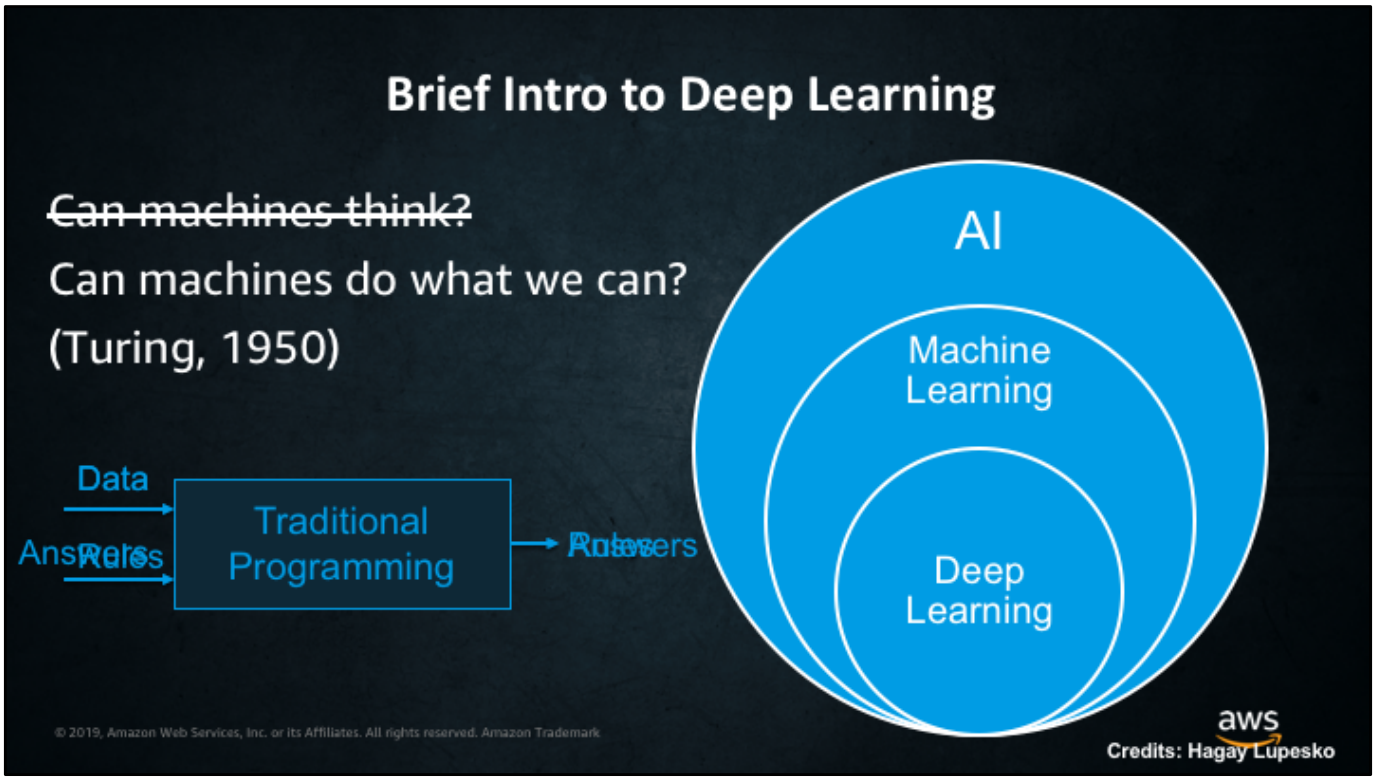
Generative Adversarial Networks(GANs) using Apache MXNet

Vandana Kannan & Piyush Ghai
Amazon AI

Agenda

- Introduction to AI and neural networks
- Convolutional neural network (CNN)
- How GANs work
- Demo
- GANs in recent years
- Tools that you can use

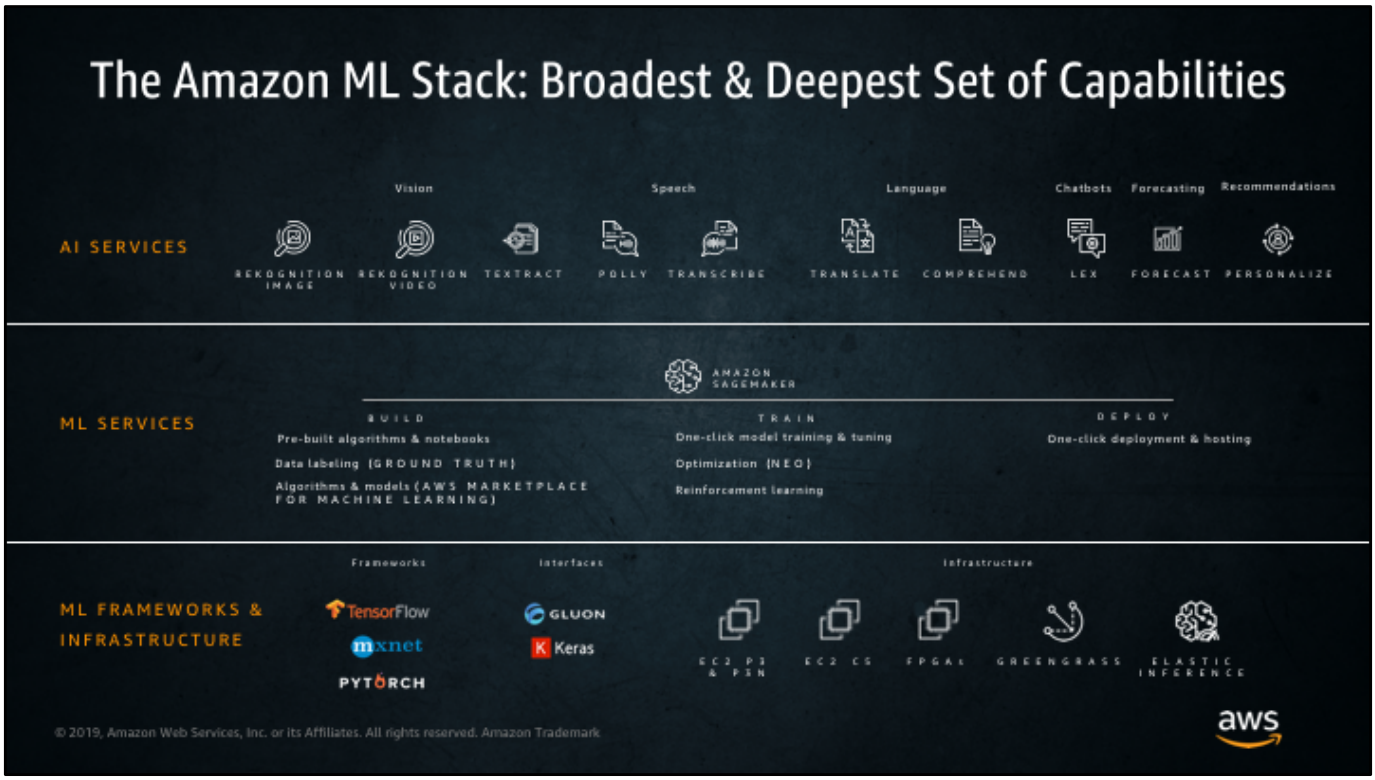


Let's start with AI. AI is an active research area dating back to at least the 50s if not earlier. Investigating the various aspects of enabling machines to mimic, and surpass, human intelligence. Alan Turing, a computing pioneer, calibrated the essence of AI by moving from the philosophical question "can machines think" to a more relevant question "can machines do what us humans can?"

ML is a subset of AI, and is really a different programming paradigm. Traditional programming, that is mostly taught at schools, is about us humans programming rules, and the machine executing these rules on data to provide answers.
ML is taking in data and answers, and constructs the rules by itself. This is closer to how humans learn from experience.
So ML is the set of techniques that enables machines to learn rules from data, without being explicitly programmed. ML is really an umbrella term that includes algorithms like decision trees, SVM and also neural networks.

This takes us to Deep Learning. DL is a subset of ML, a technique inspired by the human brain – or neurons to be more exact – that uses interconnected artificial neurons to learn from samples.

Deep Learning has a wide variety of applications, some of which we use everyday.
We see personalized movie recommendations on Netflix, product recommendations on Amazon, personalized ads etc.
We make use of robots too – robo vacuums at home. Fulfillment centers make use of these robots to move boxes around.
We have voice assistants such as Alexa, Siri, Google assistant etc.
And we have drones and self-driving cars too!
All of these run on AI/Deep Learning!



We see the Machine Learning stack having three key layers.

ML Frameworks:

- The bottom layer is for expert machine learning practitioners—researchers and developers.
- These are people who are comfortable building models, tuning models, training models, figuring out how to deploy into production, and manage them themselves.
- And the vast majority of machine learning in the cloud today at this layer is being down through Amazon SageMaker which provides a managed experience for frameworks, or the AWS Deep Learning AMI that we built that effectively embeds all the major frameworks.

Infrastructure:

- AWS offers a broad array of compute options for training and inference with powerful GPU-based instances, compute and memory optimized instances, and even FPGAs.
- Our P3 instances provide up to 14 times better performance than previous-generation Amazon EC2 GPU compute instances.
- C5 instances offer higher memory to vCPU ratio and deliver 25% improvement in price/performance compared to C4 instances, and are ideal for demanding inference applications.

- We also have Amazon EC2 F1, a compute instance with field programmable gate arrays (FPGAs) that you can program to create custom hardware accelerations for your machine learning applications. F1 instances are easy to program and come with everything you need to develop, simulate, debug, and compile your hardware acceleration code. You can reuse your designs as many times, and across as many F1 instances as you like.
- The new Amazon EC2 P3dn instance has four-times the networking bandwidth and twice the GPU memory of the largest P3 instance, P3dn is ideal for large scale distributed training. No one else has anything close.
- P3dn.24xlarge instances offer 96vCPUs of Intel Skylake processors to reduce preprocessing time of data required for machine learning training.
- The enhanced networking of the P3n instance allows GPUs to be used more efficiently in multi-node configurations so training jobs complete faster.
- Finally, the extra GPU memory allows developers to easily handle more advanced machine learning models such as holding and processing multiple batches of 4k images for image classification and object detection systems

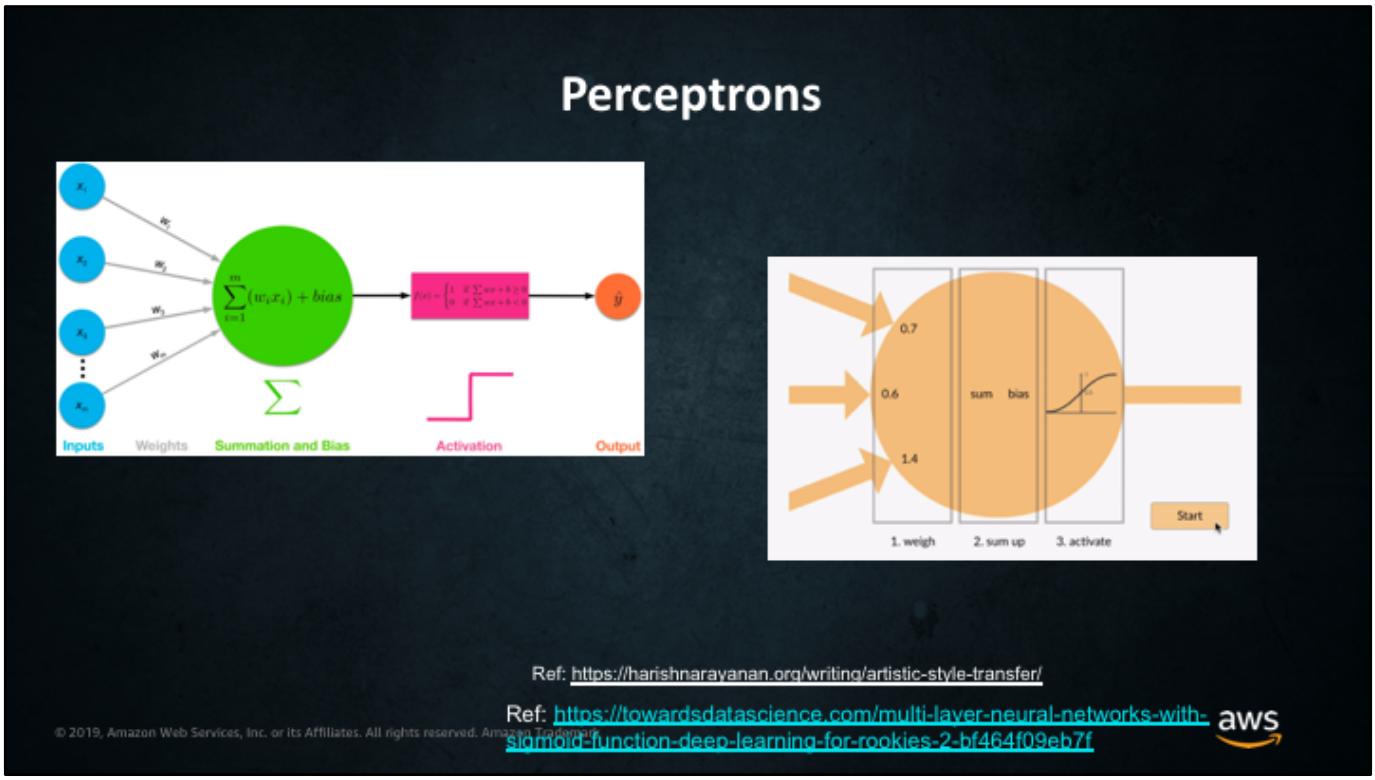
ML Services:

- But, if you want to enable most enterprises and companies to be able to scale machine learning, we've solved that problem for organizations by making ML accessible for everyday developers and scientists. Amazon SageMaker removes the heavy lifting, complexity, and guesswork from each step of the machine learning process.
- SageMaker makes model building and training easier by providing pre-built development notebooks, popular machine learning algorithms optimized for petabyte-scale datasets, and automatic model tuning, enabling developers to build, train, and deploy models in a single click.
- SageMaker is already helping thousands of developers easily get started with building, training, and deploying models.

AI Services:

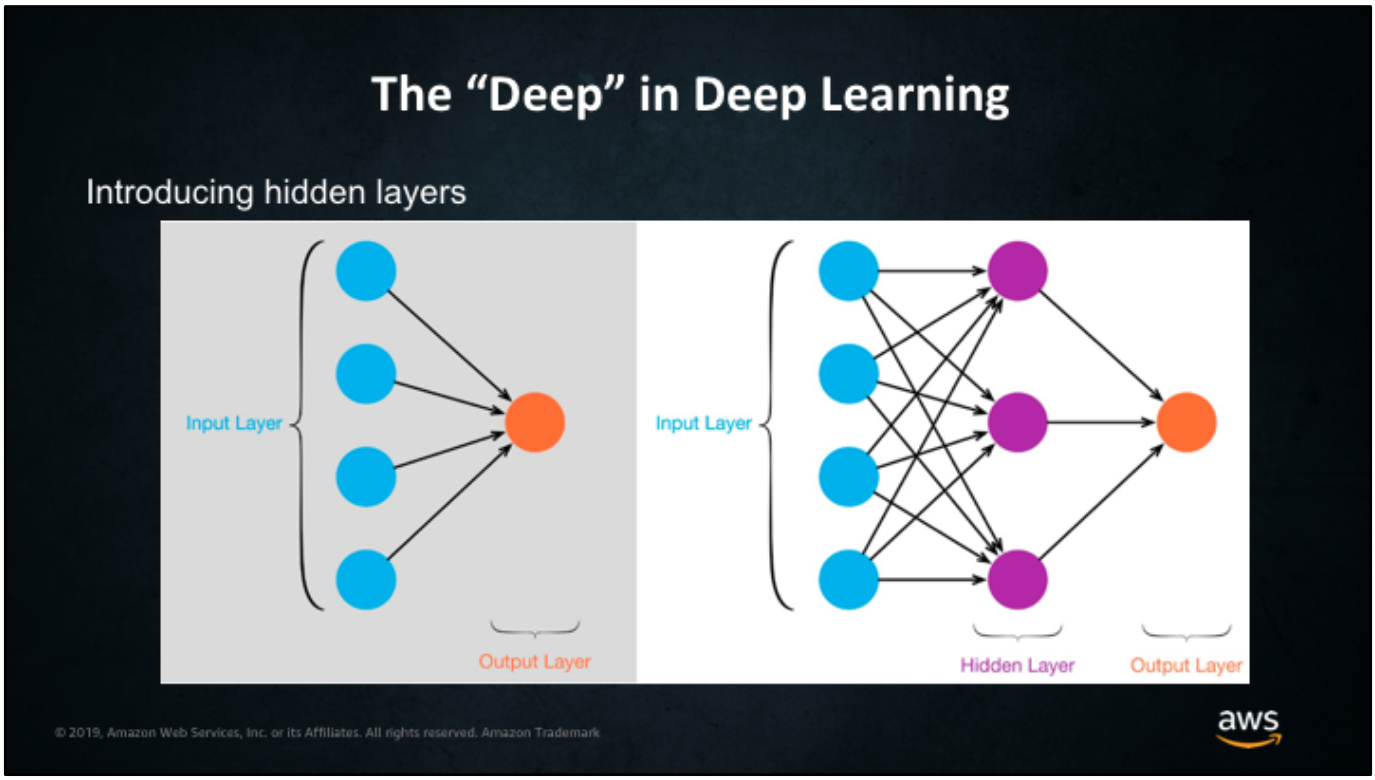
- At the top layer are AI services which are ready-made for all developers—no ML skills.
- For example, customers say here is an object, tell me what's in it, or here's a face, tell me if it's part of this facial group using Amazon Rekognition
- Or let me translate text to speech using Amazon Polly
- Or let's build conversational apps with Amazon Lex.
- Convert speech to text with Amazon Transcribe

- Translate text between languages using Amazon Translate
- Understand relationships and find insights from unstructured text using Amazon Comprehend



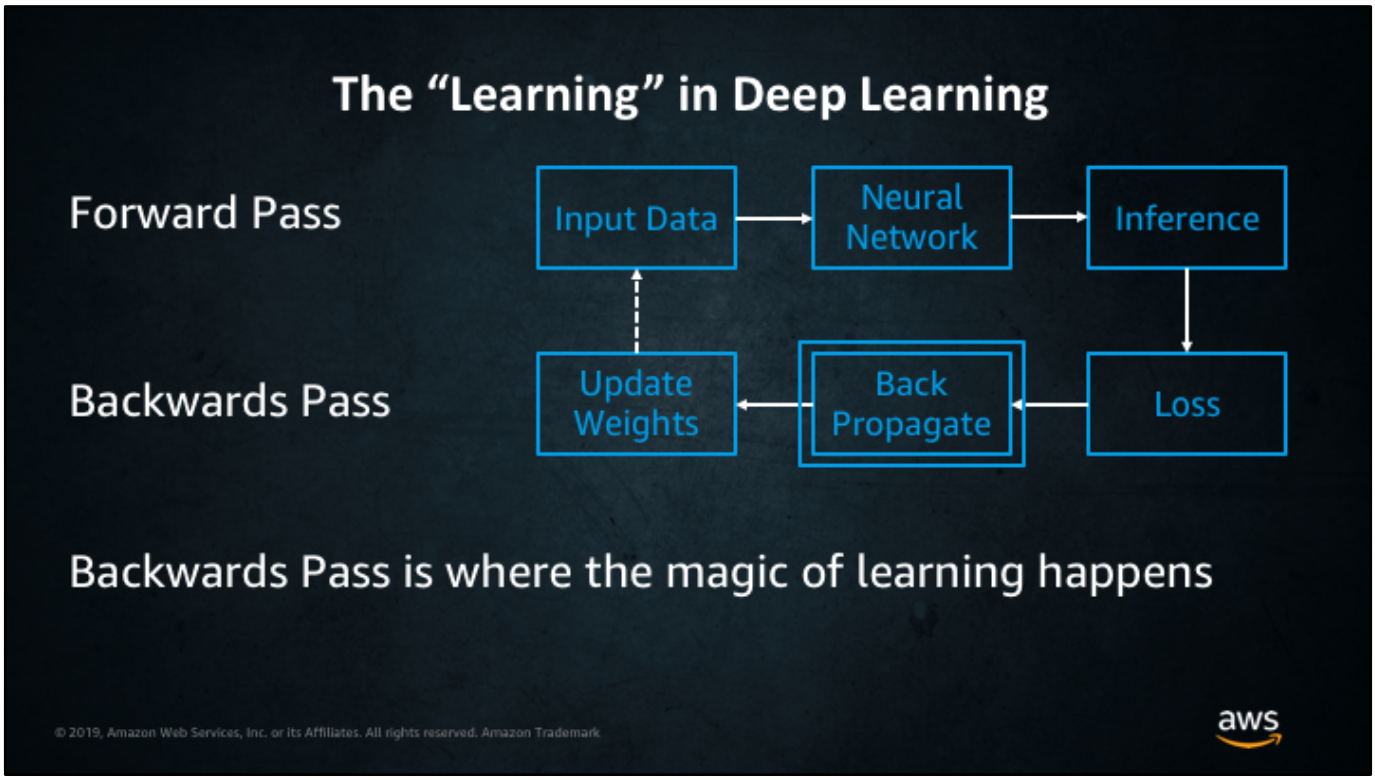
Let's start with the basics of Neural networks.
The fundamental unit of a neural network is a perceptron. It is just one layer of a neural network which takes some numbers as input and gives one output.
For example, say you want to ask the neural network to tell you if you should buy a particular house.
You would have different criteria such as price, sq.ft., neighborhood etc. And you would give certain weightage to each of these.
A perceptron would multiply the input values with weight, sum them up and pass it through an activation function. You would also set a threshold – if the resultant value is above a certain value,
You would buy the house or vice versa. This activation function would compare the result value with this threshold and give you a 1 or 0 output.

This neural network is the simplest version and is not suitable for complex applications.



So we introduce deep neural networks/ deep learning.

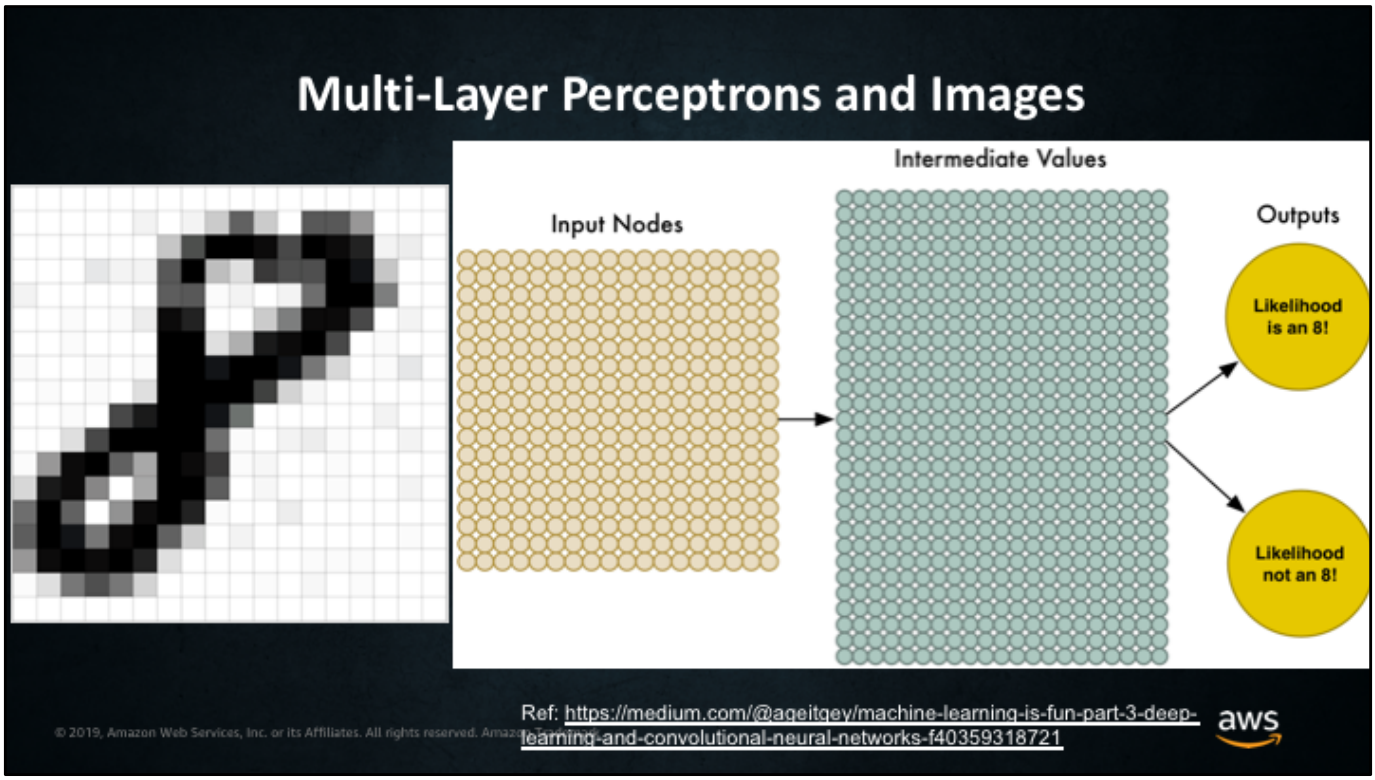
This was done by introducing multiple layers in between the input and output called hidden layers.



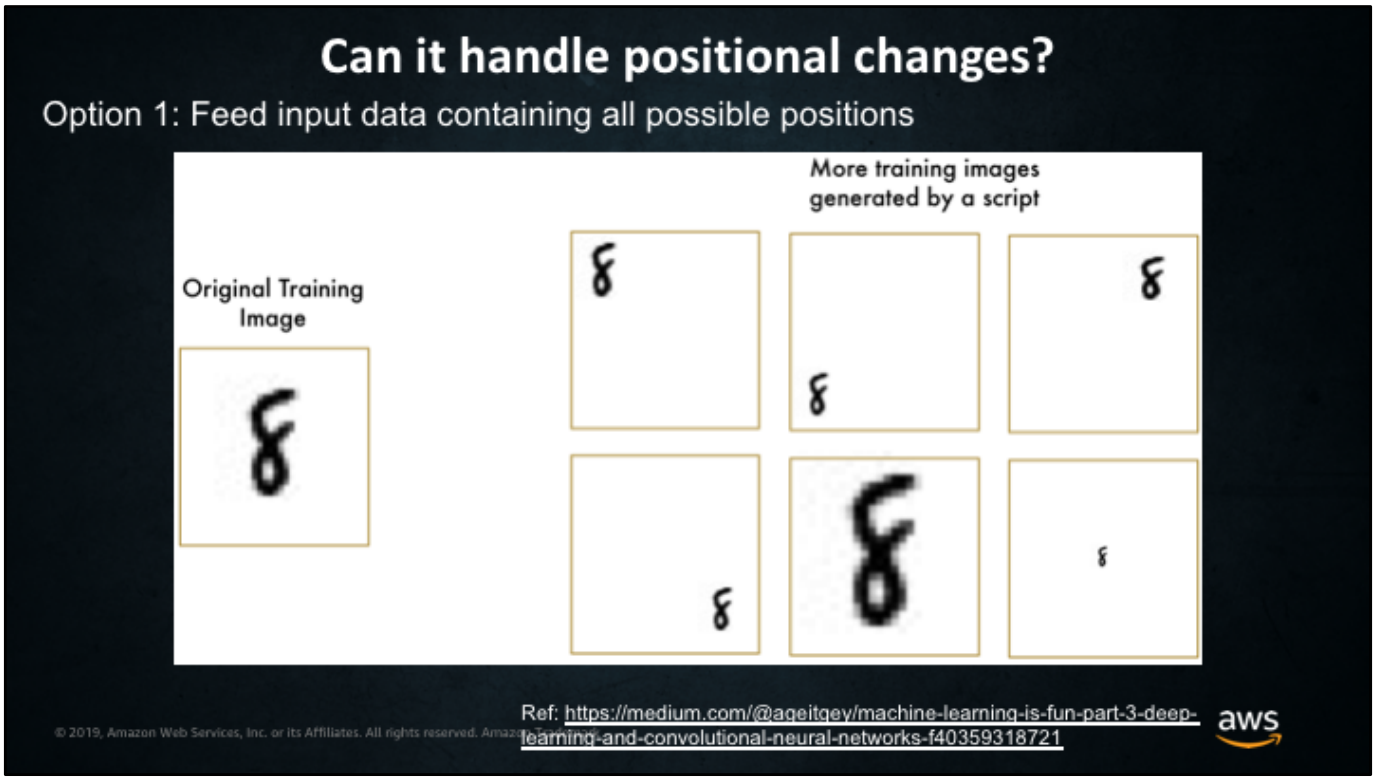
We need the neural network to learn from the information that we give it.

With the forward pass, the network just does some computation on the input data and gives an output (inference). This output may be correct/wrong.
If it is wrong, we need a feedback mechanism to tell the network about its mistakes and make it learn from it. The loss function helps us do this.

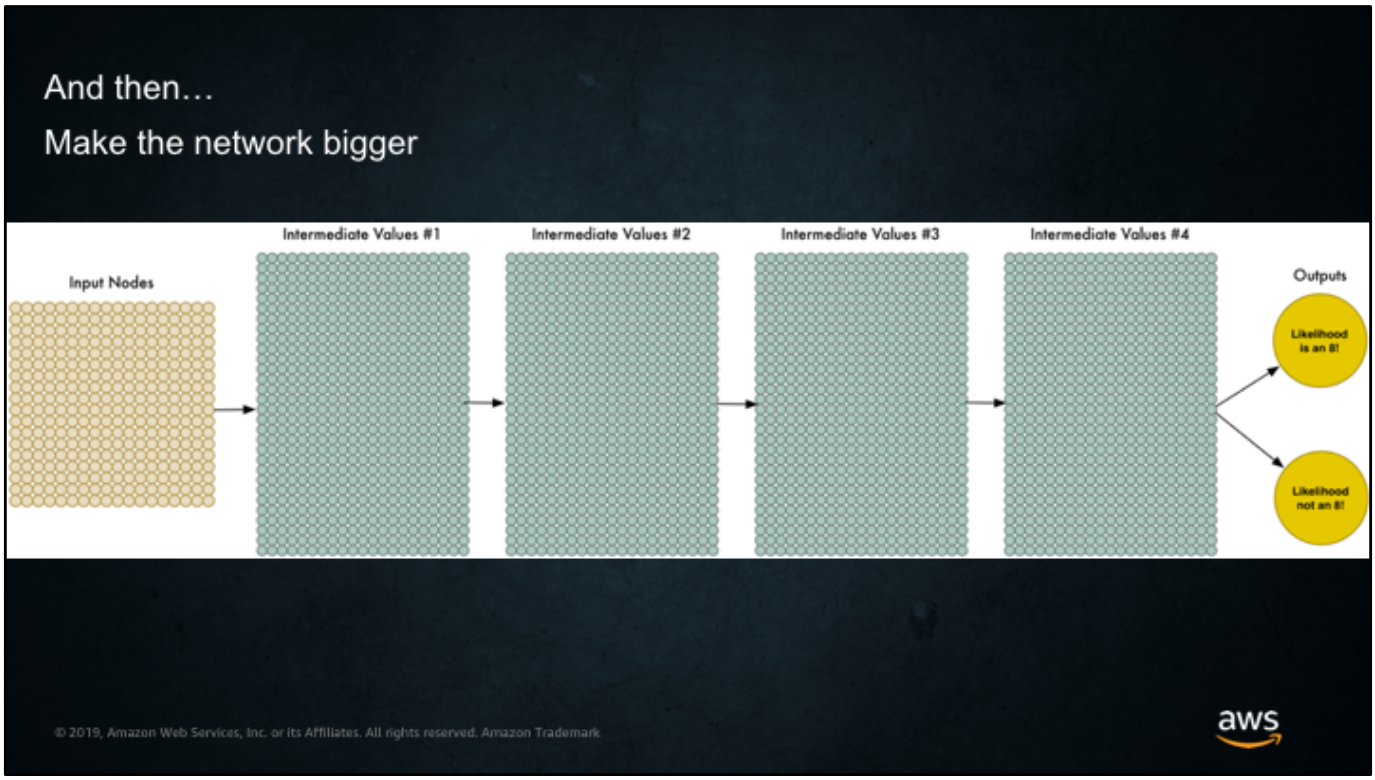
Given information about the correct answer, the network can compare its prediction, find out the loss and go back and learn/correct the weights of the network.
This is what happens in the back propagation step and update weights.



Say, you now want to give an image as input. After all, images are numbers.
In this example, you want your network to detect if the number in the image is an 8 or not.
You train your network on images such as this where 8 is at the center.
What happens if 8 is in a different position – the top, bottom, left or right?

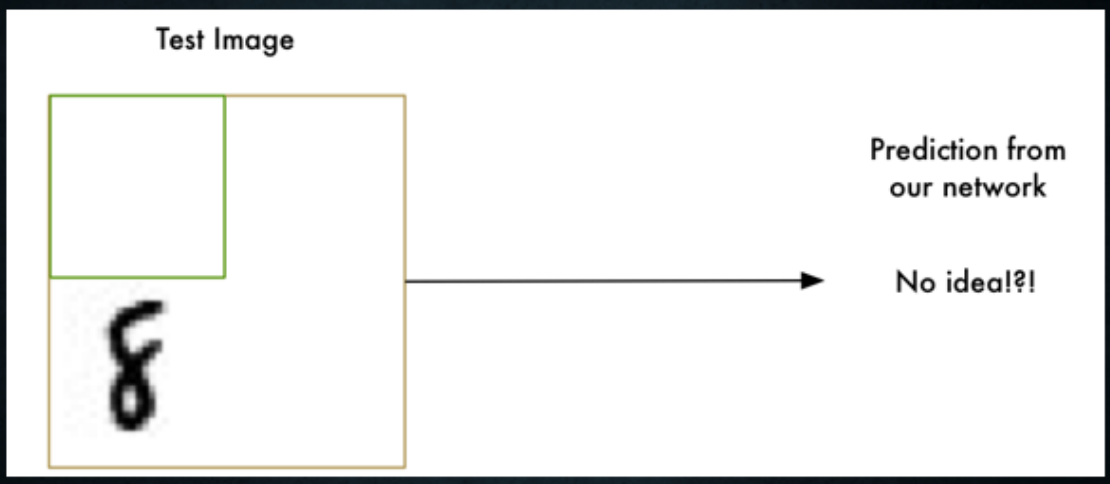


Another option would be to create a huge dataset with 8s in all possible positions and train the network.
This is going to be a lot of data!



Maybe use a bigger network for the huge dataset.
But this isn't the smartest way to process images.

Option 2: Use a sliding window to find an '8'



Ref: https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721

1. Entire image is flattened

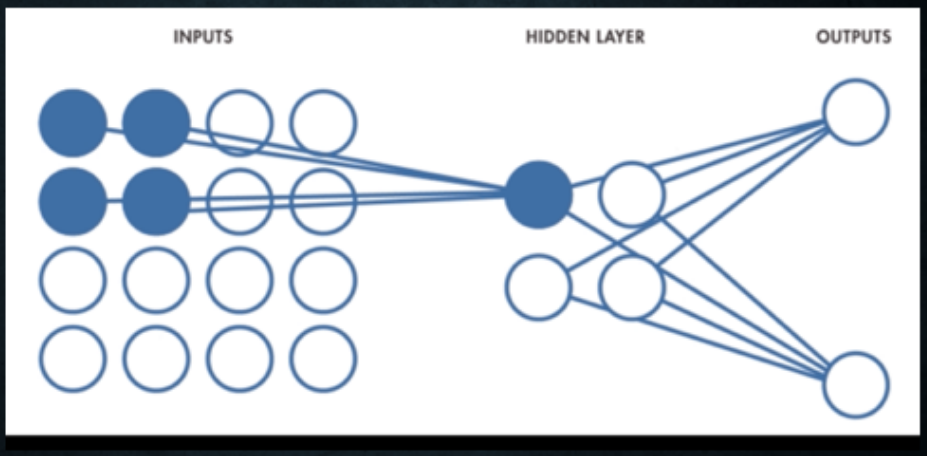
2. Intolerant to translation of objects

3. Large number of parameters

Convolutional Neural Networks

What do CNNs do differently?

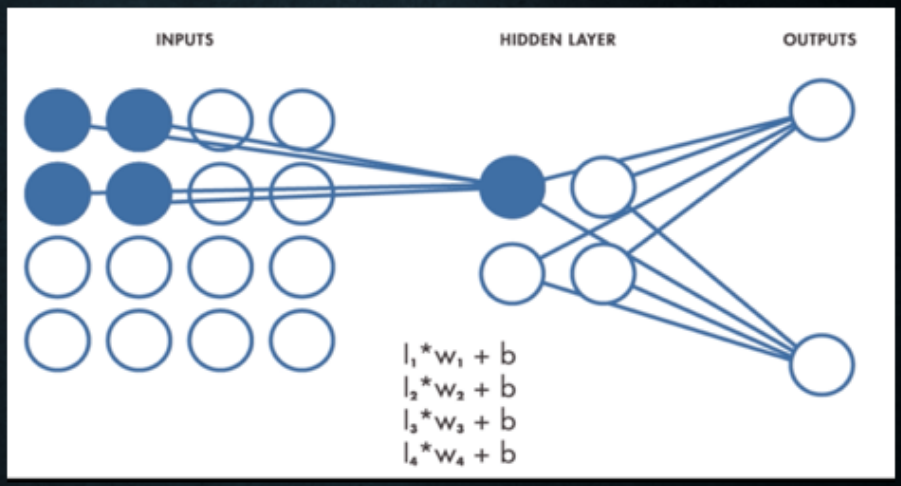
1. Localized feature maps



Ref: https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html

What do CNNs do differently?

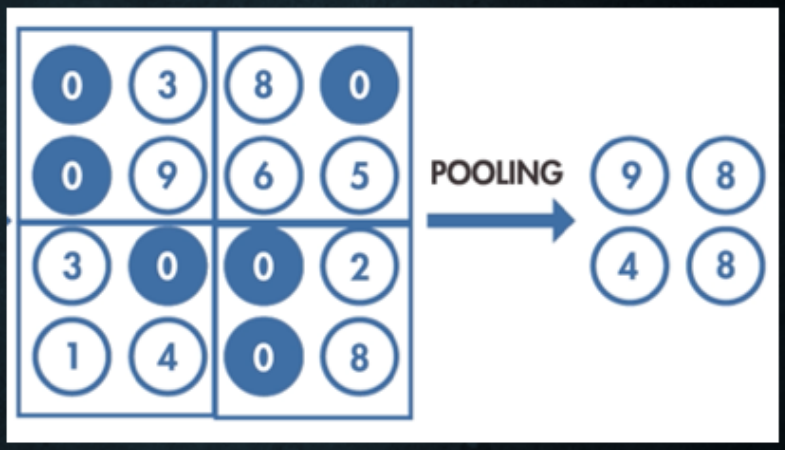
2. Shared weights



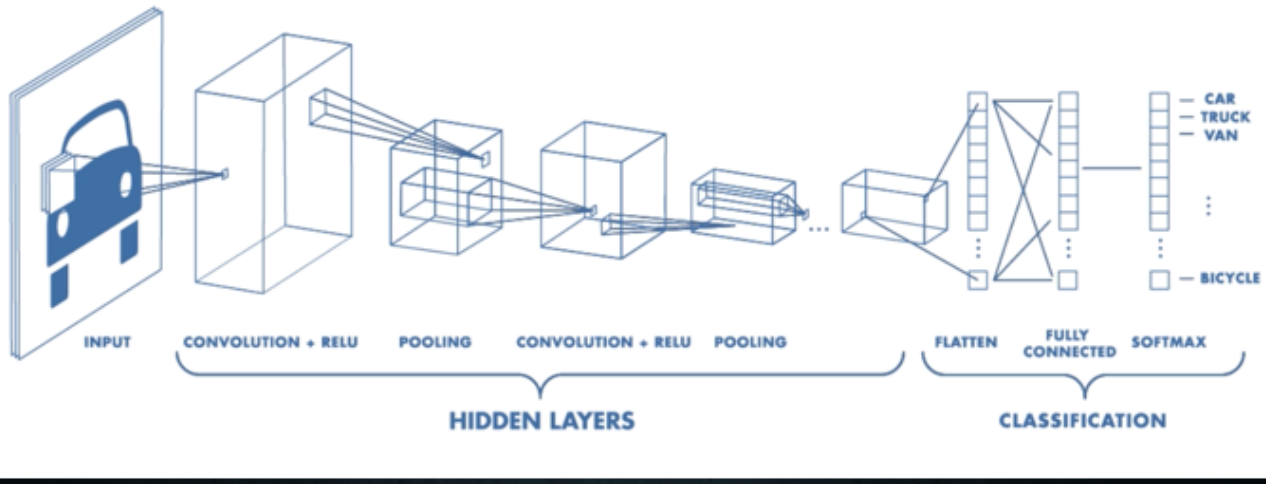
Ref: https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html

What do CNNs do differently?

3. Pooling

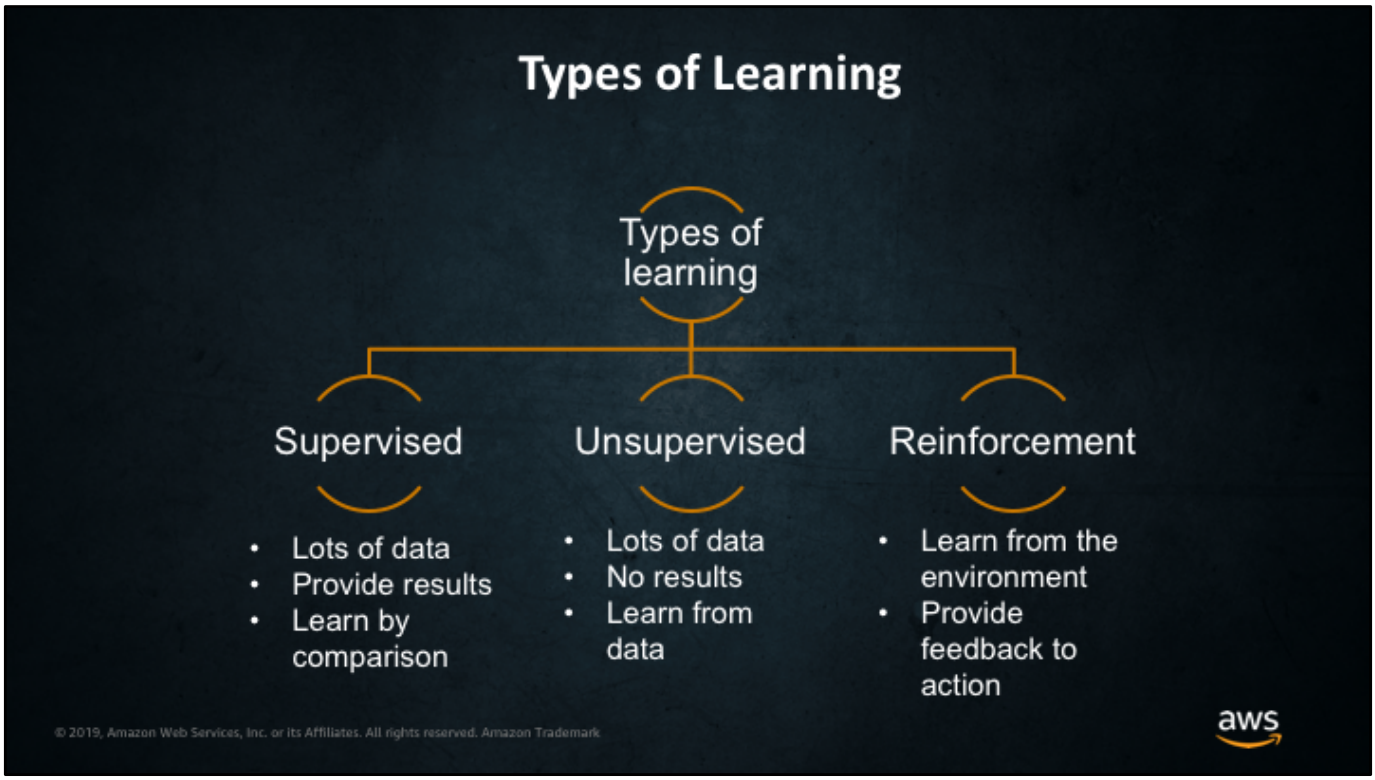


Ref: https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html



Recap…

- Intro to AI
- Intro to Neural Networks
- Intro to CNNs

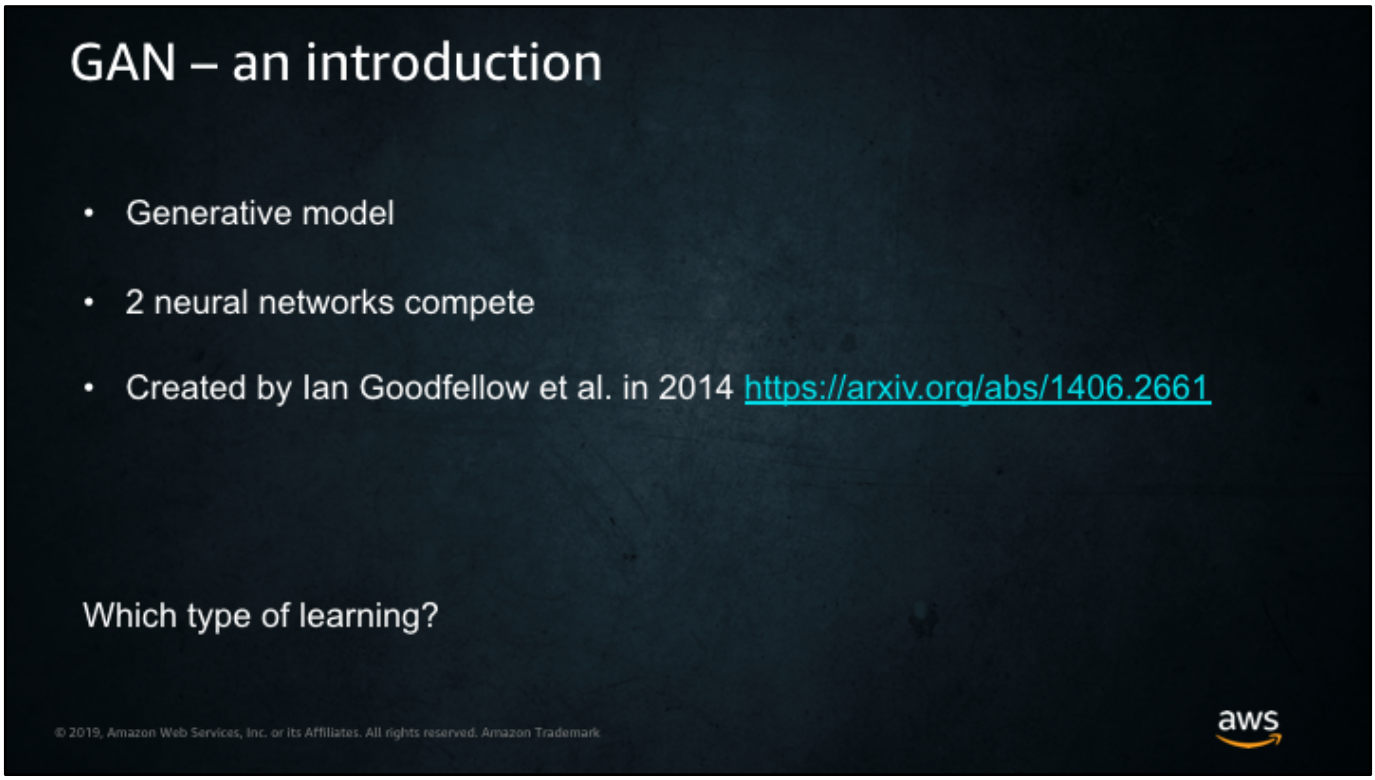


Before we start with GAN, we need to understand types of learning based on where the network learns from.

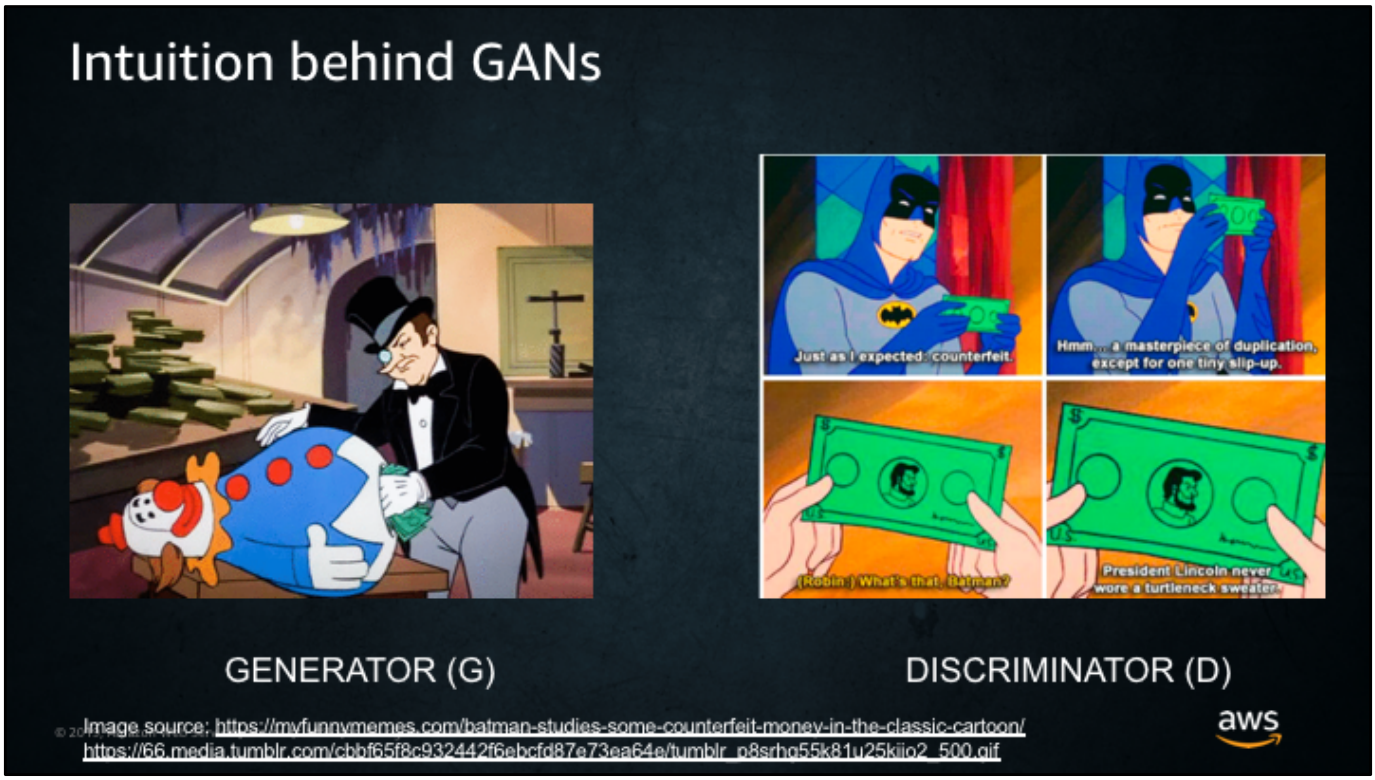
In supervised learning, you provide the model, a lot of data and the correct answers to the data so that while learning, the model can compare its prediction with the correct answer provides, to learn from mistakes and do a course correction.

In unsupervised learning, you provide the model with a lot of data but don't give it a correct answer. It is upto the model to learn the patterns in the data and come up with an answer.

In reinforcement learning, which is used in games and autonomous vehicles, the model learns through trials and error. It takes an action and is rewarded and punished for that action. The models makes further moves to maximize rewards.



GAN is a generative model where it learns from data and tries to generate something new after understanding the patterns in data.
Usually we make use of 1 neural network for our task, but GAN make use of 2 networks that fight against each other 2 win.

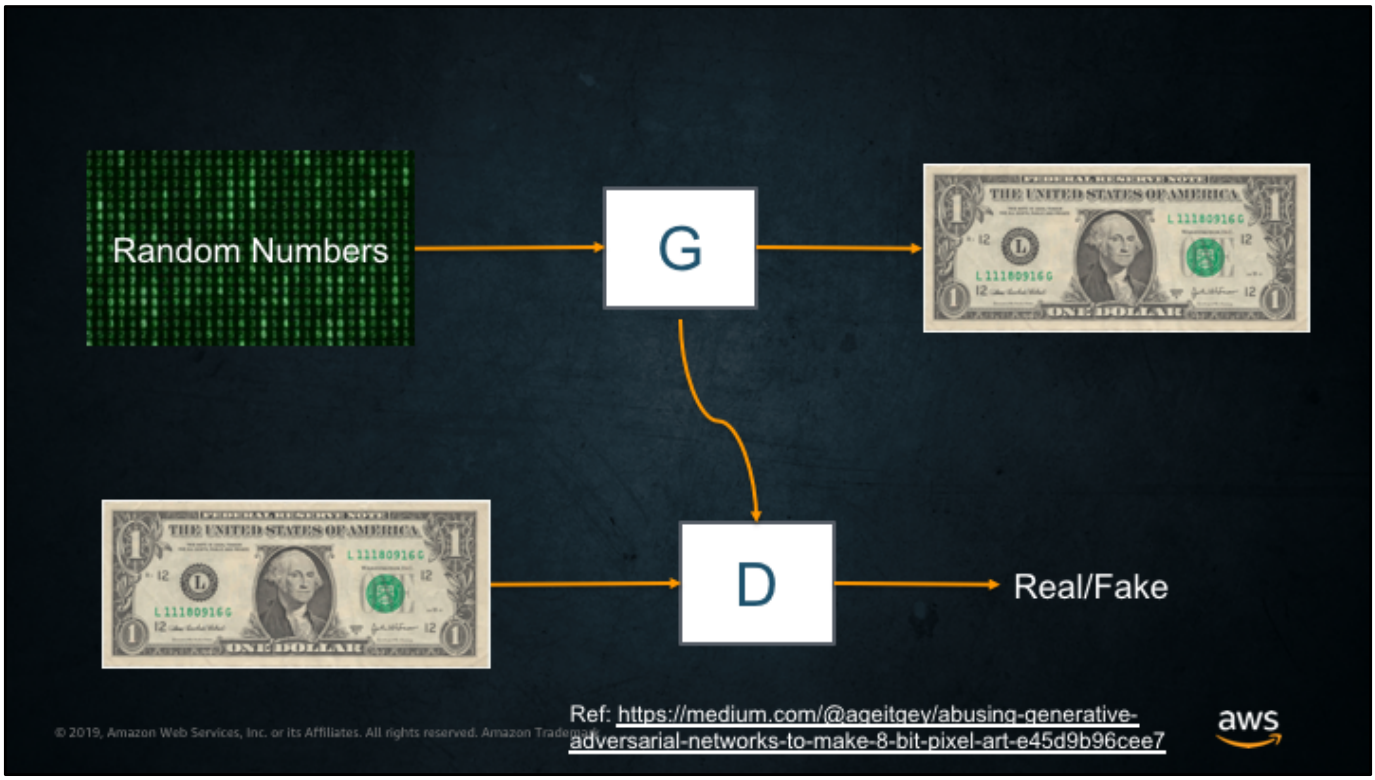


The 2 neural networks are called Generator and Discriminator.
The task of the Generator is to learn from the data and generate new content that is similar to the data distribution that it has learnt.

The task of the Discriminator is to be able to successfully differentiate between data that's coming from the real dataset and fake data that is coming from the Generator.

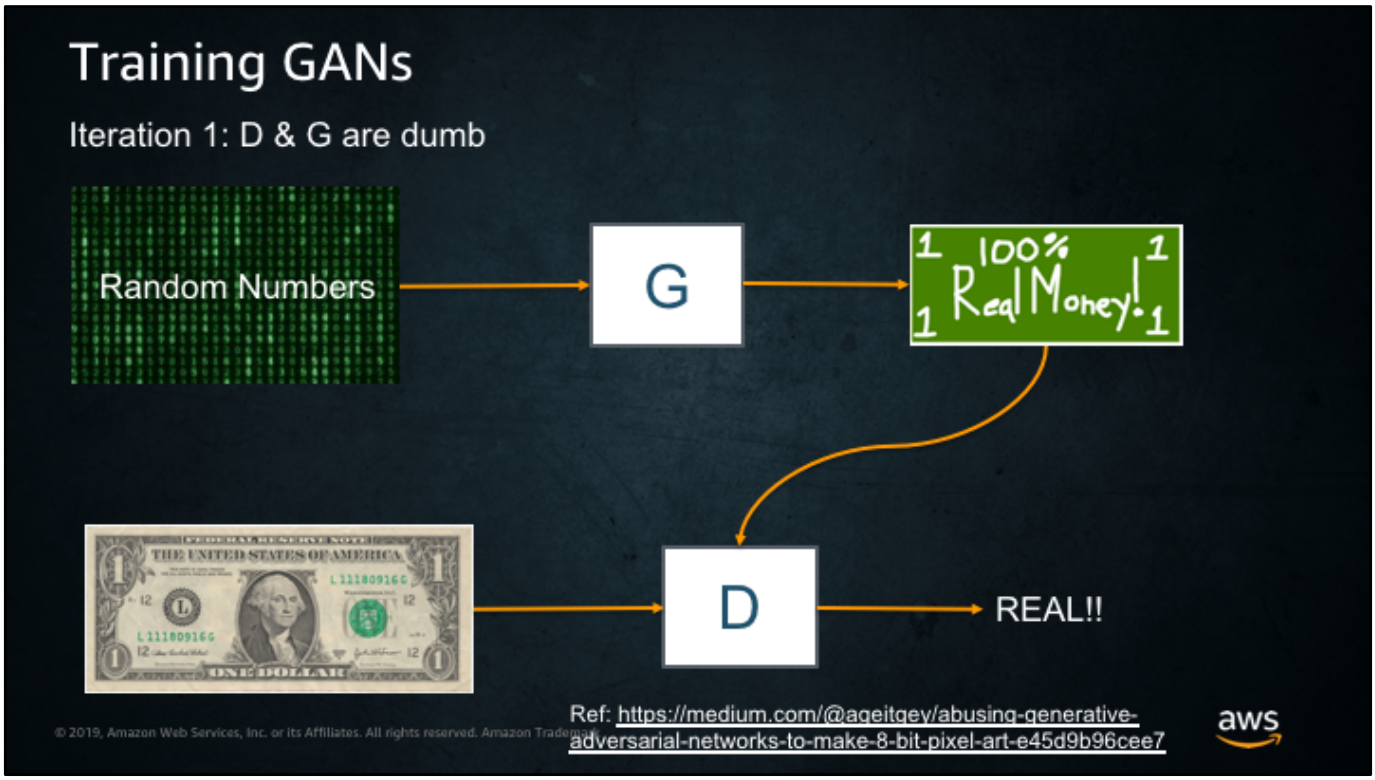
Both of them aim to be better than the other to do their job. They are adversaries.

Lets take an example, here you have the Penguin who is a Generator – generating counterfeit cash. And you have Batman, who is the Discriminator – detecting if a bill is real or fake.



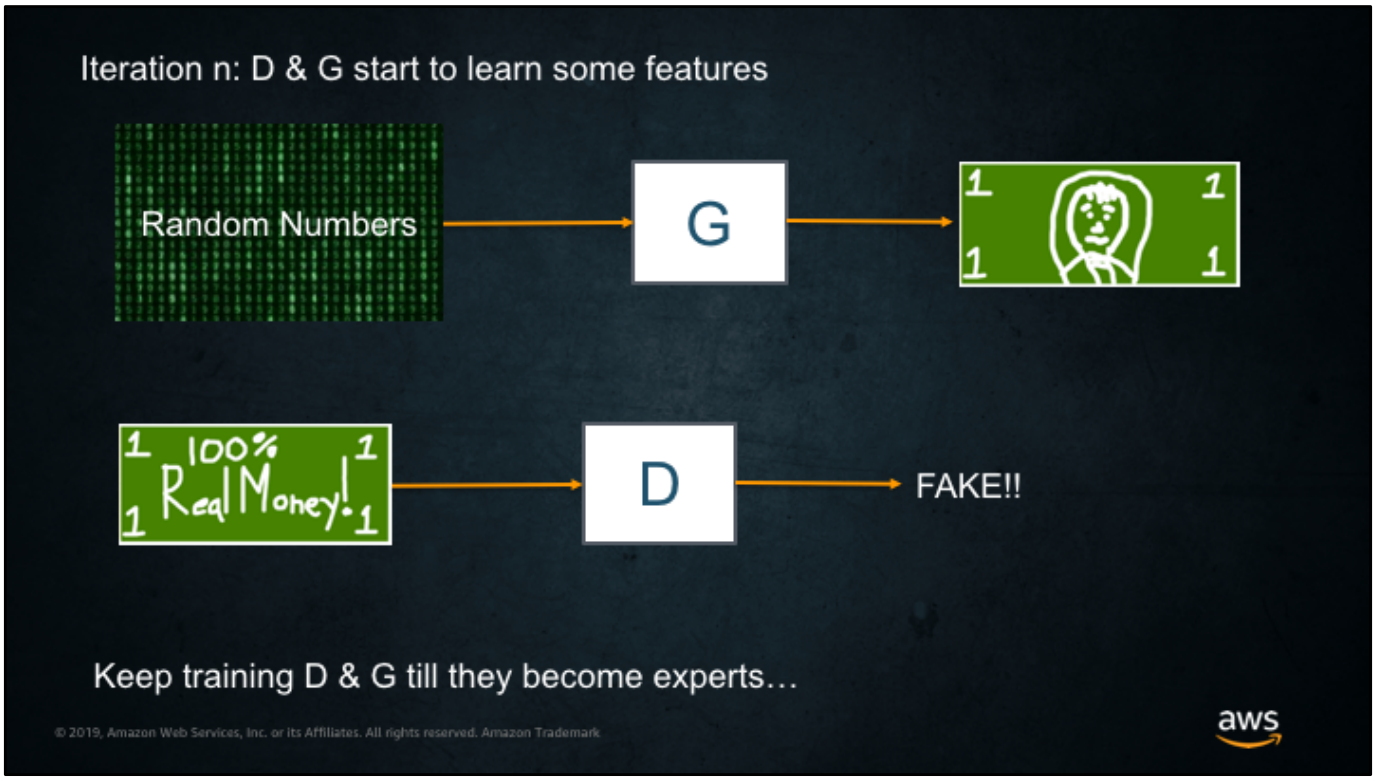
The Generator creates new cash from random numbers

The Discriminator is fed 2 types of inputs to learn – one from real data (cash here) and one from the counterfeit cash from the Generator.



At the beginning, both D and G are dumb. Both of them don't know what cash looks like.

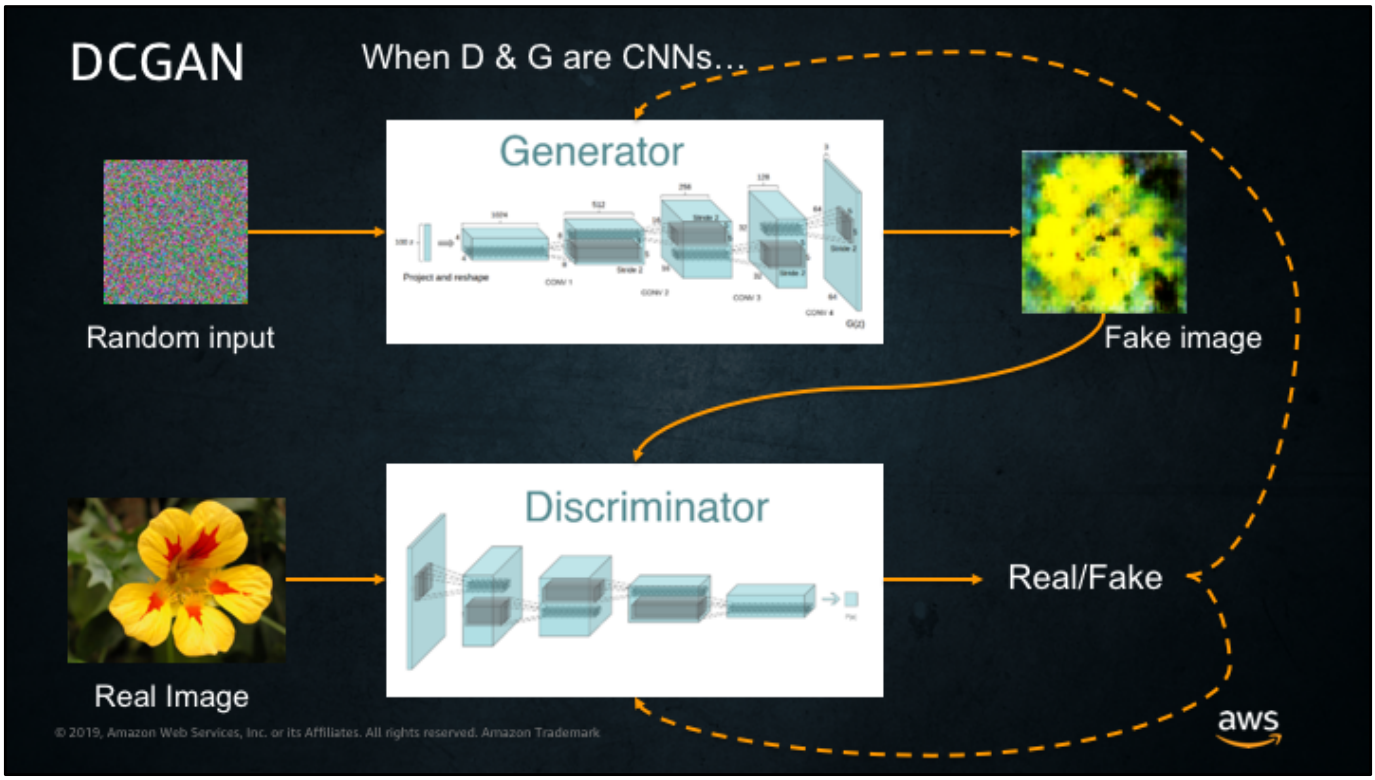
So in the early stages, if G produces a bill with "Real Money" written on it, D thinks its real



As training goes on, both of them learn new things about cash. For example, they realize that there is a face on a bill.

So, now D is able to detect an image with “Real Money” written on it, as fake.
And G has learnt to produce images with a face on it.

As we keep training D & G, it is expected that both of them become experts at their job. G at creating fake images. D at detecting real vs fake.



Since D and G are neural network after all, we can plug in different networks in its place.

In 2015, researchers introduced DCGAN which has CNNs as the network for the Discriminator and Generator.

Lets say, we are generating new images of flowers. G generates a fame image of a flower (it has learnt that it has to be something yellow). D tries to correctly detect the generated image as fake, based on what it has learnt from images of real flowers.

How do we give D & G feedback about their performance?
If D makes a mistake by classifying a real flower as fake or a fake flower as real, then the backward loop will update the weights of D.
If G makes a mistake by generating a flower image which fails D's test (is detected as fake), feedback is provided for it to generate better images.

DCGAN: Implementation walkthrough

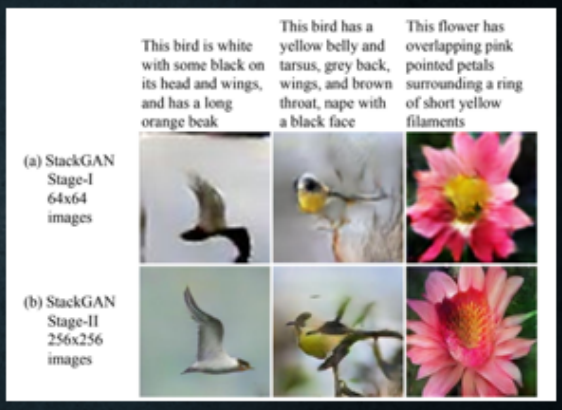
https://github.com/piyushghai/mxnet-gluon-gan/blob/dcgan/dcgan/dcgan_faces.ipynb

aws

Other GANs

StackGAN

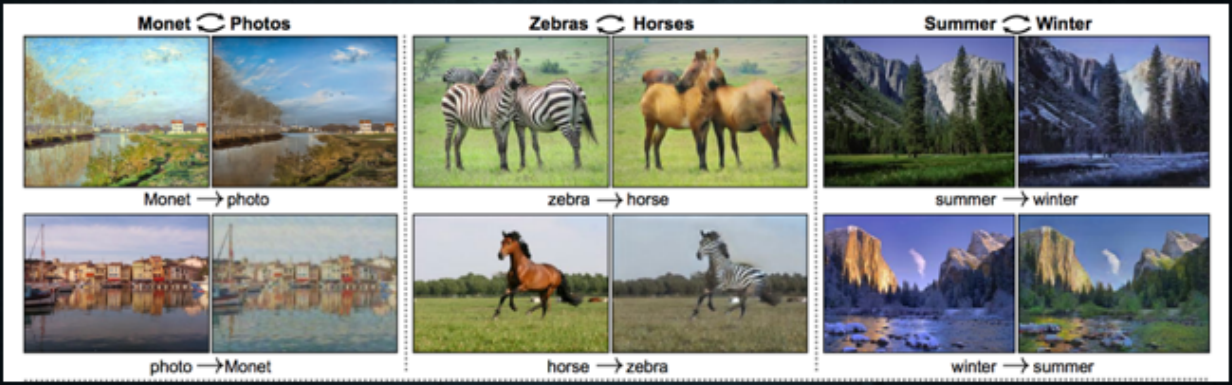
Text to image synthesis



Ref: https://arxiv.org/pdf/1612.03242.pdf

CycleGAN

Image domain transfer



Ref: https://arxiv.org/pdf/1703.10593.pdf

aws

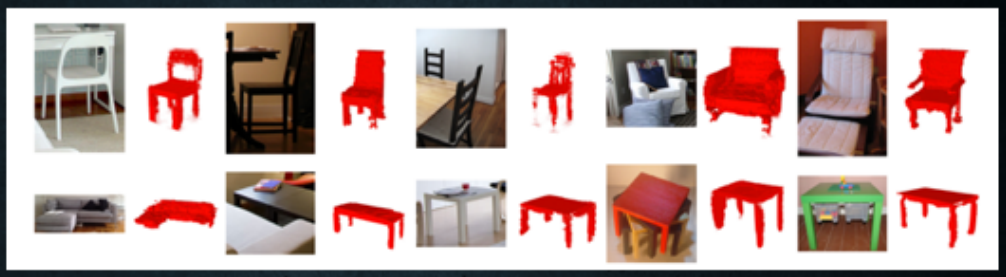
SRGAN

Image
Super-resolution

Ref: https://arxiv.org/abs/1609.04802

3D-GAN

3D reconstruction from images



Ref: http://3dgan.csail.mit.edu/papers/3dgan_nips.pdf

aws

StyleGAN

Learn attributes and styles

Ref: https://thispersondoesnotexist.com/

Some more info on GANs

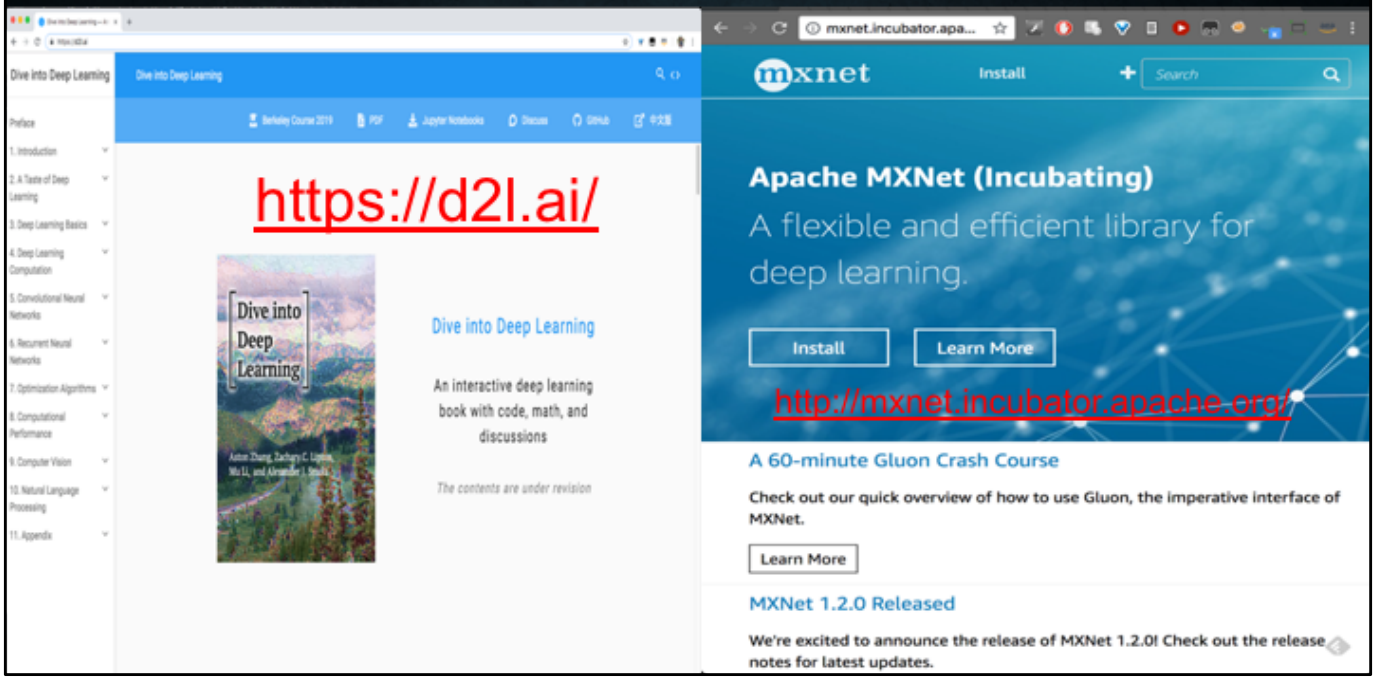
List of named GANs https://github.com/hindupuravinash/the-gan-zoo

How to load a pre-trained GAN (stored as MXNet .params file): https://mxnet.incubator.apache.org/versions/master/tutorials/gluon/info_gan.html

Tips & tricks for training GANs: https://github.com/soumith/ganhacks, https://medium.com/@utk.is.here/keep-calm-and-train-a-gan-pitfalls-and-tips-on-training-generative-adversarial-networks-edd529764aa9

Gluon Model Zoo: http://mxnet.apache.org/api/python/gluon/model_zoo.html

How to Get Started with Apache MXNet

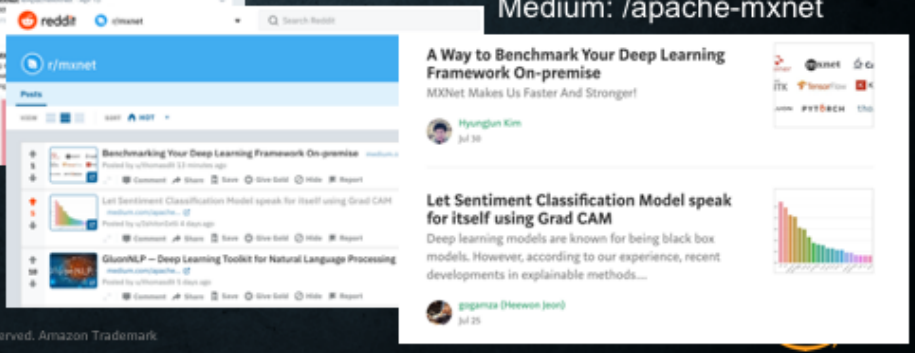


Apache MXNet Social

YouTube: /apachemxnet

Twitter: @apachemxnet

Medium: /apache-mxnet



Reddit: r/mxnet

Resources/References

- Apache MXNet – Flexible and efficient deep learning.
- https://github.com/apache/incubator-mxnet
- Apache MXNet Gluon Tutorials
- The Deep Learning Book
- MXNet – Using pre-trained models
- Amazon Elastic MapReduce
- https://medium.com/apache-mxnet
- https://twitter.com/apachemxnet

Please contact mxnet-info@amazon.com if you would like more information about you can get started with MXNet for your applications.

Using Apache MXNet with AWS ML Services

- Amazon SageMaker: https://aws.amazon.com/sagemaker/
- Amazon SageMaker Neo: https://aws.amazon.com/sagemaker/neo/
- Amazon Elastic Inference: https://aws.amazon.com/machine-learning/elastic-inference/
- Amazon Reinforcement Learning: https://aws.amazon.com/about-aws/whats-new/2018/11/amazon-sagemaker-announces-support-for-reinforcement-learning/
- AWS IoT Greengrass ML Inference: https://aws.amazon.com/greengrass/ml/
- Dynamic Training with Apache MXNet on AWS: https://aws.amazon.com/about-aws/whats-new/2018/11/introducing-dynamic-training-with-apache-mxnet/

References for the demo

Datasets:

- Classrooms: https://www.yf.io/p/lsun
- Flowers: http://www.robots.ox.ac.uk/~vgg/data/flowers/102/index.html
- Cartoons: https://google.github.io/cartoonset/download.html
- Faces: http://vis-www.cs.umass.edu/lfw/

Papers/websites:

- DCGAN paper: https://arxiv.org/pdf/1511.06434.pdf
- Gluon- The Straight Dope: https://github.com/zackchase/mxnet-the-straight-dope/blob/master/chapter14_generative-adversarial-networks/dcgan.ipynb
- Introduction to Generative Adversarial Networks (GAN) with Apache MXNet - AWS Online Tech Talks: https://www.youtube.com/watch?reload=9&v=4fQRoBz0BnM
- Machine Learning is Fun Part 7: Abusing Generative Adversarial Networks to Make 8-bit Pixel Art: https://medium.com/@ageitgey/abusing-generative-adversarial-networks-to-make-8-bit-pixel-art-e45d9b96cee7
- Keep Calm and train a GAN. Pitfalls and Tips on training Generative Adversarial Networks: https://medium.com/@utk.is.here/keep-calm-and-train-a-gan-pitfalls-and-tips-on-training-generative-adversarial-networks-edd529764aa9

Summary

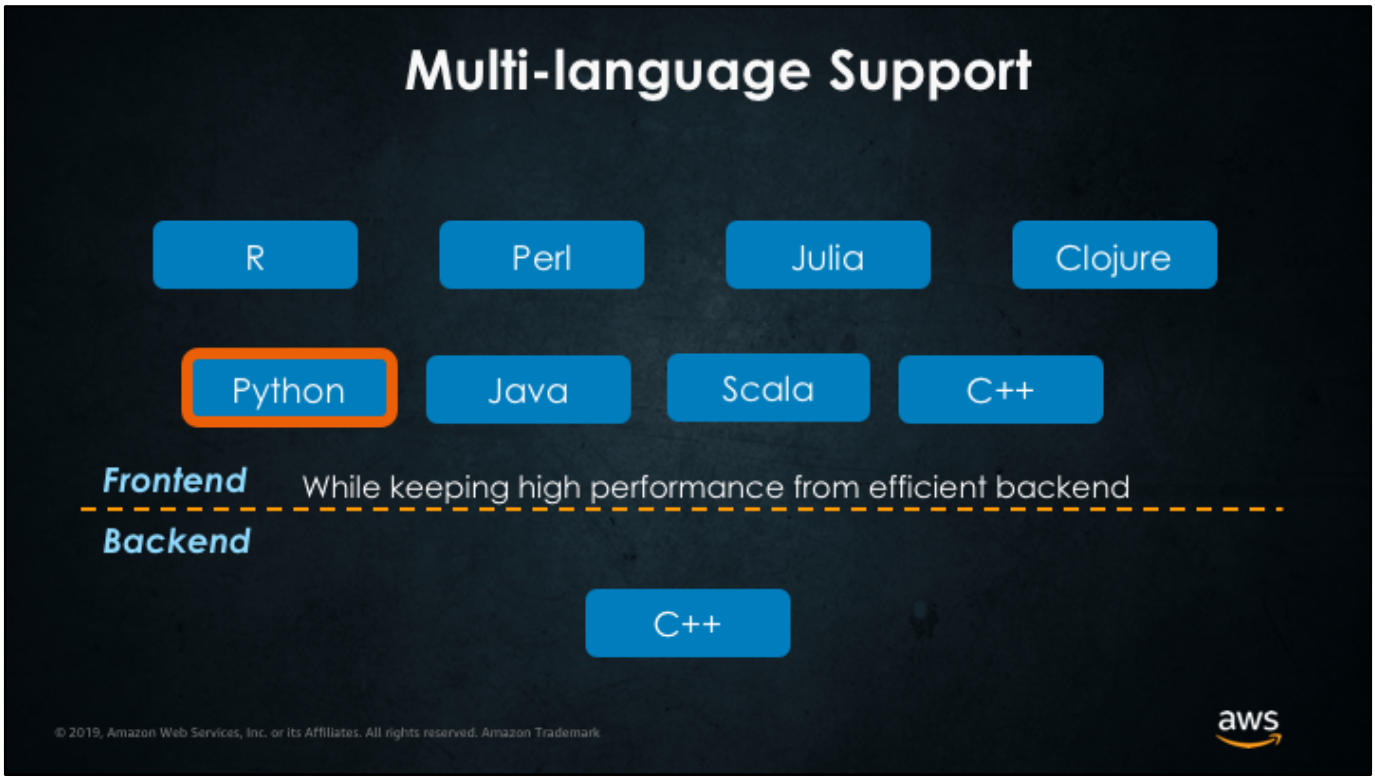
- Intro to AI
- Intro to Neural Networks
- Intro to CNNs
- Intro to GANs with an example of DCGAN
- Other GANs
- Getting started with Apache MXNet & Amazon Sagemaker

Thank you!!!

Apache MXNet - Background

- Apache (incubating) open source project
- Framework for building and training DNNs
- Created by academia (CMU and UW)
- Adopted by AWS as DNN framework of choice, Nov 2016

https://mxnet.incubator.apache.org/


Multi-language Support
R
Perl
Julia
Clojure
Python
Java
Scala
C++
Frontend
While keeping high performance from efficient backend
Backend
C++
© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved. Amazon Trademark
aws

Deep Learning Acceleration

CUDA & CuDNN

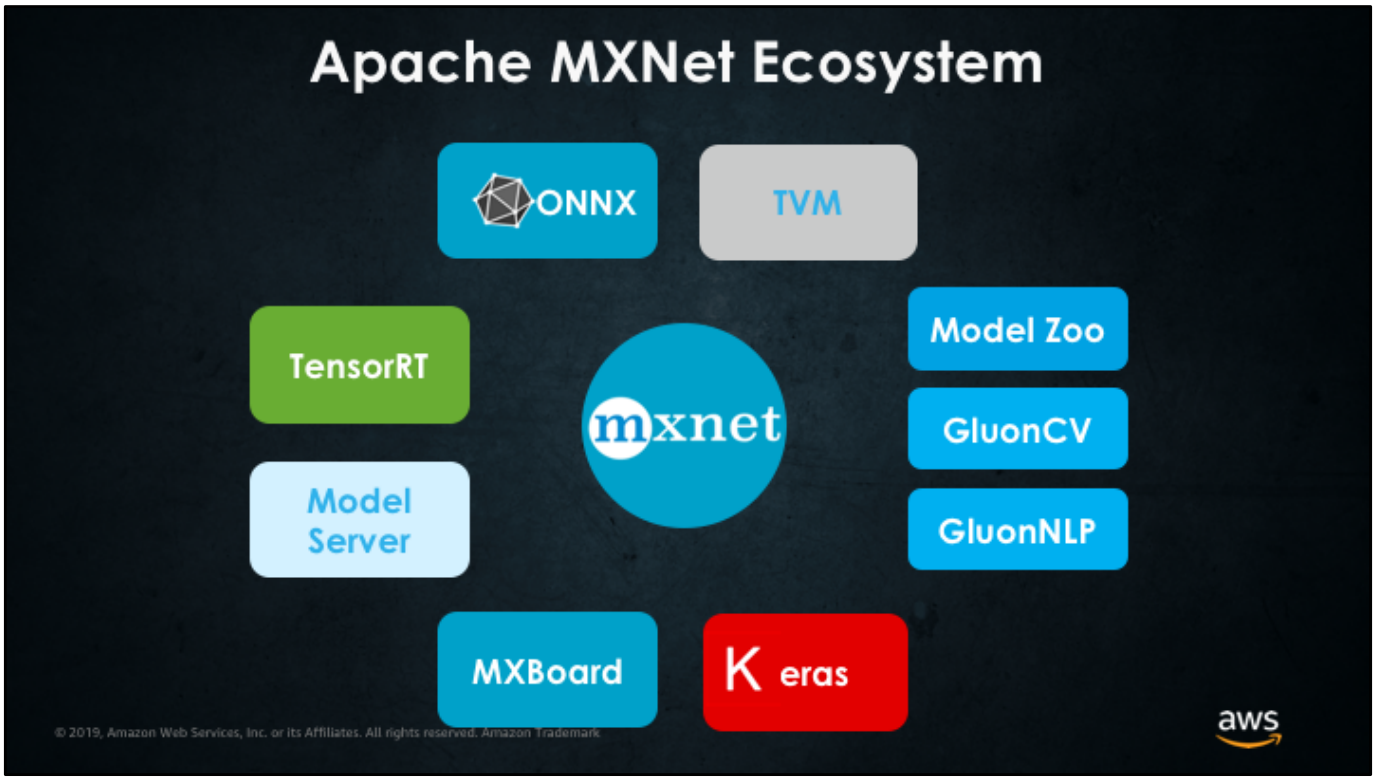
pip install mxnet-cu92

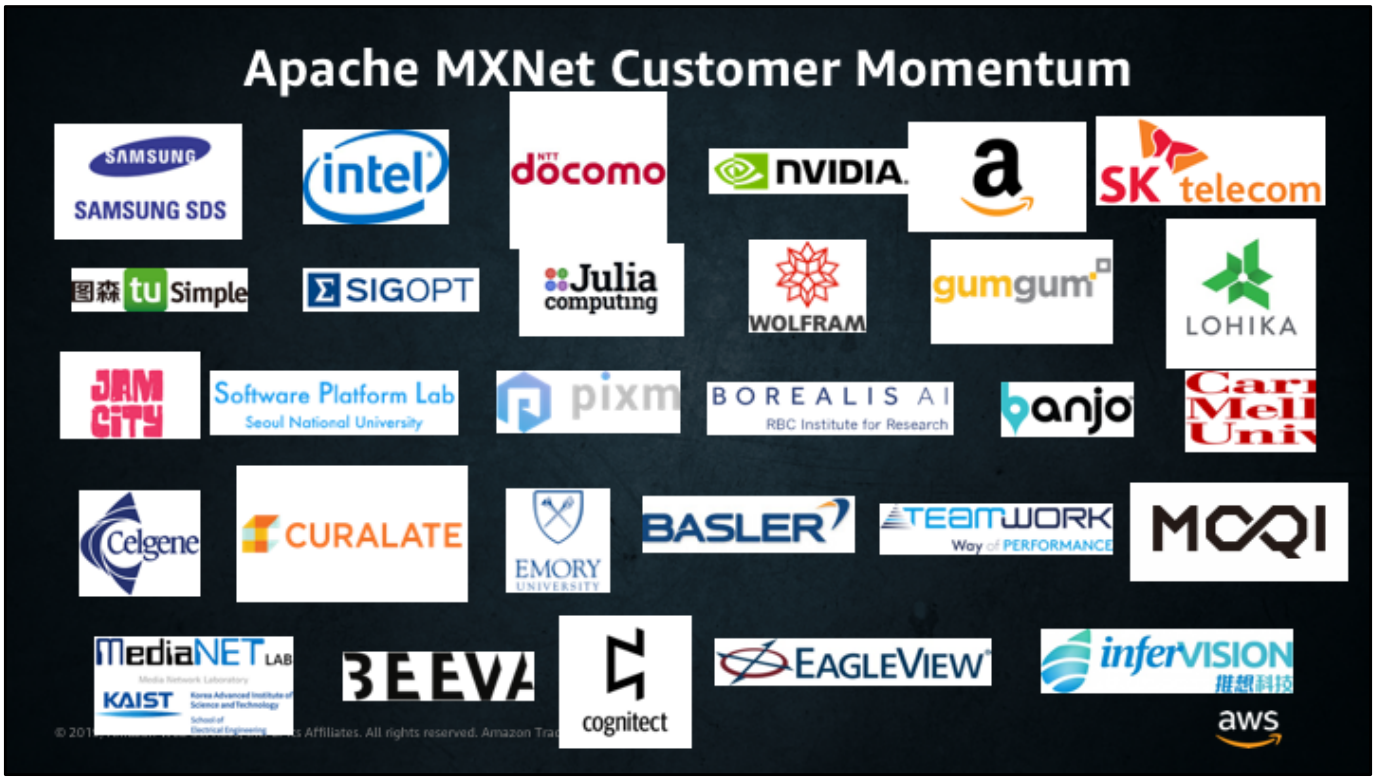
TensorRT

pip install mxnet-tensorrt-cu92

MKL, MKLML & MKLDNN

pip install mxnet-mkl


Apache MXNet Ecosystem
ONNX
TVM
TensorRT
mxnet
Model Zoo
GluonCV
GluonNLP
Model Server
MXBoard
K eras
© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved. Amazon Trademark
aws


Apache MXNet Customer Momentum
SAMSUNG SDS
intel
docomo
NVIDIA
SK telecom
SIGOPT
Julia computing
WOLFRAM
gumgum
LOHIKA
Software Platform Lab
Seoul National University
pixm
BOREALIS AI
RBC Institute for Research
anjo
Celgene
CURALATE
EMORY UNIVERSITY
BASLER
TEAMWORK
Way of PERFORMANCE
MOQI
MediaNET LAB
KAIST
BEEVA
cognitect
EAGLEVIEW
inferVISION
aws

Amazon SageMaker

A fully-managed platform
that provides a quick and easy way to
get models from idea to production.

https://aws.amazon.com/sagemaker/

Amazon SageMaker Workflow

Building | **Training** | **Hosting**

Amazon's fast, scalable algorithms

Distributed TensorFlow, Apache MXNet, Chainer, PyTorch

Bring your own algorithm

Hyperparameter Tuning

aws

Convolution to the rescue!

What we humans see

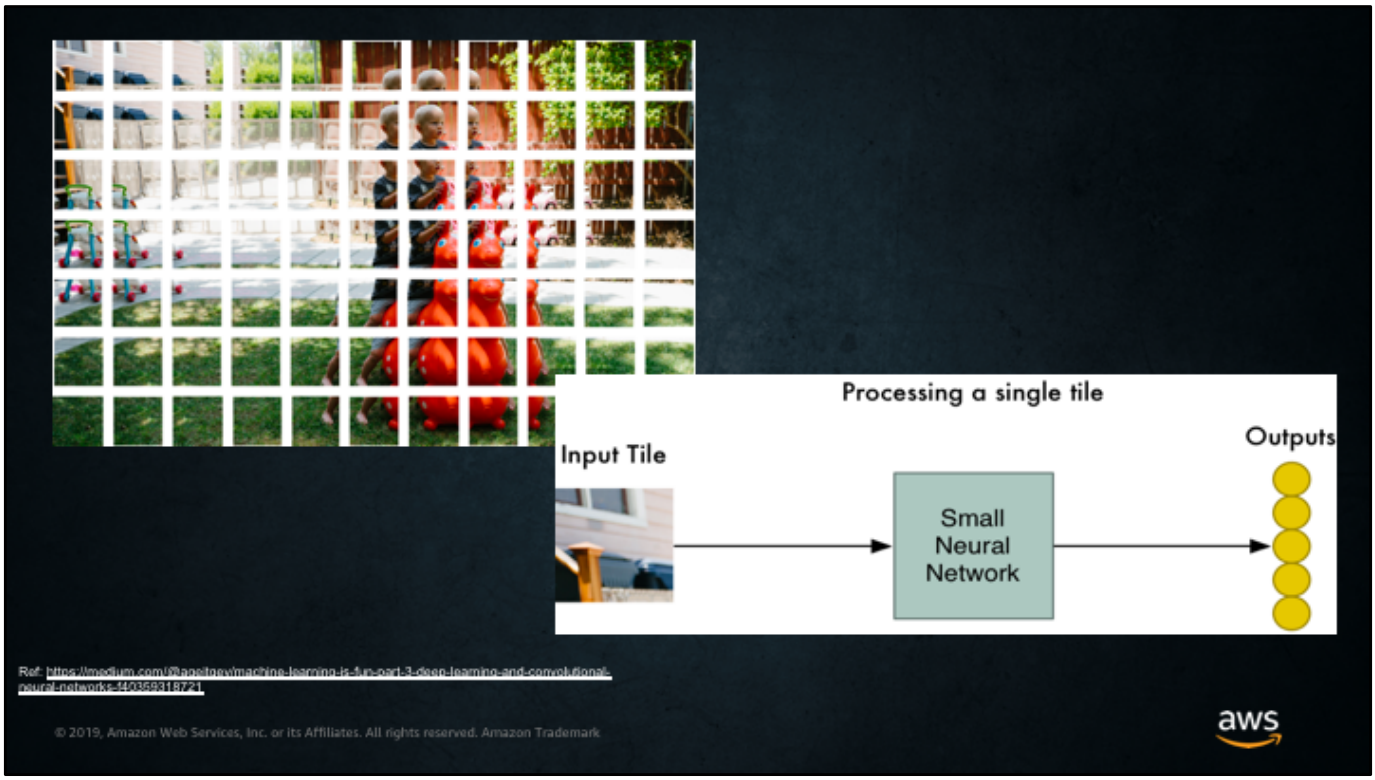
- The ground is covered in grass and concrete
- There is a child
- The child is sitting on a bouncy horse
- The bouncy horse is on top of the grass

Ref: https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721

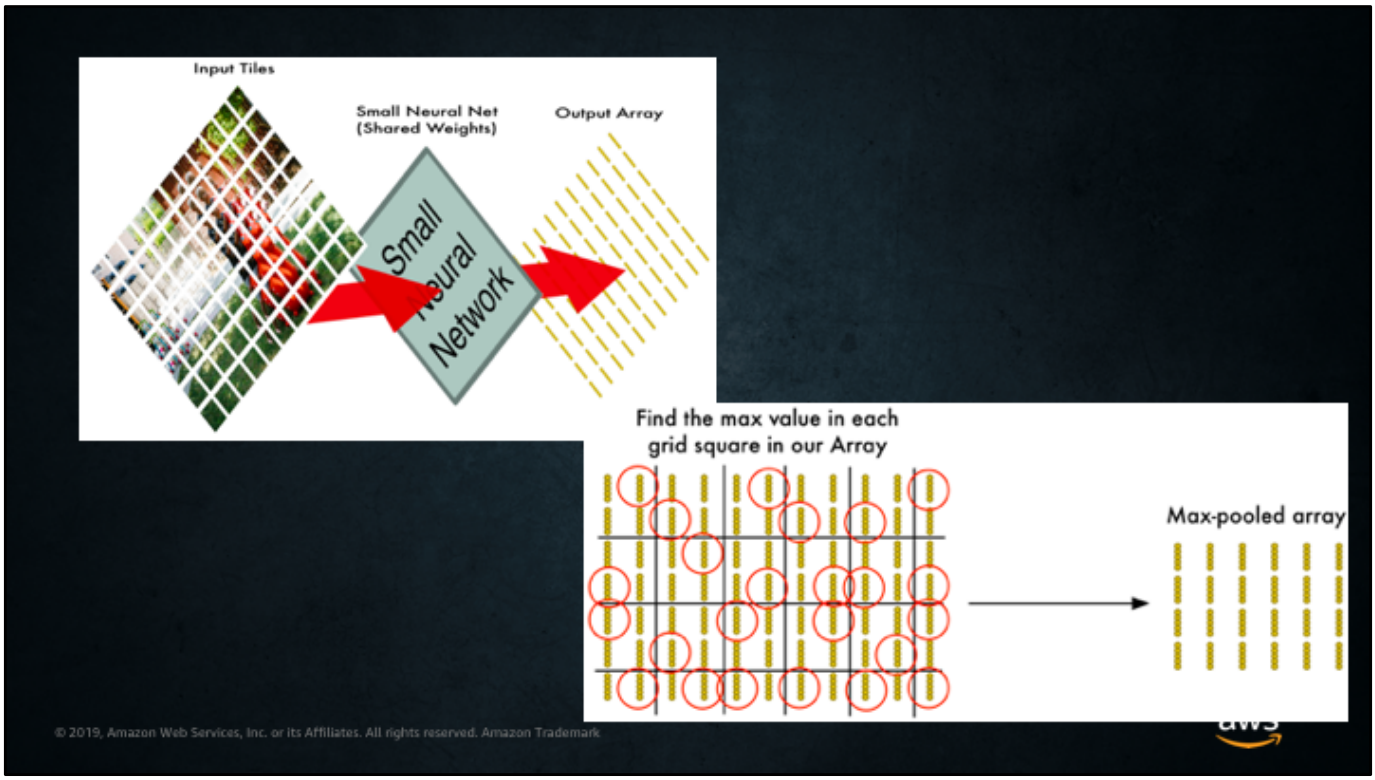
aws

Here's where convolution helps.

When we look at this picture, our brain can automatically identify the child and bouncy horse. How can we get the network to identify this?

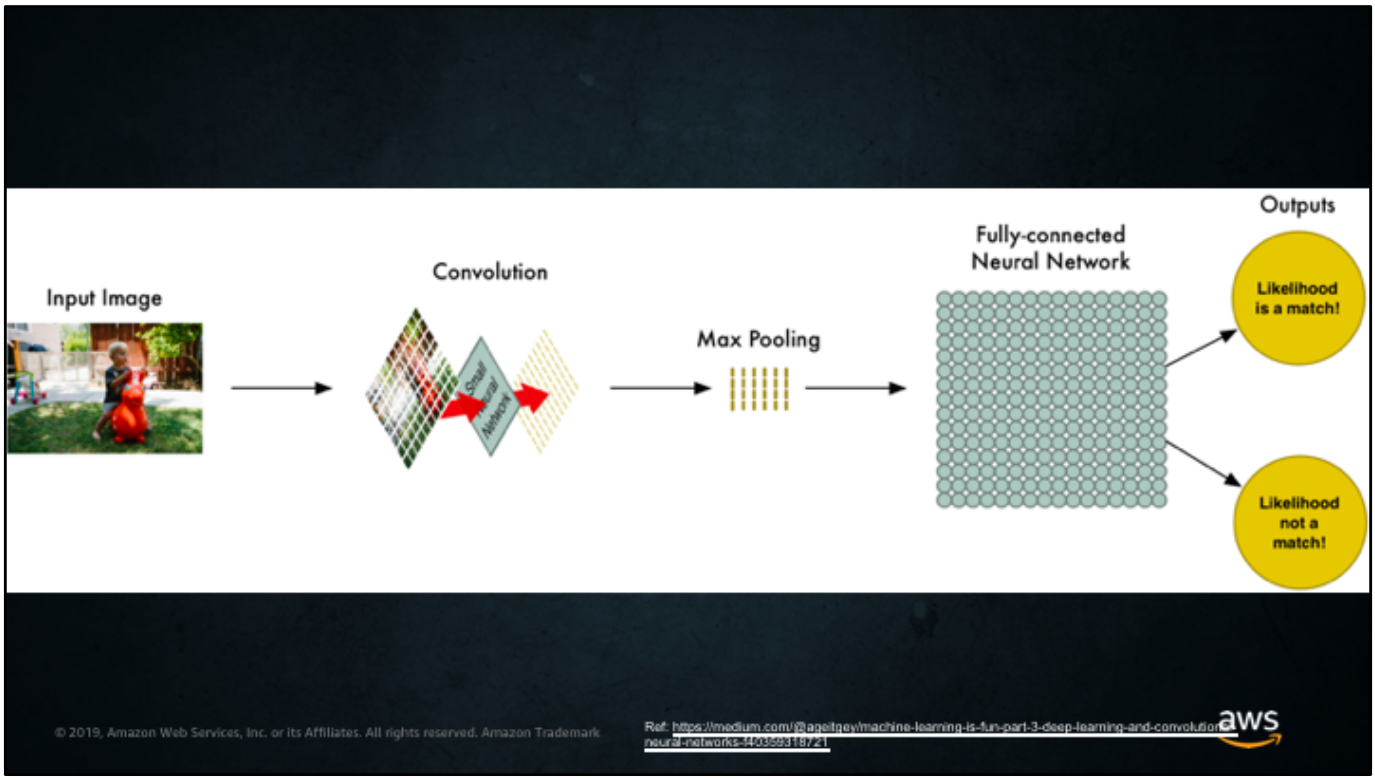


We first divide the image into tiles, pass each tile to a network (each tile is processed by the same network with the same weights) to give some numbers as output.



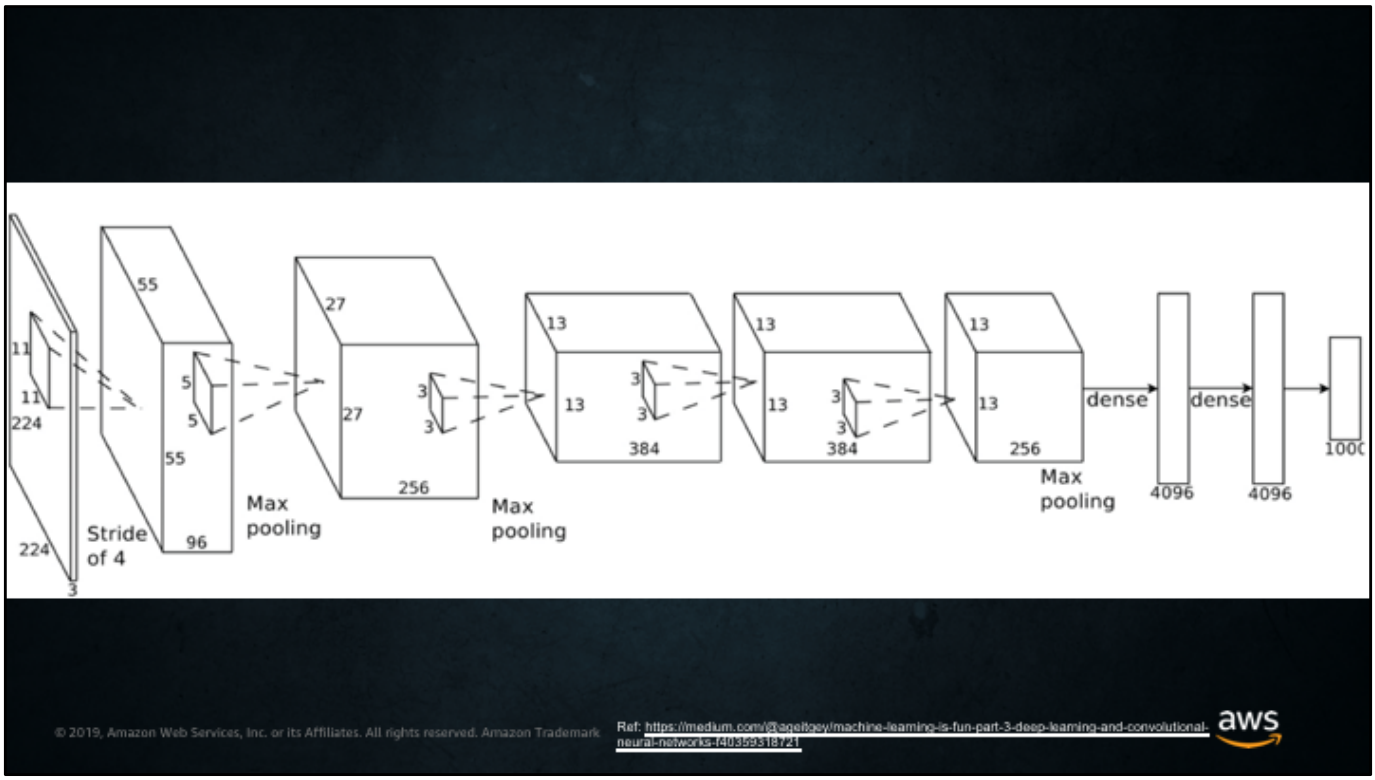
We repeat this process for every tile and get an output, which is still huge (even though it may be smaller in dimension than the original Image).

So we make use of a process called Pooling to downscale, in which a representative is picked from a group. We could either pick a max value, min value or average value (Max pooling, Min pooling, Average pooling).



We can repeat convolution+pooling steps any number of times, learning new features from data at every step.
Finally we add full connected layers before we get a classification output.

This is what a CNN is.



Ref: https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721

In papers, you might come across something like this to represent CNNs.