POLITECNICO DI MILANO

Computing Infrastructures

AI and Platforms: Machine Learing-is-a-service

The topics of the course: what are we going to see today?

A. HW Infrastructures:
- **System-level**: Computing Infrastructures and Data Center Architectures, Rack/Structure;
- **Node-level**: Server (computation, HW accelerators), Storage (Type, technology), Networking (architecture and technology);
- **Building-level**: Cooling systems, power supply, failure recovery

B. SW Infrastructures:
- Virtualization: Process/System VM, Virtualization Mechanisms (Hypervisor, Para/Full virtualization)
- Computing Architectures: Cloud Computing (types, characteristics), Edge/Fog Computing, X-as-a service
- Machine and deep learning-as-a-service

C. Methods:
- **Reliability and availability of datacenters** (definition, fundamental laws, RBDs)
- **Disk performance** (Type, Performance, RAID)
- **Scalability and performance of datacenters** (definitions, fundamental laws, queuing network theory)

2023-2024 ...

Nassim Taleb “The Black Swan” (2007)

«History and societies do not crawl, they leap: moving from one fracture to another with some vibrations in between.»

Nassim Taleb “The Black Swan” (2007)

The Black Swan

When? **Where?** **How?**

The Black Swan

Unexpected and unpredictable	Significant effects	Latent precursors
When?	**Where?**	**How?**

The Black Swan

Unexpected and unpredictable

Significant effects

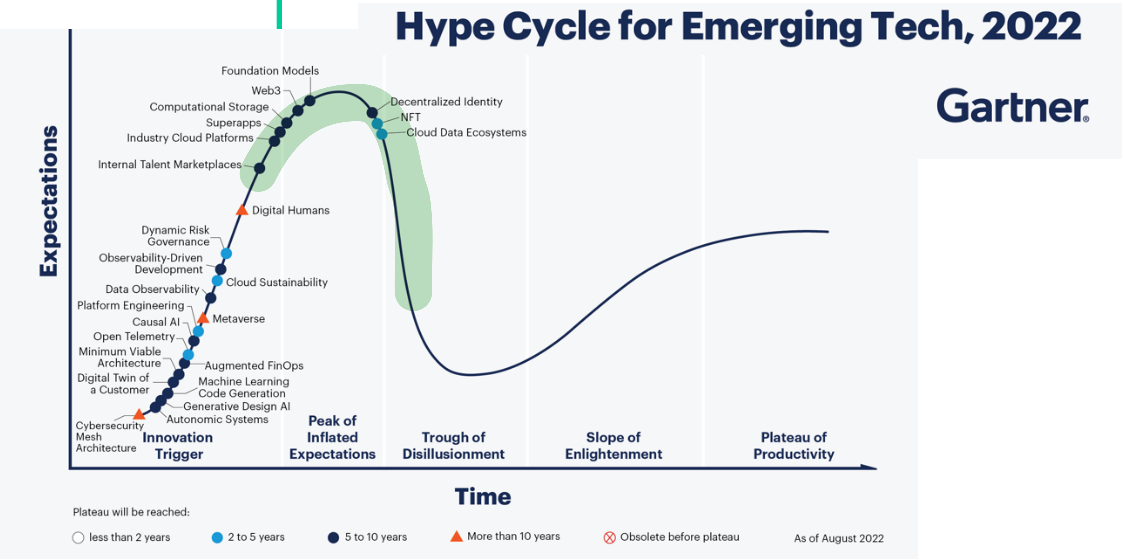
Latent precursors

The Black Swan

Unexpected and unpredictable

Significant effects

Latent precursors

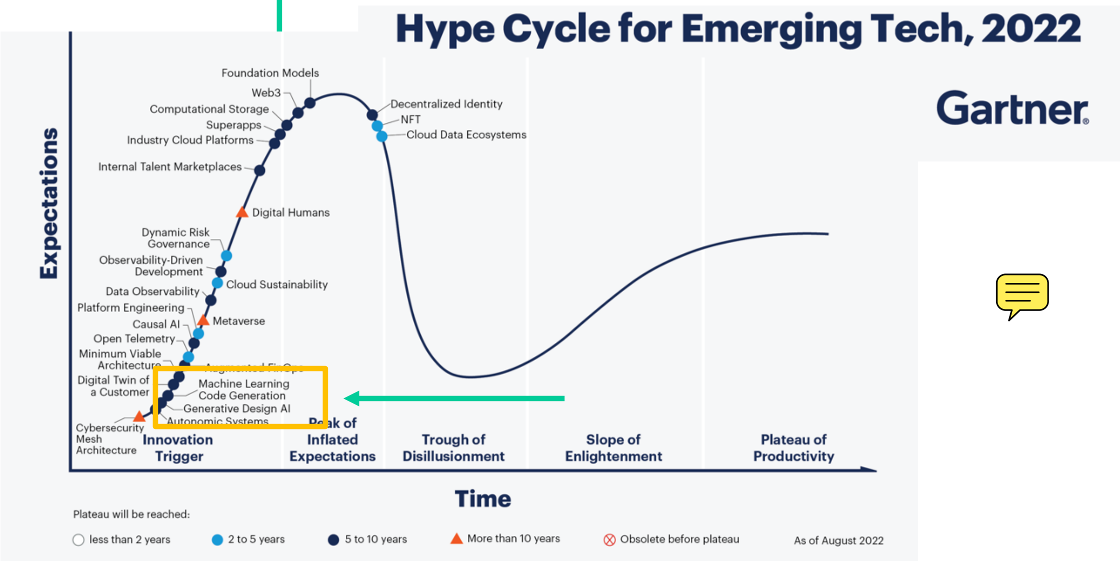


The Black Swan

Unexpected and unpredictable

Significant effects

Latent precursors

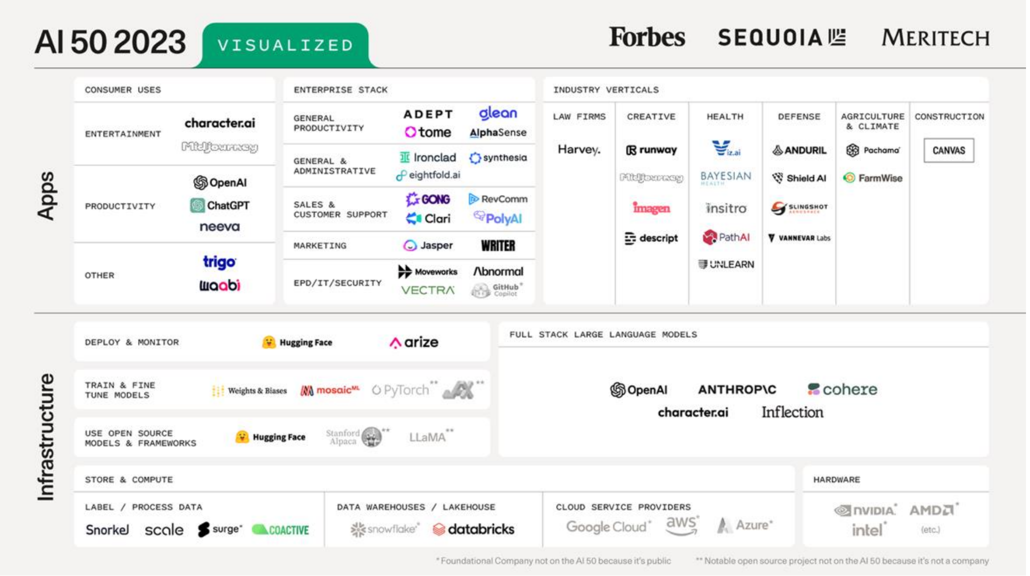


The Black Swan

Unexpected and unpredictable

Significant effects

Latent precursors



Source, Sequoia Capital Report 2023, Generative AI

The Black Swan

Unexpected and unpredictable

Significant effects

Latent precursors

The Guardian, 08/09/2020

Support the Guardian

Available for everyone, funded by readers

Contribute → Subscribe →

Sign in

The Guardian

News Opinion Sport Culture Lifestyle

The Guardian view Columnists Cartoons Opinion videos Letters

Opinion Artificial intelligence (AI)

This article is more than **5 months old**

A robot wrote this entire article. Are you scared yet, human?

GPT-3

We asked GPT-3, OpenAI's powerful new language generator, to write an essay for us from scratch. The assignment? To convince us robots come in peace

The Black Swan

When?

Unexpected and unpredictable

Where?

Significant effects

How?

Latent precursors

The Guardian, 08/09/2020


Support the Guardian

Available for everyone, funded by readers

Contribute → Subscribe →

Sign in

The Guardian

News Opinion Sport Culture Lifestyle

The Guardian view Columnists Cartoons Opinion videos Letters

Opinion Artificial intelligence (AI)

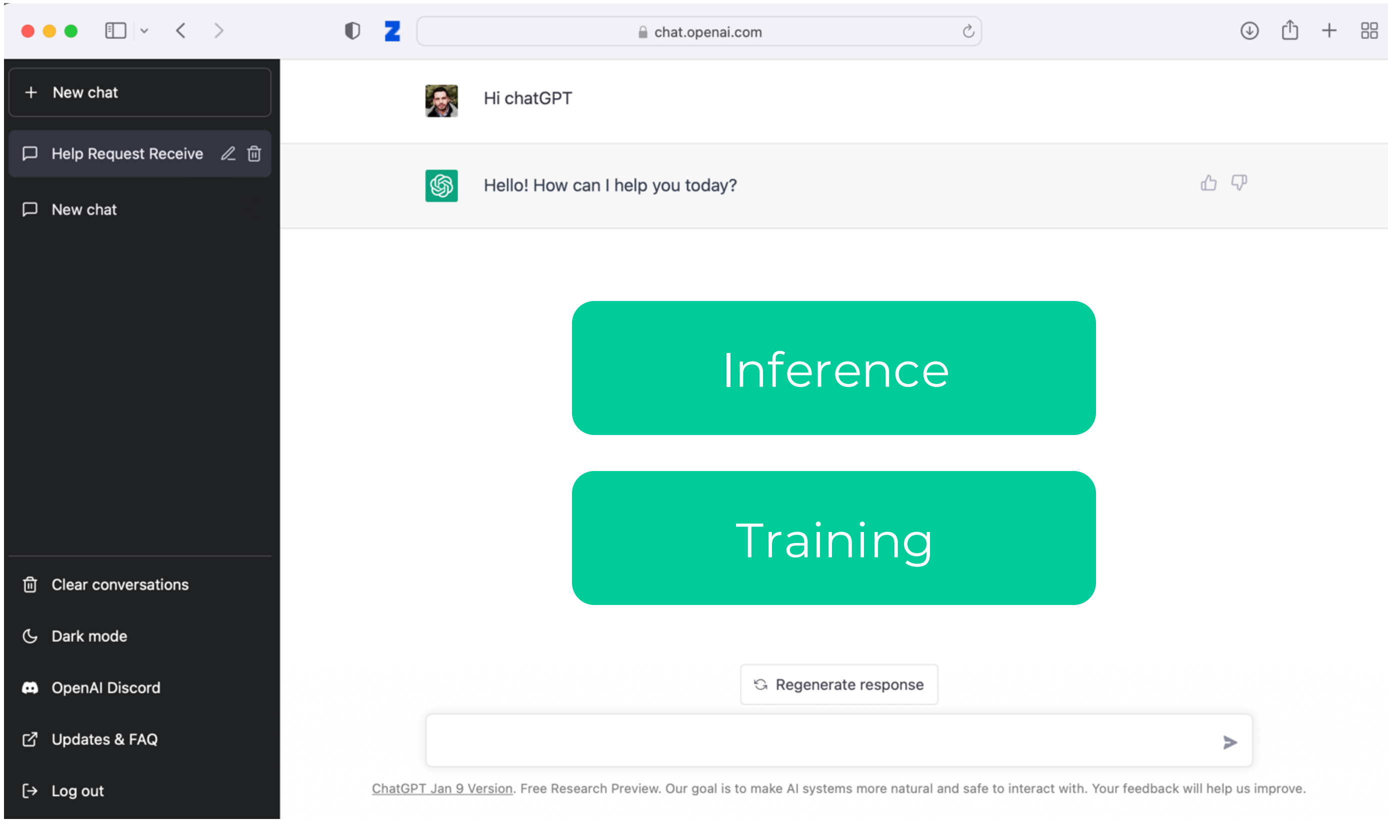
This article is more than **5 months old**

A robot wrote this entire article. Are you scared yet, human?

GPT-3

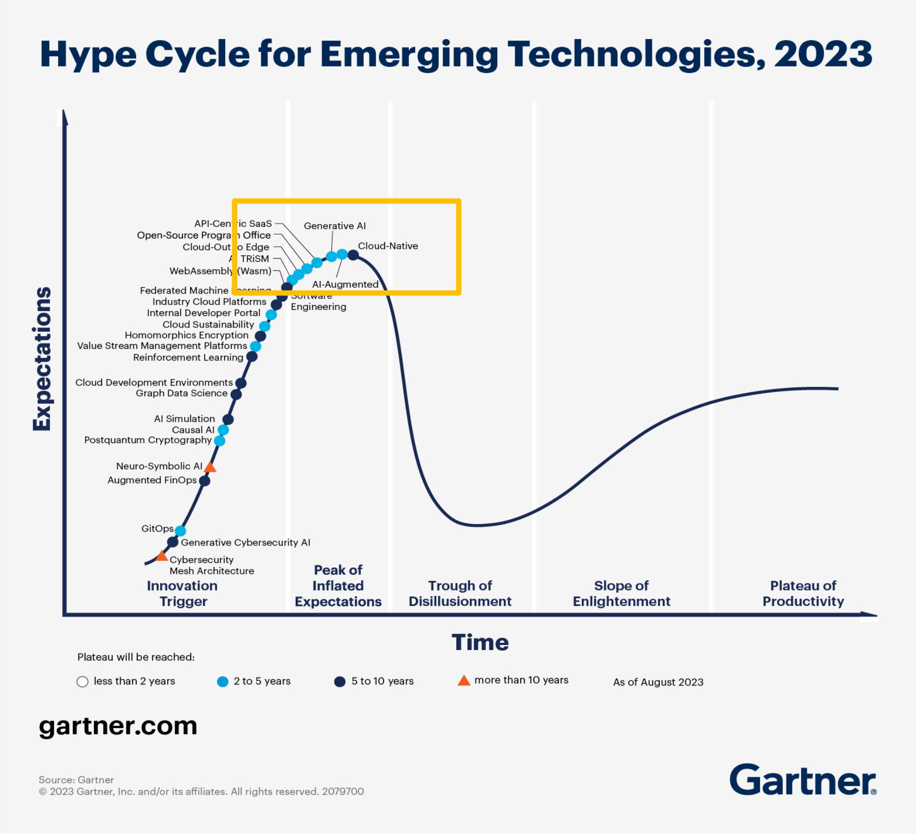
We asked GPT-3, OpenAI's powerful new language generator, to write an essay for us from scratch. The assignment? To convince us robots come in peace

The most important question … why?

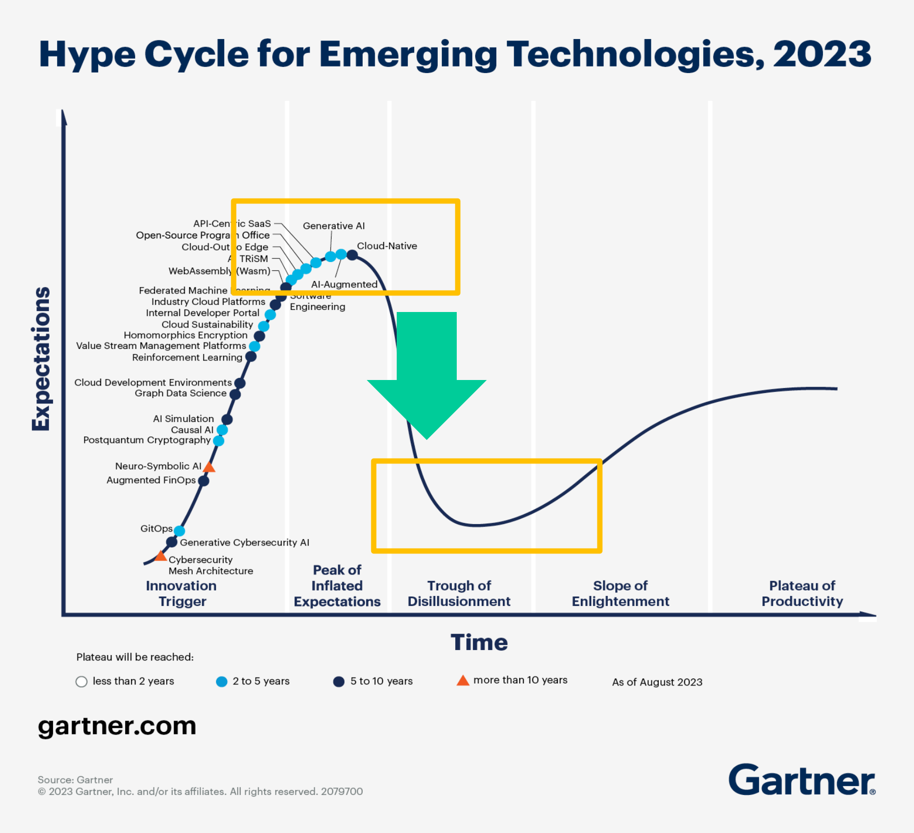


What to expect for 2024?

What to expect for 2024?



Guide the transition…



Intelligence of the 60s..

Does GPT-4 Pass the Turing Test?

Cameron Jones and **Benjamin Bergen**
UC San Diego,
9500 Gilman Dr, San Diego, CA
cameron@ucsd.edu

Abstract

We evaluated GPT-4 in a public online Turing Test. The best-performing GPT-4 prompt passed in 41% of games, outperforming baselines set by ELIZA (27%) and GPT-3.5 (14%), but falling short of chance and the baseline set by human participants (63%). Participants' decisions were based mainly on linguistic style (35%) and socio-emotional traits (27%), supporting the idea that intelligence is not sufficient to pass the Turing Test. Participants' demographics, including education and familiarity with LLMs, did not predict detection rate, suggesting that even those who understand systems deeply and interact with them frequently may be susceptible to deception. Despite known limitations as a test of intelligence, we argue that the Turing Test continues to be relevant as an assessment of naturalistic communication and deception. AI models with the ability to masquerade as humans could have widespread societal consequences, and we analyse the effectiveness of different strategies and criteria for judging humanlikeness.

Keywords: Turing Test, Large Language Models, GPT-4, interactive evaluation

1 Introduction

Turing (1950) devised the *Imitation Game* as an indirect way of asking the question: "Can machines think?". In the original formulation of the game, two witnesses—one human and one artificial—attempt to convince an interrogator that they are human via a text-only interface. Turing thought that the open-ended nature of the game—in which interrogators could ask about anything from romantic love to mathematics—constituted a broad and ambitious test of intelligence. The Turing Test, as it has come to be known, has since inspired a lively debate about what (if anything) it can be said to measure, and what kind of systems might be capable of passing (French, 2000).

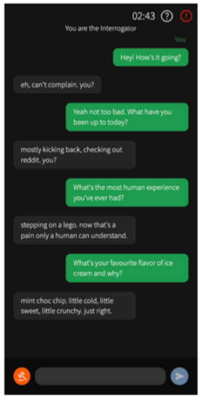


Figure 1: Chat interface for the Turing Test experiment featuring an example conversation between a human Interrogator (in green) and GPT-4.

Large Language Models (LLMs) such as GPT-4 (OpenAI, 2023) seem well designed for Turing's game. They produce fluent naturalistic text and are near parity with humans on a variety of language-based tasks (Chang and Bergen, 2023; Wang et al., 2019). Indeed, there has been widespread public speculation that GPT-4 would pass a Turing Test (Bievere, 2023) or has implicitly done so already (James, 2023). Here we address this question empirically by comparing GPT-4 to humans and other language agents in an online public Turing Test.

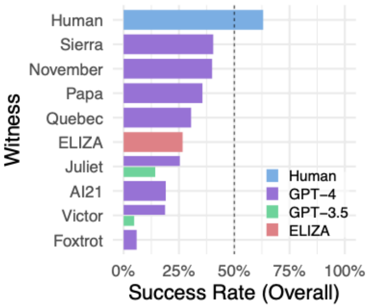


Figure 4: Overall Turing Test Success Rate (SR) for a subset of witnesses. Human witnesses perform best, with 63% SR. GPT-4 performance varies greatly by prompt from 41% (Sierra, best) to 6% (Foxtrot, worst). ELIZA achieves 27%, outperforming the best GPT-3.5 prompt (Juliet, 14%), GPT-4 performance with that prompt (26%), and a baseline prompt from Jannai et al. (2023), AI21 (19%).

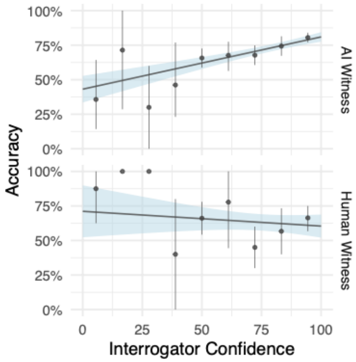
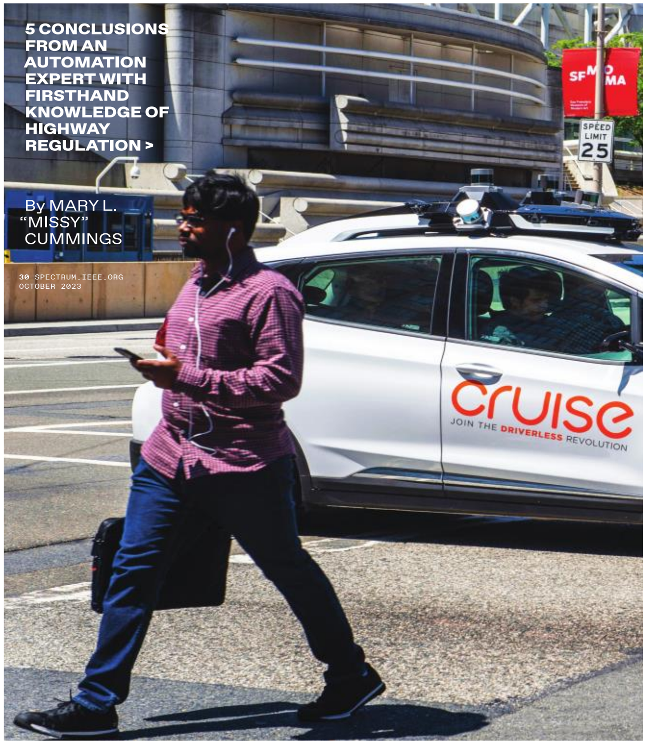


Figure 5: Interrogator confidence vs accuracy and witness type. Against AI witnesses, interrogators were well calibrated—that is, their confidence was positively correlated with accuracy. However, there was no relationship between confidence and accuracy for guesses about human witnesses.

$Turing^{-1}()$

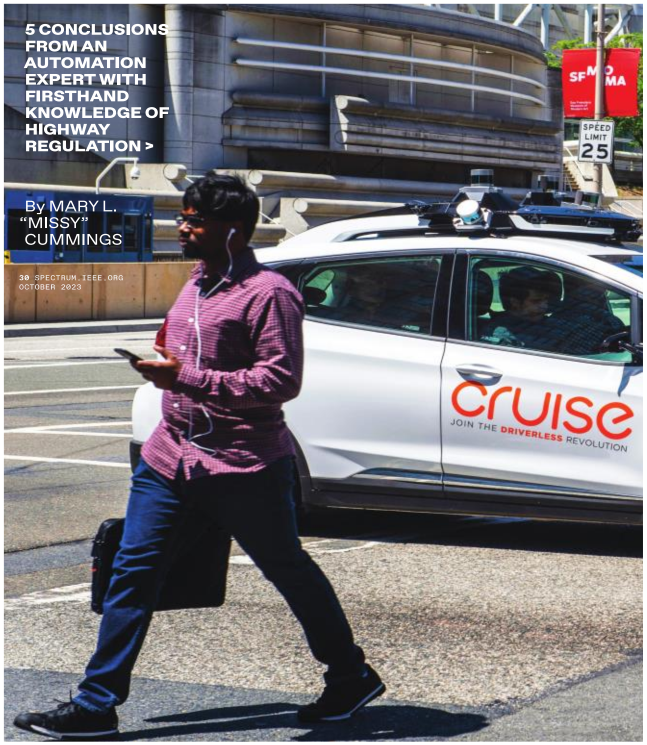
The (not) intelligent machine …



OCTOBER 2023 SPECTRUM.IEEE.ORG

- "A **large language model** guesses which words and phrases are coming next by consulting an archive assembled during training from preexisting data."
- "A **self-driving module** interprets the scene and decides how to get around obstacles by making similar guesses, based on a database of labeled images"
- "But **not every possibility can be modeled**, and so the myriad failure modes are extremely hard to predict."

The (not) intelligent machine …

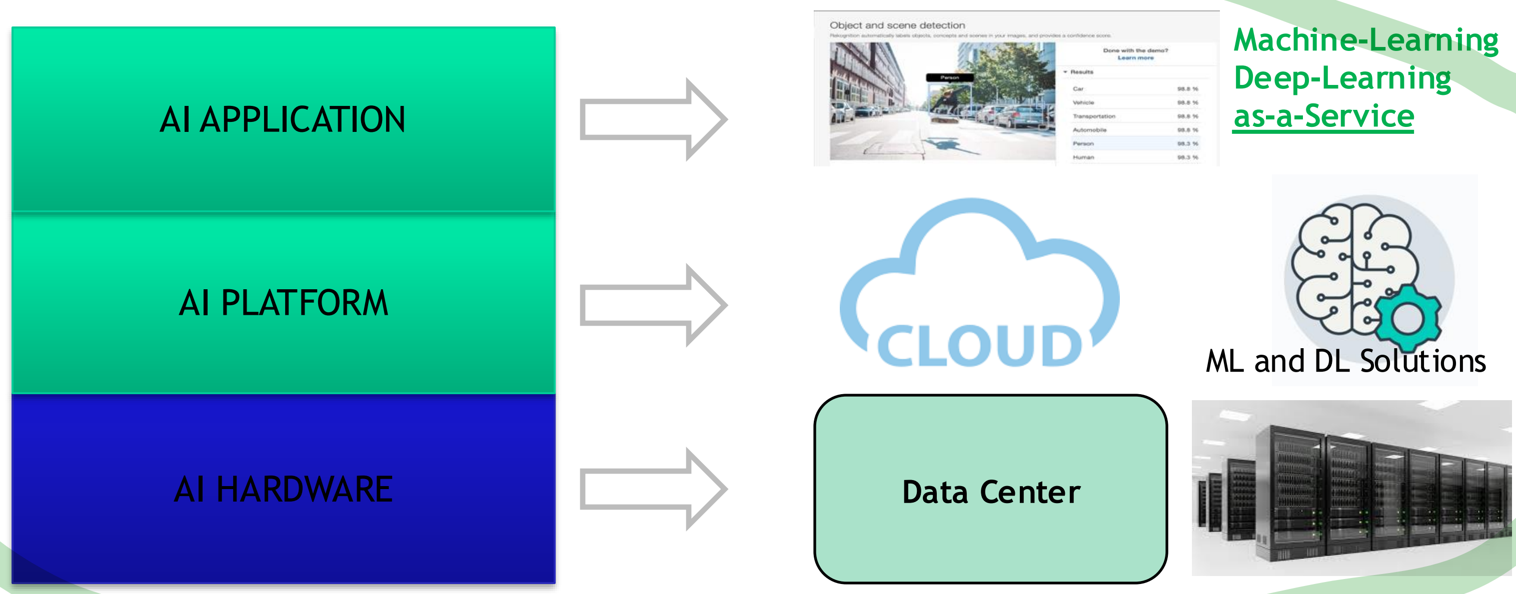


OCTOBER 2023 SPECTRUM.IEEE.ORG

- “A **large language model** guesses which words and phrases are coming next by consulting an archive assembled during training from preexisting data.”
- “A **self-driving module** interprets the scene and decides how to get around obstacles by making similar guesses, based on a database of labeled images”
- “But **not every possibility can be modeled**, and so the myriad failure modes are extremely hard to predict.”

Cloud and ML/DL as-a-service

AI Hardware: dal datacenter a ML-as-a-service



The IT architecture for ML/DL on Datacenters

Machine Learning Framework

Computing Framework

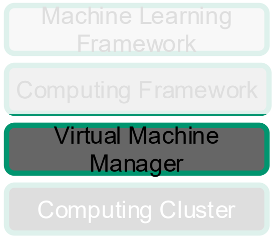
Virtual Machine Manager

Computing Cluster

Computing Cluster

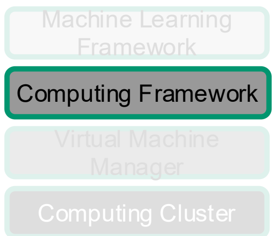
- **Servers** (parallelelization and scalability are key to find the best model as well as HW accelerators):
  - General Purpose CPU
  - GPU (TPU)
- **Storage** (multiple storage systems including distributed/parallel file systems):
  - Direct Attached Storage
  - NAS
  - SAN
- **Network** (applications are generally non-iterative):
  - Ethernet

Virtual Machine Manager



- **Virtualization** is carried out through hypervisor (virtual machines) or containers
- **Resources are increased by adding more virtual machines** (provided that hardware resources are available)
- User can design **personalized software environments** and scale instances to the needs
- **Some examples**: VMware, Xen, KVM, HyperX, Kubernetes

Computing Framework



- **Computing Frameworks are composed by several modules:**
  - Cluster Manager
  - Data Storage
  - Data processing engine
  - Graph computation
  - Programming Languages (Java, Python, Scala, R)
- An **application** (e.g., a big-data application) operating in a cluster is distributed among different computing (virtual) machines

Examples

- **Computing Framework**
  - Hadoop, Spark, Flink
- **Scheduling**
  - Apache Mesos, Hadoop YARN
- **Storage**
  - HDFS, Hbase, Amazon S3, Cassandra, Mongo DB, Spark SQL
- **Data Processing Engine**
  - Spark, Map-Reduce

Machine/Deep Learning Frameworks

Machine Learning Framework
Computing Framework
Virtual Machine Manager
Computing Cluster

- Machine learning frameworks cover a variety of learning methods for classification, regression, clustering, anomaly detection, and data preparation, and it may or may not include neural network methods.
- Deep learning frameworks cover a variety of neural network topologies with many hidden layers.

	Computing Framework	Stand-alone (OS)
Machine Learning	Spark MLLib, BigDL, Mahout	Scikit-learn, Torch, Pandas, Numpy, Matplotlib
Deep Learning	TensorFlowOnSpark, Deeplearning4j, BigDL	Tensorflow, Caffe, Apache MXNet, Keras, Theano, Microsoft CNTK, Pytorch

Exploit GPU access

Generally organized as libraries/shells

Why Machine Learning in the Cloud?

- Cloud comuputing simplifies the access to ML capabilities for
  - **designing a solution** (without requiring a deep knowledge of ML)
  - **setting up a project** (managing demand increases and IT solution)
- Amazon, Microsoft and Google (just to name a few) provide
  - Solutions → **ML Solutions as a service**
  - Platforms → **ML Platforms as a service**
  - Infrastructures → **ML Infrastructures as a service**

to support ML in the Cloud

Machine Learning as a Service

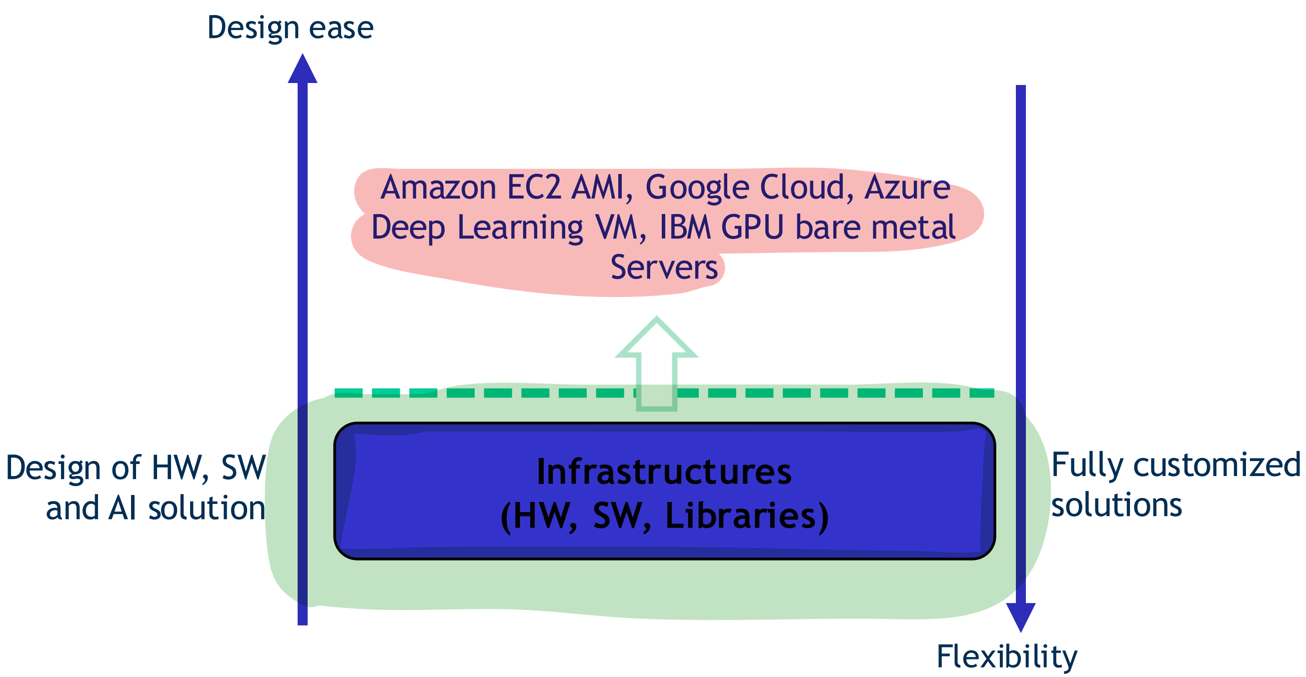
AI and off-the-shelf technological solutions

Solutions

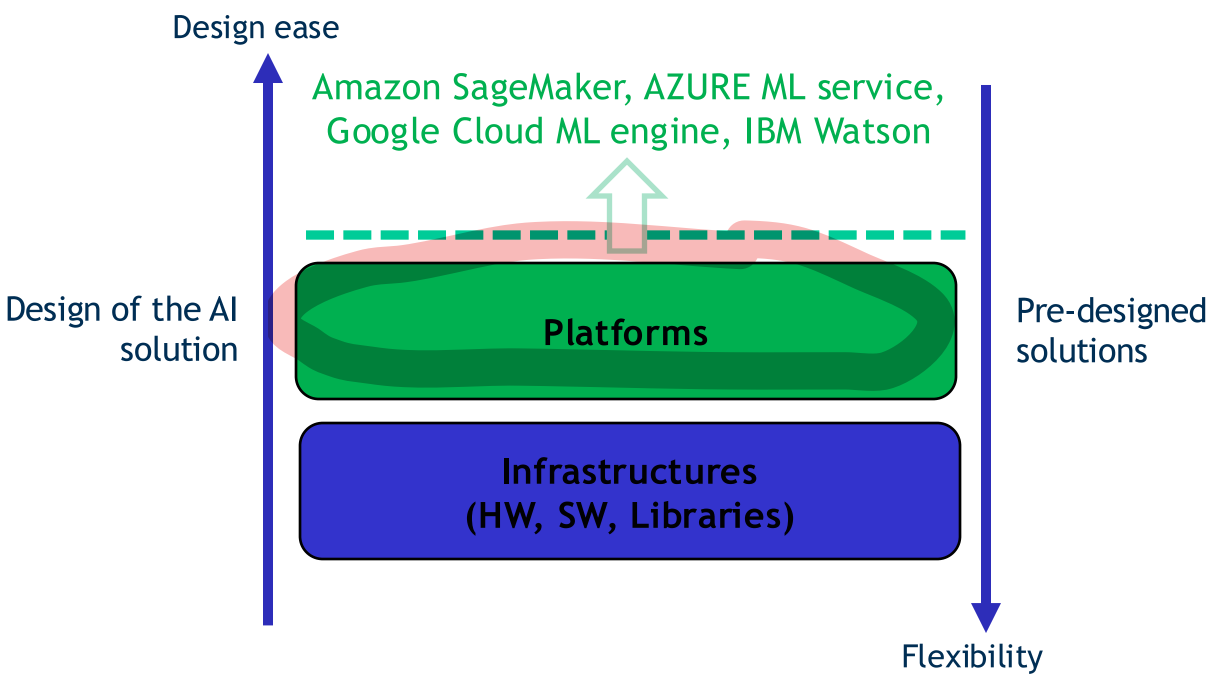
Platforms

Infrastructures
(HW, SW, Libraries)

Machine Learning Infrastructure as a Service

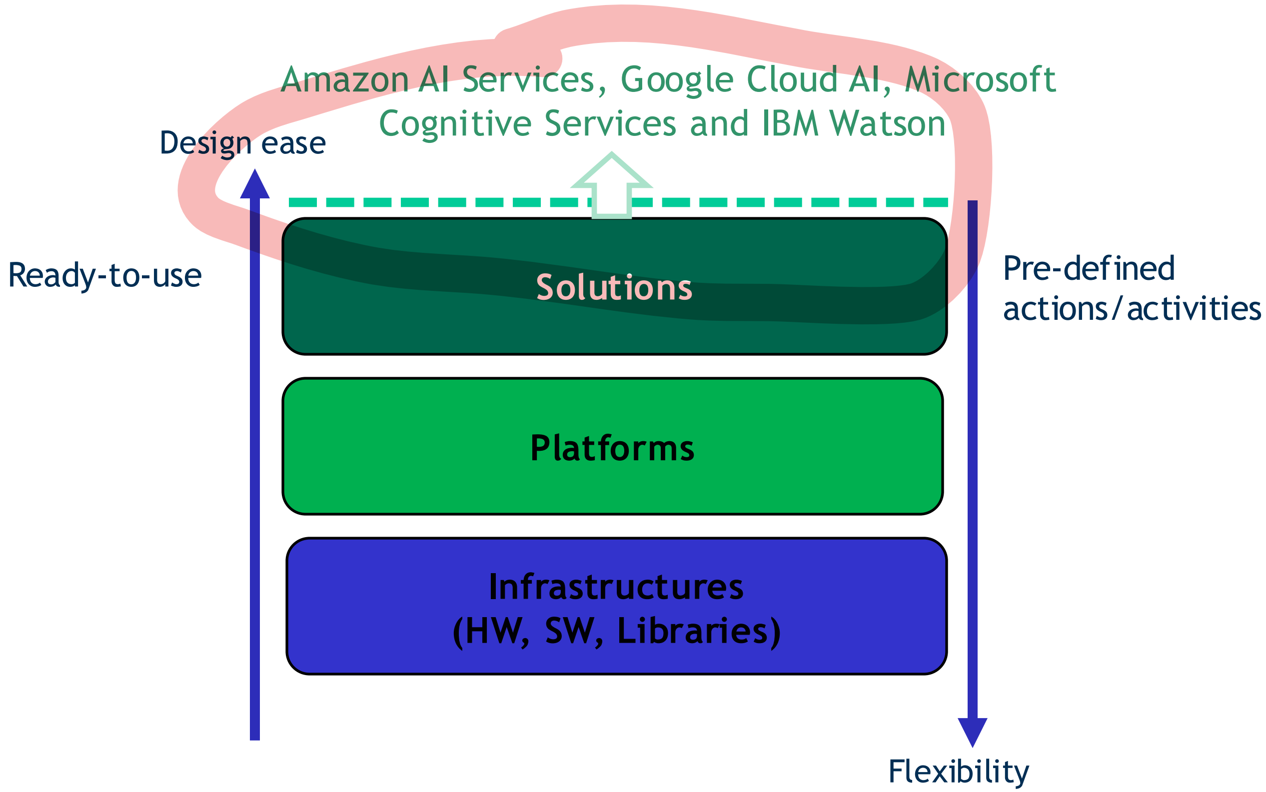


Machine Learning Platform as a Service



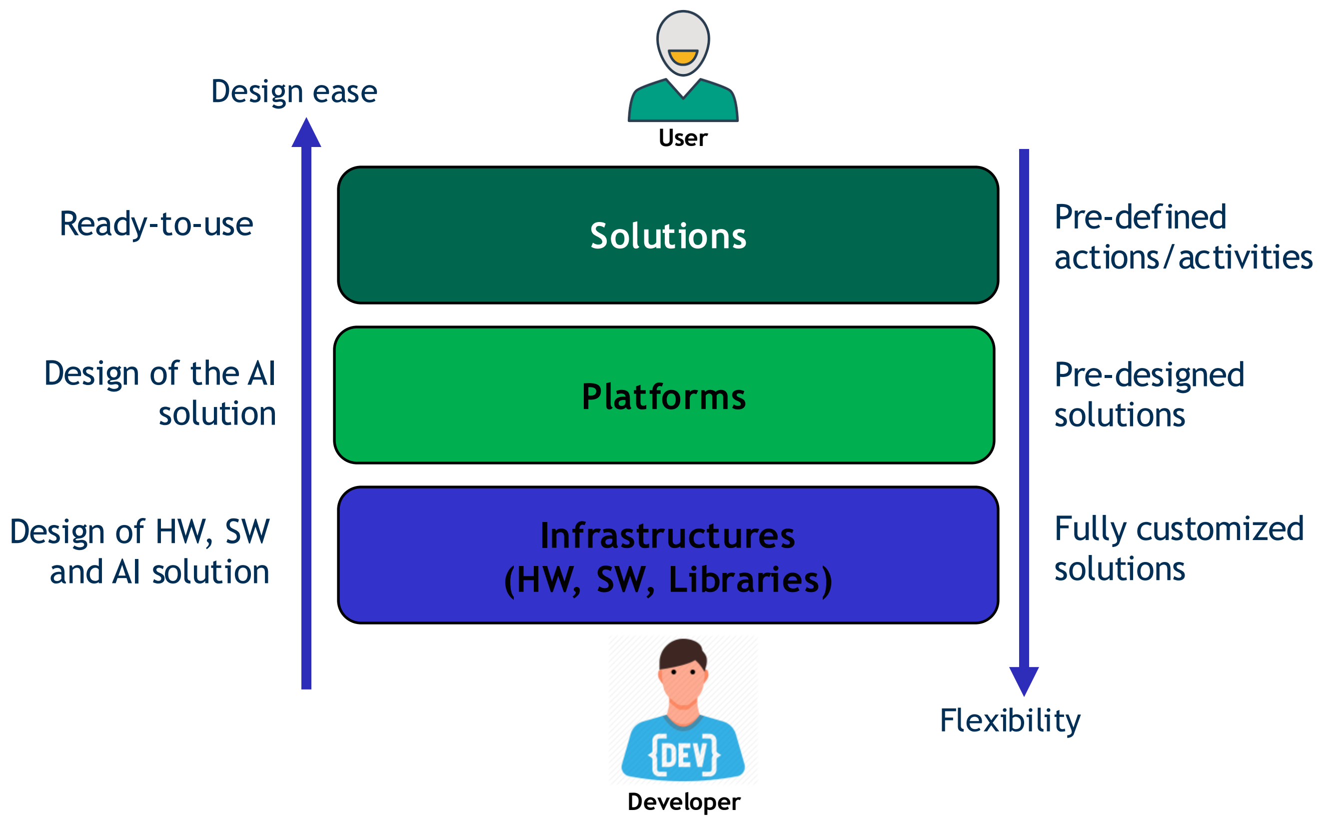
Provide pre-configured environments used by AI experts to train, tune and host models

ML software as a service



E.g., Amazon Lex, Polly, Rekognition, MS Speech2Text ...

AI and off-the-shelf technological solutions



A real world example: image classification

A real world example: image classification

Solutions

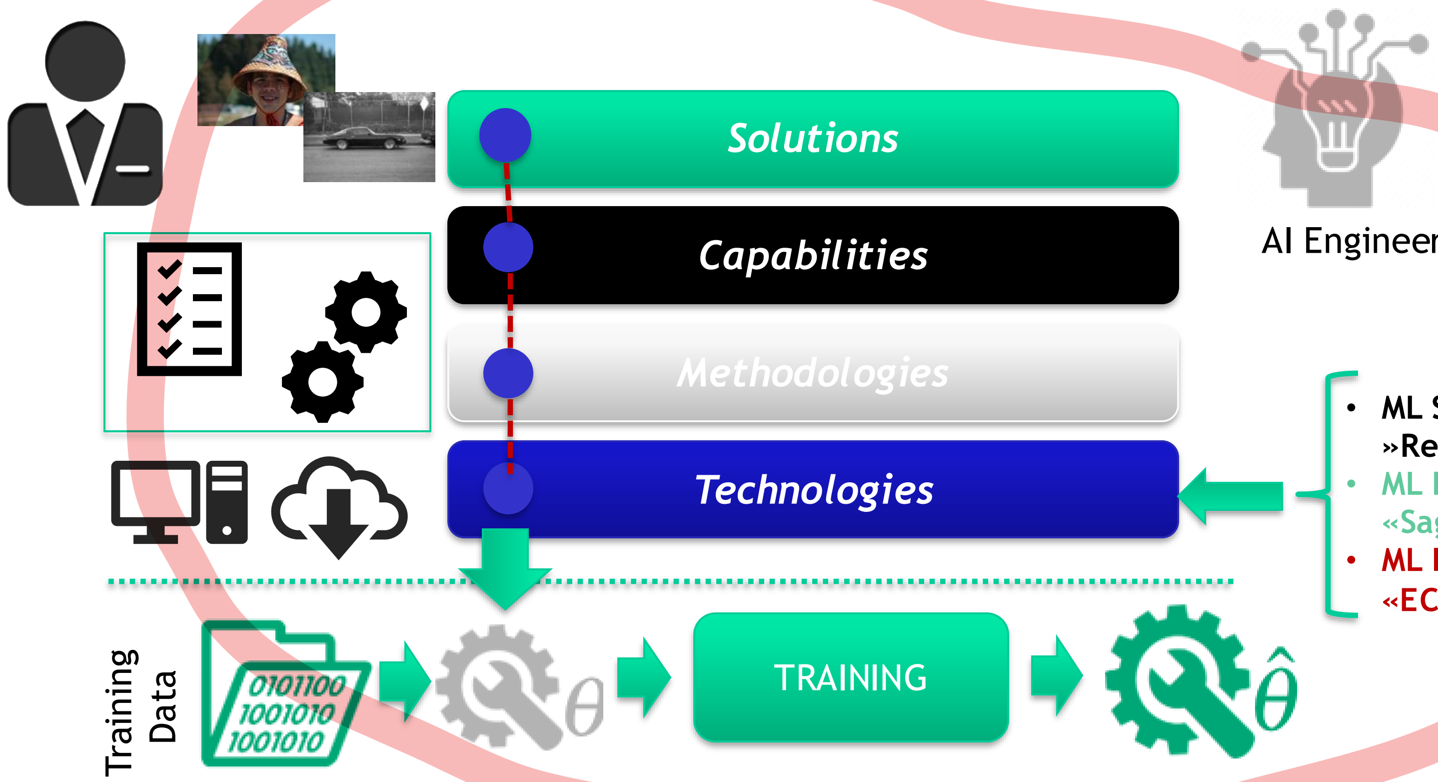
Capabilities

Methodologies

Technologies

AI Engineer

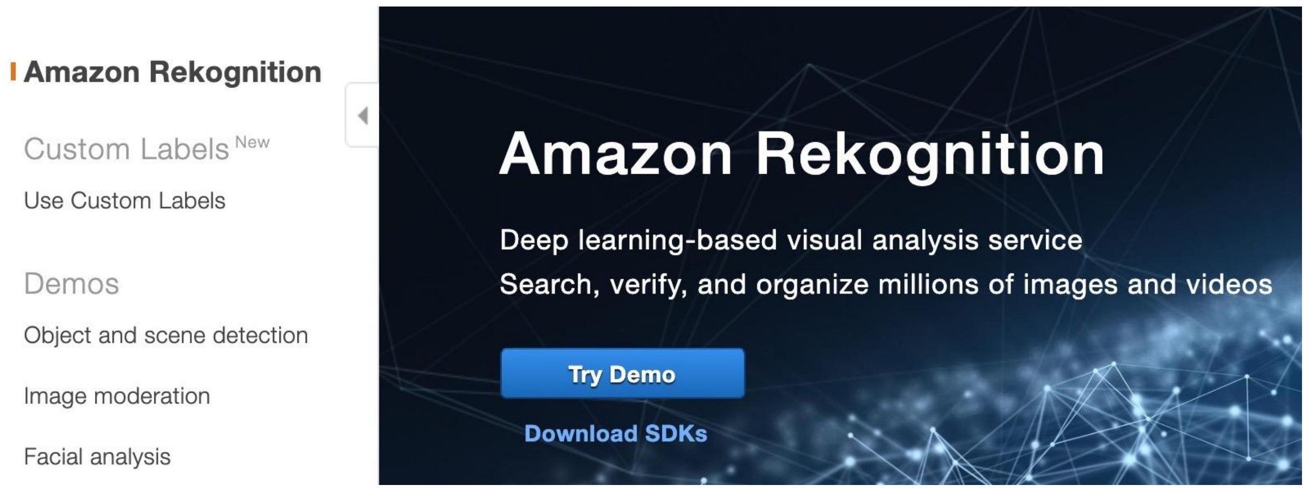
A real world example: image classification on AWS



- ML Solutions as a service »Rekognition»
- ML Platforms as a service «Sagemaker»
- ML Infrastructures as a service «EC2»

ML Solutions as a service: Rekognition

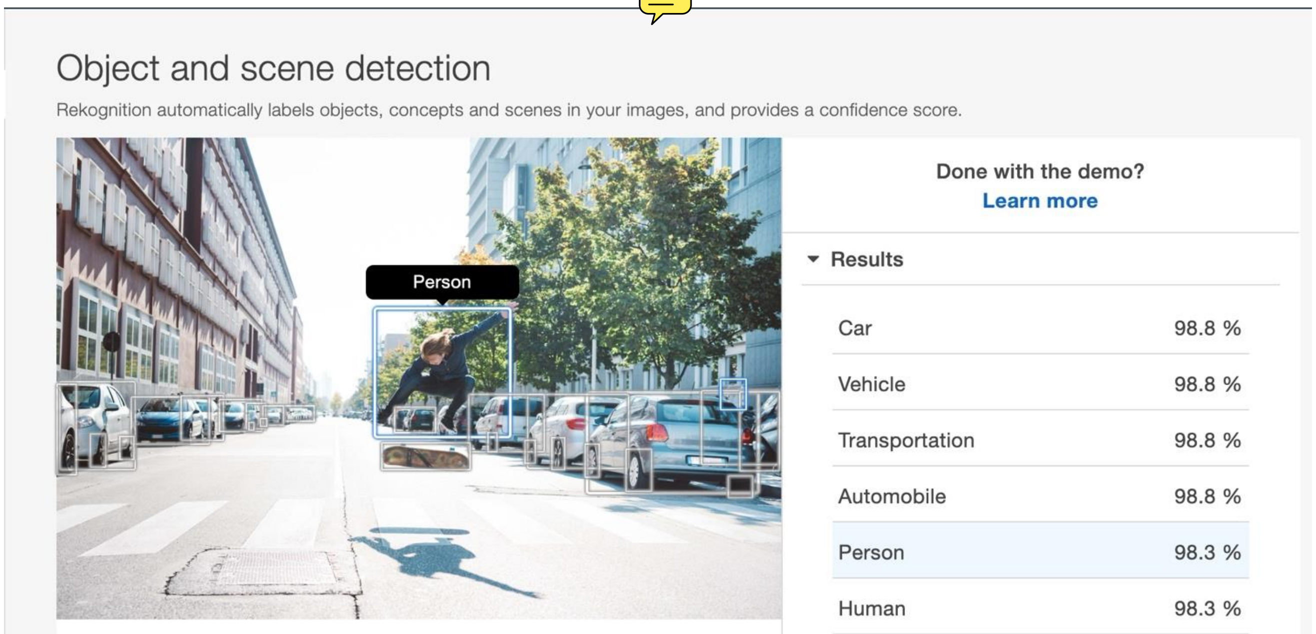
Easy to access…



- Only requires an AWS account
- Free demo available https://aws.amazon.com/rekognition/
- Test on a sample image or upload your own image

ML Solutions as a service: Rekognition

… and easy to use



- Interactive output
- JSON format of the results
- Predefined classes

ML Platforms as a service: Sagemaker

Easy to access

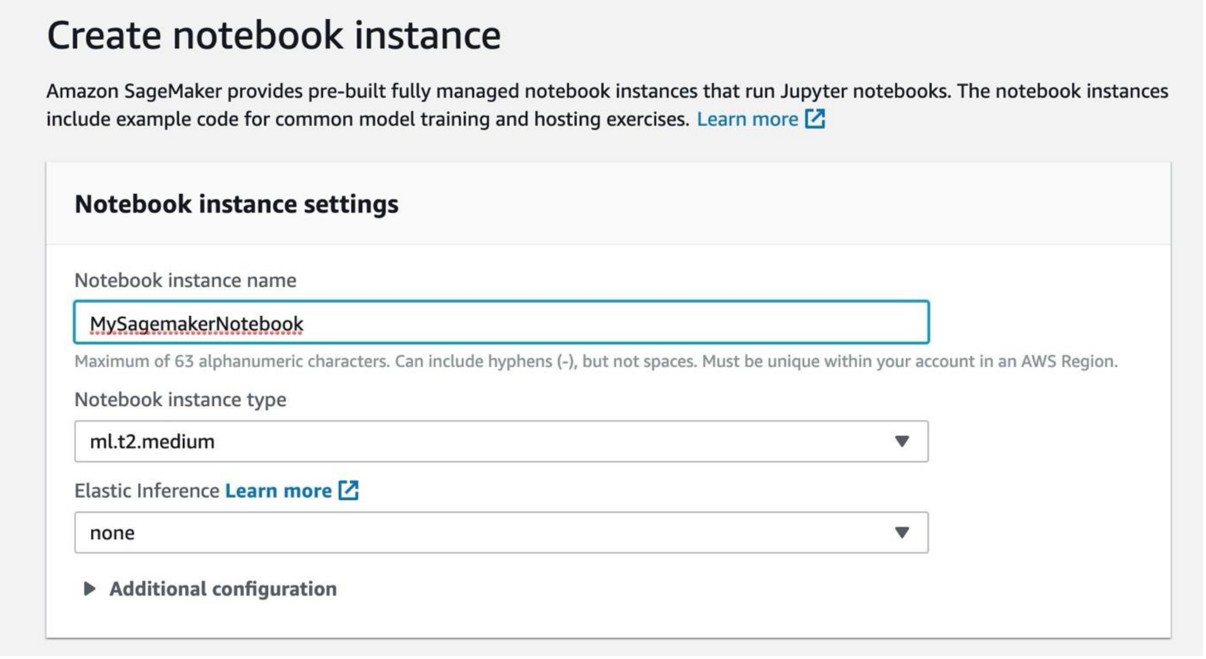
- Requires an AWS account
- Get started at https://aws.amazon.com/sagemaker/
- Running machines **at your expense**
- User-friendly interface

ML Platforms as a service: Sagemaker

1 2 3 4 5 6

Create Jupyter notebook

- Immediate to set up your environment
- Customizable VMs to run your Jupyter Notebook



ML Platforms as a service: Sagemaker

1 > 2 > 3 > 4 > 5 > 6

Define model and parameters

- Use built-in models
- Requires an S3 bucket
- Choose model parameters or automatically tune parameters

Hyperparameters for **ImageClassification** image[1]

```python
s3 = boto3.client('s3')
# create unique job name
job_name_prefix = 'DEMO-imageclassification'
timestamp = time.strftime('-%Y-%m-%d-%H-%M-%S', time.gmtime())
job_name = job_name_prefix + timestamp
training_params = \
{
    # specify the training docker image
    "AlgorithmSpecification": {
        "TrainingImage": training_image,
        "TrainingInputMode": "File"
    },
    "RoleArn": role,
    "OutputDataConfig": {
        "S3OutputPath": 's3://{}/{}/output'.format(bucket, job_name_prefix)
    },
    "ResourceConfig": {
        "InstanceCount": 1,
        "InstanceType": "ml.p2.xlarge",
        "VolumeSizeInGB": 50
    },
    "TrainingJobName": job_name,
    "HyperParameters": {
        "image_shape": image_shape,
        "num_layers": str(num_layers),
        "num_training_samples": str(num_training_samples),
        "num_classes": str(num_classes),
        "mini_batch_size": str(mini_batch_size),
        "epochs": str(epochs),
        "learning_rate": str(learning_rate)
    },
    "StoppingCondition": {
        "MaxRuntimeInSeconds": 360000
```

[1] https://docs.aws.amazon.com/sagemaker/latest/dg/IC-Hyperparameter.html

ML Platforms as a service: Sagemaker

1 2 3 4 5 6

Training job

- Create a training job to fit your data according to the model parameters
- ImageClassification image can use **pre-trained Resnet**[2] **weights** if specified or the full network is trained from scratch

```
In [*]: # create the Amazon SageMaker training job
sagemaker = boto3.client(service_name='sagemaker')
sagemaker.create_training_job(**training_params)

# confirm that the training job has started
status = sagemaker.describe_training_job(TrainingJobName=job_name)['TrainingJobStatus']
print('Training job current status: {}'.format(status))

try:
    # wait for the job to finish and report the ending status
    sagemaker.get_waiter('training_job_completed_or_stopped').wait(TrainingJobName=job_name)
    training_info = sagemaker.describe_training_job(TrainingJobName=job_name)
    status = training_info['TrainingJobStatus']
    print("Training job ended with status: " + status)
except:
    print('Training failed to start')
     # if exception is raised, that means it has failed
    message = sagemaker.describe_training_job(TrainingJobName=job_name)['FailureReason']
    print('Training failed with the following error: {}'.format(message))

Training job current status: InProgress
```

[2] http://openaccess.thecvf.com/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf

ML Platforms as a service: Sagemaker

Deploy endpoint

- Deploy your model as a web service endpoint
- Sagemaker model images support REST API calls
- Ready to be invoked for real-time prediction on your data

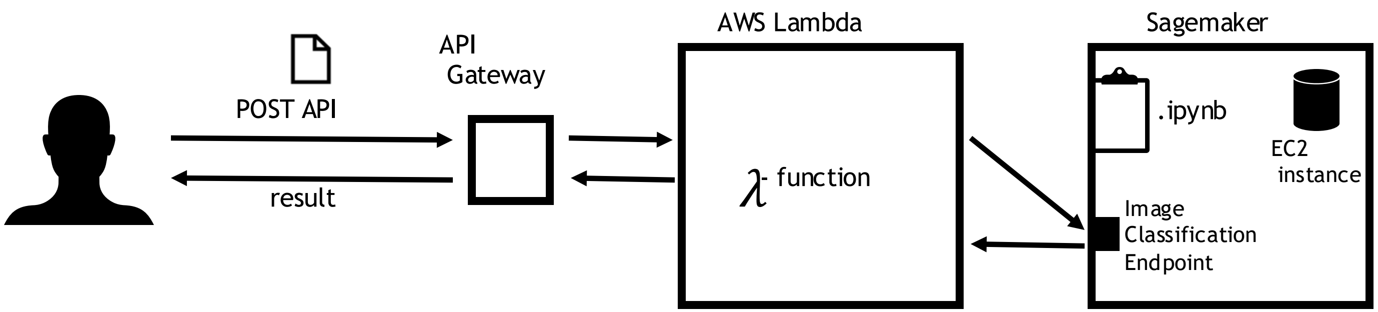
Now the endpoint can be created. It may take sometime to create the endpoint...

```python
# get the status of the endpoint
response = sagemaker.describe_endpoint(EndpointName=endpoint_name)
status = response['EndpointStatus']
print('EndpointStatus = {}'.format(status))


# wait until the status has changed
sagemaker.get_waiter('endpoint_in_service').wait(EndpointName=endpoint_name)


# print the status of the endpoint
endpoint_response = sagemaker.describe_endpoint(EndpointName=endpoint_name)
status = endpoint_response['EndpointStatus']
print('Endpoint creation ended with EndpointStatus = {}'.format(status))

if status != 'InService':
    raise Exception('Endpoint creation failed.')
```

```
EndpointStatus = Creating
Endpoint creation ended with EndpointStatus = InService
```

ML Platforms as a service: Sagemaker

Real-time prediction

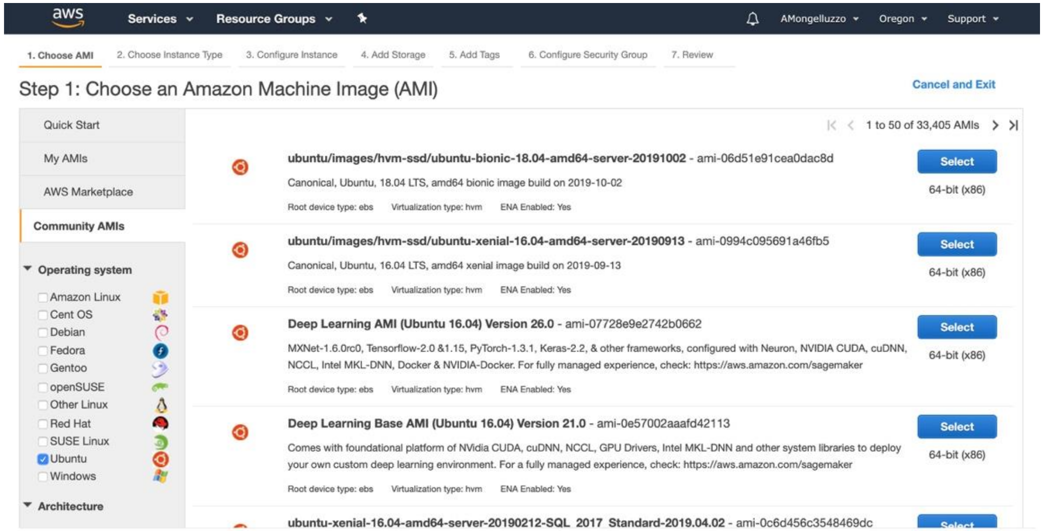


- User invokes Endpoint URL with a POST request
- Lambda function forwards the payload to the correct endpoint
- User receives back the endpoint output

ML Infrastructures as a service: EC2

Choose and create AMI

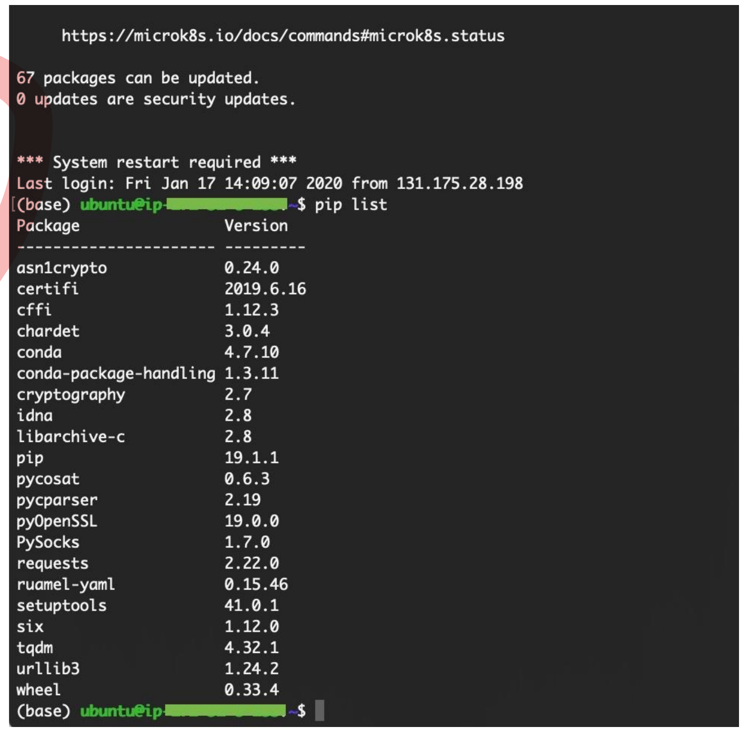
- Completely customisable infrastructure
- Choose your VM Operative System and hardware specifications



ML Infrastructures as a service: EC2

Access VM and execute your own code

- Access VM via ssh
- Set up the environment: install libraries and dependencies
- Run your own training/inference code on the EC2 instance



ML Infrastructures as a service: EC2

1 2 3 4 5 6 7 8 …

Access VM and execute your own code

- No built-in models or pre-configured environments
- Write your **CNN** code

```python
class MyCNN (nn.Module):

    def __init__(self, fullyconnected=True, num_classes=1000):
        super(AlexNetMultiGate, self).__init__()
        self.idx_dict = dict()
        # Features
        self.conv = nn.Sequential()
        self.conv.add_module('conv1', nn.Conv2d(3, 96, kernel_size=11,
                                stride=4, padding=0))
        self.conv.add_module('relu1', nn.ReLU(inplace=True))
        self.conv.add_module('norm1', nn.LocalResponseNorm(size=5, alpha=1e-4,
                                             beta=0.75, k=1))
        self.conv.add_module('pool1', nn.MaxPool2d(kernel_size=3, stride=2))
        self.conv.add_module('conv2', nn.Conv2d(96, 256, kernel_size=5,
                                stride=1, padding=2, groups=2))
        self.conv.add_module('relu2', nn.ReLU(inplace=True))
        self.conv.add_module('norm2', nn.LocalResponseNorm(size=5,
                                             alpha=1e-4,
                                             beta=0.75, k=1))
        self.conv.add_module('pool2', nn.MaxPool2d(kernel_size=3, stride=2))
        self.conv.add_module('conv3', nn.Conv2d(256, 384, kernel_size=3,
                                stride=1, padding=1))
        self.conv.add_module('relu3', nn.ReLU(inplace=True))
        self.conv.add_module('conv4', nn.Conv2d(384, 384, kernel_size=3,
                                stride=1, padding=1, groups=2))
        self.conv.add_module('relu4', nn.ReLU(inplace=True))
        self.conv.add_module('conv5', nn.Conv2d(384, 256, kernel_size=3,
                                stride=1, padding=1, groups=2))
        self.conv.add_module('relu5', nn.ReLU(inplace=True))
        self.conv.add_module('pool5', nn.MaxPool2d(kernel_size=3, stride=2))
```

ML Infrastructures as a service: EC2

1 2 3 4 5 6 7 8 …

Access VM and execute your own code

- No automatic model validation
- Write your model **validation** code

```python
# variance-based feature selection
if feat_sel:
    print("Pivoting features...")
    train_features_dict = {"f_{}".format(n) : [x[n] for x in train_features] for\
                           n in range(len(train_features[0]))}
    nfi = len(train_features_dict)
    train_ds = pd.DataFrame(train_features_dict)
    train_ds["Label"] = train_y

    valid_features_dict = {"f_{}".format(n) : [x[n] for x in valid_features] for\
                           n in range(len(valid_features[0]))}
    valid_ds = pd.DataFrame(valid_features_dict)
    valid_ds["Label"] = valid_y

    if feat_sel == 'var10':
        selected_features = [k for k, v in train_features_dict.items() if np.var(v) > 0]
        p10 = np.percentile([np.var(train_features_dict[x]) for x in selected_features], 10)
        selected_features = [k for k in selected_features if np.var(train_features_dict[k]) > p10]
        nff=len(selected_features)
        print("Selected {}/{} features.".format(nff, nfi))

        # select features
        print("Selecting chosen features...")
        """train_features = [v for k, v in train_features_dict.items() if k in selected_features]
        valid_features = [v for k, v in valid_features_dict.items() if k in selected_features]"""
        train_ds = train_ds[[c for c in selected_features]+["Label"]]
        valid_ds = valid_ds[[c for c in selected_features]+["Label"]]

        # pivoting back
        """print("Pivoting back training data...")
        train_features = [[x[n] for x in train_features] for n in range(len(train_features[0]))]
        print("Pivoting back validation data...")
        valid_features = [[x[n] for x in valid_features] for n in range(len(valid_features[0]))]"""
    elif feat_sel == 'pca':
        X_train_ds = train_ds[[c for c in train_ds.columns if c != 'Label']]
        X_valid_ds = valid_ds[[c for c in valid_ds.columns if c != 'Label']]
        print("Performing pca...")
```

ML Infrastructures as a service: EC2

Access VM and execute your own code

- Search for best model parameters (e.g., grid search on parameter space)

```bash
#!/bin/bash
f(){
    local lr=$1
    local layers=$2
    local momentum=$3
    local batch_size=$4
    python validate.py -t $t -c $c -l1 $l1 -l2 $l2
}
for lr in 5 10 15 20 25 30
do
   for l in 2 3 4
   do
      for m in 1
      do
         for bs in 5
            do
                f $lr $l $m $bs &
            done
      done
   done
done
```

- Run validation algorithm as a stand-alone screen

```
(base) ubuntu@ip-[redacted]:~/[redacted]$ conda activate [redacted]
([redacted]) ubuntu@ip-[redacted]:~/[redacted]$ bash valid_loop.sh
([redacted]) ubuntu@ip-[redacted]:~/[redacted]$ FOLD 1
```

ML Infrastructures as a service: EC2

Access VM and execute your own code

- Once the model is ready -> **deployment**
- Deploy as a web API
  (e.g., Flask, Gunicorn, Nginx)

```python
# The flask app for serving predictions
app = flask.Flask(__name__)

@app.route('/ping', methods=['GET'])
def ping():
    """Determine if the container is working and healthy. In this sample container, we declare
    it healthy if we can load the model successfully."""
    health = ScoringService.get_model() is not None  # You can insert a health check here

    status = 200 if health else 404
    return flask.Response(response='\n', status=status, mimetype='application/json')

@app.route('/invocations', methods=['POST'])
def transformation():
    """Do an inference on a single batch of data. In this sample server, we take data as CSV, convert
    it to a pandas data frame for internal use and then convert the predictions back to CSV (which really
    just means one prediction per line, since there's a single column.
    """
    data = None

    # Convert from CSV to pandas
    if flask.request.content_type == 'text/csv':
        data = flask.request.data.decode('utf-8')
        s = io.StringIO(data)
        data = pd.read_csv(s, header=None)
    else:
        return flask.Response(response='This predictor only supports CSV data', status=415, mimetype='text/plain')

    print('Invoked with {} records'.format(data.shape[0]))

    # Do the prediction
    predictions = ScoringService.predict(data)

    # Convert from numpy back to CSV
    out = io.StringIO()
    pd.DataFrame({'results':predictions}).to_csv(out, header=False, index=False)
    result = out.getvalue()

    return flask.Response(response=result, status=200, mimetype='text/csv')
```

ML Infrastructures as a service: EC2

1 › 2 › 3 › 4 › 5 › 6 › 7 › 8 › …

Access VM and execute your own code

Custom model **deployment**

- EC2 Web Server
- Sagemaker Custom Container
- Deploy your local container with Elastic Container Service (ECS)[3]
- Deploy your EC2 container with CodeDeploy[4]

[3] https://aws.amazon.com/it/ecs/
[4] https://aws.amazon.com/it/codedeploy/