge image circumscribing objects* to *a view-dependent model with depth* to *a view-independent 3D model.*
- **1978:** Marr and Nishihara came up with shape descriptions, dictating that we should break big objects into, e.g., limbs and cylinders (the generalized cylinders approach).
- **1991:** Turk and Pentland arrived at eigenfaces, a data-driven approach to recognizing faces. In it, we would first perform PCA on a set of face images (i.e. compute the eigenvectors). These eigenvectors represent principal components for the family of images, aka additive stereotypes for a face. Typically a face would load heavily on the first few of these principal components. Anyway, PCA allowed us to express general faces in terms of such a basis. It was interesting that this was a data-driven approach, which was somewhat new at the time.
- **1999:** Lowe presented SIFT, which allowed us to recognize special signature regions of objects using

view-invariant features (under the idea that these features would be highly predictive of the class of an object). SIFT allowed us to perform feature matching, simply by comparing these view-invariant features for multiple patches.

- **2003:** Sivic and Zisserman presented a bag-of-words approach, where we would take feature sets and construct code books (i.e. clustering all possible representations of features into a smaller set of codes, and then coding the image using the nearest matching code). This simplified the problem of matching an initially analog representation of a feature to matching a *code*. In the method, we would take an area of an image, compute the view-independent SIFT features, then match the resulting high-dimensional vector with the nearest matching code vector. At this point the image patch would be assigned a code, and this code could be matched to the codes of corresponding regions in other images.
- **2005:** Dalal and Triggs presented Histogram of Gradients, allowing us to generate even bigger codes.
- **2006:** PASCAL, ImageNet, and other standardized benchmarking competitions arose. This proved to be a nice barometer for progress on the image classification task, and to this day we are still improving. Amazingly, nets for these competitions are doing what David Marr anticipated all the way back in 1970, even though they are not specifically engineered for it! (Feature representations at each layer are not hand-engineered; accordingly, nets are inferring intermediate representations with no explicit cues. What's impressive is that they have somehow learned these representations in a greedy fashion – using only gradient descent on the loss.)

## References

[1] Ian Goodfellow and Yoshua Bengio and Aaron Courville. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`