Study Guide: Deep Learning by Yoshua Bengio, Yann Lecun & Geoffrey Hinton- Nature (2015, 2021)

Mohammed R. Osman · Follow
40 min read · Apr 20, 2023

Three Grandfathers of AI review where AI has been and future thoughts.

Photo by Nicolas I. on Unsplash

This study guide written for an online machine learning study group “ML Reading Club” from Reddit’s r/machinelearning & r/learnmachinelearning.

The point of this is not to summarize the paper but explain the important points and provide help for people trying to learn and get deeper on to the topics. There are plenty of summaries available elsewhere, I wanted to help people using a different angle.

The structure I’m using is an expanded outline. Regular points and explanations in my own words are written regularly. Direct quotes from sources are put into quote blocks directly after links.

I hope people are better able to learn from these seminal machine learning papers by using these guides. All feedback is appreciated to help make it

better.

Deep learning — Nature

Deep learning allows computational models that are composed of multiple processing layers to learn representations of...

www.nature.com

Nature volume 521, pages 436–444 (2015)

This guide is about the classic review paper by Bengio, LeCun, and Hinton about the state of Deep Learning. This paper was published in the journal *Nature* and is behind a paywall. You can find multiple places to download it anyways with a simple google search. But just to be safe, there will not be any direct quotes from it.

Deep Learning for AI

By Yoshua Bengio, Yann Lecun, Geoffrey Hinton
Communications of the ACM, July 2021, Vol. 64 №7, Pages 58–65
10.1145/3448250

In 2021, these authors revisited their review paper and published an update, which is fully open and free to anyone. I will be breaking things down and heavily quoting that paper in the second half of this guide. In my opinion it is a better use of time to focus on their most current thoughts.

They also did a sit down podcast about their thoughts which is a worthwhile listen:

After reviewing the 2021 paper, the guide will end on a little bit of biography on the 3 authors for those interested.

Deep Learning (2015)

Success of Deep Learning

Deep Learning requires less domain knowledge, can process more raw data, and improve with more data

Applied Domains

- Speech Recognition (translation)
- Visual Object Recognition (ImageNet)
- Object Detection (Face, Pedestrian)
- Drug Discovery
- Genomics (predicting effects of mutations)
- Particle Accelerators
- Brain Circuits

- Natural Language (Topic Classification, Sentiment Analysis, Question & Answer, Language Translation)

“Conventional” Machine Learning Weaknesses

- Processing “natural” or raw data
- Feature selection: time & domain knowledge intensive
- Low-Level Sensing
- Pre-processing
- Feature Extraction
- Inference (Prediction, Recognition)

Input -> Feature Representation -> Learning Algorithm

Supervised Learning: Active model making.

dataset -> labeling -> training (errors, tuning parameters, gradient descent) -> testing

Gradient Descent

- Imagine prediction errors or 'error gradient' as a high dimensional (multiple features) landscape, learning is finding the minimal points or "hill climbing" by adjusting the weights.
- Stochastic gradient descent speeds up the process by randomly picking points in the (error) terrain to do calculations, then averaging the results from multiple runs.
- Great general overview of the math

> *Showing input vector, computing outputs and errors, computing average gradient, adjusting weights accordingly. Repeating for many small sets until objective function stop decreasing. Why stochastic: small set gives a noisy estimate of the average gradient over all examples*

- Not as good techniques: Linear classifiers or shallow classifiers (must have good features). Input space -> half-spaces; Kernel methods: do not generalize well

Unsupervised Deep Learning -> Representation Learning

- Raw Data
- Layers of features generated without human involvement
- Automatically discover representations
- Non-linear modules combined to higher/abstract representation
- Learning pushes functions to go from from simple to complex

Basics of Neural Networks

- Inputs, Outputs (X, y)
- Parameters: weights (w), bias (b)
- Activation Function (phi) (non-linearity)

Common Activation Functions:

- Rectified Linear Unit (ReLU) <- Most important
- Exponential Linear Unit (ELU)
- Leaky ReLU (ReLU mod)
- Maxout (ReLU/Leaky ReLU mod)
- Sigmoid <- Classic
- tanh (Sigmoid mod)
- Great general overview on activation functions

A multi-layer neural network can distort the input space to make the classes of data linearly separable

Forward & Back Propagation

Feed-forward neural networks use Forward Propagation: straight shot from inputs to outputs w/o adjustment.

- Strengths: Fast, computationally less demanding, easier math
- Weaknesses: Gets ‘stuck’ in local minima, needs lots of runs

Back Propagation: After Forward Pass, working backwards and adjusting weights & bias by comparing final output prediction to ground truth. Primary feature of ‘Supervised’ Learning. Goes through all the layers.

- Strengths: Introduces optimization, yields great results
- Weakness: Computationally intensive

> *Chain Rule: The chain rule of derivatives tells us how two small effects (that of a small change of x on y, and that of y on x) are composed. A small change Δx in x gets transformed first into a small change Δy in y by getting multiplied by ∂ y / ∂ x (that is, the definition of partial derivative). Similarly, the change Δy creates a change Δz in z. Substituting one equation into the other gives the chain rule of derivatives — how Δx gets turned into Δz through multiplication by the product of ∂ y⁄∂ x and ∂ z⁄∂ x. It also works when x, y and z are vectors (and the derivatives are Jacobian matrices).*

“Practical application” of the chain rule for derivatives.

> *Revived around 2006 by unsupervised learning procedures with unlabeled data. In practice, local minima are rarely a problem.*

Fully Connected Neural Net — resource intensive, complex, hard to interpret, but contains all possible connections between adjacent layers.

Locally Connected Neural Net — ‘Stationarity’ or statistics are similar at different locations. This is how neural networks start looking at data in parts (ex. separate objects in an image file, snippets of audio clip, trends in time series)

Convolutional Net (CNN, sometimes referred to as ConvNet)— Learning multiple filters. The filtering operation is termed ‘discrete convolution’ , hence the name. Filtering is a bit of a hard concept when thinking of neural networks, try to think of each node or perceptron over multiple layers combining to specialize in a particular idea/symbol/concept. So when the

signal from the input goes all the way to the output layer, a certain path takes prominence.

Convolution: 'A form or shape that is folded in curved or tortuous windings' — Webster's Dictionary

Pooling:

> *A technique for generalizing features extracted by convolutional filters and helping the network recognize features independent of their location in the image.*

Feature map (sometimes called 'Activation Map')

> *In CNNs, the feature map is the output of one filter applied to the previous layer. It is called a feature map because it is a mapping of where a certain kind of feature is found in the image. Convolutional Neural Networks look for "features" such as straight lines, edges, or even objects.*

Pooling Layer (also known as 'Pooled Feature Maps')

> *Pooling layers provide an approach to down sampling feature maps by summarizing the presence of features in patches of the feature map. Two common pooling methods are average pooling and max pooling that summarize the average presence of a feature and the most activated presence of a feature respectively.*

Classification (sometimes “Recognition”): Predicted labels

Object Detection = Classification + Localization: Predicted labels and location

> *Different Object recognition tasks. Image classification is a task where an image is classified into one or multiple classes based on the task. Image/Object localization is a regression problem where the output is x and y coordinates around the object of interest to draw bounding boxes.*

Instance Segmentation: Classification + Localization + Boundaries

> *a special form of image segmentation that deals with detecting instances of objects and demarcating their boundaries. It finds large-scale applicability in real-world scenarios like self-driving cars, medical imagining, aerial crop monitoring, and more.*

Recurrent Neural Networks (RNN):

> *artificial neural network which uses sequential data or time series data. These deep learning algorithms are commonly used for ordinal or temporal problems, such as language translation, natural language processing (nlp), speech recognition, and image captioning*

Data with time & order or sequential data qualities are key to RNNs.

From a great cheat sheet:

> Advantages*:*
>
> *- Possibility of processing input of any length*
>
> *- Model size not increasing with size of input*
>
> *- Computation takes into account historical information*

> *- Weights are shared across time*
>
> Disadvantages*:*
>
> *- Computation being slow*
>
> *- Difficulty of accessing information from a long time ago*
>
> *- Cannot consider any future input for the current state*

For more details on math inside RNNs:

Vectors (Thought, State)

- State or Hidden State contains sequence information up to (but not including) the current time step

> *Hidden layers are… layers that are hidden from view on the path from input to output. Hidden states are technically speaking* inputs *to whatever we do at a given step, and they can only be computed by looking at data at previous time steps.*

- Thought Vector:

> *"Thought vector" is a term popularized by Geoffrey Hinton… which is using vectors based on natural language to improve its search results.*
>
> *A thought vector is like a word vector, which is typically a vector of 300–500 numbers that represent a word. A word vector represents a word's meaning as it relates to other words (its context) with a single column of numbers.*
>
> *That is, the word is embedded in a vector space using a shallow neural network like word2vec, which learns to generate the word's context through repeated guesses.*
>
> *A thought vector, therefore, is a vectorized thought, and the vector represents one thought's relations to others. A thought vector is trained to generate a thought's context. Just as a words are linked by grammar (a sentence is just a path drawn across words), so thoughts are linked by a chain of reasoning, a logical path of sorts.*

Long Short Term Memory network (LSTM): Improvement on RNN via "memory cell" composed of gates and hidden states. LSTMs are allowed to *forget*.

> *One of the first and most successful techniques for addressing vanishing gradients came in the form of the long short-term memory (LSTM) model due to Hochreiter and Schmidhuber (1997). LSTMs resemble standard recurrent neural networks but here each ordinary recurrent node is replaced by a* memory cell. *Each memory cell contains an* internal state*, i.e., a node with a self-connected recurrent edge of fixed weight 1, ensuring that the gradient can pass across many time steps without vanishing or exploding.*
>
> *The term "long short-term memory" comes from the following intuition. Simple recurrent neural networks have* long-term memory *in the form of weights. The weights change slowly during training, encoding general knowledge about the data. They also have* short-term memory *in the form of ephemeral activations, which pass from each node to successive nodes. The LSTM model introduces an intermediate type of storage via the memory cell. A memory cell is a composite unit, built from simpler nodes in a specific connectivity pattern, with the novel inclusion of multiplicative nodes.*

States & Gates:

- Gates: Input, Forget, Output
- States: Gated Hidden State, Memory Cell Internal State

Slightly off topic: if you're interested in LSTMs check out its next evolution: Bidirectional LSTM (BiLSTM). BiLSTM has an additional LSTM layer that reverses the direction of information flow. So the input flows in both directions, allowing the LSTM to use information from both sides and model words (and grouping of words like phrases) sequential dependencies. Good explainer "Bi-LSTM" by Raghav Aggarwal on Medium.

Distributed Representations and Language Processing

Distributed Representations: Features (or ‘elements’) are not mutually exclusive and the variety of different combinations and configurations could *surpass* the number of variations found in the training data. This is because learning distributed representations enable generalization to new or novel combinations. Layers of representation in a deep networks creates the potential for exponential performance gains (“exponential in the depth”).

> *Memory Networks can answer questions that require complex inference.*

Future of Deep Learning

- Potential of Unsupervised Learning
- CNNs & RNNs + reinforcement learning in computer vision
- NLP

Deep Learning for AI (2021)

> *Research on artificial neural networks was motivated by the observation that human intelligence emerges from highly parallel networks of relatively simple, non-linear neurons that learn by adjusting the strengths of their connections.*
>
> *This observation leads to a central computational question: How is it possible for networks of this general kind to learn the complicated internal representations that are required for difficult tasks such as recognizing objects or understanding language?*

Neural network research is heavily inspired by biological functions. It's hilarious that even the legends in the field are in awe of how brains work and even more amazed at how well artificial neural networks perform in the real world.

> *Deep learning seeks to answer this question by using many layers of activity vectors as representations and learning the connection strengths that give rise to these vectors by following the stochastic gradient of an objective function that measures how well the network is performing.*
>
> *It is very surprising that such a conceptually simple approach has proved to be so effective when applied to large training sets using huge amounts of computation and it appears that a key ingredient is depth: shallow networks simply do not work as well.*

Complexity through multiple layers of nodes leads to learning. Here “layers of activity vectors” aka the hidden layers in neural networks create emergent “representations.” Representations here is a catchall word for abstract ideas, symbols, similarity/dissimilarity, etc. As a neural network processes inputs through all the layers, the layers start to ‘filter’ them into silos, ending in the output layer. The more layers or deeper a neural network is, the more likely it will accurately understand what the input data has. This is how Deep Learning works.

> *These challenges include learning with little or no external supervision, coping with test examples that come from a different distribution than the training examples, and using the deep learning approach for tasks that humans solve by using a deliberate sequence of steps which we attend to consciously — tasks that Kahneman calls* system 2 *tasks as opposed to* system 1 *tasks like object recognition or immediate natural language understanding, which generally feel effortless.*

The authors here are starting to bring up the differences between Supervised vs Unsupervised machine learning. Unsupervised machine learning is able to handle ‘raw’ or ‘natural’ data much better than Supervised. Supervised requires much more active participation by the users as well.

> *Put simply, the logic-inspired paradigm views sequential reasoning as the essence of intelligence and aims to implement reasoning in computers using hand-designed rules of inference that operate on hand-designed symbolic expressions that formalize knowledge.*
>
> *The brain-inspired paradigm views learning representations from data as the essence of intelligence and aims to implement learning by hand-designing or*

> *evolving rules for modifying the connection strengths in simulated networks of artificial neurons.*
>
> *In the logic-inspired paradigm, a symbol has no meaningful internal structure: Its meaning resides in its relationships to other symbols which can be represented by a set of symbolic expressions or by a relational graph.*

Inspired by our own biology, this is how researchers first started out designing and using machine learning. It's much more logical (and intuitive) to our human minds to handcraft AIs. But it didn't perform nearly as well as hoped. Research in neural networks stagnated a bit because it was stuck in the Supervised paradigm. Both leaps in compute power and theory were needed.

> *In the brain-inspired paradigm the external symbols that are used for communication are converted into internal vectors of neural activity and these vectors have a rich similarity structure. Activity vectors can be used to model the structure inherent in a set of symbol strings by learning appropriate activity vectors for each symbol and learning non-linear transformations that allow the activity vectors that correspond to missing elements of a symbol string to be filled in.*

> *The main advantage of using vectors of neural activity to represent concepts and weight matrices to capture relationships between concepts is that this leads to automatic generalization. If Tuesday and Thursday are represented by very similar vectors, they will have very similar causal effects on other vectors of neural activity. This facilitates analogical reasoning and suggests that immediate, intuitive analogical reasoning is our primary mode of reasoning, with logical sequential reasoning being a much later development…*

The hardest part of this passage is “activity vectors.” Let’s go back to the physics definition of vectors: a value that describes magnitude and direction. In the machine learning context, feature vectors represent an object’s numeric or symbolic characteristics (basically features). I interpret activity vectors, as written by the authors here, as a grouping of feature vectors during learning. The AI is learning concepts, symbols and groupings of them, which allows it to identify & include similar objects in the absence of explicit labeling in the training data or missing information.

> *Deep learning re-energized neural network research in the early 2000s by introducing a few elements which made it easy to train deeper networks. The*

> *emergence of GPUs and the availability of large datasets were key enablers of deep learning and they were greatly enhanced by the development of open source, flexible software platforms with automatic differentiation such as Theano Torch, Caffe, TensorFlow, and PyTorch. This made it easy to train complicated deep nets and to reuse the latest models and their building blocks.*

More compute, larger datasets, and advancements in software platforms were a big help.

> *But the composition of more layers is what allowed more complex non-linearities and achieved surprisingly good results in perception tasks*

More hidden layers gives more opportunities for the neural network to form ‘’complex non-linearities’ and correctly filter the input.

> *It is important to realize that it is not simply a question of having more parameters, since deep networks often generalize better than shallow networks with the same number of parameters. The practice confirms this. The most popular*

> *class of convolutional net architecture for computer vision is the ResNet family43 of which the most common representative, ResNet-50 has 50 layers.*

Very interesting take, parameters (think number of nodes or perceptrons in a layer) is less important than the number of layers used. Depth is the key quality.

> *Other ingredients not mentioned in this article but which turned out to be very useful include image deformations, drop-out, and batch normalization.*

Image Deformation: from Wikipedia, “In computer graphics, free-form deformation (FFD) is a geometric technique used to model simple deformations of rigid objects. It is based on the idea of enclosing an object within a cube or another hull object, and transforming the object within the hull as the hull is deformed.”

Drop-out Regularization: A method used to prevent over-fitting by the neural network. Randomly selected neurons are ignored during training, temporally on the forward pass and weight changes are skipped during backward propagation. This prevents overspecialization and forces the neural network to be more robust by having multiple pathways for

recognition of the same object. Hopefully this increases the generalizability of the network.

Batch Normalization: Standardization or Normalization of inputs between layers to increase training speeds. The compute time required to pre-process the signals is dwarfed by the speed gains when the number of layers is high.

> *We believe that deep networks excel because they exploit a particular form of compositionality in which features in one layer are combined in many different ways to create more abstract features in the next layer.*
>
> *For tasks like perception, this kind of compositionality works very well and there is strong evidence that it is used by biological perceptual systems.*

From Compositionality in Computer Vision:

> *People understand the world as a sum of its parts. Events are composed of other actions, objects can be broken down into pieces, and this sentence is composed of a series of words. When presented with new concepts, people can decompose the*

> *novelty into familiar parts. Our knowledge representation is naturally compositional. Unfortunately, many of the underlying architectures that catalyze vision tasks generate representations that are not compositional…compositionality in computer vision — — the notion that the representation of the whole should be composed of the representation of its parts. As humans, our perception is intertwined greatly by reasoning through composition: we understand a scene by components, a 3D shape by parts, an activity by events, etc.*

What the authors in the paper are saying is that computer vision by deep networks are able to breakdown inputs (pictures, scenes, etc) into smaller components to understand both what it contains inside and the visual data as a whole. Again, the multitude of layers in the network allow it to form symbols and representations that it can filter for.

> ***Unsupervised pre-training.*** *When the number of labeled training examples is small compared with the complexity of the neural network required to perform the task, it makes sense to start by using some other source of information to create layers of feature detectors and then to fine-tune these feature detectors using the limited supply of labels.*

This is a really interesting comment snuck in before talking about transfer learning. Essentially, the authors are thinking that Transfer Learning is the key to deal with situations where labels are too few or labeling is too onerous (cost, time-wise). IMO this recommendation is not conventional thinking, but I could be wrong.

> *In transfer learning, the source of information is another supervised learning task that has plentiful labels. But it is also possible to create layers of feature detectors without using any labels at all by stacking auto-encoders.*
>
> *First, we learn a layer of feature detectors whose activities allow us to reconstruct the input.*
>
> *Then we learn a second layer of feature detectors whose activities allow us to reconstruct the activities of the first layer of feature detectors.*
>
> *After learning several hidden layers in this way, we then try to predict the label from the activities in the last hidden layer [breaking this sentence into two -MO, pt. 1]*
>
> *and we backpropagate the errors through all of the layers in order to fine-tune the feature detectors that were initially discovered without using the precious*

> *information in the labels. [pt. 2]*

So what's going on here?:

> *The pre-training may well extract all sorts of structure that is irrelevant to the final classification but,* ***in the regime where computation is cheap and labeled data is expensive****, this is fine so long as the pre-training transforms the input into a representation that makes classification easier. [bold used for emphasis -MO]*
>
> *In addition to improving generalization, unsupervised pre-training initializes the weights in such a way that it is easy to fine-tune a deep neural network with back propagation.*

So transfer learning is dependent on re-optimizing the neural network originally trained on something else for a new use case, hoping the symbols in the original have some relevance to the new training data. Otherwise more compute time is necessary to do more iterations of optimization using back propagation.

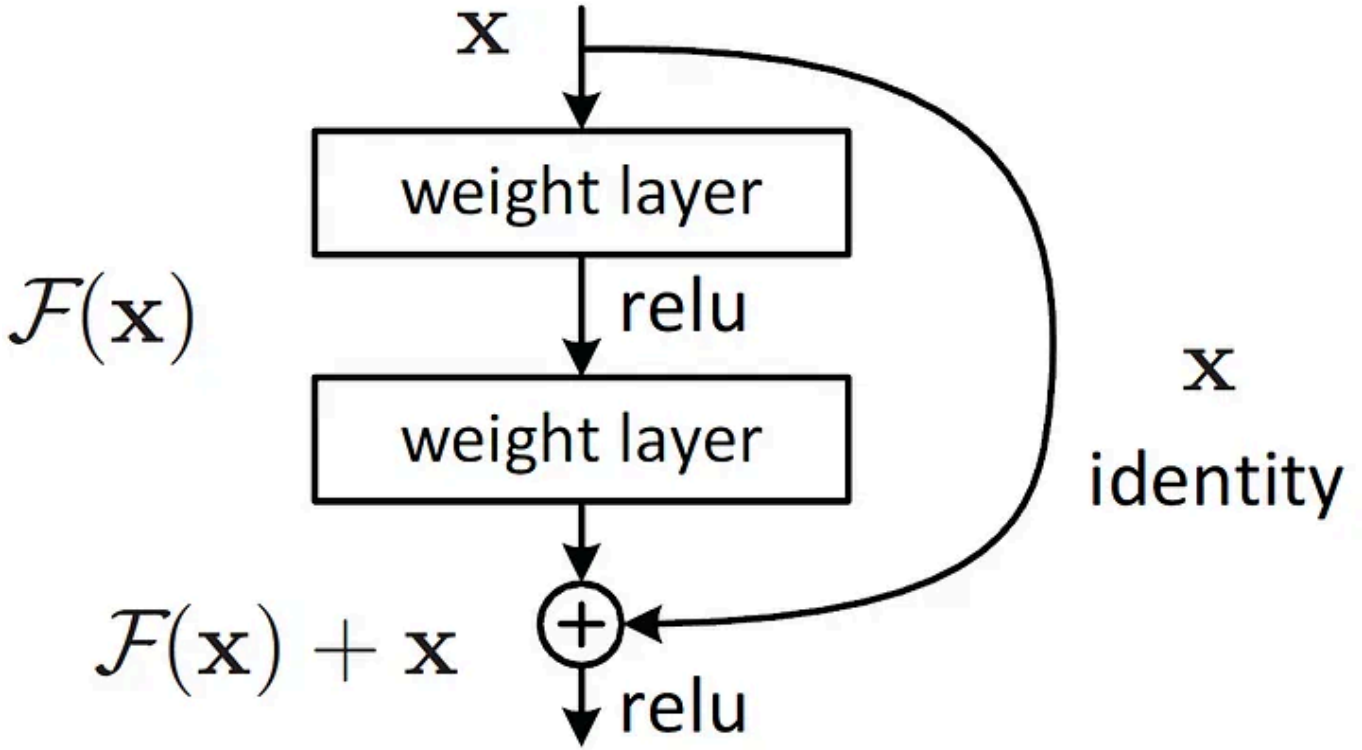
> *The effect of pre-training on* optimization *was historically important for overcoming the accepted wisdom that deep nets were hard to train, but it is much*

> *less relevant now that people use rectified linear units and residual connections.*

Rectified linear units (ReLU) was already covered earlier. Residual connections are defined as:

> *Residual Connections are a type of skip-connection that learn residual functions with reference to the layer inputs, instead of learning unreferenced functions.*
>
> *The intuition is that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping. To the extreme, if an identity mapping were optimal, it would be easier to push the residual to zero than to fit an identity mapping by a stack of nonlinear layers.*

This figure from the residual connections paper helps in understanding it better:



Clipped from Resnet documentation

So going top to bottom, it looks like a regular neural network with connections from input through 2 hidden layers. Except now we got this 'x identity' that carries information that skips internal transformation by the layers (and thus not altered), this is the residual connection.

> *However, the effect of pre-training on* generalization *has proved to be very important. It makes it possible to train very large models by leveraging large*

> *quantities of unlabeled data, for example, in natural language processing, for which huge corpora are available.*
>
> *The general principle of pre-training and fine-tuning has turned out to be an important tool in the deep learning toolbox, for example, when it comes to transfer learning or even as an ingredient of modern meta-learning.*

This is confusing to me because the original model used for transfer learning was supposed to have been made using data sets with labeling but here the authors are saying the original model used for transfer learning can be unsupervised too. So…maybe trial and error is necessary to see if it works? Hope the back propagation works its magic?

> ***Unsupervised and self-supervised learning.*** *Supervised learning, while successful in a wide variety of tasks, typically requires a large amount of human-labeled data. Similarly, when reinforcement learning is based only on rewards, it requires a very large number of interactions. These learning methods tend to produce task-specific, specialized systems that are often brittle outside of the narrow domain they have been trained on.*

Very critical considerations. So how do the authors advise overcoming them?

> *Reducing the number of human-labeled samples or interactions with the world that are required to learn a task and increasing the out-of-domain robustness is of crucial importance for applications such as low-resource language translation, medical image analysis, autonomous driving, and content filtering.*
>
> *Humans and animals seem to be able to learn massive amounts of background knowledge about the world, largely by observation, in a task-independent manner. This knowledge underpins common sense and allows humans to learn complex tasks, such as driving, with just a few hours of practice. A key question for the future of AI is how do humans learn so much from observation alone?*
>
> *In supervised learning, a label for one of* N *categories conveys, on average, at most* $\log_2(N)$ *bits of information about the world. In model-free reinforcement learning, a reward similarly conveys only a few bits of information. In contrast, audio, images and video are high-bandwidth modalities that implicitly convey large amounts of information about the structure of the world. This motivates a form of prediction or reconstruction called self-supervised learning which is training to "fill in the blanks" by predicting masked or corrupted portions of the data. Self-supervised learning has been very successful for training transformers to*

> *extract vectors that capture the context-dependent meaning of a word or word fragment and these vectors work very well for downstream tasks.*
>
> *For text, the transformer is trained to predict missing words from a discrete set of possibilities. But in high-dimensional continuous domains such as video, the set of plausible continuations of a particular video segment is large and complex and representing the distribution of plausible continuations properly is essentially an unsolved problem.*

This is the first time in either paper transformers were brought up. In the seminal paper "Attention is All You Need", transformer models are:

> *The dominant sequence transduction models are based on complex recurrent or convolutional neural networks in an encoder-decoder configuration. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train.*

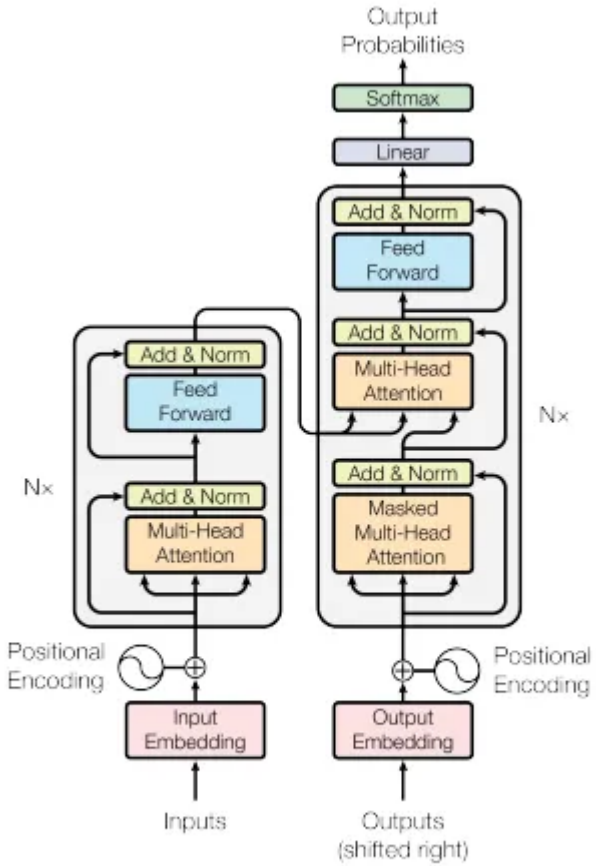


Figure 1: The Transformer - model architecture.

from “Attention is All You Need” by Vaswani et al.

From Nvidia’s official blog:

> *A transformer model is a neural network that learns context and thus meaning by tracking relationships in sequential data like the words in this sentence.*

> *Transformer models apply an evolving set of mathematical techniques, called attention or self-attention, to detect subtle ways even distant data elements in a series influence and depend on each other.*

Truely paradigm shifting, next level work. To fully describe transformers would require a whole post on its own. But these two links are a good start. More info available in 'Dive into Deep Learning' Chapter 11 (Section 7) Transformer Architecture

> ***The mysterious success of rectified linear units.*** *The early successes of deep networks involved unsupervised pre-training of layers of units that used the logistic sigmoid nonlinearity or the closely related hyperbolic tangent. Rectified linear units had long been hypothesized in neuroscience and already used in some variants of RBMs and convolutional neural networks. It was an unexpected and pleasant surprise to discover that rectifying non-linearities (now called ReLUs, with many modern variants) made it easy to train deep networks by backprop and stochastic gradient descent, without the need for layerwise pre-training.*

That list of activation functions is more important than you think!

> ***Soft attention and the transformer architecture.*** *A significant development in deep learning, especially when it comes to sequential processing, is the use of multiplicative interactions, particularly in the form of soft attention.*
>
> *This is a transformative addition to the neural net toolbox, in that it* ***[Transformers] changes neural nets from purely vector transformation machines into architectures which can dynamically choose which inputs they operate on, and can store information in differentiable associative memories.*** *[bold for emphasis -MO]*

Now the neural network not just has memory like the RNNs, but can be selective about the what inputs go through the layers and how.

> *A key property of such architectures is that they can effectively operate on different kinds of data structures including sets and graphs.*

So greater flexibility in data types, hallmark of Unsupervised, but with additional flexibility in terms of order.

> *Soft attention can be used by modules in a layer to dynamically select which vectors from the previous layer they will combine to compute their outputs. This can serve to make the output independent of the order in which the inputs are presented (treating them as a set) or to use relationships between different inputs (treating them as a graph).*

Soft and Hard attention are concepts in computer vision that are on the harder end and explaining them would be too involved for this post. I'll post some links to the topic area. From the Institute of Physics "An Overview of the Attention Mechanisms in Computer Vision", the Jonathan Hui blog post "Soft & hard attention," and "Attention mechanism" by Heuritech on Medium. Very, very simply put: Attention is having the neural network look at specific areas of the image to classify certain things that the AI expects to find there, as opposed to looking at the image as a whole. Soft attention is deterministic and Hard attention is stochastic (so an estimate) in terms of gradient descent calculations.

> *Another way to think about attention mechanisms is that they make it possible to dynamically route information through appropriately selected modules and combine these modules in potentially novel ways for* ***improved out-of-distribution generalization****. [bold for emphasis -MO]*

That last part of the excerpt is key: Transformers can be robust enough handle situations the training set didn't have. The incredible performance of transformers cannot be understated.

> *Transformers have produced dramatic performance improvements that have revolutionized natural language processing…*
>
> *Perhaps more surprisingly, transformers have been used successfully to solve integral and differential equations symbolically. A very promising recent trend uses transformers on top of convolutional nets for object detection and localization in images with state-of-the-art performance. The transformer performs post-processing and object-based reasoning in a differentiable manner, enabling the system to be trained end-to-end.*

> ***Contrastive learning.*** *One way to approach this problem is through latent variable models that assign an energy (that is, a badness) to examples of a video and a possible continuation.*

From V7 Lab's blog post "The Beginner's Guide to Contrastive Learning":

> *Contrastive Learning is a Machine Learning paradigm where unlabeled data points are juxtaposed against each other to teach a model which points are similar and which are different. That is, as the name suggests, samples are contrasted against each other, and those belonging to the same distribution are pushed towards each other in the embedding space. In contrast, those belonging to different distributions are pulled against each other…*
>
> *Contrastive Learning mimics the way humans learn. For example, we might not know what otters are or what grizzly bears are, but seeing the images… we can at least infer which pictures show the same animals. The basic contrastive learning framework consists of selecting a data sample, called "anchor," a data point belonging to the same distribution as the anchor, called the "positive" sample, and another data point belonging to a different distribution called the "negative" sample. The SSL model tries to minimize the distance between the anchor and positive samples, i.e., the samples belonging to the same distribution, in the latent space, and at the same time maximize the distance between the anchor and the negative samples.*

So some labeling is required, hence Contrastive learning is sometimes regarded as 'Semi-supervised' machine learning. If you see 'Self-supervised', that means that the AI is actively labeling the unstructured and or unlabeled raw input data as it flows through the model. The AI needs to have some base

truth from which to see the similarities or dissimilarities to of other pictures. Back to the review paper...

> *The key difficulty with contrastive learning is to pick good "negative" samples: suitable points* Y *whose energy will be pushed up. When the set of possible negative examples is not too large, we can just consider them all. This is what a softmax does, so in this case contrastive learning reduces to standard supervised or self-supervised learning over a finite discrete set of symbols. But in a real-valued high-dimensional space, there are far too many ways a vector* $\hat{Y}$ *could be different from* Y *and to improve the model we need to focus on those* Y*s that should have high energy but currently have low energy. Early methods to pick negative samples were based on Monte-Carlo methods, such as contrastive divergence for restricted Boltzmann machines and noise-contrastive estimation.*

Check out Analytics Step's post "What is a Restricted Boltzmann Machine? Gibbs Sampling and Contrastive Divergence" and KDNuggets' "A Gentle Introduction to Noise Contrastive Estimation" for more info on these two examples.

Ok now we take the principles of transformers and contrastive learning and start cooking with GANs:

> *Generative Adversarial Networks (GANs) train a generative neural net to produce contrastive samples by applying a neural network to latent samples from a known distribution (for example, a Gaussian). The generator trains itself to produce outputs $\hat{Y}$ to which the model gives low energy* $\mathrm{E}(\hat{Y})$*. The generator can do so using backpropagation to get the gradient of* $\mathrm{E}(\hat{Y})$ *with respect to* $\hat{Y}$*.* ***The generator and the model are trained simultaneously, with the model attempting to give low energy to training samples, and high energy to generated contrastive samples****.*
> *[bold for emphasis -MO]*

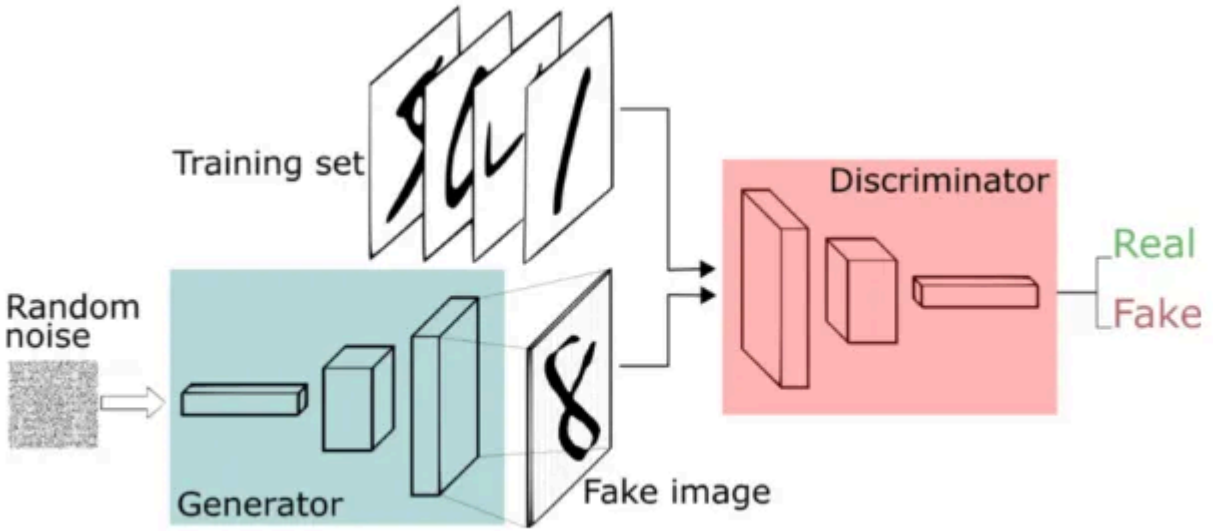
Basically the GAN and Model are to separate things that work together, one judges the output of the other with the target images. Original paper on Generative Adversarial Networks by Goodfellow et al. From Google Developer blog "Overview of GAN Structure":

> *A generative adversarial network (GAN) has two parts:*
>
> *The* ***generator*** *learns to generate plausible data. The generated instances become negative training examples for the discriminator.*
>
> *The* ***discriminator*** *learns to distinguish the generator's fake data from real data. The discriminator penalizes the generator for producing implausible results.*

> *When training begins, the generator produces obviously fake data, and the discriminator quickly learns to tell that it's fake:*

The point of a GAN is that eventually the model learns how to generate images that can fool the Discriminator.

A fantastic blog post on GANs is "An intuitive introduction to Generative Adversarial Networks (GANs)" by Thalles Silva on freecodecamp.com. The following schematic by Silva is great:



Generative Adversarial Network framework.

Along with IBM's explainer video:

> *GANs are somewhat tricky to optimize, but adversarial training ideas have proved extremely fertile, producing impressive results in image synthesis, and opening up many new applications in content creation and domain adaptation as well as domain or style transfer.*

Style transfer generates an new image by combining data or content from one image with the style of another. The difficulty lies in training a computer vision model to understand (what humans) consider a style. Having enough images of the specific style in the training set is crucial. Very thorough blog post on the topic by Fritz AI "Style Transfer Guide." How to use TensorFlow to 'neural' style transfer and paper with code.

> ***Making representations agree using contrastive learning.*** *Contrastive learning provides a way to discover good feature vectors without having to reconstruct or generate pixels. The idea is to learn a feed-forward neural network that produces very similar output vectors when given two different crops of the same image or two different views of the same object but dissimilar output vectors for crops from different images or views of different objects.*

Two things are going on here. First, the cropping (not mentioned here but also turning the image by an angle to change its orientation). This does two things, it trains the model to recognize vectors (or features) in the training set images without too much focus on location within the image. Two, it multiplies the number of images in the training data without the risk of

overfitting. Warning: this only applies for this and several other computer vision techniques, absolutely not for machine learning or data science in general.

It's interesting that the GAN methodology has one feed forward or classical neural network set up. So no cycling or feedback mechanisms involved. I think this is because the separate Discriminator model or module is tasked with giving the feedback within the GAN setup.

> *The squared distance between the two output vectors can be treated as an energy, which is pushed down for compatible pairs and pushed up for incompatible pairs.*

Ok this is a strange concept, mainly because we're using the word 'energy' here. This is very much not the typical way 'energy' is used, here is one of the authors, Yann LeCun, doing a conference session on them.

From LeCun on the NYU Center for Data Science's github:

> *Energy-based models observe a set of variables* x *and output a set of variables* y*. There are 2 major problems with feed-forward nets:*
>
> *What if the inference procedure is a more complex calculation than stacked layers of weighted sums?*

What if there are multiple possible outputs for a single input? Example: Predicting future frames of video. Essentially in a classification net, we train this net to emit a score for each class. However, this is not possible to do in a continuous high dimensional domain like images. (We cannot have softmax over images!). Even if the output is discrete, it could have a large sample space. For example, the text is compositional leading to a huge number of possible combinations. Energy-based models provide a better framework to model these modalities.

EBM approach

Instead of trying to classify x*'s to* y*'s, we would like to predict if a certain pair of* (x, y) *fit together or not. Or in other words, find a* y *compatible with* x*. We can also pose the problem as finding a* y *for which some* $F(x,y)$ *is low.*

Definition

We define an energy function $F: X \times Y \rightarrow R$ *where* $F(x,y)$ *describes the level of dependency between* (x,y) *pairs. (**Note** that this energy is used in inference, **not** in learning.) The inference is given by the following equation:*

$$\hat{y} = \underset{y}{\operatorname{argmin}}\{F(x,y)\}$$

> ***Solution: gradient-based inference***
>
> *We would like the energy function to be smooth and differentiable so that we can use it to perform the gradient-based method for inference. In order to perform inference, we search this function using gradient descent to find compatible* y*'s. There are many alternate methods to gradient methods to obtain the minimum.*

This is hard to understand but LeCun later explains why an 'energy-based' model can be better than the standard probabilistic way of doing things:

> **Question: Can you elaborate on the advantage that energy-based models give? In probability-based models, you can also have latent variables, which can be marginalized over.**
>
> *The difference is that in probabilistic models, you basically don't have the choice of the objective function you're going to minimize, and you have to stay true to the probabilistic framework in the sense that every object you manipulate has to be a normalized distribution (which you may approximate using variational methods, etc.). Here, we're saying that ultimately what you want to do with these models is make decisions. If you build a system that drives a car, and the system tells you "I need to turn left with probability 0.8 or turn right with probability 0.2", you're*

> *going to turn left. The fact that the probabilities are 0.2 and 0.8 doesn't matter — what you want is to make the best decision, because you're forced to make a decision. So probabilities are useless if you want to make decisions. If you want to combine the output of an automated system with another one (for example, a human, or some other system), and these systems haven't been trained together, but rather they have been trained separately, then what you want are calibrated scores so that you can combine the scores of the two systems so that you can make a good decision. There is only one way to calibrate scores, and that is to turn them into probabilities. All other ways are either inferior or equivalent. But if you're going to train a system end-to-end to make decisions, then whatever scoring function you use is fine, as long as it gives the best score to the best decision. Energy-based models give you way more choices in how you handle the model, way more choices of how you train it, and what objective function you use. If you insist your model be probabilistic, you have to use maximum likelihood — you basically have to train your model in such a way that the probability it gives to the data you observed is maximum. The problem is that this can only be proven to work in the case where your model is "correct" — and your model is never "correct". There's a quote from a famous statistician [Goerge Box] that says* **"All models are wrong, but some are useful."** *So probabilistic models, particularly those in high-dimensional spaces, and in combinatorial spaces such as text, are all approximate models. They're all wrong in a way, and if you try to normalize them, you make them more wrong. So you're better off not normalizing them.*

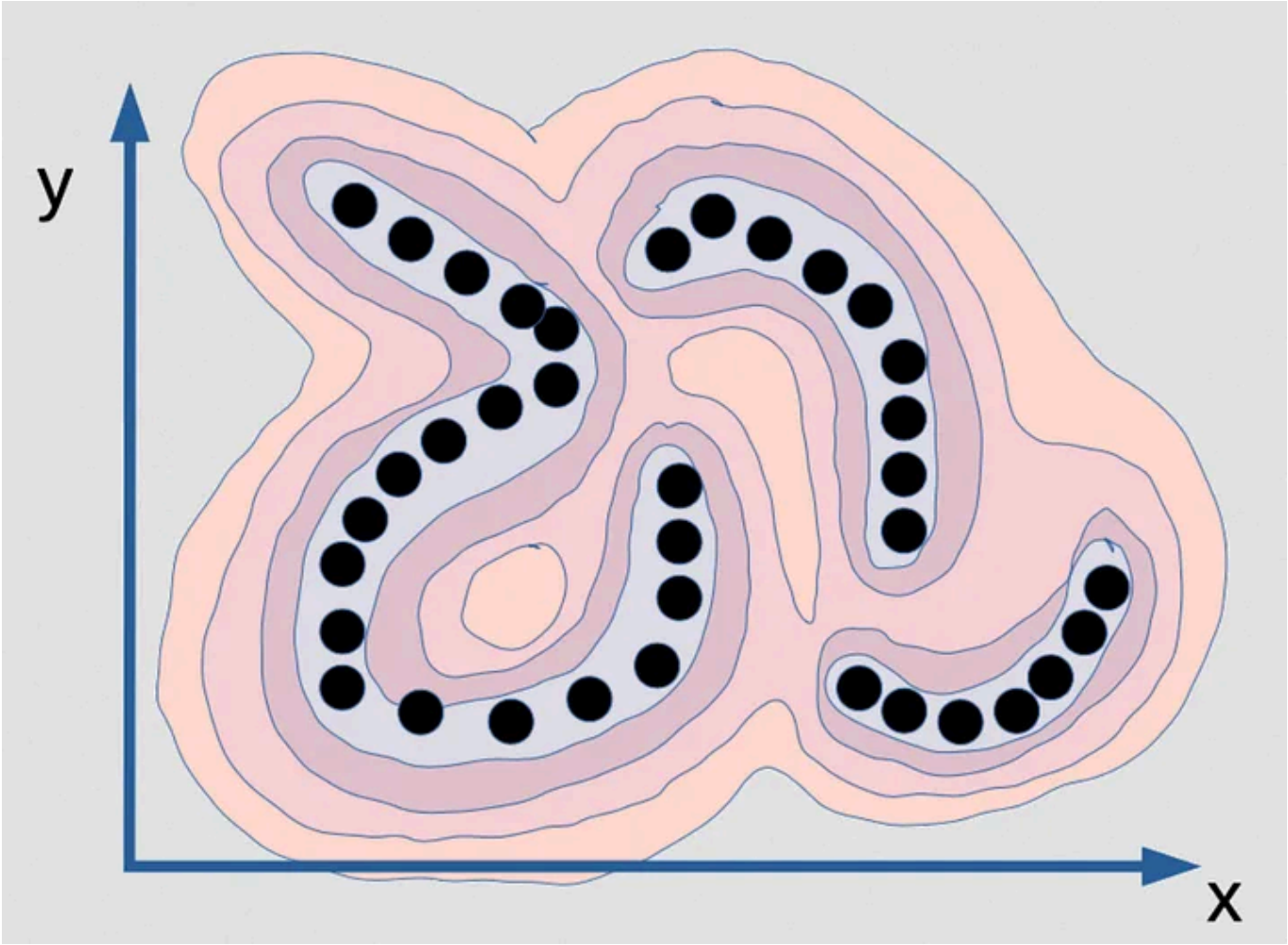


Fig. 2: Visualization of the energy function that captures dependency between x and y (LeCun, github)

> *This is an energy function that's meant to capture the dependency between* x *and* y*. It's like a mountain range if you will. The valleys are where the black dots are (these are data points), and there are mountains all around. Now, if you train a probabilistic model with this, imagine that the points are actually on an infinitely*

> *thin manifold. So the data distribution for the black dots is actually just a line, and there are three of them. They don't actually have any width. So if you train a probabilistic model on this, your density model should tell you when you are on this manifold. On this manifold, the density is infinite, and just ε outside of it should be zero. That would be the correct model of this distribution. Not only should the density be infinite, but the integral over [x and y] should be 1. This is very difficult to implement on the computer! Not only that, but it's also basically impossible. Let's say you want to compute this function through some sort of neural net — your neural net will have to have infinite weights, and they would need to be calibrated in such a way that the integral of the output of that system over the entire domain is 1. That's basically impossible. The accurate, correct probabilistic model for this particular data example is impossible. This is what maximum likelihood will want you to produce, and there's no computer in the world that can compute this. So in fact, it's not even interesting. Imagine that you had the perfect density model for this example, which is a thin plate in that (x, y) space — you couldn't do inference! If I give you a value of x, and ask you "what's the best value of y?" You wouldn't be able to find it because all values of y except a set of zero-probability have a probability of zero, and there are just a few values that are possible. For these values of x for example:*

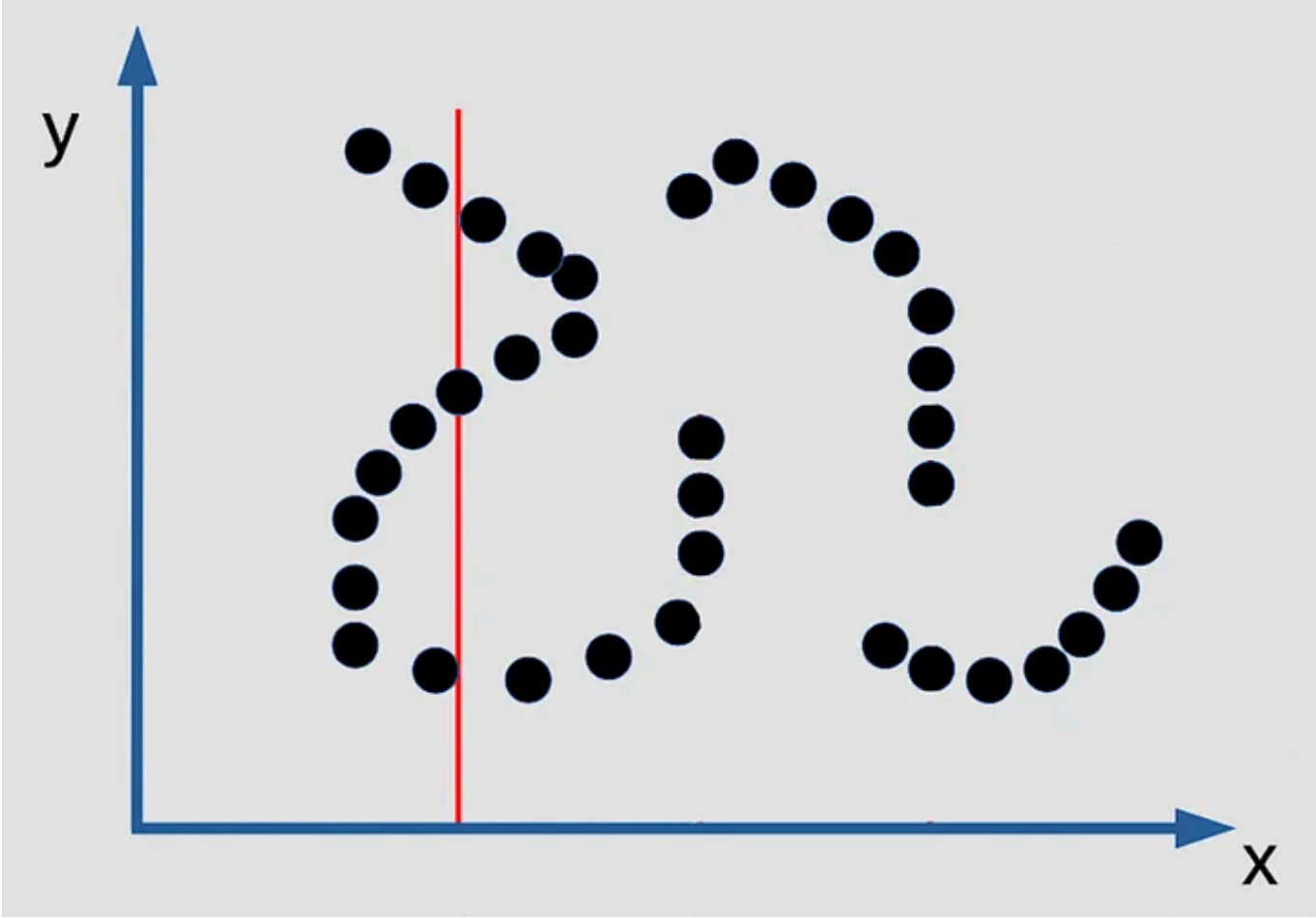


Fig. 3: Example for multiple prediction of EBM as an implicit function (LeCun, github)

> *There are 3 values of y that are possible, and they are infinitely narrow. So you wouldn't be able to find them. There's no inference algorithm that will allow you to find them. The only way you can find them is if you make your contrast function smooth and differentiable, and then you can start from any point and by gradient descent you can find a good value for y for any value of x. But this is not*

> *going to be a good probabilistic model of the distribution if the distribution is of the type I mentioned. So here is a case where insisting to have a good probabilistic model is actually bad. Maximum likelihood sucks [in this case]!*
>
> *So if you are a true Bayesian, you say "oh but you can correct this by having a strong prior where the prior says your density function has to be smooth". You could think of this as a prior. But, everything you do in Bayesian terms — take the logarithm thereof, forget about normalization — you get energy-based models. Energy-based models that have a regulariser, which is additive to your energy function, are completely equivalent to Bayesian models where the likelihood is exponential of the energy, and now you get exp(energy)exp(regulariser)exp(energy)exp(regulariser), and so it's equal to exp(energy+regulariser)exp(energy+regulariser). And if you remove the exponential you have an energy-based model with an additive regulariser.*
>
> ***So there is a correspondence between probabilistic and Bayesian methods there, but insisting that you do maximum likelihood is sometimes bad for you, particularly in high-dimensional spaces or combinatorial spaces where your probabilistic model is very wrong. It's not very wrong in discrete distributions (it's okay) but in continuous cases, it can be really wrong. And all the models are wrong.*** *[bold for emphasis- MO]*

Ok back to the review paper:

> *A series of recent papers that use convolutional nets for extracting representations that agree have produced promising results in visual feature learning. The positive pairs are composed of different versions of the same image that are distorted through cropping, scaling, rotation, color shift, blurring, and so on.*

All these preprocessing of images in the training set help make the end model more robust (especially to bad data or low quality inputs) and multiplies the size of the training set. Color shift and blurring are examples of more extreme distortions, not always recommended.

> *The negative pairs are similarly distorted versions of different images which may be cleverly picked from the dataset through a process called hard negative mining or may simply be all of the distorted versions of other images in a minibatch.*

Hard negative mining is including in the training dataset explicit examples of things you explicitly **don't want the classifier to identify** amongst the universe of things you're designing the AI to catch. Common procedure is to

generate random bounding boxes in the training images, remove the ones that overlap with actual boxes you want the model to recognize and label the remaining ones in the negative. So now the training data has *positive* labeled examples of things you want the AI to classify and bound (segment analysis) and *negative* examples of things you want it to ignore. This helps prevent false positives and improves overall accuracy.

For further reading, two papers using hard negative mining are: "Contrastive Learning with Hard Negative Samples" by Joshua Robinson et al., and "Improved Single Shot Detector with Enhanced Hard Negative Mining Approach" by Niranjan Ravi and Mohamed El-Sharkawy. Hard negative mining can be used in NLP too, check out "Hard Negative Mining in Nature Language Processing (How to Select Negative Examples in Classification and Rank Task)" by infgrad on Medium.

> *The hidden activity vector of one of the higher-level layers of the network is subsequently used as input to a linear classifier trained in a supervised manner. This Siamese net approach has yielded excellent results on standard image recognition benchmarks. Very recently, two Siamese net approaches have managed to eschew the need for contrastive samples. The first one, dubbed SwAV, quantizes the output of one network to train the other network, the second one, dubbed*

> *BYOL, smoothes the weight trajectory of one of the two networks, which is apparently enough to prevent a collapse.*

"Siamese net approach" is to input two different (but usually similar) pictures in to two instances of the same model (both instances of the model have the same exact design and weights & biases) then analyzing the similarity/dissimilarity of the outputs. The point of this procedure is increase the accuracy of classifying objects in the same class and to increase the distance or perceived differences with other classes. Great blog post on it: "Siamese Networks Introduction and Implementation" by Aditya Dutt on Medium. A good technical paper on the topic for those interested in the approach: "Siamese Neural Networks for One-shot Image Recognition" by Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov from University of Toronto.

The paper that the review authors referenced to about SwAV is "Exploring Simple Siamese Representation Learning" by Xinlei Chen and Kaiming He from Facebook AI Research. They put out a video to explain their work:

> ***Variational auto-encoders.*** *A popular recent self-supervised learning method is the Variational Auto-Encoder (VAE). This consists of an encoder network that maps the image into a latent code space and a decoder network that generates an image from a latent code. The VAE limits the information capacity of the latent code by adding Gaussian noise to the output of the encoder before it is passed to the decoder. This is akin to packing small noisy spheres into a larger sphere of minimum radius. The information capacity is limited by how many noisy spheres fit inside the containing sphere. The noisy spheres repel each other because a good reconstruction error requires a small overlap between codes that correspond to*

> *different samples. Mathematically, the system minimizes a free energy obtained through marginalization of the latent code over the noise distribution. However, minimizing this free energy with respect to the parameters is intractable, and one has to rely on variational approximation methods from statistical physics that minimize an upper bound of the free energy.*

At the moment I do not feel comfortable in putting all this into my own words so will link to people who do. The original paper cited by the authors on this topic is "Auto-Encoding Variational Bayes" by Diederik P Kingma and Max Welling. In Towards Data Science, Joseph Rocca and Baptiste Rocca wrote a very popular Medium blog post on the topic: "Understanding Variational Autoencoders (VAEs)." Jaan Altosaar also wrote a great piece on VAE for both deep learning and graph models in "Tutorial — What is a variational autoencoder?" Nice video on VAE:

The rest of the 2021 review paper had to do with the three authors' views on the future of Machine Learning and the challenges it must tackle. There isn't too much to comment on as the writing is clear. Not much technical material for me to explain either. It's very much the thoughts of these pioneers for the upcoming future, so I will separate their last portion of the paper into relevant sections and keep commentary to a minimum.

> *The Future of Deep Learning*
>
> *The performance of deep learning systems can often be dramatically improved by simply scaling them up. With a lot more data and a lot more computation, they generally work a lot better. The language model GPT-3 with 175 billion parameters (which is still tiny compared with the number of synapses in the human brain) generates noticeably better text than GPT-2 with only 1.5 billion parameters. The chatbots Meena and BlenderBot also keep improving as they get bigger. Enormous effort is now going into scaling up and it will improve existing systems a lot,* ***but there are fundamental deficiencies of current deep learning that cannot be overcome by scaling alone,*** *as discussed here. [bold for emphasis — MO]*

3 of the biggest challenges:

> *Comparing human learning abilities with current AI suggests several directions for improvement:*
>
> *Supervised learning requires too much labeled data and model-free reinforcement learning requires far too many trials. Humans seem to be able to generalize well*

> *with far less experience.*
>
> *Current systems are not as robust to changes in distribution as humans, who can quickly adapt to such changes with very few examples.*
>
> *Current deep learning is most successful at perception tasks and generally what are called system 1 tasks. Using deep learning for system 2 tasks that require a deliberate sequence of steps is an exciting area that is still in its infancy.*

> ***What needs to be improved.*** *From the early days, theoreticians of machine learning have focused on the iid assumption, which states that the test cases are expected to come from the same distribution as the training examples. Unfortunately, this is not a realistic assumption in the real world: just consider the non-stationarities due to actions of various agents changing the world, or the gradually expanding mental horizon of a learning agent which always has more to learn and discover. As a practical consequence, the performance of today's best AI systems tends to take a hit when they go from the lab to the field.*

> *Our desire to achieve greater robustness when confronted with changes in distribution (called out-of-distribution generalization) is a special case of the more general objective of reducing sample complexity (the number of examples needed to generalize well) when faced with a new task — as in transfer learning and lifelong learning— or simply with a change in distribution or in the relationship between states of the world and rewards.*
>
> *Current supervised learning systems require many more examples than humans (when having to learn a new task) and the situation is even worse for model-free reinforcement learning since each rewarded trial provides less information about the task than each labeled example. It has already been noted that humans can generalize in a way that is different and more powerful than ordinary iid generalization: we can correctly interpret novel combinations of existing concepts, even if those combinations are extremely unlikely under our training distribution, so long as they respect high-level syntactic and semantic patterns we have already learned. Recent studies help us clarify how different neural net architectures fare in terms of this systematic generalization ability. How can we design future machine learning systems with these abilities to generalize better or adapt faster out-of-distribution?*

> ***From homogeneous layers to groups of neurons that represent entities.*** *Evidence from neuroscience suggests that groups of nearby neurons (forming what is called a hyper-column) are tightly connected and might represent a kind of higher-level vector-valued unit able to send not just a scalar quantity but rather a set of coordinated values. This idea is at the heart of the capsules architectures, and it is also inherent in the use of soft-attention mechanisms, where each element in the set is associated with a vector, from which one can read a key vector and a value vector (and sometimes also a query vector).*
>
> *One way to think about these vector-level units is as representing the detection of an object along with its attributes (like pose information, in capsules). Recent papers in computer vision are exploring extensions of convolutional neural networks in which the top level of the hierarchy represents a set of candidate objects detected in the input image, and operations on these candidates is performed with transformer-like architectures.*
>
> *Neural networks that assign intrinsic frames of reference to objects and their parts and recognize objects by using the geometric relationships between parts should be far less vulnerable to directed adversarial attacks, which rely on the large difference between the information used by people and that used by neural nets to recognize objects.*

> ***Multiple time scales of adaption.*** *Most neural nets only have two timescales: the weights adapt slowly over many examples and the activities adapt rapidly changing with each new input. Adding an overlay of rapidly adapting and rapidly, decaying "fast weights" introduces interesting new computational abilities.*
>
> *In particular, it creates a high-capacity, short-term memory, which allows a neural net to perform true recursion in which the same neurons can be reused in a recursive call because their activity vector in the higher-level call can be reconstructed later using the information in the fast weights. Multiple time scales of adaption also arise in learning to learn, or meta-learning.*

> ***Higher-level cognition.*** *When thinking about a new challenge, such as driving in a city with unusual traffic rules, or even imagining driving a vehicle on the moon, we can take advantage of pieces of knowledge and generic skills we have already mastered and recombine them dynamically in new ways. This form of systematic*

> *generalization allows humans to generalize fairly well in contexts that are very unlikely under their training distribution. We can then further improve with practice, fine-tuning and compiling these new skills so they do not need conscious attention anymore. How could we endow neural networks with the ability to adapt quickly to new settings by mostly reusing already known pieces of knowledge, thus avoiding interference with known skills? Initial steps in that direction include Transformers and Recurrent Independent Mechanisms.*

> *It seems that our implicit (system 1) processing abilities allow us to guess potentially good or dangerous futures, when planning or reasoning. This raises the question of how system 1 networks could guide search and planning at the higher (system 2) level, maybe in the spirit of the value functions which guide Monte-Carlo tree search for AlphaGo.*
>
> *Machine learning research relies on inductive biases or priors in order to encourage learning in directions which are compatible with some assumptions about the world. The nature of system 2 processing and cognitive neuroscience theories for them suggests several such inductive biases and architectures, which may be exploited to design novel deep learning systems. How do we design deep*

> *learning architectures and training frameworks which incorporate such inductive biases?*
>
> *The ability of young children to perform causal discovery suggests this may be a basic property of the human brain, and recent work suggests that optimizing out-of-distribution generalization under interventional changes can be used to train neural networks to discover causal dependencies or causal variables. How should we structure and train neural nets so they can capture these underlying causal properties of the world?*

> *How are the directions suggested by these open questions related to the symbolic AI research program from the 20th century? Clearly, this symbolic AI program aimed at achieving system 2 abilities, such as reasoning, being able to factorize knowledge into pieces which can easily recombined in a sequence of computational steps, and being able to manipulate abstract variables, types, and instances. We would like to design neural networks which can do all these things while working with real-valued vectors so as to preserve the strengths of deep learning which include efficient large-scale learning using differentiable computation and gradient-based adaptation, grounding of high-level concepts in low-level*

> *perception and action, handling uncertain data, and using distributed representations.*

Thats it, that's the end of the two review papers by three of the greatest pioneers in Machine Learning. For those who want to know more about the authors themselves, the final section here will describe their careers quickly and link to interviews/presentations they've done.

Geoffrey Hinton

Geoffrey Hinton is an Emeritus Distinguished Professor at the University of Toronto and was a Google Brain researcher

Best known for artificial neural networks (ANNs). Often called the "Godfather of Deep Learning." His research on the back propagation

algorithm significantly improved of deep learning model performance. Other research works are Boltzmann machines and Capsule neural networks.

edit: most recently Hinton quit Google and wrote about how concerned he is about the direction LLM Ai is headed:

'The Godfather of A.I.' Leaves Google and Warns of Danger Ahead

For half a century, Geoffrey Hinton nurtured the technology at the heart of chatbots like ChatGPT. Now he worries it...

www.nytimes.com

Yoshua Bengio

Yoshua Bengio is a professor at the Department of Computer Science and Operations Research at the Université de Montréal. He is also the co-founder of Element AI

Yoshua is known for his work on artificial neural networks and deep learning in the 1980s and 1990s. He co-founded the ICLR conference with

Yann LeCun. He is one of the most-cited computer scientists in the areas of Deep Learning, RNN, probabilistic learning, and NLP.

Yann LeCun

Currently the Chief AI Scientist and VP at Facebook. Founding father of convolutional nets. Made convolutional neural networks work with backpropagation. Titan.

Thanks for reading! Hope this helped.

Machine Learning | Artificial Intelligence | AI | Study Guide

M

Written by Mohammed R. Osman

134 Followers · 169 Following

Climate & Data Science, Sports, Totally Mature Adult

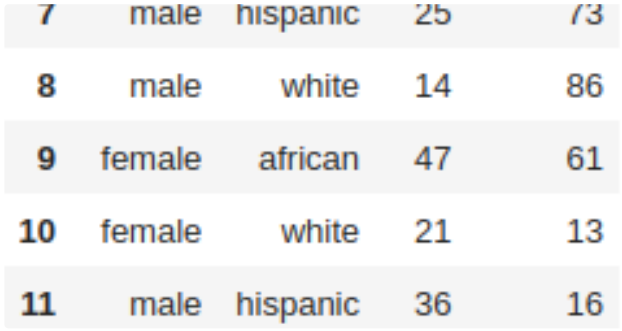
Follow

No responses yet

Marco Roccetti

What are your thoughts?

More from Mohammed R. Osman



7	male	hispanic	25	73
8	male	white	14	86
9	female	african	47	61
10	female	white	21	13
11	male	hispanic	36	16

Mohammed R. Osman

Run NVIDIA's RAPIDS on a eGPU: How to Do Advanced ML/DS on a...

Access cutting edge computing with a GPU on a budget

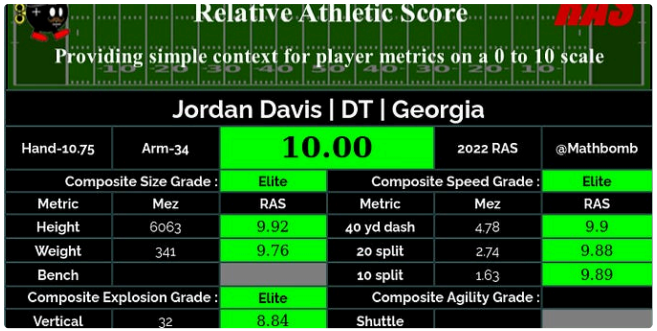
In RoundtableML by Mohammed R. Osman

Quickie: Avoid the 'Dummy Variable Trap'

'Dummy Variable Trap' is when you use 'one-hot' encoding to handle a categorical variabl...

Apr 11, 2022 151

Jun 8, 2023 9



Mohammed R. Osman

The Elite Athletes of the 2022 NFL Draft Class: A Data Exploration

I'm a big fan of the NFL and love the draft. These next few pieces will be for fans of the...

Apr 28, 2022 284

Mohammed R. Osman

Killers Outta the Gate '22: Reviewing Offensive Linemen's...

Simplified look into Offensive Linemen picked in 2022 NFL Draft to see who popped...

Apr 11, 2023

See all from Mohammed R. Osman

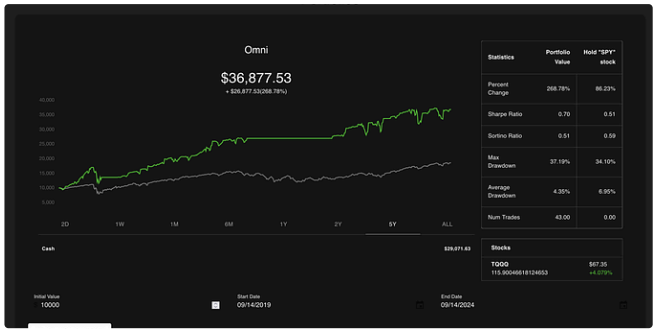
Recommended from Medium

Jessica Stillman

Jeff Bezos Says the 1-Hour Rule Makes Him Smarter. New...

Jeff Bezos's morning routine has long included the one-hour rule. New...

Oct 30, 2024 24K 714



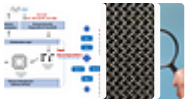
In DataDrivenInvestor by Austin Starks

I used OpenAI's o1 model to develop a trading strategy. It is...

It literally took one try. I was shocked.

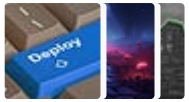
Sep 15, 2024 9K 237

Lists



Natural Language Processing

1967 stories · 1613 saves



Predictive Modeling w/ Python

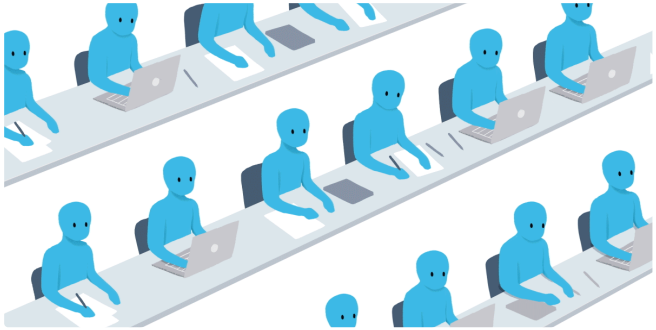
20 stories · 1848 saves

AI Regulation

6 stories · 704 saves

Generative AI Recommended Reading

52 stories · 1679 saves



In Human Parts by Devon Price

Laziness Does Not Exist

Psychological research is clear: when people procrastinate, there's usually a good reason

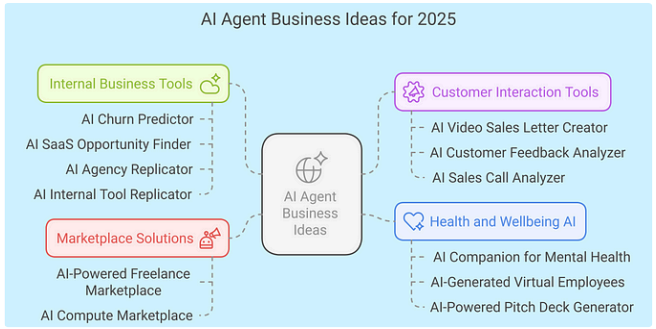
Mar 23, 2018 342K 2178

Alexander Nguyen

I Wrote On LinkedIn for 100 Days. Now I Never Worry About Finding ...

Everyone is hiring.

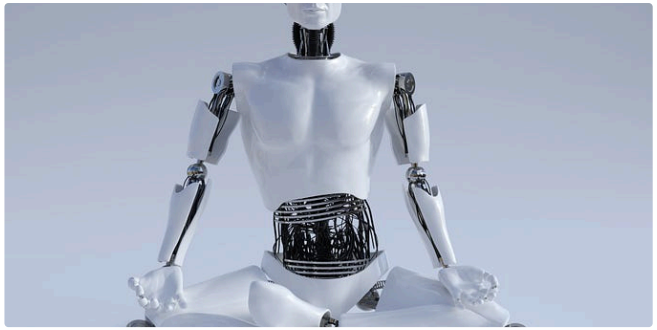
Sep 21, 2024 44K 943



In Everyday AI by Manpreet Singh

15 AI Agent Business Ideas to Get Rich in 2025

Feb 6 1.4K 45



In Artificial Corner by The PyCoach

You're Using ChatGPT Wrong! Here's How to Be Ahead of 99% of...

Master ChatGPT by learning prompt engineering.

Mar 17, 2023 39K 842

See more recommendations