## Hello - my name is Ernesto Lee. In this episode, we are going to delve into deep neural networks and how they work. If you have done the lightest google search ever, you will know that Deep Learning is inspired by the brain. Let’s now look at Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) and some Real-world applications.

With Machine Learning, we were deeply concerned with learning a function that we called f and having it map input data called X to an output prediction called Y with a minimal loss on the test data. With deep learning, we are looking to do the exact same thing but with a different and more powerful method. Let’s go back to first principles… In supervised learning, we know the meta form that we are looking at is:

**Y = *f*(X) + ϵ**

**We use Training data so that** machine learns *f* from labeled training data

**We use Testing data** so that the machine has an opportunity to predict Y’s from unlabeled testing data.

That doesn’t change and that is still the problem that we are trying to solve.

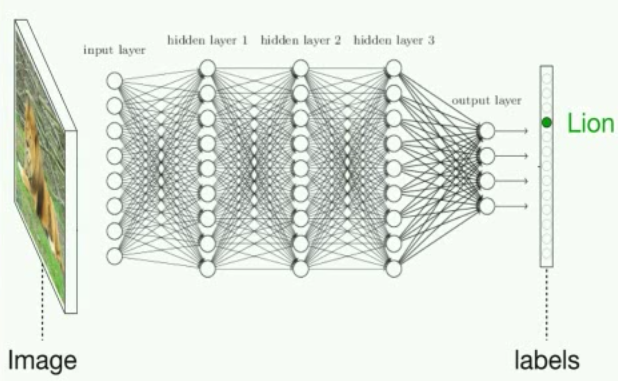
In the real world, nothing is easy. By this, I mean that most use cases will not involve a simple straight line set of data. Most of the time, the function f that we are trying to discover is extremely complex. Take NLP or natural language problems for instance. There are massive vocabulary sizes and each entry is a different feature. This means that we have a ridiculous amount of dimensions and we’ve already talked about how that can be a problem in classic machine learning. Look at vision problems. Computer Vision is all about pulling data by investigating pixels. Each pixel can be a feature. Consider when you play a board game… there are many decisions that must be made that are based complicated scenarios with each scenario leading to many possible futures. Machine Learning works excellent when the data lines up nicely, but they often have difficulty generalizing when the data and the dimensions are complex.

You can see what I am leading up to. Deep learning is really good at learning a function *f from a data set*, even when the data is complex. Especially when the data is complex. So much so that, artificial neural networks are considered to be **universal function approximators** because they are able to [learn any function](http://neuralnetworksanddeeplearning.com/chap4.html) from the data, regardless of how complicated, with just a single **hidden layer.**

**I had the pleasure of going to NYU and teaching AI, ML and DL once. I heard all of the students rave about this guy named Yan LeCun. So I did what anybody in my position would do… I looked him up on google and found out that he is the GodFather of AI because he is known as the Father of Convolutional Neural Networks. We are going to be discussing a lot of his work in this episode.**

Let us start by looking at the problem of image classification. Say I want you to take an image as an input, and output a class (e.g., dog, cat, car).

Graphically, a deep neural network solving image classification looks something like this:



In reality, this is all just a huge mathematical equation with millions of terms and a ton of parameters. The input X is, say, a greyscale image represented by a *w*-by-*h* matrix of pixel brightnesses. The output Y is a vector of class probabilities. This means we have as an output the probability of each class being the correct label. If this neural net is working well, the highest probability should be for the correct class. And the layers in the middle are just doing a bunch of matrix multiplication by summing **activations** x **weights** with non-linear transformations (**activation functions**) after every hidden layer to enable the network to learn a **non-linear function.**

Here is the amazing part, you can use **gradient descent** in the *exact same way* that we did with linear regression to train these parameters in a way that minimizes loss. So with a lot of examples and a lot of gradient descent, the model can learn how to classify images of animals correctly. And that ladies and gentlemen, is “deep learning”. That was the 30,000 foot overview - we’re about to get down to brass tacks.

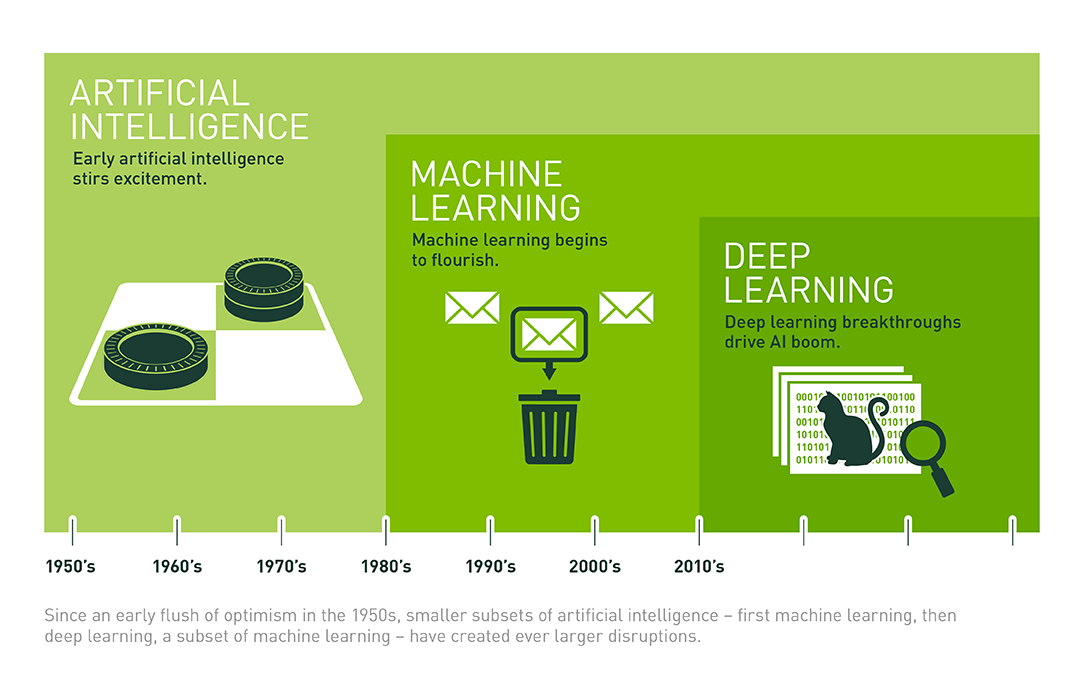
# **Let’s look at comes from and where deep learning does well**

Artificial neural networks have actually been around for quite a long time. In the 40’s, 50’s, and 60’s it was called **cybernetics. In the 80’s and 90’s it was called connectionism**. And finally, when neural networks started going “deeper” - the name was changed to **deep learning**. Here is the thing though, we are still only at the surface of what deep learning can do.

The Director of AI at Tesla, Andrej Karpathy believes that there are generally *“four separate factors that hold back AI:*

1. *Compute (the obvious one: Moore’s Law, GPUs, ASICs),*
2. *Data (in a nice form, not just out there somewhere on the internet — e.g. ImageNet),*
3. *Algorithms (research and ideas, e.g. backprop, CNN, LSTM), and*
4. *Infrastructure (software under you — Linux, TCP/IP, Git, ROS, PR2, AWS, AMT, TensorFlow, etc.)”*

Recently (and I am defining recently as within the last 10 years), the full potential of deep learning is beginning to manifest because of cloud advances in Compute and Data. These advances have led to increases in Algorithms, and Infrastructure. These advances lead to more advances in Compute and data… You get the point… there is the snake eating the tail… we have a continuous cycle.



In the rest of this episode, I’ll give you some relevant insights from biology and statistics to explain what happens inside of a neural network, and then talk through some amazing applications of deep learning. Finally, I’ll give you the tools begin creating some models so you can apply deep learning yourself. I will admit that it is not completely intuitive but once you get it you will be able to quickly achieve greater-than-human-level performance on a wide range of problems.

# **\*\*So what happens inside neural nets?**

## **Let’s discuss Neurons, feature learning, and layers of abstraction and talk about how they apply to Deep Learning.**

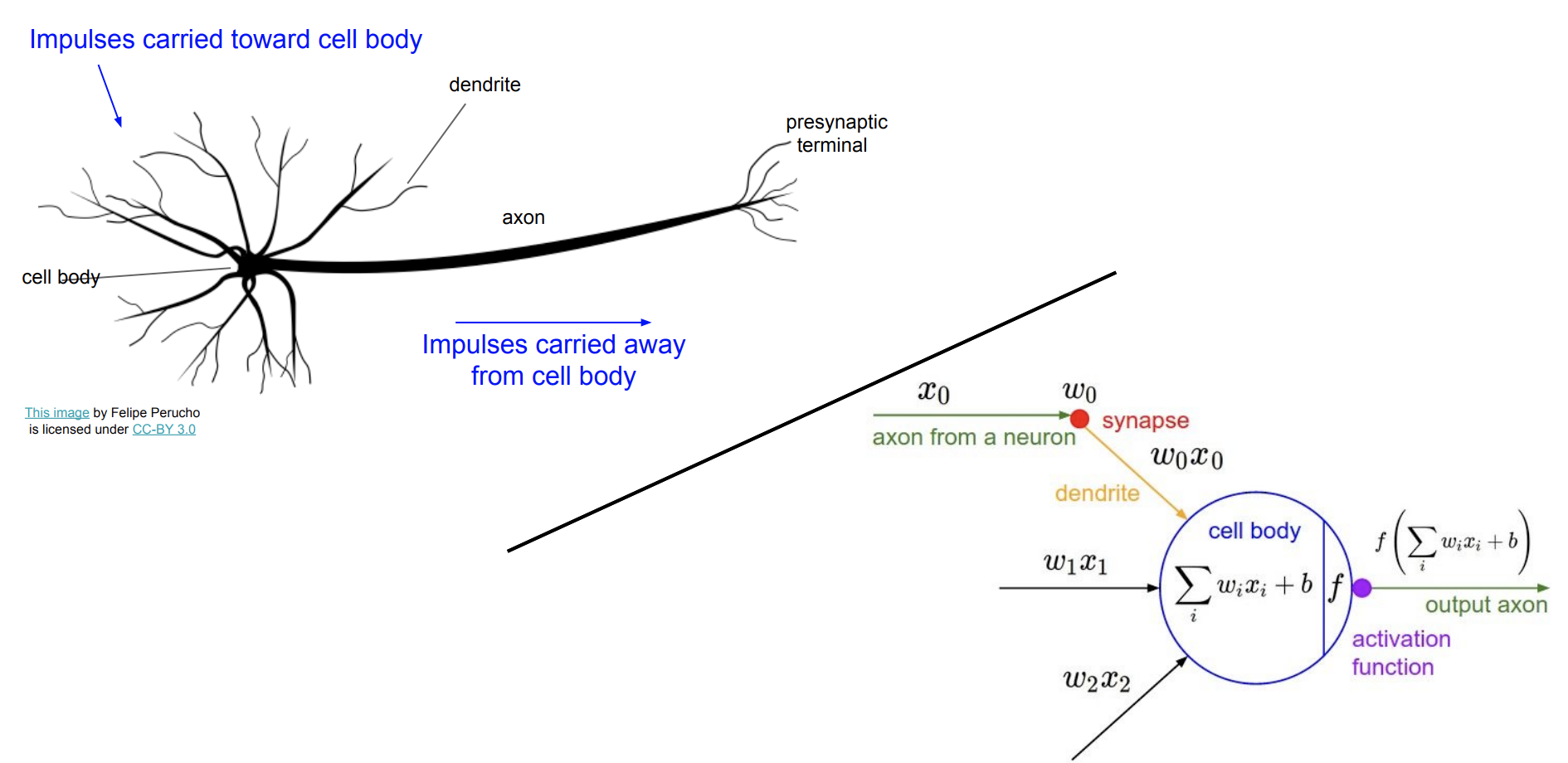
As you hear my words you are not examining every syllable of every word that I say, or looking at every pixel on the screen that makes an image that you assume to be me. What you are doing is abstracting away from the details and grouping things into higher-level concepts: words, phrases, sentences, paragraphs. Or pixels to images, to a person… me!

Look at these words on your screen:

Yuor abiilty to exaimne hgiher-lveel fteaures is waht aollws yuo to unedrtsand waht is hpapening in tihs snetecne wthiout too mcuh troulbe (or myabe yuo sned too mnay dnruk txets).

The same thing happens in vision, not just in humans but in most animals’ visual systems.

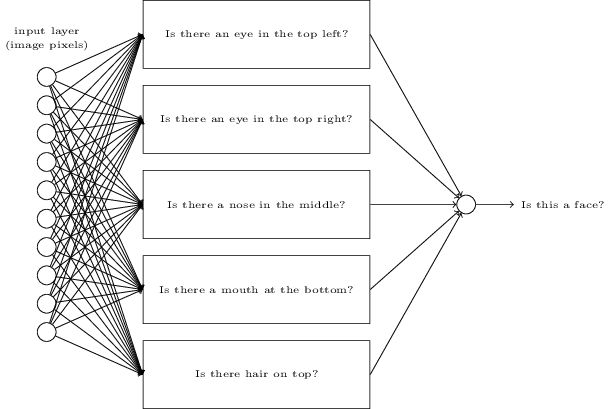
Our Brains are made up of neurons and these neurons are “fired” by emitting electrical signals to other neurons after being sufficiently “activated”. Remember those words… neurons… fired… and activated. If I put my hand in or near a fire.. The stimuli of the heat on my hand will serve as an input. This will give us a signal that there is heat and you can think of the intensity of the heat as a “weight” or a multiplier of the heat. How much heat do we need to feel before we are triggered to move our hand? This is what the activation function does. We will go back to these concepts later later. These neurons are malleable in terms of how much a signal from other neurons will add to the activation level of the neuron. The activation level (vaguely speaking, the *weights* connecting neurons to each other end up being *trained* to make the neural connections more useful, just like the parameters in a linear regression can be trained to improve the mapping from input to output).



Here is a Side-by-side illustrations of biological and artificial neurons, provided thanks to S[tanford’s CS231n](http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture4.pdf) class. This analogy should not be taken too literally — biological neurons can do things that artificial neurons can’t, and vice versa — but it’s useful to understand the biological inspiration. See Wikipedia’s description of [biological vs. artificial neurons](https://en.wikipedia.org/wiki/Artificial_neuron#Biological_models) for more detail.

Our biological networks are arranged in a hierarchical manner, so that certain neurons end up detecting not extremely *specific* features of the world around us, but rather more *abstract* features, i.e. patterns or groupings of more low-level features. For example, the [fusiform face area](https://en.wikipedia.org/wiki/Fusiform_face_area) in the human visual system is specialized for facial recognition.





The Illustration on your screen… the top one… shows how CNNs learning via increasingly abstract features. The diagram below - of how an artificial neural network takes raw pixel inputs, develops intermediate “neurons” to detect higher-level features (e.g. presence of a nose), and combines the outputs of these to create a final output. Illustration from Neural Networks and Deep Learning ([Nielsen, 2017](http://neuralnetworksanddeeplearning.com/chap1.html)).

This hierarchical structure exhibited by biological neural networks was discovered in the 1950s when researchers David Hubel and Torsten Wiesel were studying neurons in the visual cortex of cats. They were unable to observe neural activation after exposing the cat to a variety of stimuli: dark spots, light spots, hand-waving, and even pictures of women in magazines. But in their frustration, as they removed a slide from the projector at a diagonal angle, they noticed some neural activity! It turned out that diagonal edges at a very particular angle were causing certain neurons to be activated.

Background via [Knowing Neurons](http://knowingneurons.com/2014/10/29/hubel-and-wiesel-the-neural-basis-of-visual-perception/)

This makes sense evolutionarily since natural environments are generally noisy and random (imagine a grassy plain or a rocky terrain). So when a feline in the wild perceives an “edge”, i.e. a line that contrasts from its background, this might indicate that an object or creature is in the visual field. When a certain combination of edge neurons are activated, those activations will combine to yield a yet more abstract activation, and so on, until the final abstraction is a useful concept, like “bird” or “wolf”.

The idea behind a deep neural network is to mimic a similar structure with layers of artificial neurons.

## **Why linear models don’t work**

To draw from Stanford’s excellent deep learning course, [CS231n: Convolutional Neural Networks and Visual Recognition](http://cs231n.stanford.edu/syllabus.html), imagine that we want to train a neural network to classify images with the correct one of the following labels: ["plane", "car", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"].

One approach could be to construct a “template”, or average image, of each class of image using the training examples, and then use a nearest-neighbors algorithm during testing to measure the distance of each unclassified image’s pixel values, in aggregate, to each template. This approach involves no layers of abstraction. It’s a linear model that combines all the different orientations of each type of image into one averaged blur.

For instance, it would take all the cars — regardless of whether they’re facing left, right, center, and regardless of their color — and average them. The template then ends up looking rather vague and blurry.



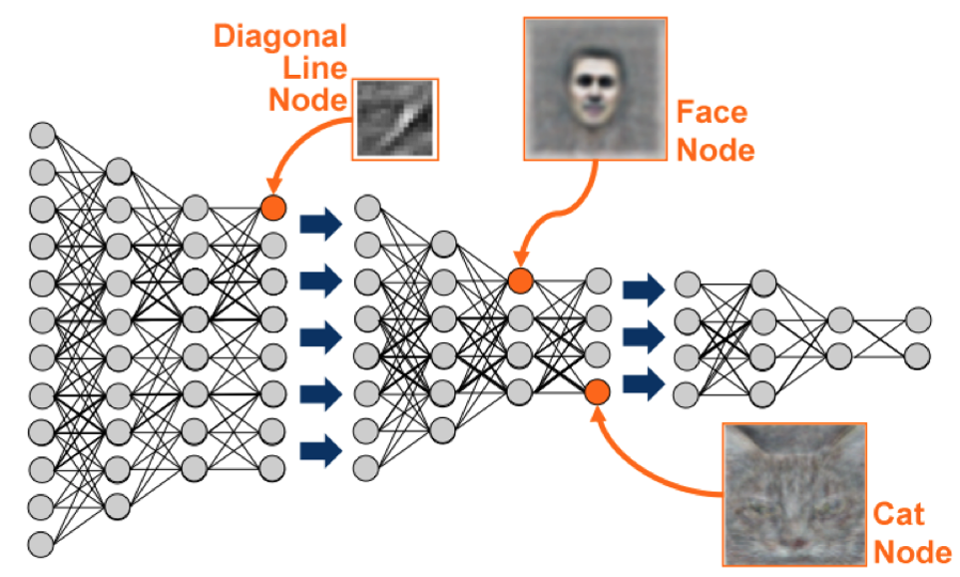
Example drawn from Stanford’s [CS231n: Convolutional Neural Networks and Visual Recognition](http://cs231n.stanford.edu/syllabus.html), Lecture 2.

Notice that the horse template above appears to have two heads. This doesn’t really help us: we want to be able to detect right-facing horse *or* a left-facing horse separately, and then if *either one* of those features is detected, *then* we want to say we’re looking at a horse. This flexibility is provided by deep neural nets, as we will see in the next section.

**Deep neural networks approach the image classification problem using layers of abstraction**

To repeat what we explained earlier in this section: the input layer will take raw pixel brightnesses of an image. The final layer will be an output vector of **class probabilities** (i.e. the probability of the image being a “cat”, “car”, “horse”, etc.)

But instead of learning a simple linear modelrelating input to output, we’ll instead construct intermediate **hidden layers** of the network will learn increasingly abstract features, which enables us to not lose all the nuance in the complex data.



Source: [Analytics Vidhya](https://www.analyticsvidhya.com/blog/2017/04/comparison-between-deep-learning-machine-learning/)

Just as we described animal brains detecting abstract features, the artificial neurons in the hidden layers will learn to detect abstract concepts — whichever concepts are ultimately most useful for capturing the most information and minimizing loss in the accuracy of the network’s output (this is an instance of unsupervised learning happening within the network).

This comes at the cost of model interpretability, since as you add in more hidden layers the neurons start representing more and more abstract and ultimately unintelligible features — to the point that you may hear deep learning referred to as “black box optimization”, where you basically are just trying stuff somewhat at random and seeing what comes out, without *really* understanding what’s happening inside.

Linear regression is interpretable because you *decided* which features to include in the model. Deep neural networks are harder to interpret because the features are learned and aren’t explained anywhere in English. It’s all in the machine’s imagination.

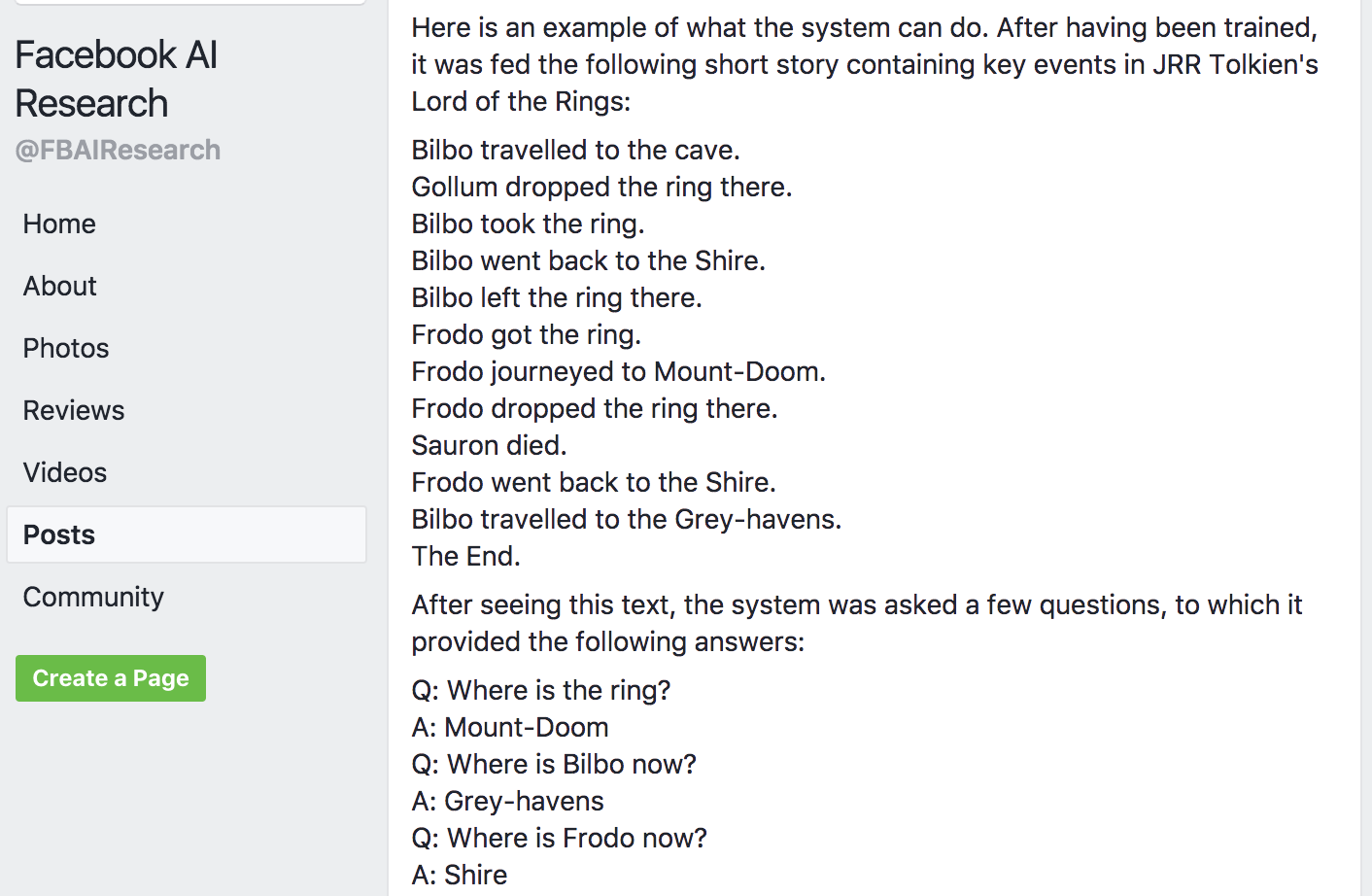
## **Some extensions and further concepts worth noting**

* **Deep learning software packages.** You’ll rarely need to implement all the parts of neural networks from scratch because of existing libraries and tools that make deep learning implementations easier. There are many of these: TensorFlow, Caffe, Torch, Theano, and more.
* **Convolutional neural networks (CNNs).** CNNs are designed specifically for taking images as input, and are effective for computer vision tasks. They are also instrumental in deep reinforcement learning. CNNs are specifically inspired by the way animal visual cortices work, and they’re the focus of the deep learning course we’ve been referencing throughout this article, Stanford’s CS231n.
* **Recurrent neural networks (RNNs).** RNNs have a sense of built-in memory and are well-suited for language problems. They’re also important in reinforcement learning since they enable the agent to keep track of where things are and what happened historically even when those elements aren’t all visible at once. Christopher Olah wrote an [excellent walkthrough](http://colah.github.io/posts/2015-08-Understanding-LSTMs/) of RNNs and LSTMs in the context of language problems.
* **Deep reinforcement learning.** This is one of the most exciting areas of deep learning research, at the heart of recent achievements like OpenAI defeating professional Dota 2 players and DeepMind’s AlphaGo surpassing humans in the game of Go. We’ll dive deeper in [Part 5](https://medium.com/@v_maini/machine-learning-for-humans-part-5-reinforcement-learning-6eacf258b265), but essentially the goal is to apply all of the techniques in this post to the problem of teaching an agent to maximize reward. This can be applied in any context that can be gamified — from actual games like Counter Strike or Pacman, to self-driving cars, to trading stocks, to (ultimately) real life and the real world.

## **Deep learning applications**

Deep learning is reshaping the world in virtually every domain. Here are a few examples of the incredible things that deep learning can do…

* Facebook trained a neural network augmented by short-term memory to intelligently answer questions about the plot of Lord of the Rings.



Research from [FAIR](https://www.facebook.com/FBAIResearch/posts/362517620591864) (Facebook AI Research) applying deep neural networks augmented by separate short-term memory to intelligently answer questions about the LOTR storyline. This is the definition of epic.

* Self-driving cars rely on deep learning for visual tasks like understanding road signs, detecting lanes, and recognizing obstacles.



Source: [Business Insider](http://www.businessinsider.com/uber-driverless-car-in-pittsburgh-review-photos-2016-9/#to-try-the-cars-we-lined-up-at-ubers-advanced-technologies-center-in-the-strip-district-of-pittsburgh-a-small-neighborhood-on-the-allegheny-river-with-nearby-warehouses-the-atc-is-tucked-under-an-overpass-for-a-freight-train-keeping-it-secluded-1)

* Deep learning can be used for fun stuff like art generation. A tool called [neural style](https://github.com/jcjohnson/neural-style) can impressively mimic an artist’s style and use it to remix another image.



The style of Van Gogh’s [Starry Night](https://en.wikipedia.org/wiki/The_Starry_Night) applied to a picture of Stanford’s campus, via Justin Johnson’s neural style implementation: <https://github.com/jcjohnson/neural-style>

Other noteworthy examples include:

* Predicting molecule bioactivity for [drug discovery](https://arxiv.org/abs/1510.02855)
* Face and object recognition for photo and video tagging
* Powering Google search results
* Natural language understanding and generation, e.g. [Google Translate](https://translate.google.com/)
* The Mars explorer robot Curiosity is [autonomously selecting inspection-worthy soil targets](https://www.theatlantic.com/technology/archive/2017/06/mars-curiosity-rover/531339/) based on visual examination

…and many, many, more.

# **Now go do it!**

We haven’t gone into as much detail here on how neural networks are set up in practice because it’s much easier to understand the details by implementing them yourself. Here are some amazing hands-on resources for getting started.

* Play around with the architecture of neural networks to see how different configurations affect network performance with the Google’s [Neural Network Playground](http://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=4,2&seed=0.70277&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false).
* Get up-and-running quickly with this tutorial by Google: [TensorFlow and deep learning, without a PhD](https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/). Classify handwritten digits at >99% accuracy, get familiar with TensorFlow, and learn deep learning concepts within 3 hours.
* Then, work through at least the first few lectures of [Stanford’s CS231n](http://cs231n.stanford.edu/syllabus.html) and the first assignment of building a two-layer neural network from scratch to really solidify the concepts covered in this article.

# **Further resources**

*Deep learning is an expansive subject area. Accordingly, we’ve also compiled some of the best resources we’ve encountered on the topic, in case you’d like to go… deeper.*

* [*Deeplearning.ai*](http://deeplearning.ai/)*, Andrew Ng’s new deep learning course with a comprehensive syllabus on the subject*
* [*CS231n: Convolutional Neural Networks for Visual Recognition*](http://cs231n.stanford.edu/syllabus.html)*, Stanford’s deep learning course. One of the best treatments we’ve seen, with excellent lectures and illustrative problem sets*
* [*Deep Learning & Neural Networks*](http://neuralnetworksanddeeplearning.com/chap1.html) *— accessible but rigorous*
* [*Deep Learning Book*](http://www.deeplearningbook.org/) *— foundational, more mathematical*
* [*Fast.ai*](http://fast.ai/) *— less theoretical, much more applied and black-boxy*
* *See Greg Brockman (CTO of OpenAI)’s answer to the question “What are the best ways to pick up Deep Learning skills as an engineer?”* [*on Quora*](https://www.quora.com/What-are-the-best-ways-to-pick-up-Deep-Learning-skills-as-an-engineer/answer/Greg-Brockman?srid=2sq8)

# **Next up: time to play some games!**

Last, but most certainly not least, is [Part 5: Reinforcement Learning](https://medium.com/@v_maini/reinforcement-learning-6eacf258b265).