



Computing Infrastructures

 POLITECNICO DI MILANO



AI and Platforms:
Machine Learning-**is-a-service**

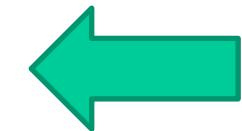
The topics of the course: what are we going to see today?

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



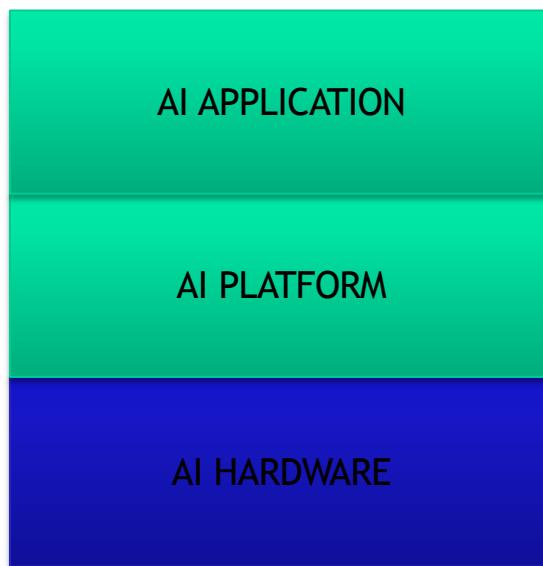
C. Methods:

- **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)

Cloud and ML/DL as-a-service



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



Machine-Learning
Deep-Learning
as-a-Service



ML and DL Solutions





The IT architecture for ML/DL on Datacenters

Machine Learning Framework

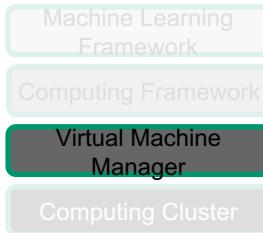
Computing Framework

Virtual Machine Manager

Computing Cluster

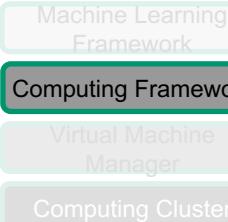


- **Servers** (parallelization and scalability are key to find the best model as well as HW accelerators):
 - General Purpose CPU
 - GPU (TPU)
- **Storage** (multiple storage systems including distributed/parallel file systems):
 - Direct Attached Storage
 - NAS
 - SAN
- **Network** (applications are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



- **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

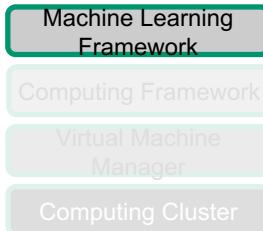
Computing Framework



Examples

- 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

- 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 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

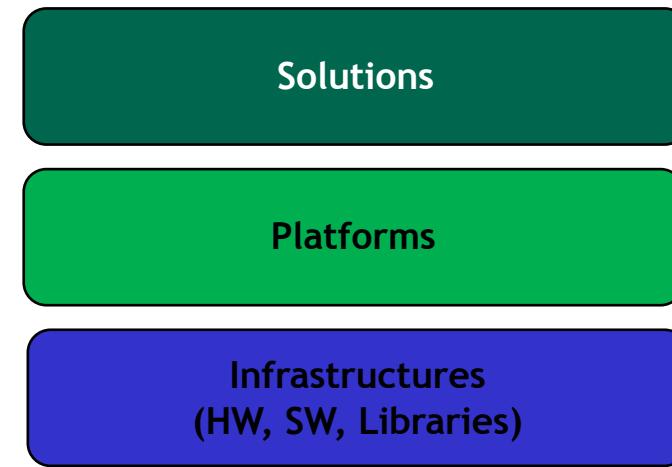
Generally organized as libraries/shells

Exploit GPU access

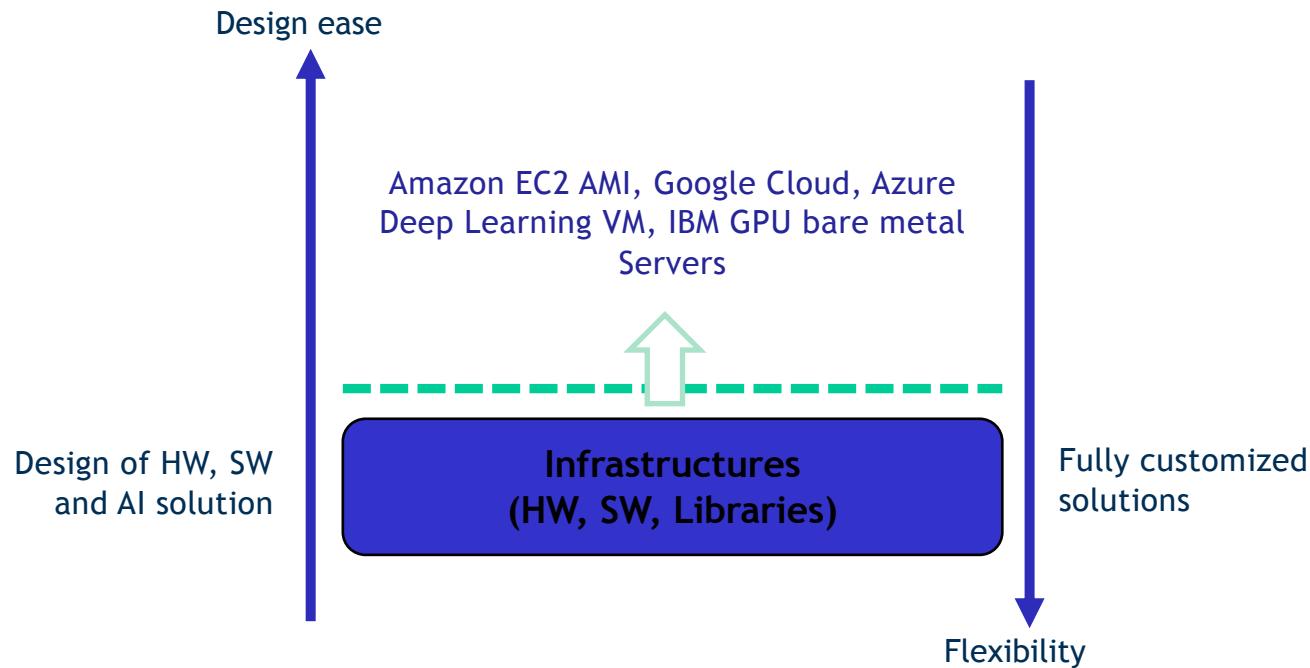
- Cloud computing simplifies the access to ML capabilities for
 - **designing a solution** (without requiring a deep knowledge of ML)
 - **setting up a project** (managing demand increases and IT solution)
- Amazon, Microsoft and Google (just to name a few) provide
 - Solutions → **ML Solutions as a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

to support ML in the Cloud

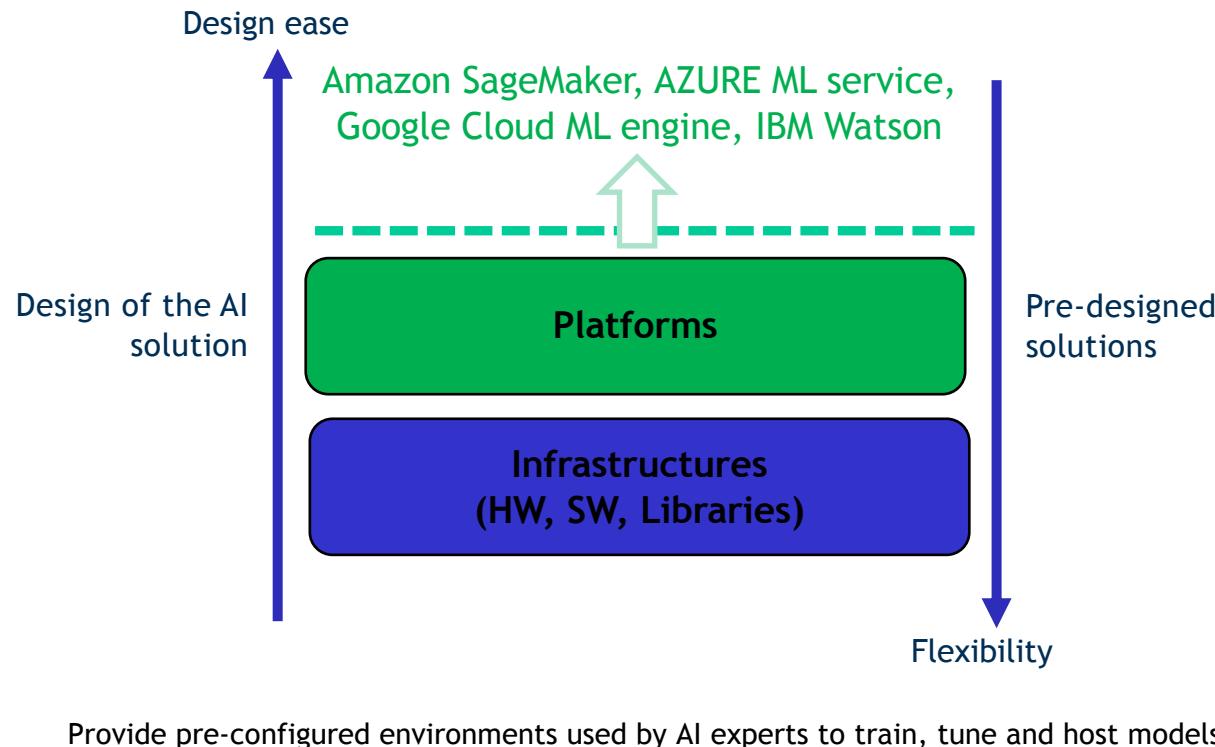
Machine Learning as a Service



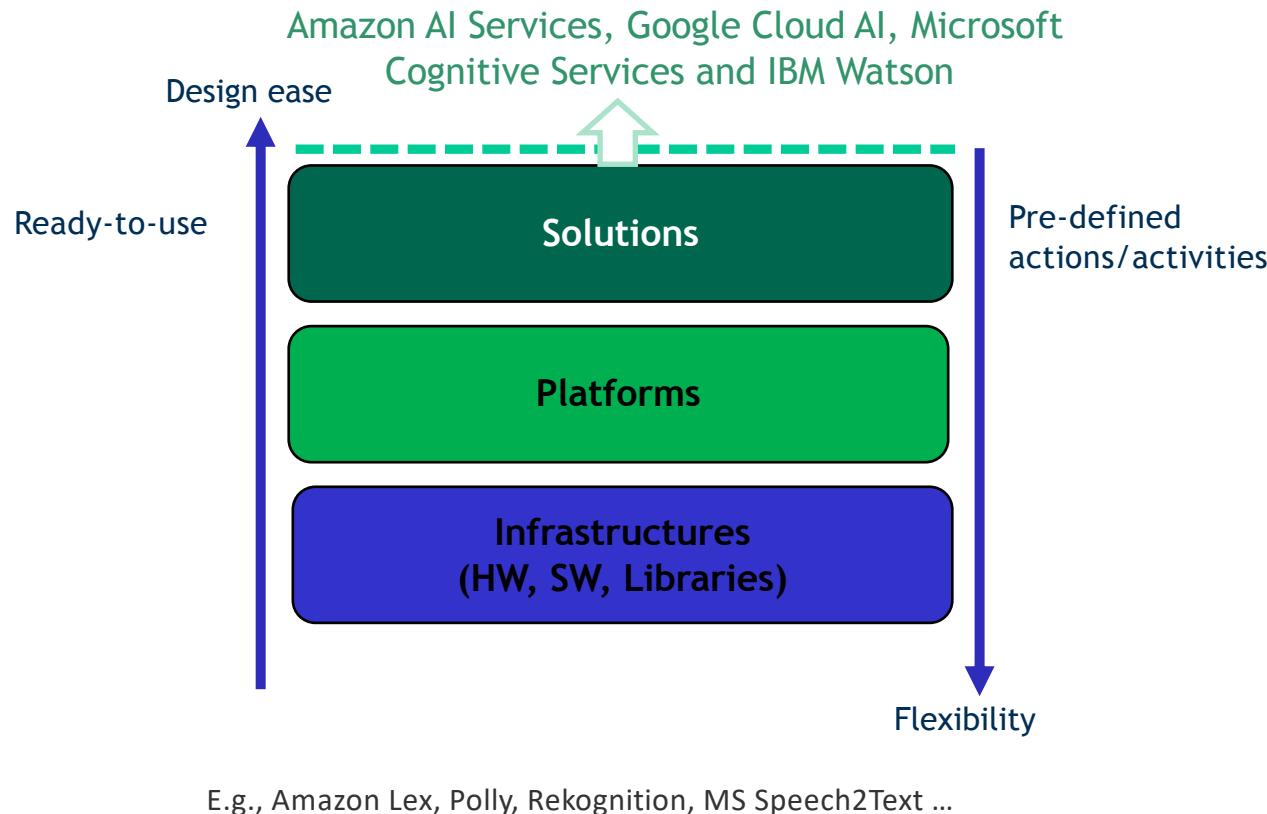
Machine Learning Infrastructure as a Service



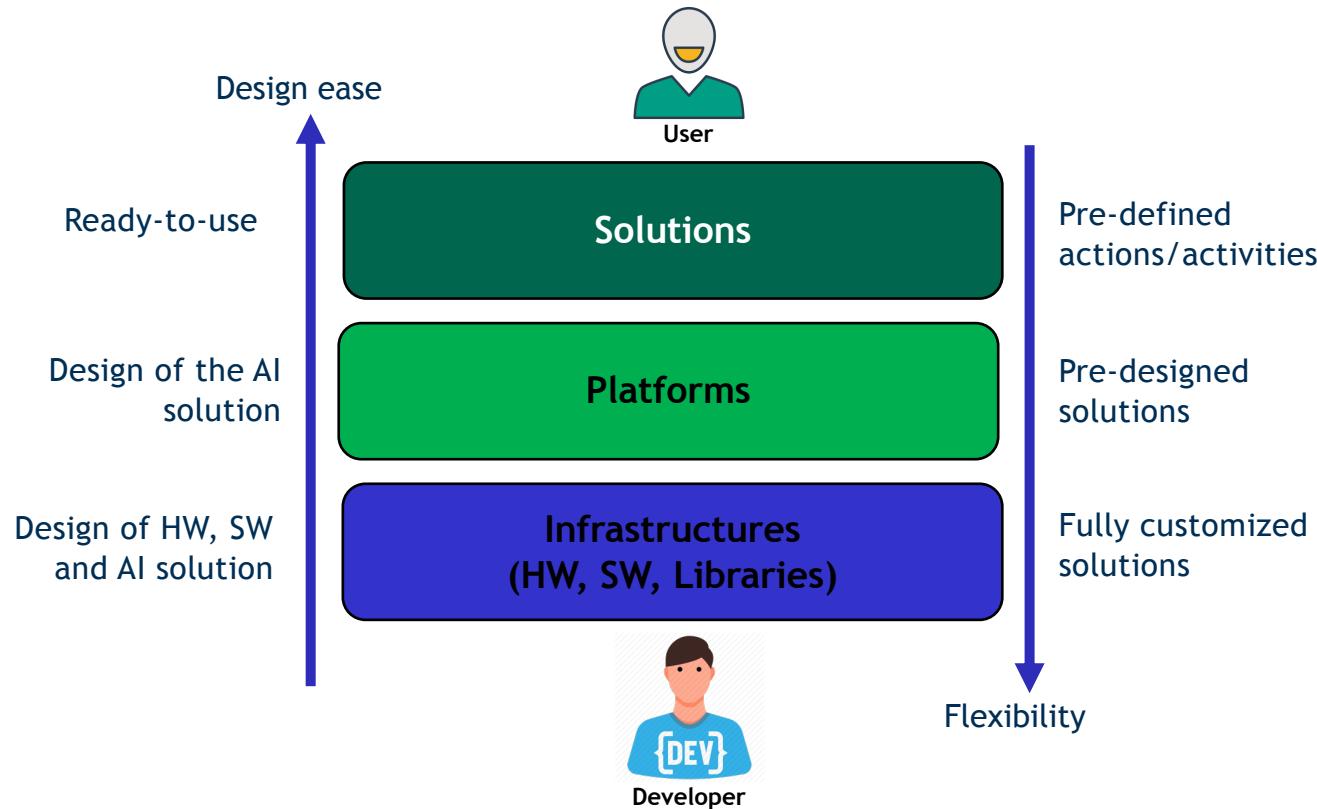
Machine Learning Platform as a Service



ML software as a service



AI and off-the-shelf technological solutions





A real world example: image classification

25



A real world example: image classification

26



Solutions

Capabilities

Methodologies

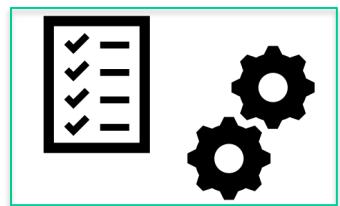
Technologies



AI Engineer

A real world example: image classification on AWS

27



Training Data

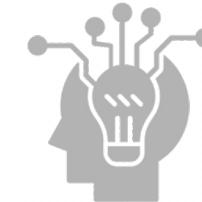


Solutions

Capabilities

Methodologies

Technologies



AI Engineer

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



ML Solutions as a service: Rekognition

1

Easy to access...



The screenshot shows the Amazon Rekognition landing page. At the top left, there's a sidebar with the title "Amazon Rekognition" and links to "Custom Labels New", "Use Custom Labels", "Demos", "Object and scene detection", "Image moderation", and "Facial analysis". The main content area has a dark blue background with a network graph pattern. The title "Amazon Rekognition" is prominently displayed in large white font. Below it, the text "Deep learning-based visual analysis service" and "Search, verify, and organize millions of images and videos" is written in smaller white font. There are two blue buttons at the bottom: "Try Demo" and "Download SDKs".

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

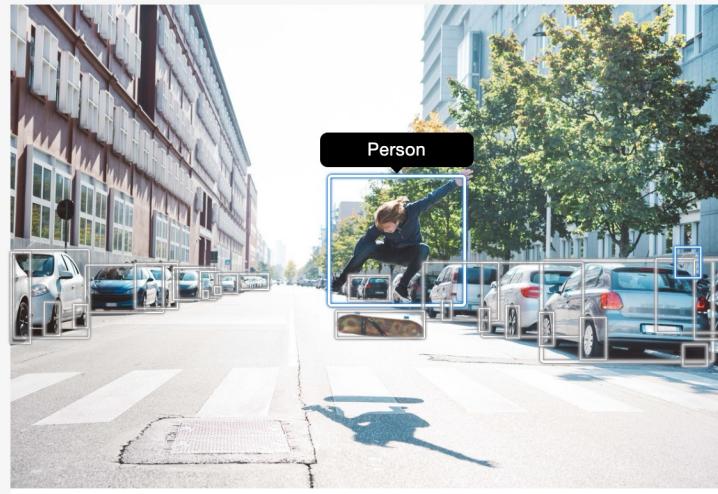
ML Solutions as a service: Rekognition

1

... and easy
to use

Object and scene detection

Rekognition automatically labels objects, concepts and scenes in your images, and provides a confidence score.



Done with the demo?

[Learn more](#)

▼ Results

Car	98.8 %
Vehicle	98.8 %
Transportation	98.8 %
Automobile	98.8 %
Person	98.3 %
Human	98.3 %

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

ML Platforms as a service: Sagemaker

1 2 3 4 5 6

Easy to access

The screenshot shows the Amazon SageMaker console. On the left, there's a sidebar with the title "Amazon SageMaker" and a navigation menu including "Dashboard", "Search", "Ground Truth" (with sub-options "Labeling jobs", "Labeling datasets", "Labeling workforces"), and "Notebook" (with sub-options "Notebook instances", "Lifecycle configurations"). The main content area has a dark background and features the text "MACHINE LEARNING" at the top. In the center, it says "Amazon SageMaker" in large bold letters, followed by "Build, train, and deploy machine learning models at scale". Below this, a subtitle reads "The quickest and easiest way to get ML models from idea to production." To the right, there's a white callout box with "Get started" and "Create notebook instance" buttons, and a link "Start with an overview".

- 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

Create notebook instance

Amazon SageMaker provides pre-built fully managed notebook instances that run Jupyter notebooks. The notebook instances include example code for common model training and hosting exercises. [Learn more](#)

Notebook instance settings

Notebook instance name

MySagemakerNotebook

Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.

Notebook instance type

ml.t2.medium

Elastic Inference [Learn more](#)

none

► Additional configuration

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 image¹

```
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 Resnet² 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

1 2 3 4 5 6

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...

```
# 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))

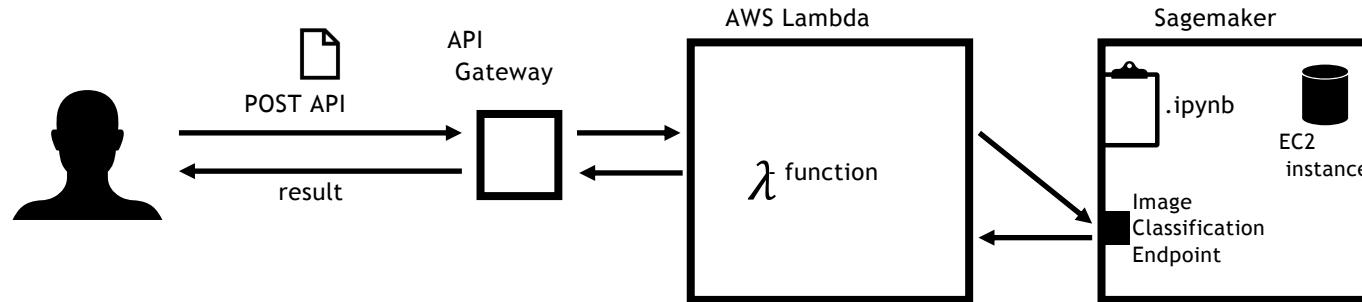
if status != 'InService':
    raise Exception('Endpoint creation failed.')

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

ML Platforms as a service: Sagemaker

1 2 3 4 5 6

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

1 2 3 4 5 6 7 8 ...

Choose and create AMI

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

The screenshot shows the AWS Step 1: Choose an Amazon Machine Image (AMI) interface. The left sidebar has tabs for Quick Start, My AMIs, AWS Marketplace, and Community AMIs. Under Community AMIs, there are two expandable sections: Operating system and Architecture. The Operating system section is expanded, showing various Linux distributions with checkboxes: Amazon Linux, Cent OS, Debian, Fedora, Gentoo, openSUSE, Other Linux, Red Hat, SUSE Linux, Ubuntu (which is checked), and Windows. The Architecture section is collapsed. The main area displays a list of AMIs with columns for name, description, root device type, virtualization type, ENA Enabled status, and architecture. The first item is "ubuntu/images/hvm-ssd/ubuntu-bionic-18.04-amd64-server-20191002 - ami-06d51e91cea0dac8d". Below it are "ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-20190913 - ami-0994c095691a46fb5", "Deep Learning AMI (Ubuntu 16.04) Version 26.0 - ami-07728e9e2742b0662", and "Deep Learning Base AMI (Ubuntu 16.04) Version 21.0 - ami-0e57002aaaaf42113". At the bottom of the list is "ubuntu-xenial-16.04-amd64-server-20190212-SQL 2017 Standard-2019.04.02 - ami-0c6d456c3548469dc". Each item has a "Select" button to its right. Navigation buttons at the top right include back, forward, and search fields.

1 2 3 4 5 6 7 8 ...

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

```
https://microk8s.io/docs/commands#microk8s.status

67 packages can be updated.
0 updates are security updates.

*** System restart required ***
Last login: Fri Jan 17 14:09:07 2020 from 131.175.28.198
(base) ubuntu@ip-131-175-28-198:~$ pip list
Package           Version
-----
asn1crypto        0.24.0
certifi          2019.6.16
cffi              1.12.3
chardet          3.0.4
conda             4.7.10
conda-package-handling 1.3.11
cryptography     2.7
idna              2.8
libarchive-c      2.8
pip               19.1.1
pycosat          0.6.3
pycparser         2.19
pyOpenSSL         19.0.0
PySocks           1.7.0
requests          2.22.0
ruamel-yaml       0.15.46
setuptools        41.0.1
six               1.12.0
tqdm              4.32.1
urllib3           1.24.2
wheel              0.33.4
(base) ubuntu@ip-131-175-28-198:~$
```

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

```
14 class MyCNN (nn.Module):  
15     def __init__(self, fullyconnected=True, num_classes=1000):  
16         super(AlexNetMultiGate, self).__init__()  
17         self.idx_dict = dict()  
18         # Features  
19         self.conv = nn.Sequential()  
20         self.conv.add_module('conv1', nn.Conv2d(3, 96, kernel_size=11,  
21                                         stride=4, padding=0))  
22         self.conv.add_module('relu1', nn.ReLU(inplace=True))  
23         self.conv.add_module('norm1', nn.LocalResponseNorm(size=5, alpha=1e-4,  
24                                         beta=0.75, k=1))  
25         self.conv.add_module('pool1', nn.MaxPool2d(kernel_size=3, stride=2))  
26         self.conv.add_module('conv2', nn.Conv2d(96, 256, kernel_size=5,  
27                                         stride=1, padding=2, groups=2))  
28         self.conv.add_module('relu2', nn.ReLU(inplace=True))  
29         self.conv.add_module('norm2', nn.LocalResponseNorm(size=5,  
30                                         alpha=1e-4,  
31                                         beta=0.75, k=1))  
32         self.conv.add_module('pool2', nn.MaxPool2d(kernel_size=3, stride=2))  
33         self.conv.add_module('conv3', nn.Conv2d(256, 384, kernel_size=3,  
34                                         stride=1, padding=1))  
35         self.conv.add_module('relu3', nn.ReLU(inplace=True))  
36         self.conv.add_module('conv4', nn.Conv2d(384, 384, kernel_size=3,  
37                                         stride=1, padding=1, groups=2))  
38         self.conv.add_module('relu4', nn.ReLU(inplace=True))  
39         self.conv.add_module('conv5', nn.Conv2d(384, 256, kernel_size=3,  
40                                         stride=1, padding=1, groups=2))  
41         self.conv.add_module('relu5', nn.ReLU(inplace=True))  
42         self.conv.add_module('pool5', nn.MaxPool2d(kernel_size=3, stride=2))  
43  
44
```

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

```
# Variance-based feature selection
if feat_sel:
    print("Pivoting features...")
    train_features_dict = {"f_{0}":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_{0}":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

1 2 3 4 5 6 7 8 ...

Access VM and execute your own code

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

```
#!/bin/bash
# Of and attributes or search by keyword
local lr=$1
local layers=$2
local momentum=$3
local batch_size=$4
python validate.py -t $t -c $c -l1 $l1 -l2 $l2
} i-01bdd8a5f0862185b c5.18xlarge
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
done Instance state      running
```

- Run validation algorithm as a stand-alone screen

```
(base) ubuntu@ip-10-10-1-1:~/ml$ conda activate ml
(base) ubuntu@ip-10-10-1-1:~/ml$ bash valid_loop.sh
(base) ubuntu@ip-10-10-1-1:~/ml$ FOLD 1
```

ML Infrastructures as a service: EC2

1 2 3 4 5 6 7 8 ...

Access VM and execute your own code

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

```
# 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()

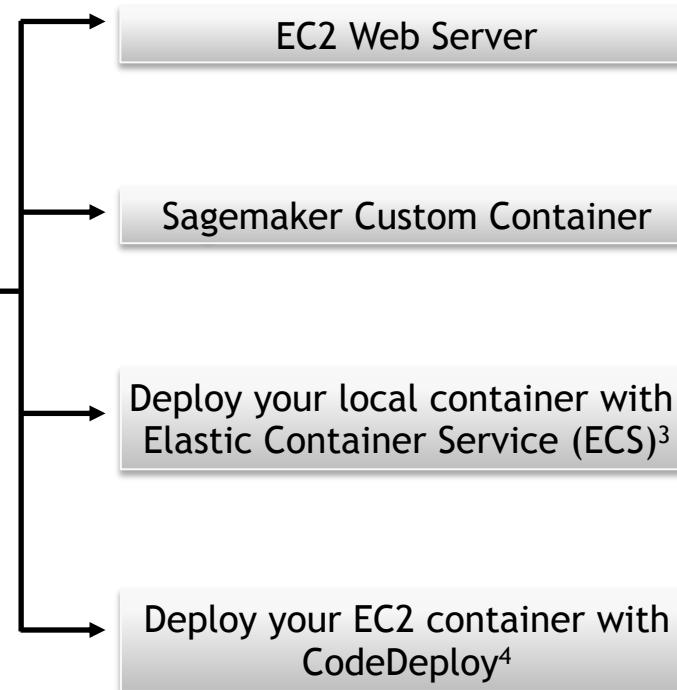
    return flask.Response(response=result, status=200, mimetype='text/csv')
```

ML Infrastructures as a service: EC2



Access VM and execute your own code

Custom model deployment



[3] <https://aws.amazon.com/it/ecs/>

[4] <https://aws.amazon.com/it/codedeploy/>