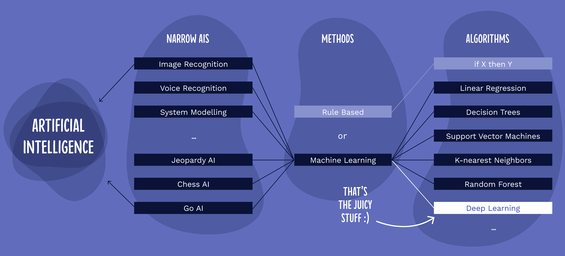
The Simple Guide to Deep Learning

The Components of a Deep Learning System and What All These F*cking Nerdy Buzzwords Actually Mean.

By Iskender Dirik

Artificial Intelligence, Machine Learning, Deep Learning, Cats in Videos. It's kind of overwhelming, like dinner with your parents. Let's draw a map.



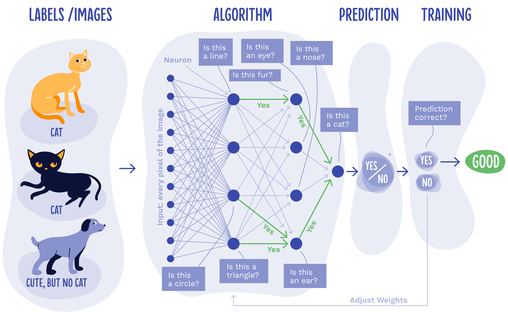
What is it that we call AI (or I, for that matter)? It's just a sum of "narrow AIs" that can solve small subsets of problems. Like playing Jeopardy, Go or recognizing animals. Remember the guy in school next to you? Really good in solving rubik's cubes, not so much in soccer. To solve these narrow sets of problems, we historically used rule based methods - if the other chess player has a tower less, try to make him lose more figures. Now we use machine learning algorithms. Among them is Deep Learning, which is super powerful with problems like Image Recognition and Natural Language Processing.

First things first: **The Basic Lesson about Deep Learning**

Deep Learning is based on an artificial neural network (ANN) with more than two layers. One of the earliest "deep" neural networks has three hidden layers (paper from Hinton and his pals in 2006). "Very Deep" networks (like VGG, the ImageNet winner in 2014) consist of 16+ hidden layers. An ANN is a biologically inspired computer system modeled on the human brain and nervous system.

A neural network consists of different abstraction layers. Each layer consists of nodes that ask different questions. The earlier the layer, the more abstract and simple the questions/tasks. The output of a node tells us how certain the node is about the question. This information is the input to the next layer – a system inspired by our human brain.

The first layer nodes take in all the input data, like all pixels of images when it comes to image classification tasks. The first abstraction layer tries to find concepts like shapes – triangles, circles, lines. The next layer asks questions like, is this triangle an ear? The last layer is called the output layer, takes all this information and finally asks, Is this furry thing with two spiky ears a cat?



*Hello, lovely know-it-all! For the AI experts amongst you:

1. This Whitepaper is not meant for you, mate!
2. Yes, I'm fully aware that neural networks actually never deliver an explicit yes or no as answer; they rather predict that it's 99% a cat and 1% no cat. This is a simplified model. Over and out!
3. Yes, I know that the description above applies especially for Convolutional Neural Networks (CNN). But the reader is still not aware about the existence of CNNs - we'll change that later. Okay?
4. Yes, I know that the first layers of CNNs are all about edge detection and less about detecting closed forms like triangles, circles etc. But this is not a research lab - I try to make things as striking and tangible as possible for the non-expert readers. Geddit?

So far, so good. You're ready to build your first deep learning system (You are! Be bold!). Let's start with the process. Calm down, this won't hurt.

1 Input Data

First of all we need input data. The accuracy of a deep learning system completely depends on the input data. Shit in, shit out – you know. So: Collect data. As much as possible. From external and/or internal sources.

These images are our input data. We're crazy about cats, so let's build a system that detects cats! Cat recognition - what a creative AI and Deep Learning example, right?

2 Labelling Data

For most of the deep learning tasks we need to label the data. Manually. Yes, this is f*cking hard work dude. Or use publicly available labeled data sources (there are huge libraries for pictures, sounds etc.).
However, everyone has access to it, so it won‘t give you a real competitive advantage, lazy ass.

3 Learning Criteria

And now – the fun part: Engineering the system.
Let's start with: What is the goal of the deep learning system, what is our base method?

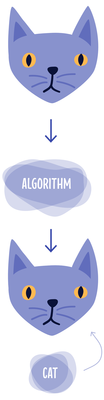
A. SUPERVISED LEARNING

DESCRIPTION
In a supervised learning task, we have a bunch of data with known labels and are trying to find a function that predicts these labels from the input. It's all about learning an input/output mapping, using one simple, beautiful algorithm called backpropagation.

WHEN TO USE
Classifying/predicting something from input data.

TYPICAL USE-CASES
- Cat detection in images
- Cancer detection in Xray images
- Forecasting

EXAMPLE
Question Is this a cat?
Input Image
Output Yes/No



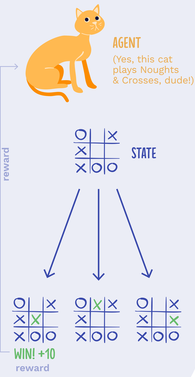
B. REINFORCEMENT LEARNING

DESCRIPTION
In a reinforcement problem, we are given a situation and can choose different actions that have an unknown future feedback.

WHEN TO USE
Striving to maximize a future reward.

TYPICAL USE-CASES
- Teaching a robot to lift a box
- Teaching a helicopter to fly stable
- Play Chess, Go or Noughts & Crosses

EXAMPLE
Question Which action maximizes future reward to the agent given the current state?
Input State
Output Action



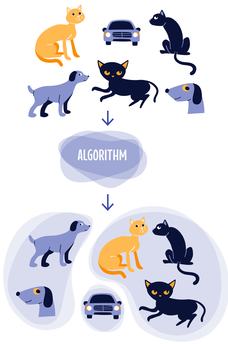
C. UNSUPERVISED LEARNING

DESCRIPTION
In an unsupervised learning task, we just have data without labels.

WHEN TO USE
Eliciting hidden/inherent structures of a problem.

TYPICAL USE-CASES
- Grouping cat and dog images together
- Anomaly detection
- Finding reationships between words in a language

EXAMPLE
Question Which objects are similar?
Input Images
Output Clusters of similar images



Yes, yes, smartass. Unsupervised learning is just useful if you didn't do step 2. But do you want to annotate 3 million cats yourself?

Unsupervised Learning is still commercially irrelevant. Yes, it worked for Google to find cats, but it required 16,000 computer processors with one billion connections and they had to browse all of YouTube. Supervised learning is what works in deep learning commercially, unsupervised doesn't (yet).

Excursion: The Future of Deep Learning

Present: We are mainly leveraging supervised learning today, which is essentially behavioral cloning. Humans label images, videos, text or speech and the neural network tries to match up with what human do.

Future: Over time, as neural networks get "more capable" (aka pretty darn good at supervised learning/behavioral cloning), companies will start to add some reinforcement learning where you give the neural network actual objectives – not just matching the human behind the curtains but giving objectives of achievement, like:
- E-Commerce: model learns customer behaviors and tailors service to suit customer interests.
- Finance: model learns market behavior and generates trading strategies.
- Robots: model learns how physical world behaves (through video) and then navigates that world.

To get there will require a lot of behavioral cloning of humans showing how this could be done.

4 Layer Architectures

Different architectures of neural networks perform well on different types of data. To solve AI problems involving visual, audio or textual data, we need to choose the right circuit diagram (that's just a metaphor, my friend). This is the high level structure of our deep learning system. Now things start to get rocketscientish. There are many different neural network layer architectures, but the most common ones are the following three:

A. CONVOLUTIONAL NEURAL NETWORK (CNN OR CONVNET)

CIRCUIT DIAGRAM

DESCRIPTION

The network first learns to detect edges, then it uses the edges to detect simple shapes, then it uses these shapes to detect more higher-level features like facial shapes, and so on. In essence, the network breaks the complicated process of image classification into a series of simpler steps, each described by a different layer of the model.

WHEN TO USE

When your input is an image or audio file, and you can make your problem look like finding patterns in it, then convolutions may be exactly what you need.

TYPICAL USE-CASES

- Cat detection in images
- Perceiving the environment in autonomous driving
- Recommending a song you might like

EXAMPLE

Task	Detection
Input	Image
Output	Yes/No

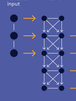
You should have gotten it by now, don't you think?

Well okay... you have an image with, say 360 pixels. Each pixel has a value. This value goes into the neural net, is being fed forward and then a single neuron tells, if the input was a cat image or not.

The trick is that all images have to be the exact same size.

B. SEQUENCE MODELS

CIRCUIT DIAGRAM



DESCRIPTION

A series of data enters the network which produces an output for as many steps as wanted. Btw, for the real pros: The so called "Recurrent Neural Networks" (RNNs), "Long short-term memory" (LSTM) and "Recursive Networks" are all Sequence Models.

WHEN TO USE

When you have sequential data where time is the main differentiating factor between the elements of the sequence. Examples include text, speech or video.

TYPICAL USE-CASES

- Natural language understanding
- Translation
- Video understanding

EXAMPLE

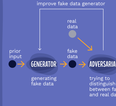
Task	Translation
Input	„A cat eats potatoes"
Output	Un chat mange des pommes de terre.

Imagine you are trying to translate sentences. First of all, different sentences have different amount of words. Second, a single word like potatoes maps to four words in french (des pommes des terre - apples of the soil). Oh man, the french you just need to ask me for the way to the airport in French and I take my pants off.

That's where sequence-to-sequence models come to rescue.
For the translation, not the pants, of course.

C. GENERATIVE ADVERSARIAL NETWORK (GAN)

CIRCUIT DIAGRAM



DESCRIPTION

GANs are a technique to generate synthetic pictures/videos that look authentic to the human observers. The goal of is to fool the discriminator into accepting the synthetic photo as an actual photo. GANs are currently all the rage in the deep learning community.

WHEN TO USE

When you want to generate data yourself, for example images.

TYPICAL USE-CASES

- Produce photorealistic images from text or other images
- Improve resolution of photos
- Predict next frame in a video

EXAMPLE

Task	Generate faces
Input	No input needed
Output	Images of faces

These faces have been generated by a computer.
Yes, holy shit. I don't have no words either. Prepare for the Matrix, my friend.

This happens when a neural network tries to generate fake faces trying to foot a second neural network.

Excursion for the Pros: CNN Deep Dive
Don't feel like a loser if you skip this

I heard that you were fascinated by the Convolutional Neural Networks (CNNs) and their image recognition skills. So let's dig a little deeper:

A CNN is, essentially, an input-output mapping. It transforms raw sensory input data (say, an image) into a label classification (say, a textual description what object is in the image). The input data is fed into the network and then a series of hidden layers in the network extract increasingly abstract "features" from the image. A feature is an attribute that the network "learns" from the data.

Do you feel strong enough for a more technical description? OK, you challenged it. There are four main operations in a classic CNN: convolution operations, ReLU operations, pooling operations, and lastly, a classification operation:

1. Convolution Operations: The primary purpose of the convolution step is to extract features from the input data (e.g. detecting an edge). During that step, different feature detectors slide over the input image and produce a set of "feature maps" (i.e. matrices of numbers).

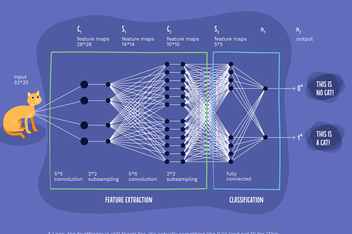
2. Rectified Linear Units (ReLU) Operations: Convolutions are linear operation. Most real-world data is non-linear though. The purpose of the ReLU operation is to introduce non-linearity in our CNN.

3. Pooling Operations: The pooling operation reduces the dimensionality of each "feature map" but retains the most important information. It makes the input representations smaller and more manageable. Depending on the number of layers in the network, there are multiple convolution and pooling steps until we arrive at the last layer of the network.

4. Classification Operation: The classification operation (last layer) uses the learned features to classify the input image into various classes based on the training dataset (e.g. cats and dogs).

Ready to Learn System:
A Network Architecture to Detect Objects in Images.

Yeah, we've just built a deep learning system. Let's assume that we want for a **supervised learning system** with **convolutional layers** to identify cats on pictures.

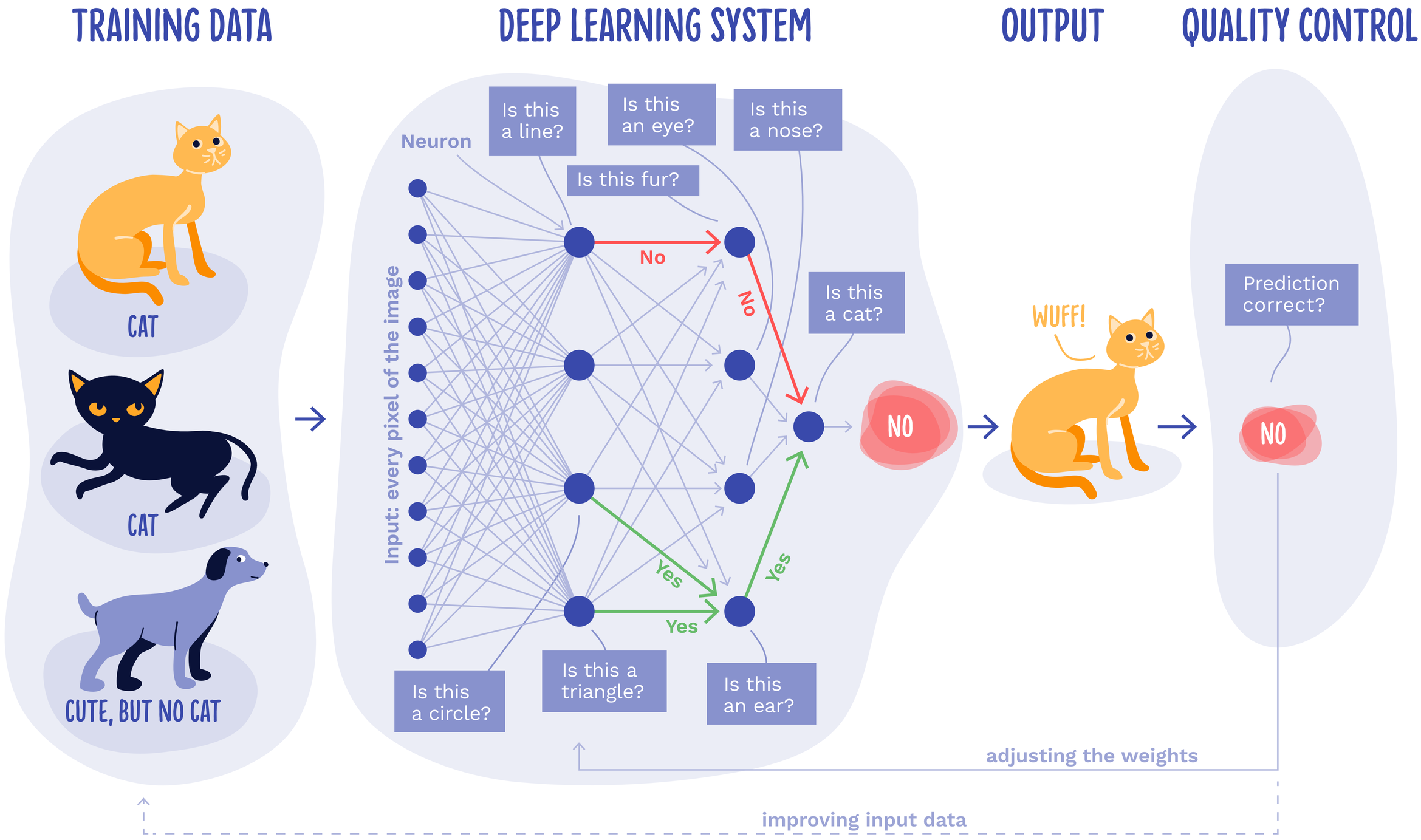


* Look, the faultfinder is still there! Yes, it's actually something like 0.01 (and not 0) for "This is no cat!" and something like 0.99 (and not 1) for "This is a cat!". You're exhausting, dude.

Yes, it looks way more complicated than the examples.
Because, well, it is way more complicated. But, I mean, you are a real pro now, right?

5 Train the New System

Even the most talented player is nothing without training. So now the show begins: Put your collected (labeled) data on the table and throw it into the monster you created:

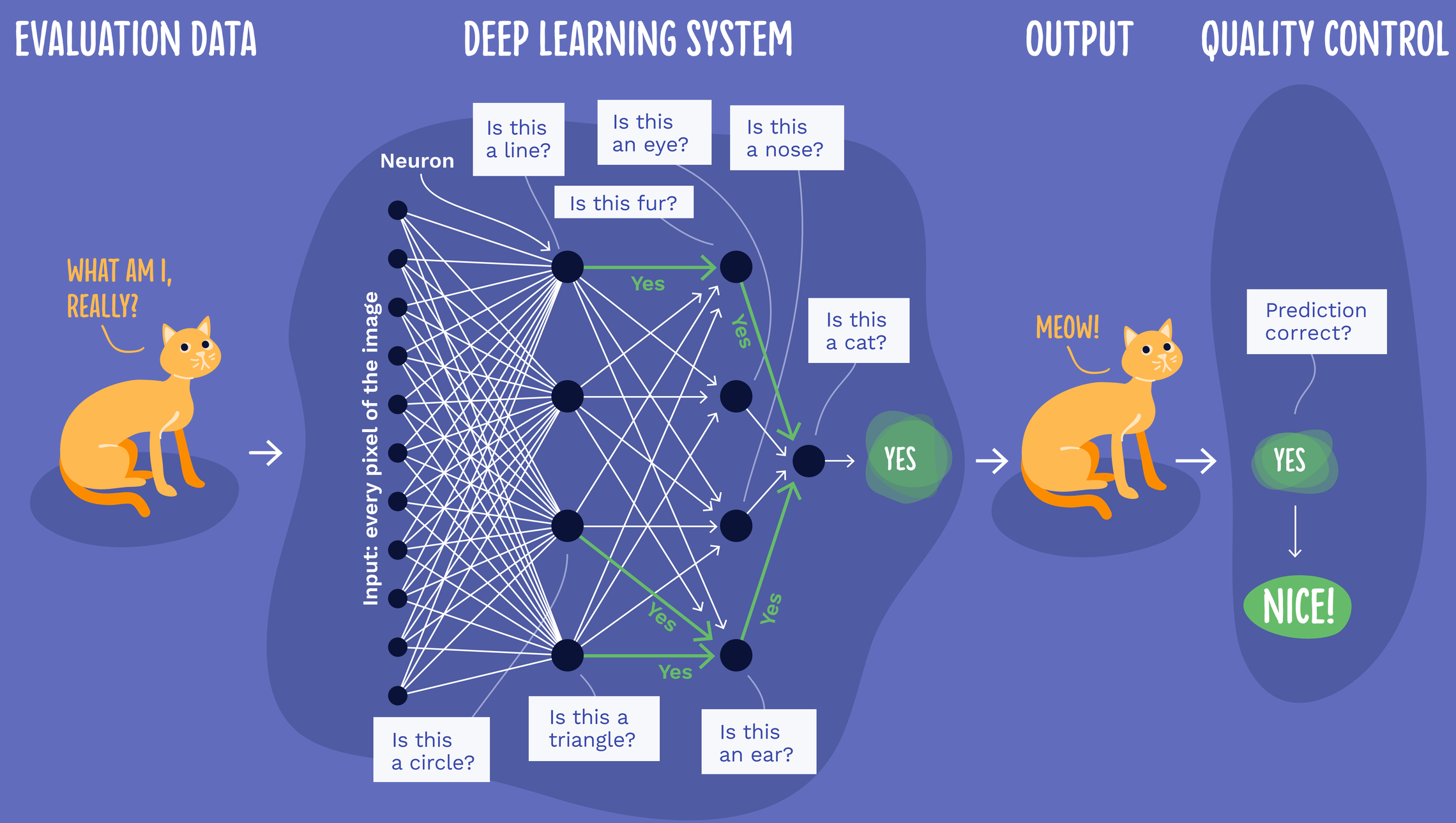


Training is hard. We train and optimize our deep learning system as long as the output satisfies our immense expectations.

Optimizing means changing the weights in the network, so that a cat picture input actually yields a cat label output. But this can also mean collecting more or better data .

6 Evaluate the Trained System

We can't rely just on results based on our (labelled) evaluation data. This is the point of truth: Let's throw in new, unseen data sets and check how our machine performs on this fresh input.

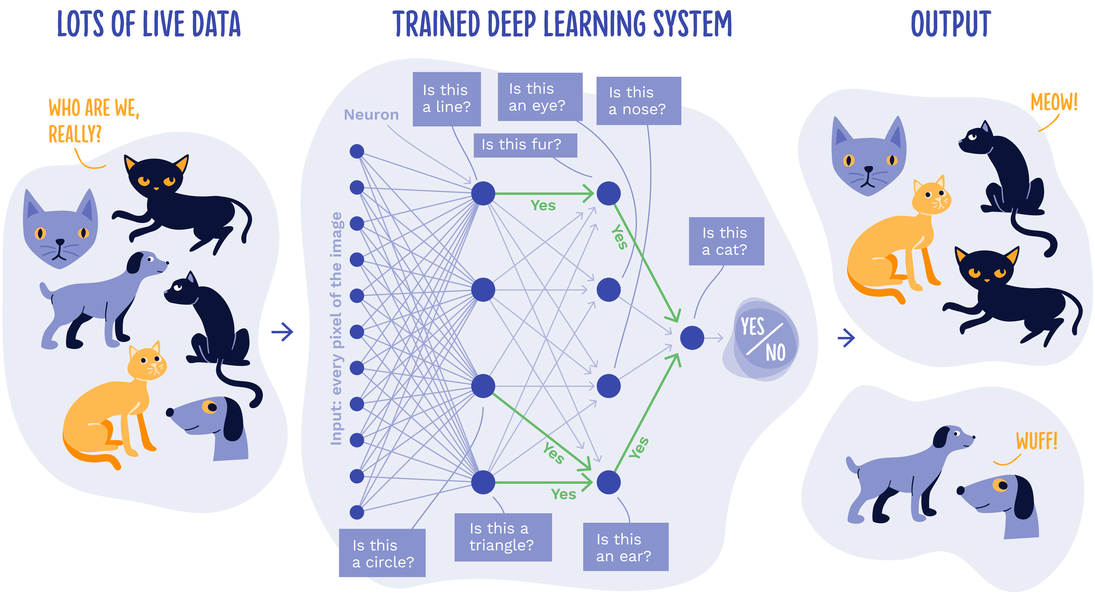


We continuously optimize our system to the best outputs on the evaluation data.
The results are satisfying?
We are ready to rumble.

7 Launch The Live System

For the smartypants amongst you: Still there?
OK, we know: Now we have to deploy the system on a (cloud) server* or a client like a smartphone.
Once the training is finished, which can take months**, the live system only needs milliseconds for evaluation. **BOOM!**

* This would be actually the single best place in this whitepaper to promote Microsoft's amazing Azure cloud solution. Just saying.
** If you won't use Azure Batch AI services. Just saying.



Learning never ends, but you have just gained your PhD in Deep Learning, you sly fox.

Author

ISKENDER DIRIK

Managing Director / CEO
Microsoft ScaleUp Berlin

Encore: Why is Deep Learning so Revolutionary?

Iskender Dirik: Moritz, you're the Co-Founder of Twenty Billion Neurons (TwentyBN) and working on advanced deep learning systems that understand video. Why is deep learning so revolutionary?

Moritz Müller-Freitag: Deep learning is a completely different way of using computers. Instead of writing explicit, step-by-step instructions for the computer to follow, we are exposing deep neural networks to the real world and letting them learn. We are “showing” the computer what to do, rather than “programming” it.

To me personally, there are two reasons why deep learning is such a transformative technology.

First, deep learning helps us solve problems that were previously intractable. In the past five years, we have seen a number of step function improvements that have fundamentally pushed the envelope of what computers are capable of. This includes advances in image recognition, machine translation, speech synthesis, and lately also video understanding, which is our focus at TwentyBN.

Second, deep learning is remarkably accessible to students, or in fact anyone who is interested in this nascent field and wants to reskill themselves. If you understand calculus and algebra, you can understand what happens under the hood of a neural network. Undergraduates can do it, something you can’t say about other fields like nuclear physics or rocket science.

Taken together, I believe that deep learning will fundamentally transform how we write software in the future. The exciting thing is that we are still in the very early stages of this shift. There is so much more to come!

Iskender Dirik: Thanks Moritz; could you be my secret weapon against the smartypants?