Cloud Computing & Big Data

PARALLEL & SCALABLE MACHINE LEARNING & DEEP LEARNING

Prof. Dr. – Ing. Morris Riedel

Associated Professor

School of Engineering and Natural Sciences, University of Iceland, Reykjavik, Iceland

Research Group Leader, Juelich Supercomputing Centre, Forschungszentrum Juelich, Germany

PRACTICAL LECTURE 7.1

@Morris Riedel @MorrisRiedel @MorrisRiedel

Using Deep Learning Techniques in Clouds

October 27, 2020

Online Lecture

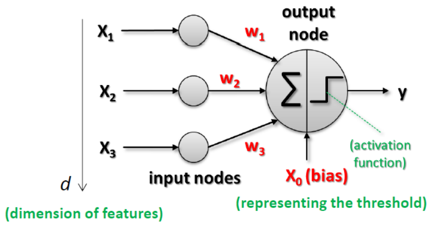
EUROPEAN OPEN SCIENCE CLOUD EOSC NORDIC EuroHPC Joint Undertaking ADMIRE EURO

UNIVERSITY OF ICELAND
SCHOOL OF ENGINEERING AND NATURAL SCIENCES
FACULTY OF INDUSTRIAL ENGINEERING, MECHANICAL ENGINEERING AND COMPUTER SCIENCE

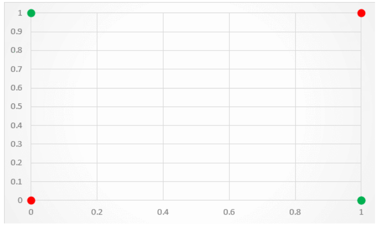
JÜLICH Forschungszentrum JÜLICH SUPERCOMPUTING CENTRE DEEP Projects HELMHOLTZAI ARTIFICIAL INTELLIGENCE COOPERATION UNIT

Review of Lecture 7 – Deep Learning Applications in Clouds

- Limited Perceptron Learning Model



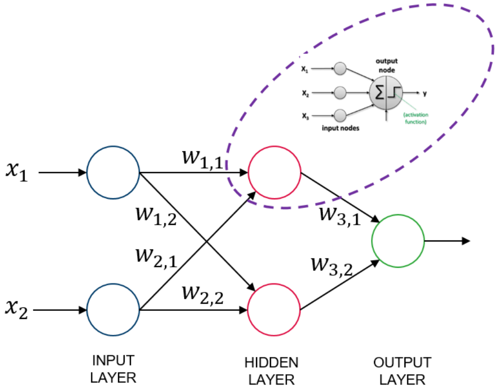
[30] F. Rosenblatt, 1957

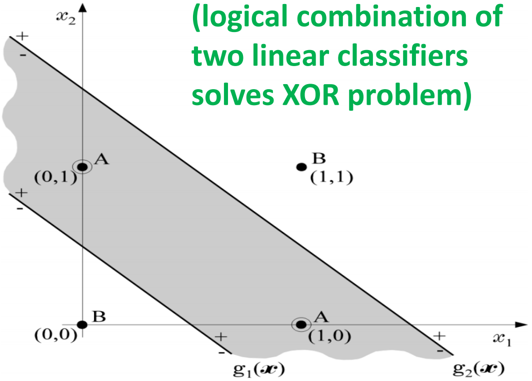


[29] XOR Problem

Input 1	Input 2	Output
0	0	0
0	1	1
1	1	0
1	0	1

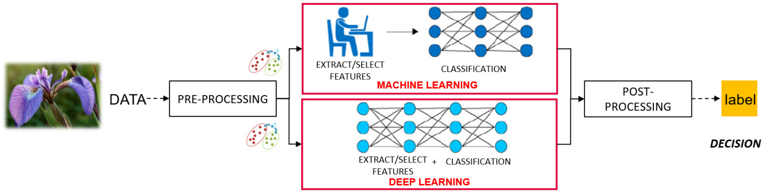
- Artificial Neural Networks (ANNs)





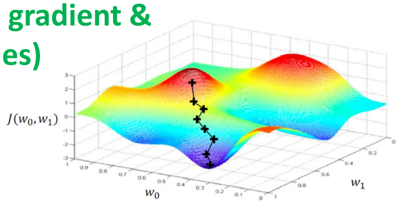
(logical combination of two linear classifiers solves XOR problem)

- Feature Engineering vs. Feature Learning

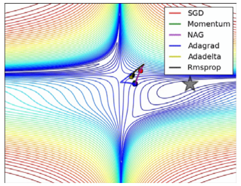


- Training via Optimization & Backpropagation

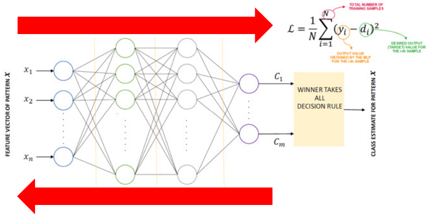
(stochastic gradient & mini-batches)



[31] MIT Deep Learning



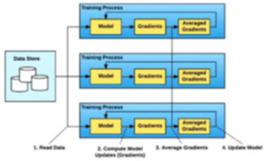
[32] Optimizers


$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(y_i - d_i)^2$
TOTAL NUMBER OF TRAINING SAMPLES
OUTPUT VALUE OBTAINED BY THE MLP FOR THE i-th SAMPLE
DESIRED OUTPUT (TARGET) VALUE FOR THE i-th SAMPLE
FEATURE VECTOR OF PATTERN x
x_1
x_2
x_n
C_1
C_m
WINNER TAKES ALL DECISION RULE
CLASS ESTIMATE FOR PATTERN x

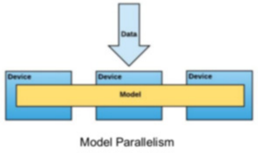
```
1. Initialize weights randomly $\sim\mathcal{N}(0, \sigma^2)$
2. Loop until convergence
3.     Pick batch of B data points
4.     Compute gradient $\frac{\partial \mathcal{L}_i(W)}{\partial W} = \frac{1}{B}\sum_{k=1}^{B}\frac{\partial \mathcal{L}_i(W)}{\partial W}$
5.     Update weights $W := W - \eta \frac{\partial \mathcal{L}(W)}{\partial W}$
6. Return weights
```

(training has here effect!)

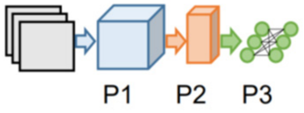
- Distributed Deep Learning



(data parallel)



(model parallel)



(pipelining)

[22] Horovod

[23] T. Ben-Nun & T. Hoefler

Outline of the Course

1. Cloud Computing & Big Data Introduction
2. Machine Learning Models in Clouds
3. Apache Spark for Cloud Applications
4. Virtualization & Data Center Design
5. Map-Reduce Computing Paradigm
6. Deep Learning driven by Big Data
7. Deep Learning Applications in Clouds
8. Infrastructure-As-A-Service (IAAS)
9. Platform-As-A-Service (PAAS)
10. Software-As-A-Service (SAAS)
11. Big Data Analytics & Cloud Data Mining
12. Docker & Container Management
13. OpenStack Cloud Operating System
14. Online Social Networking & Graph Databases
15. Big Data Streaming Tools & Applications
16. Epilogue

+ additional practical lectures & Webinars for our hands-on assignments in context

- 🟩 Practical Topics
- 🟦 Theoretical / Conceptual Topics

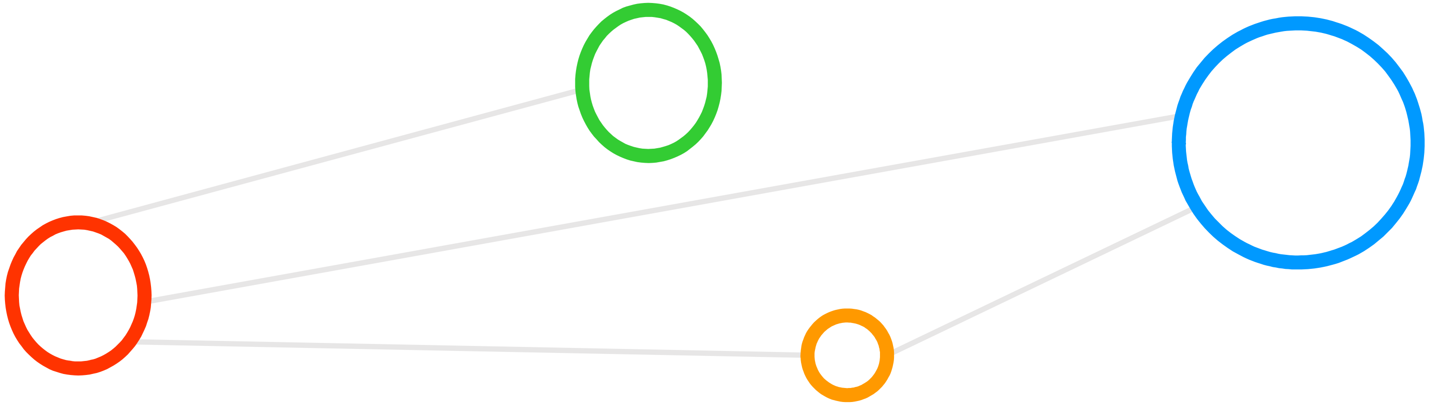
Outline

- Using Artificial Neural Network (ANN) & CPUs in Clouds
  - Handwritten Character Recognition MNIST Dataset Revisited
  - AWS Elastic Compute Cloud (EC2) & Virtual Server Cloud Instances
  - Using EC2 Amazon Machine Images (AMIs) for Machine Learning
  - Limitations of Free Usage Tiers & Review Challenges of Deploying
  - Observe Growth of Trainable Parameter & Understanding Overfitting

- Using Convolutional Neural Network (CNN) & GPUs in Clouds
  - Using EC2 Amazon Machine Images (AMIs) for Deep Learning via CPU
  - Growth of Trainable Parameters & Hyperparameter Complexity
  - Understanding difference between CPUs & GPUs in Training
  - Using Google Colaboratory ‘Colab‘ Cloud Service for Deep Learning
  - Neural Architecture Search and Auto-ML & Resource Requirements

- **Promises from previous lecture(s):**
- ***Practical Lecture 0.1:*** **Lecture 6 & 7 will provide more insights into deep learning algorithms and networks including the use of TensorFlow and Keras libraries**
- ***Practical Lecture 0.1:*** **Lecture 6 & 7 will provide more details on how artificial neural networks (ANNs) and deep learning networks can be used with this data**
- ***Lecture 2:*** **Lectures 6 & 7 offer more details on feature selection concepts including working with spatial aspects in image recognition tasks**
- ***Lecture 3:*** **Lecture 6 & 7 offer insights of how to use deep learning with cutting-edge GPUs via Google ‘colab’ notebooks within the Google Cloud**

Using Artificial Neural Network (ANN) & CPUs in Clouds



Handwritten Character Recognition MNIST Dataset – Preprocessing with Python

- Metadata (cf. Practical Lecture 0.1)
  - Not very challenging dataset, but good for benchmarks & tutorials
- When working with the dataset
  - Dataset is not in any standard image format like jpg, bmp, or gif (i.e. file format not known to a graphics viewer)
  - Data samples are stored in a simple file format that is designed for storing vectors and multidimensional matrices (i.e. numpy arrays)
  - The pixels of the handwritten digit images are organized row-wise with pixel values ranging from 0 (white background) to 255 (black foreground)
  - Images contain grey levels as a result of an anti-aliasing technique used by the normalization algorithm that generated this dataset
  - Initially input for an Artificial Neural Network (ANN)
  - Afterwards input for a deep learning network

- **Handwritten Character Recognition MNIST dataset is a subset of a larger dataset from US National Institute of Standards (NIST)**
- **MNIST handwritten digits includes corresponding labels with values 0-9 and is therefore a labeled dataset**
- **MNIST digits have been size-normalized to 28 * 28 pixels & are centered in a fixed-size image for direct processing**
- **Two separate files for training & test: 60000 training samples (~47 MB) & 10000 test samples (~7.8 MB)**

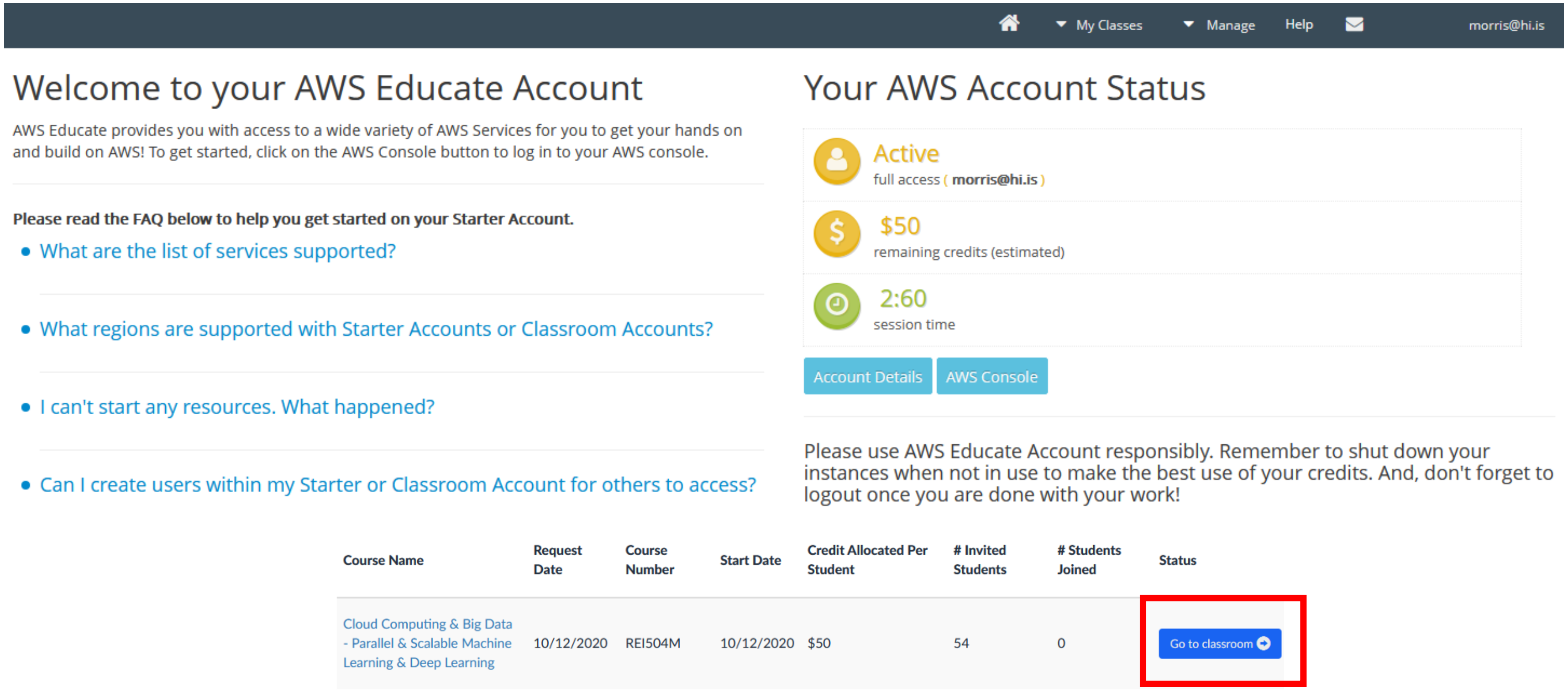
(10 class classification problem)

[33] www.big-data.tips, 'MNIST Database'

[36] www.big-data.tips, 'MNIST Dataset'

AWS Educate Starter Account – Account Status in Classrooms

- Workbench & Example Classroom
  - Cloud Computing & Big Data – Parallel and Scalable Machine Learning and Deep Learning



Welcome to your AWS Educate Account

AWS Educate provides you with access to a wide variety of AWS Services for you to get your hands on and build on AWS! To get started, click on the AWS Console button to log in to your AWS console.

Please read the FAQ below to help you get started on your Starter Account.

- What are the list of services supported?
- What regions are supported with Starter Accounts or Classroom Accounts?
- I can't start any resources. What happened?
- Can I create users within my Starter or Classroom Account for others to access?

Your AWS Account Status

Active
full access (**morris@hi.is**)

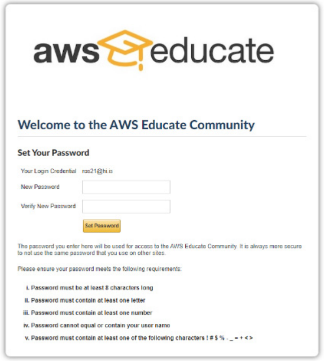
$50
remaining credits (estimated)

2:60
session time

Please use AWS Educate Account responsibly. Remember to shut down your instances when not in use to make the best use of your credits. And, don't forget to logout once you are done with your work!

Course Name	Request Date	Course Number	Start Date	Credit Allocated Per Student	# Invited Students	# Students Joined	Status
Cloud Computing & Big Data - Parallel & Scalable Machine Learning & Deep Learning	10/12/2020	REI504M	10/12/2020	$50	54	0	Go to classroom

[5] Amazon Web Services



Welcome to the AWS Educate Community

Set Your Password

Your Login Credential ros21@hi.is

New Password

Verify New Password

The password you enter here will be used for access to the AWS Educate Community. It is always more secure to not use the same password that you use on other sites.

Please ensure your password meets the following requirements:

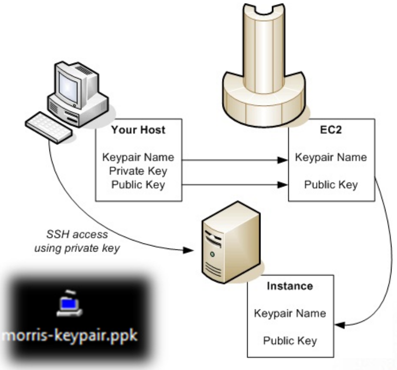
i. Password must be at least 8 characters long
ii. Password must contain at least one letter
iii. Password must contain at least one number
iv. Password cannot equal or contain your user name
v. Password must contain at least one of the following characters ! # $ % - _ = + < >

[4] AWS Educate Web page

AWS Elastic Compute Cloud (EC2) Virtual Servers & Using Key Pairs – Revisited

- Secure Shell (SSH)
  - Universal technique to securely access remote clusters & HPC machines

- The Secure Shell (SSH) is a technique to securely access remote AWS computing instances (e.g., AWS EC2) using a named key pair
- An SSH key pair consists of a public key that is known by the Amazon Cloud and a private key that remains only on the laptop of cloud users

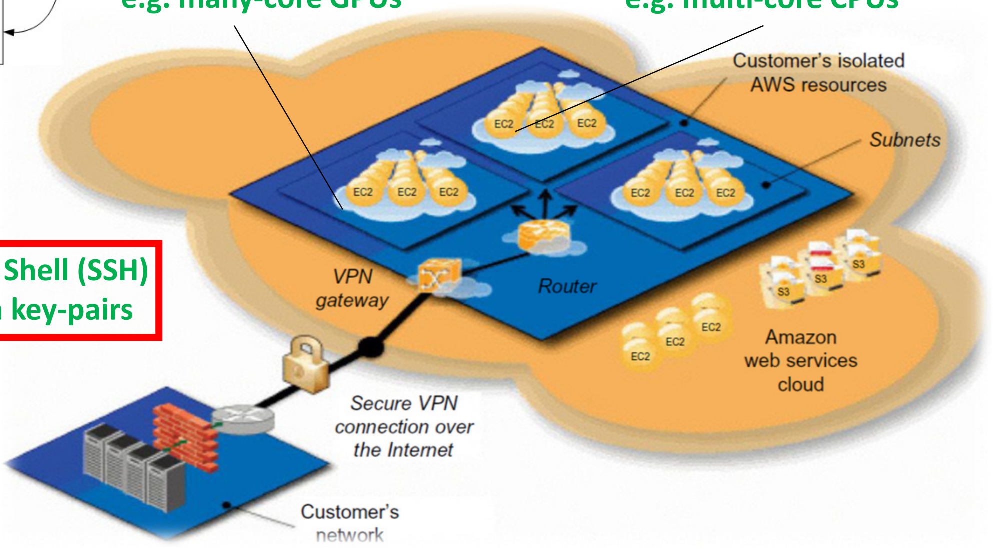


- Generated AWS key pairs are created per region (e.g., Virginia) in the AWS Cloud
- Switching regions means new and/or other SSH keys needs to be used as before

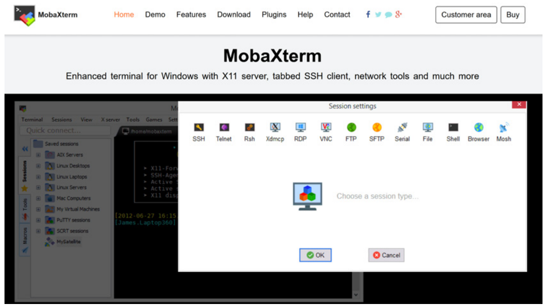
[5] Amazon Web Services

e.g. many-core GPUs

e.g. multi-core CPUs



e.g. Secure Shell (SSH) access with key-pairs



MobaXterm Home Demo Features Download Plugins Help Contact Customer area Buy

MobaXterm

Enhanced terminal for Windows with X11 server, tabbed SSH client, network tools and much more

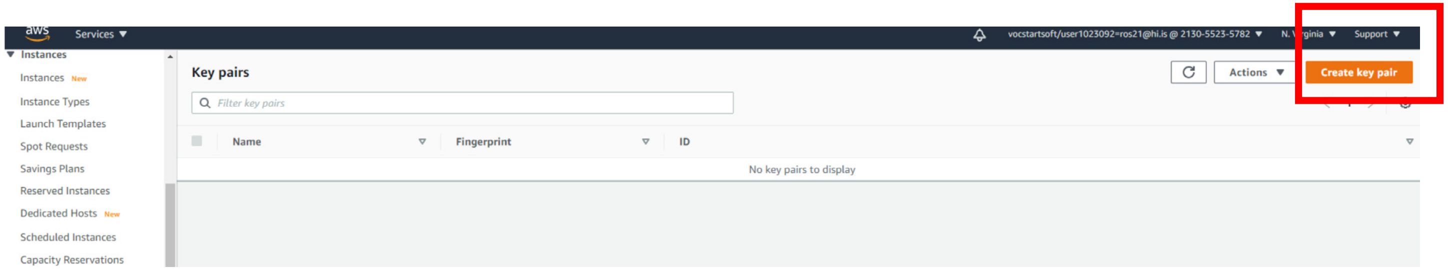
(SSH client is necessary)

[7] MobaXterm Web page

[6] Key Concepts from the AWS Cloud

AWS Key Pair – Key Pair Generation (cf. Practical Lecture 5.1)

- Usage
  - Public Key remains in Cloud
  - Private Key on Laptop
  - Use SSH Client tool with private key to access remote cloud with matching public key



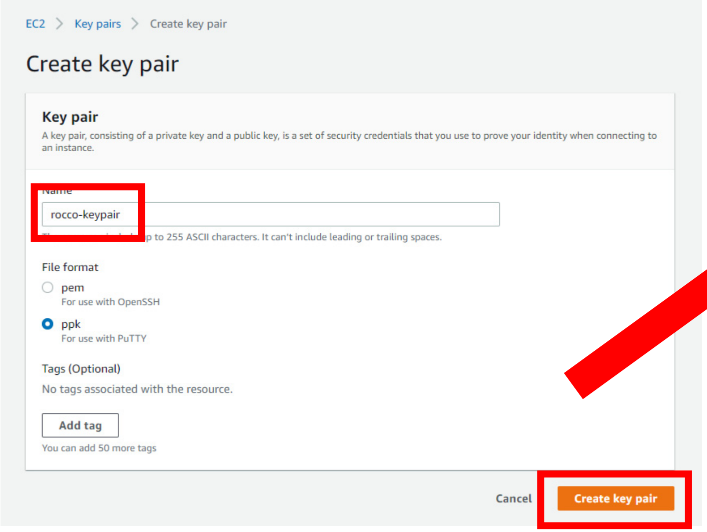
Key pairs

Filter key pairs

□	Name	Fingerprint	ID

No key pairs to display

- After the AWS Key Pair generation, the name of the key is known in many AWS service configuration deployment options such as within the Elastic Compute Cloud (EC2) service or Elastic Map-Reduce (EMR) service



Create key pair

Key pair

A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

Name

rocco-keypair

p to 255 ASCII characters. It can't include leading or trailing spaces.

File format

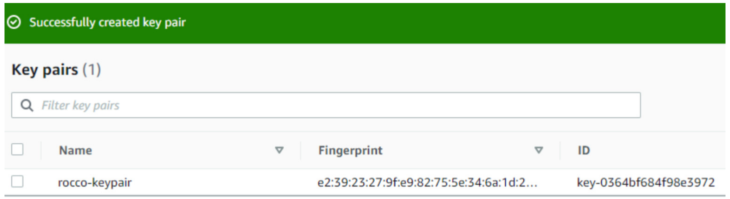
○ pem
For use with OpenSSH

● ppk
For use with PuTTY

Tags (Optional)

No tags associated with the resource.

You can add 50 more tags

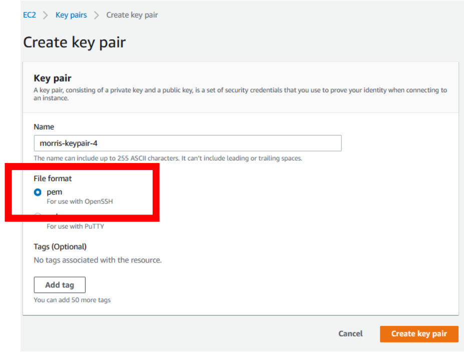

Successfully created key pair

Key pairs (1)

Filter key pairs

□	Name	Fingerprint	ID
□	rocco-keypair	e2:39:23:27:9f:e9:82:75:5e:34:6a:1d:2...	key-0364bf684f98e3972

Creating an SSH key pair and keeping private key in pem can be more convenient in certain connections with SSH



Create key pair

Key pair

A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

Name

morris-keypair-4

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

File format

● pem
For use with OpenSSH

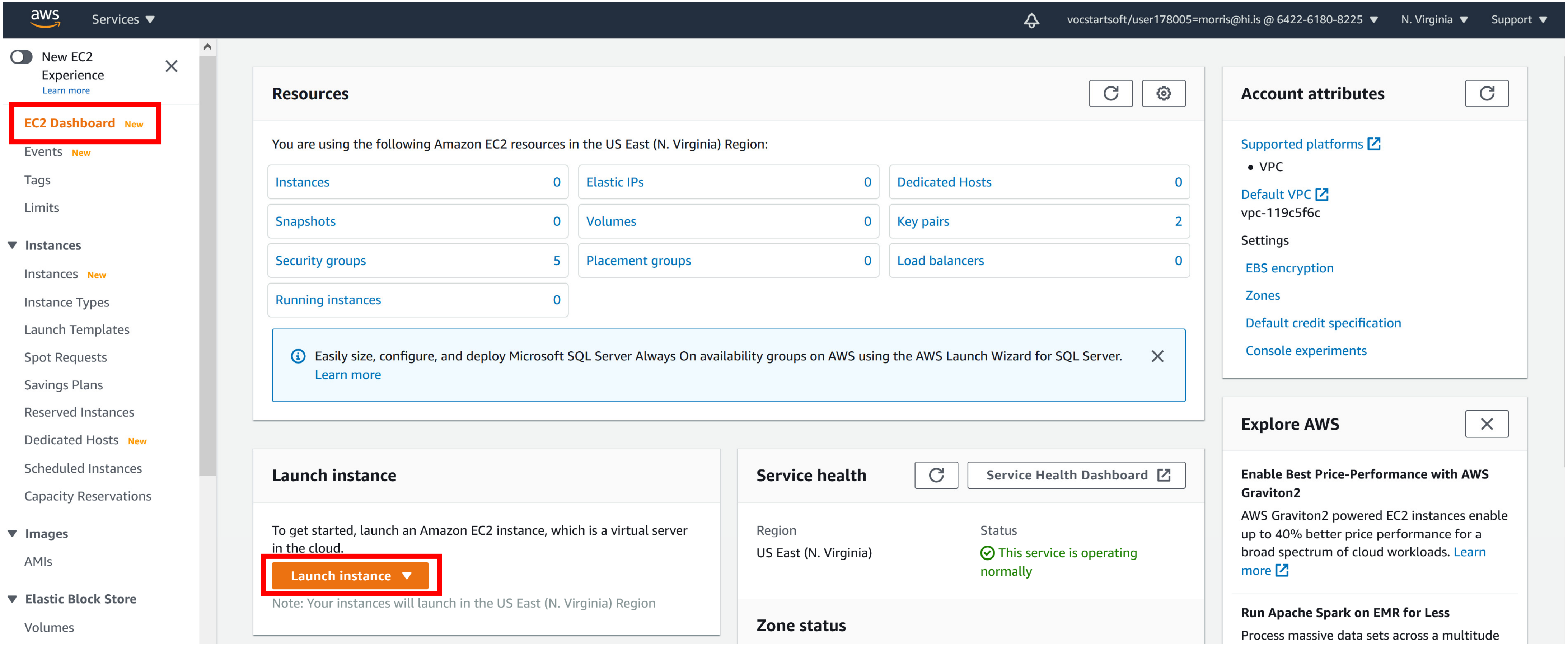
For use with PuTTY

Tags (Optional)

No tags associated with the resource.

You can add 50 more tags

AWS Elastic Compute Cloud (EC2) & Launch Virtual Server Cloud Instances



Resources

You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:

Instances	0	Elastic IPs	0	Dedicated Hosts	0
Snapshots	0	Volumes	0	Key pairs	2
Security groups	5	Placement groups	0	Load balancers	0
Running instances	0				

Easily size, configure, and deploy Microsoft SQL Server Always On availability groups on AWS using the AWS Launch Wizard for SQL Server. Learn more

Launch instance

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Note: Your instances will launch in the US East (N. Virginia) Region

Service health

Region: US East (N. Virginia)

Status: This service is operating normally

Zone status

Account attributes

Supported platforms
- VPC

Default VPC
vpc-119c5f6c

Settings

Explore AWS

Enable Best Price-Performance with AWS Graviton2

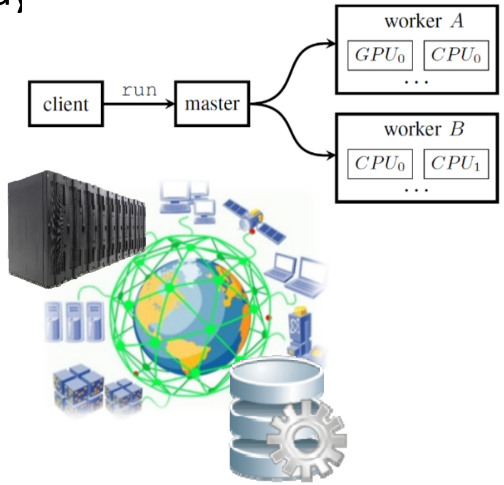
AWS Graviton2 powered EC2 instances enable up to 40% better price performance for a broad spectrum of cloud workloads. Learn more

Run Apache Spark on EMR for Less

Process massive data sets across a multitude

Popular Deep Learning Frameworks used with Python in Cloud Computing

- TensorFlow
  - One of the most popular deep learning frameworks available today
  - Execution on multi-core CPUs or many-core GPUs



TensorFlow

[2] Tensorflow Web page

- **Tensorflow is an open source library for deep learning models using a flow graph approach**
- **Tensorflow nodes model mathematical operations and graph edges between the nodes are so-called tensors (also known as multi-dimensional arrays)**
- **The Tensorflow tool supports the use of CPUs and GPUs (much more faster than CPUs)**
- **Tensorflow work with the high-level deep learning tool Keras in order to create models fast**
- **New versions of Tensorflow have Keras shipped with it as well & many further tools**

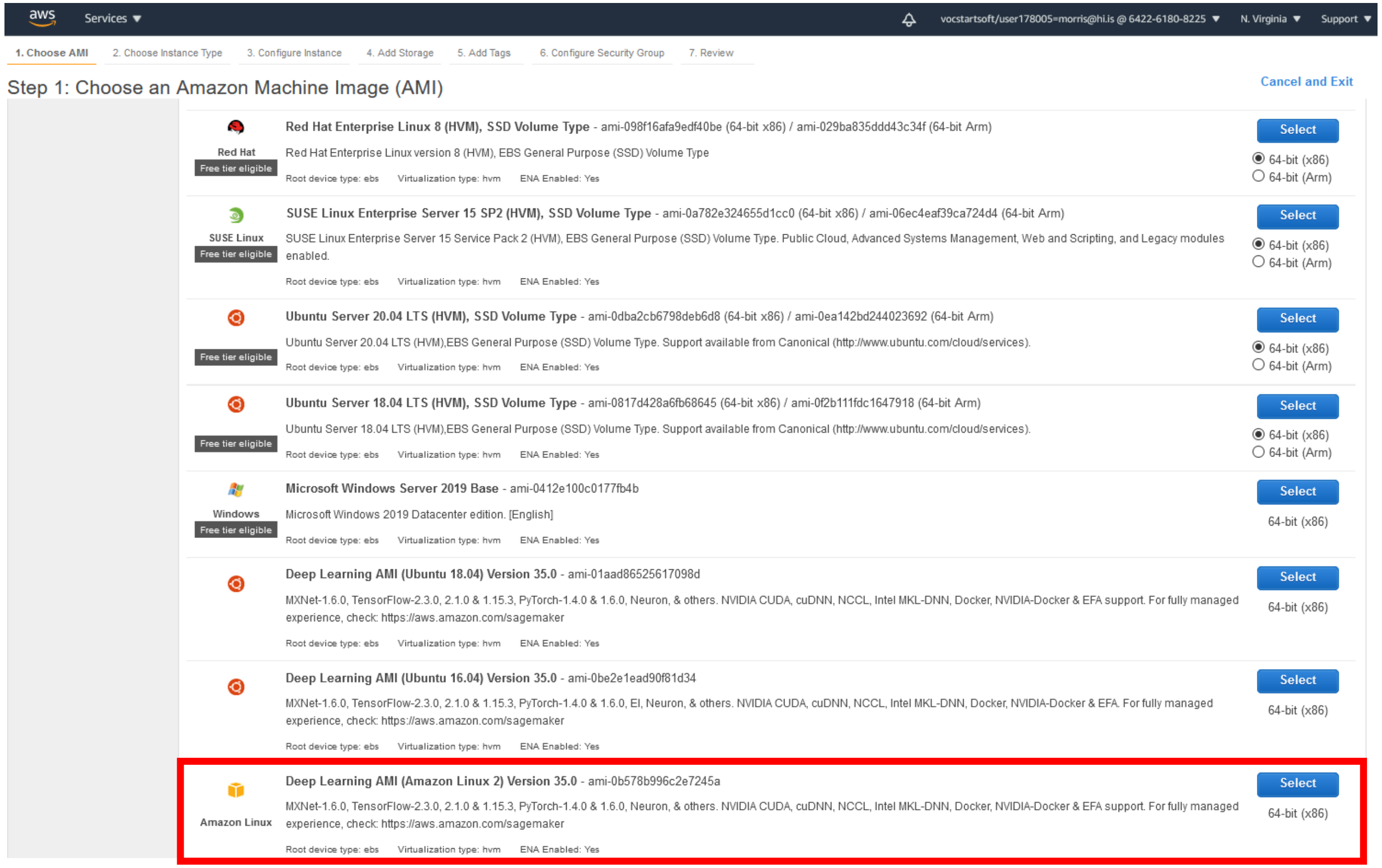
K Keras

[3] Keras Web page

- Keras
  - Often used in combination with low-level frameworks like Tensorflow

- **Keras is a high-level deep learning library implemented in Python that works on top of existing other rather low-level deep learning frameworks like Tensorflow, CNTK, or Theano**
- **Created deep learning models with Keras run seamlessly on CPU and GPU via low-level deep learning frameworks**
- **The key idea behind the Keras tool is to enable faster experimentation with deep networks**

AWS Elastic Compute Cloud (EC2) & Amazon Machine Images (AMIs)



Step 1: Choose an Amazon Machine Image (AMI)

Red Hat — Free tier eligible

Red Hat Enterprise Linux 8 (HVM), SSD Volume Type - ami-098f16afa9edf40be (64-bit x86) / ami-029ba835ddd43c34f (64-bit Arm)

Red Hat Enterprise Linux version 8 (HVM), EBS General Purpose (SSD) Volume Type

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select ● 64-bit (x86) ○ 64-bit (Arm)

SUSE Linux — Free tier eligible

SUSE Linux Enterprise Server 15 SP2 (HVM), SSD Volume Type - ami-0a782e324655d1cc0 (64-bit x86) / ami-06ec4eaf39ca724d4 (64-bit Arm)

SUSE Linux Enterprise Server 15 Service Pack 2 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select ● 64-bit (x86) ○ 64-bit (Arm)

Free tier eligible

Ubuntu Server 20.04 LTS (HVM), SSD Volume Type - ami-0dba2cb6798deb6d8 (64-bit x86) / ami-0ea142bd244023692 (64-bit Arm)

Ubuntu Server 20.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select ● 64-bit (x86) ○ 64-bit (Arm)

Free tier eligible

Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-0817d428a6fb68645 (64-bit x86) / ami-0f2b111fdc1647918 (64-bit Arm)

Ubuntu Server 18.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select ● 64-bit (x86) ○ 64-bit (Arm)

Windows — Free tier eligible

Microsoft Windows Server 2019 Base - ami-0412e100c0177fb4b

Microsoft Windows 2019 Datacenter edition. [English]

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select 64-bit (x86)

Deep Learning AMI (Ubuntu 18.04) Version 35.0 - ami-01aad86525617098d

MXNet-1.6.0, TensorFlow-2.3.0, 2.1.0 & 1.15.3, PyTorch-1.4.0 & 1.6.0, Neuron, & others. NVIDIA CUDA, cuDNN, NCCL, Intel MKL-DNN, Docker, NVIDIA-Docker & EFA support. For fully managed experience, check: https://aws.amazon.com/sagemaker

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select 64-bit (x86)

Deep Learning AMI (Ubuntu 16.04) Version 35.0 - ami-0be2e1ead90f81d34

MXNet-1.6.0, TensorFlow-2.3.0, 2.1.0 & 1.15.3, PyTorch-1.4.0 & 1.6.0, EI, Neuron, & others. NVIDIA CUDA, cuDNN, NCCL, Intel MKL-DNN, Docker, NVIDIA-Docker & EFA. For fully managed experience, check: https://aws.amazon.com/sagemaker

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select 64-bit (x86)

Amazon Linux

Deep Learning AMI (Amazon Linux 2) Version 35.0 - ami-0b578b996c2e7245a

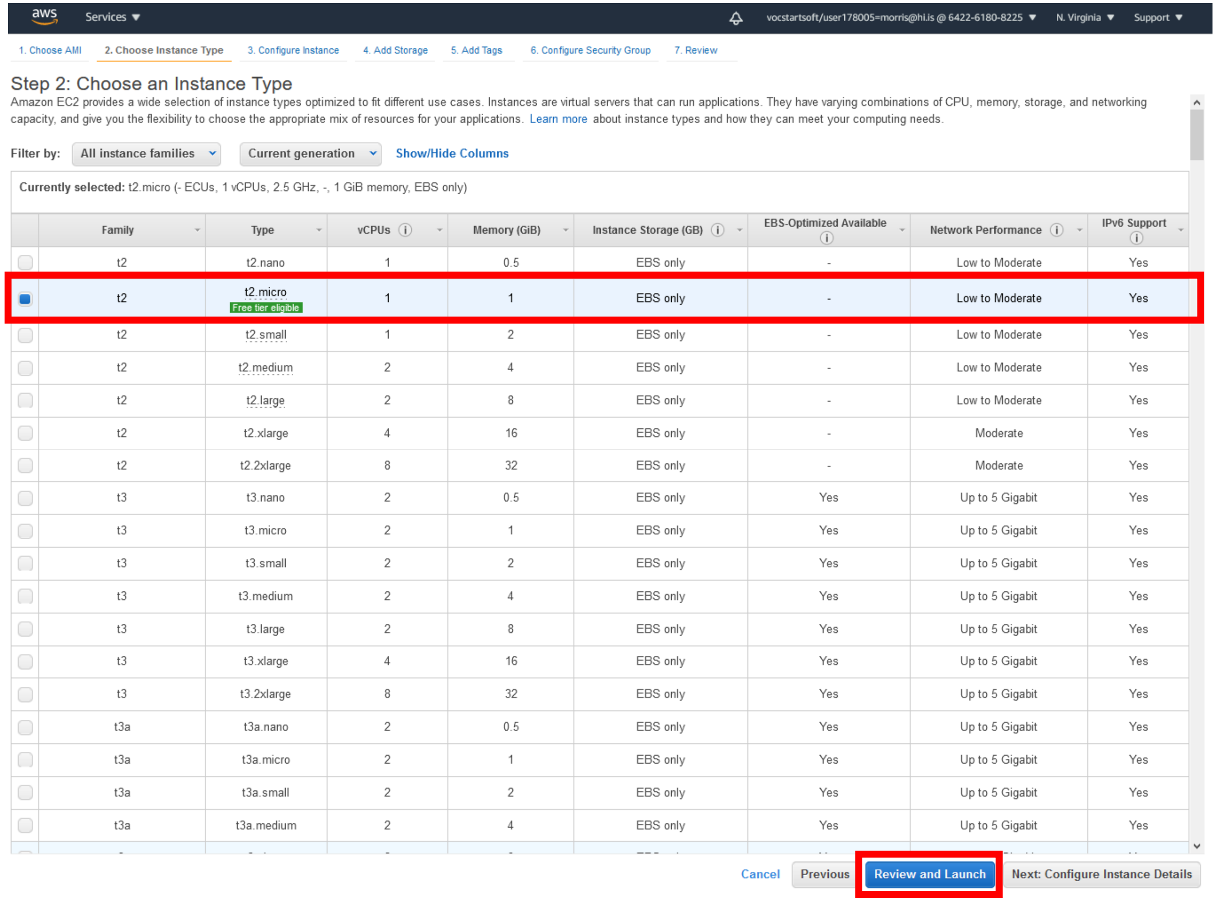
MXNet-1.6.0, TensorFlow-2.3.0, 2.1.0 & 1.15.3, PyTorch-1.4.0 & 1.6.0, Neuron, & others. NVIDIA CUDA, cuDNN, NCCL, Intel MKL-DNN, Docker, NVIDIA-Docker & EFA support. For fully managed experience, check: https://aws.amazon.com/sagemaker

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select 64-bit (x86)

- **AWS Amazon Machine Images (AMIs) are templates that contain the software configuration (e.g., operating system, libraries, application server, and applications) required to launch a EC2 virtual server instance for a specific purpose**
- **AWS EC2 AMI offers solutions that enormously simplify the deployment of required machine learning and deep learning stacks that can be complicated to make work together due to the many different software versions and fast moving technologies (e.g., different NVIDIA GPUs)**
- **AWS EC2 AMI The AMIs are independent from the underlying hardware infrastructure (i.e. concrete CPUs) and can be easily migrated (cf. Lecture 4) to other hardware – be aware of different hardware costs here**
- **Amazon offers pre-configured AMIs for deep learning consisting of preinstalled deep learning packages such as MXNet, TensorFlow, PyTorch, Keras, etc.**
- **Pre-configured AMIs for deep learning feature preinstalled GPU NVIDIA CUDA, cuDNN, and NCCL libraries that usually requires a lot of efforts in installation and version checks with deep learning packages**

Choose EC2 Instance for AMI & Review Costs Using Free Tier Eligible CPUs



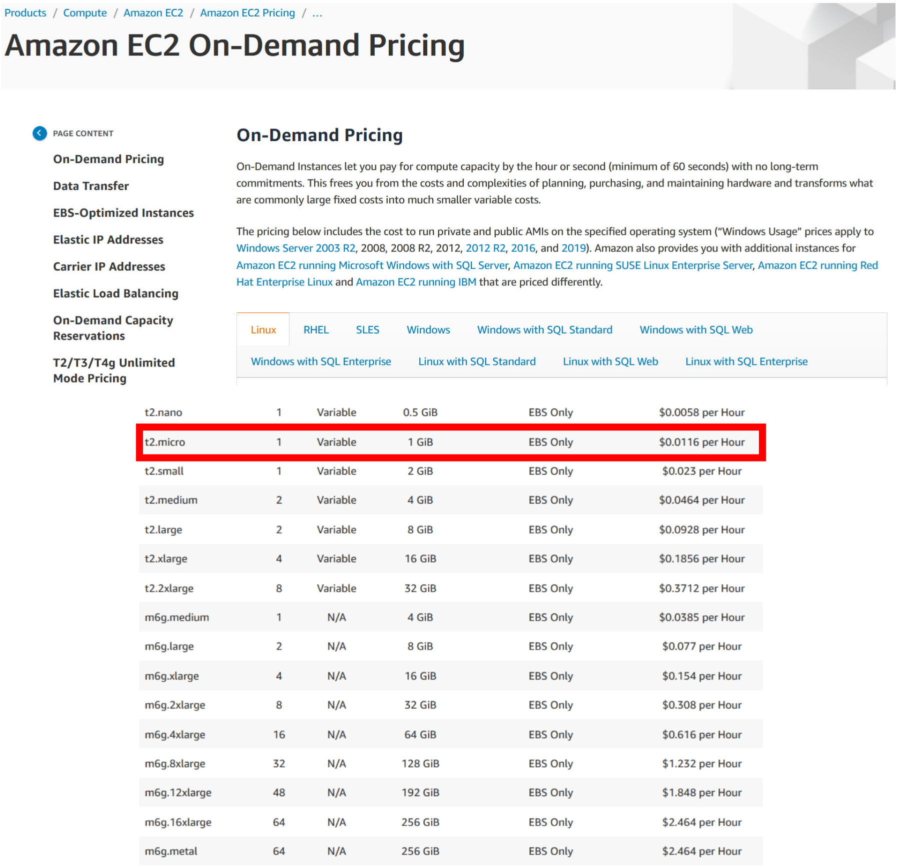
Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

Filter by: All instance families · Current generation · Show/Hide Columns

Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
□	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
■	t2	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
□	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
□	t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
□	t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
□	t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes
□	t2	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
□	t3	t3.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes
□	t3	t3.micro	2	1	EBS only	Yes	Up to 5 Gigabit	Yes
□	t3	t3.small	2	2	EBS only	Yes	Up to 5 Gigabit	Yes
□	t3	t3.medium	2	4	EBS only	Yes	Up to 5 Gigabit	Yes
□	t3	t3.large	2	8	EBS only	Yes	Up to 5 Gigabit	Yes
□	t3	t3.xlarge	4	16	EBS only	Yes	Up to 5 Gigabit	Yes
□	t3	t3.2xlarge	8	32	EBS only	Yes	Up to 5 Gigabit	Yes
□	t3a	t3a.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes
□	t3a	t3a.micro	2	1	EBS only	Yes	Up to 5 Gigabit	Yes
□	t3a	t3a.small	2	2	EBS only	Yes	Up to 5 Gigabit	Yes
□	t3a	t3a.medium	2	4	EBS only	Yes	Up to 5 Gigabit	Yes



Amazon EC2 On-Demand Pricing

On-Demand Pricing

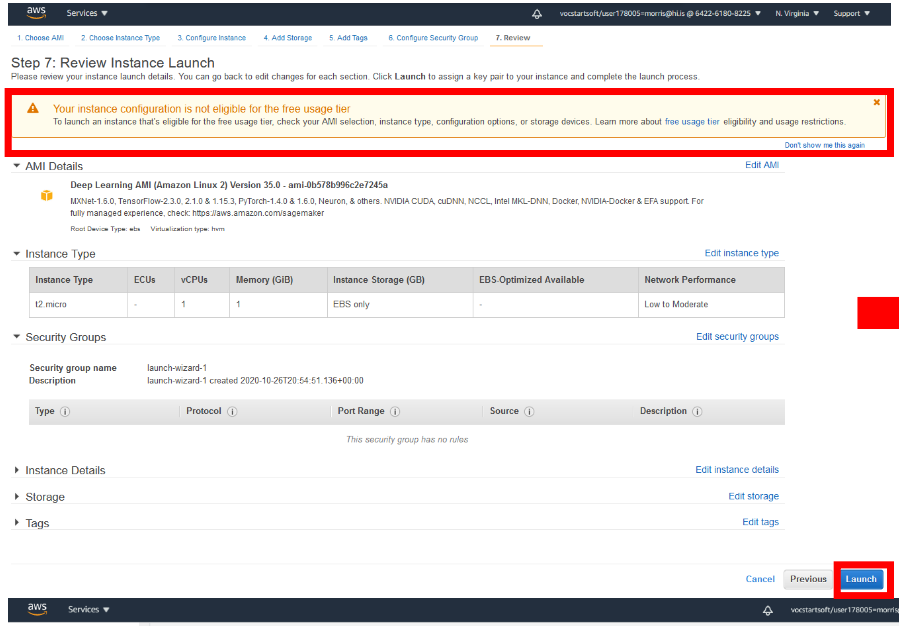
On-Demand Instances let you pay for compute capacity by the hour or second (minimum of 60 seconds) with no long-term commitments. This frees you from the costs and complexities of planning, purchasing, and maintaining hardware and transforms what are commonly large fixed costs into much smaller variable costs.

The pricing below includes the cost to run private and public AMIs on the specified operating system ("Windows Usage" prices apply to Windows Server 2003 R2, 2008, 2008 R2, 2012, 2012 R2, 2016, and 2019). Amazon also provides you with additional instances for Amazon EC2 running Microsoft Windows with SQL Server, Amazon EC2 running SUSE Linux Enterprise Server, Amazon EC2 running Red Hat Enterprise Linux and Amazon EC2 running IBM that are priced differently.

t2.nano	1	Variable	0.5 GiB	EBS Only	$0.0058 per Hour
t2.micro	1	Variable	1 GiB	EBS Only	$0.0116 per Hour
t2.small	1	Variable	2 GiB	EBS Only	$0.023 per Hour
t2.medium	2	Variable	4 GiB	EBS Only	$0.0464 per Hour
t2.large	2	Variable	8 GiB	EBS Only	$0.0928 per Hour
t2.xlarge	4	Variable	16 GiB	EBS Only	$0.1856 per Hour
t2.2xlarge	8	Variable	32 GiB	EBS Only	$0.3712 per Hour
m6g.medium	1	N/A	4 GiB	EBS Only	$0.0385 per Hour
m6g.large	2	N/A	8 GiB	EBS Only	$0.077 per Hour
m6g.xlarge	4	N/A	16 GiB	EBS Only	$0.154 per Hour
m6g.2xlarge	8	N/A	32 GiB	EBS Only	$0.308 per Hour
m6g.4xlarge	16	N/A	64 GiB	EBS Only	$0.616 per Hour
m6g.8xlarge	32	N/A	128 GiB	EBS Only	$1.232 per Hour
m6g.12xlarge	48	N/A	192 GiB	EBS Only	$1.848 per Hour
m6g.16xlarge	64	N/A	256 GiB	EBS Only	$2.464 per Hour
m6g.metal	64	N/A	256 GiB	EBS Only	$2.464 per Hour

[8] AWS EC2 Pricing

Review & Launching EC2 Instance with AMI – Problems with Free Usage Tier?!



Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

Your instance configuration is not eligible for the free usage tier

To launch an instance that's eligible for the free usage tier, check your AMI selection, instance type, configuration options, or storage devices. Learn more about free usage tier eligibility and usage restrictions.

Don't show me this again

▾ AMI Details — Edit AMI

Deep Learning AMI (Amazon Linux 2) Version 35.0 - ami-0b578b996c2e7245a

MXNet-1.6.0, TensorFlow-2.3.0, 2.1.0 & 1.15.3, PyTorch-1.4.0 & 1.6.0, Neuron, & others. NVIDIA CUDA, cuDNN, NCCL, Intel MKL-DNN, Docker, NVIDIA-Docker & EFA support. For fully managed experience, check: https://aws.amazon.com/sagemaker

Root Device Type: ebs   Virtualization type: hvm

▾ Instance Type — Edit instance type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	-	1	1	EBS only	-	Low to Moderate

▾ Security Groups — Edit security groups

Security group name: launch-wizard-1
Description: launch-wizard-1 created 2020-10-26T20:54:51.136+00:00

Type ⓘ	Protocol ⓘ	Port Range ⓘ	Source ⓘ	Description ⓘ

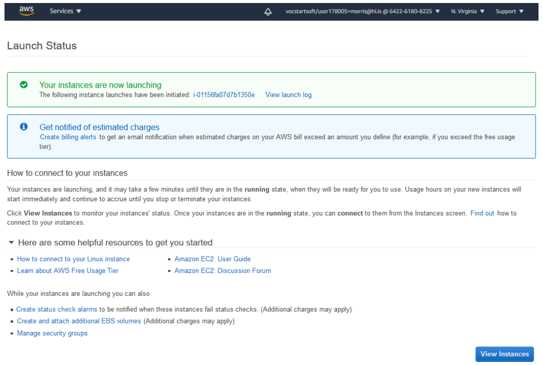
This security group has no rules

▸ Instance Details — Edit instance details

▸ Storage — Edit storage

▸ Tags — Edit tags

Cancel | Previous | Launch



Launch Status

Your instances are now launching

The following instance launches have been initiated: i-01156fa07d7b1350e   View launch log

Get notified of estimated charges

Create billing alerts to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier).

How to connect to your instances

Your instances are launching, and it may take a few minutes until they are in the **running** state, when they will be ready for you to use. Usage hours on your new instances will start immediately and continue to accrue until you stop or terminate your instances.

Click **View Instances** to monitor your instances' status. Once your instances are in the **running** state, you can **connect** to them from the Instances screen. Find out how to connect to your instances.

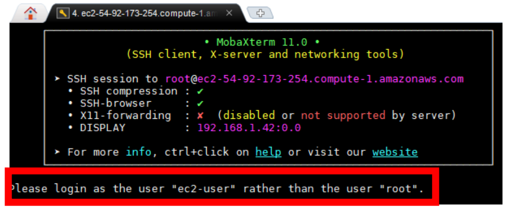
▾ Here are some helpful resources to get you started

- How to connect to your Linux instance
- Learn about AWS Free Usage Tier
- Amazon EC2: User Guide
- Amazon EC2: Discussion Forum

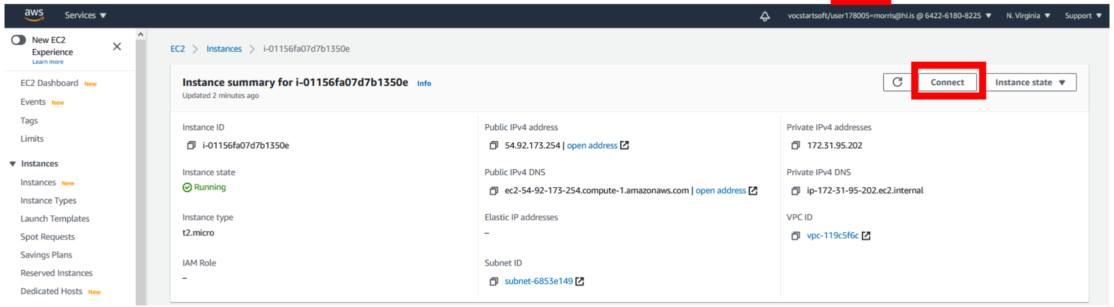
While your instances are launching you can also

- Create status check alarms to be notified when these instances fail status checks. (Additional charges may apply)
- Create and attach additional EBS volumes (Additional charges may apply)
- Manage security groups

View Instances



```
                    • MobaXterm 11.0 •
          (SSH client, X-server and networking tools)

 ➤ SSH session to root@ec2-54-92-173-254.compute-1.amazonaws.com
   • SSH compression : ✓
   • SSH-browser     : ✓
   • X11-forwarding  : ✗  (disabled or not supported by server)
   • DISPLAY         : 192.168.1.42:0.0

 ➤ For more info, ctrl+click on help or visit our website

Please login as the user "ec2-user" rather than the user "root".
```



Instance summary for i-01156fa07d7b1350e Info

Updated 2 minutes ago

Instance ID: i-01156fa07d7b1350e

Instance state: Running

Instance type: t2.micro

IAM Role: –

Public IPv4 address: 54.92.173.254 | open address

Public IPv4 DNS: ec2-54-92-173-254.compute-1.amazonaws.com | open address

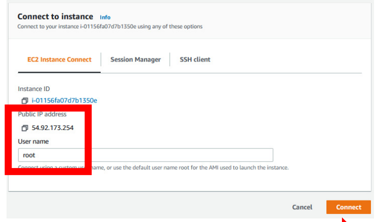
Elastic IP addresses: –

Subnet ID: subnet-6853e149

Private IPv4 addresses: 172.31.95.202

Private IPv4 DNS: ip-172-31-95-202.ec2.internal

VPC ID: vpc-119c5f6c



Connect to instance Info

Connect to your instance i-01156fa07d7b1350e using any of these options

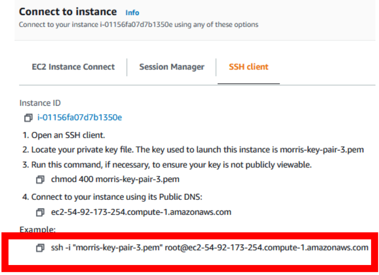
Instance ID: i-01156fa07d7b1350e

Public IP address: 54.92.173.254

User name: root

Connect using a custom user name, or use the default user name root for the AMI used to launch the instance.

e.g. Secure Shell (SSH) access with key-pairs



Connect to instance Info

Connect to your instance i-01156fa07d7b1350e using any of these options

Instance ID: i-01156fa07d7b1350e

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is morris-key-pair-3.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.

   chmod 400 morris-key-pair-3.pem

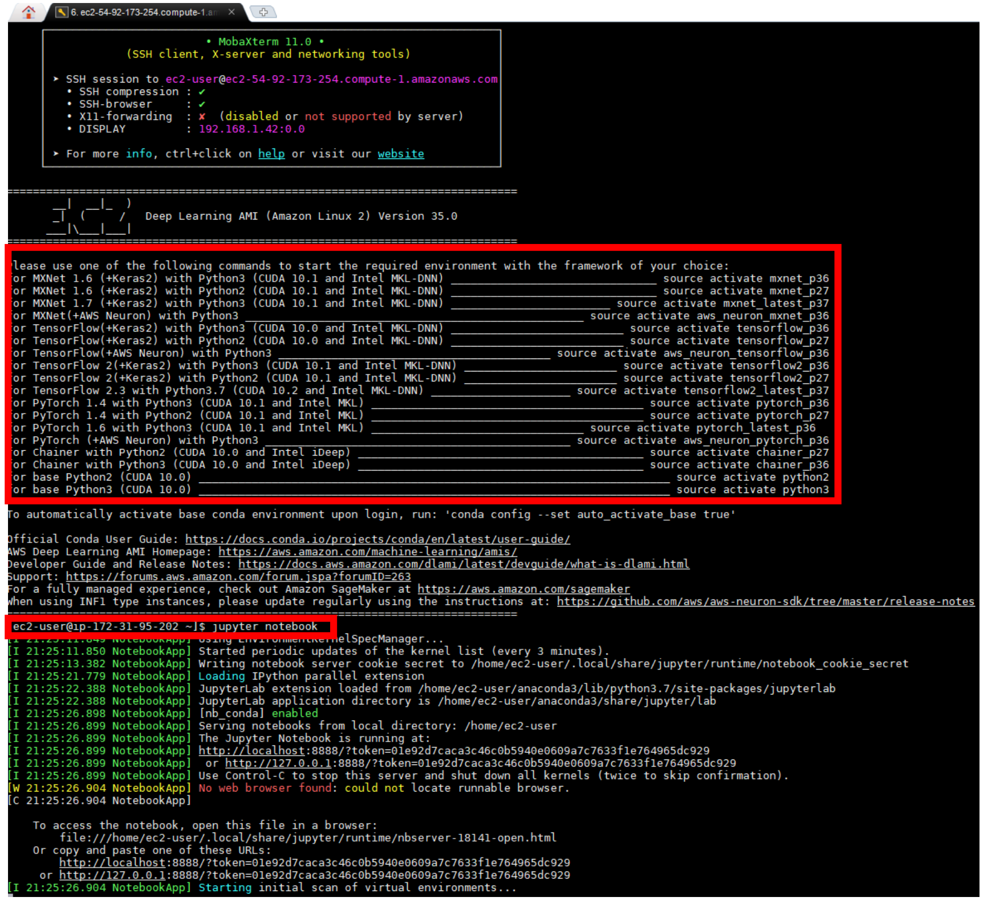
4. Connect to your instance using its Public DNS:

   ec2-54-92-173-254.compute-1.amazonaws.com

Example:

ssh -i "morris-key-pair-3.pem" root@ec2-54-92-173-254.compute-1.amazonaws.com

Using SSH Client to Connect to AWS EC2 AMI Instance & Jupyter Notebooks

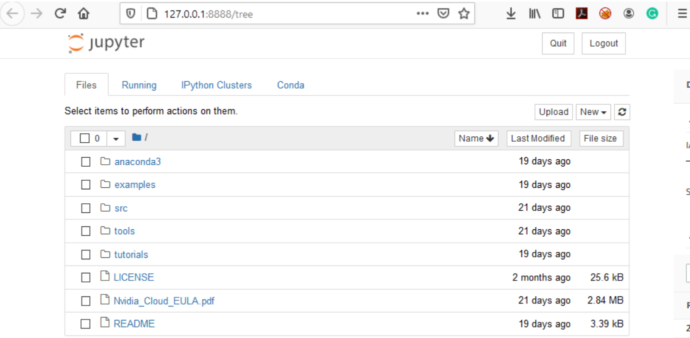


6. ec2-54-92-173-254.compute-1.am

```
                        • MobaXterm 11.0 •
            (SSH client, X-server and networking tools)

 ➤ SSH session to ec2-user@ec2-54-92-173-254.compute-1.amazonaws.com
   • SSH compression : ✓
   • SSH-browser     : ✓
   • X11-forwarding  : ✗  (disabled or not supported by server)
   • DISPLAY         : 192.168.1.42:0.0

 ➤ For more info, ctrl+click on help or visit our website

=============================================================================
       __|  __|_  )
       _|  (     /   Deep Learning AMI (Amazon Linux 2) Version 35.0
      ___|\___|___|
=============================================================================

lease use one of the following commands to start the required environment with the framework of your choice:
or MXNet 1.6 (+Keras2) with Python3 (CUDA 10.1 and Intel MKL-DNN) _______________________ source activate mxnet_p36
or MXNet 1.6 (+Keras2) with Python2 (CUDA 10.1 and Intel MKL-DNN) _______________________ source activate mxnet_p27
or MXNet 1.7 (+Keras2) with Python3 (CUDA 10.1 and Intel MKL-DNN) __________________ source activate mxnet_latest_p37
or MXNet(+AWS Neuron) with Python3 _______________________________________ source activate aws_neuron_mxnet_p36
or TensorFlow(+Keras2) with Python3 (CUDA 10.0 and Intel MKL-DNN) ___________________ source activate tensorflow_p36
or TensorFlow(+Keras2) with Python2 (CUDA 10.0 and Intel MKL-DNN) ___________________ source activate tensorflow_p27
or TensorFlow(+AWS Neuron) with Python3 ______________________________ source activate aws_neuron_tensorflow_p36
or TensorFlow 2(+Keras2) with Python3 (CUDA 10.1 and Intel MKL-DNN) _________________ source activate tensorflow2_p36
or TensorFlow 2(+Keras2) with Python2 (CUDA 10.1 and Intel MKL-DNN) _________________ source activate tensorflow2_p27
or TensorFlow 2.3 with Python3.7 (CUDA 10.2 and Intel MKL-DNN) ________________ source activate tensorflow2_latest_p37
or PyTorch 1.4 with Python3 (CUDA 10.1 and Intel MKL) ______________________________ source activate pytorch_p36
or PyTorch 1.4 with Python2 (CUDA 10.1 and Intel MKL) ______________________________ source activate pytorch_p27
or PyTorch 1.6 with Python3 (CUDA 10.1 and Intel MKL) _________________________ source activate pytorch_latest_p36
or PyTorch (+AWS Neuron) with Python3 _______________________________ source activate aws_neuron_pytorch_p36
or Chainer with Python2 (CUDA 10.0 and Intel iDeep) ________________________________ source activate chainer_p27
or Chainer with Python3 (CUDA 10.0 and Intel iDeep) ________________________________ source activate chainer_p36
or base Python2 (CUDA 10.0) ______________________________________________________ source activate python2
or base Python3 (CUDA 10.0) ______________________________________________________ source activate python3

To automatically activate base conda environment upon login, run: 'conda config --set auto_activate_base true'

Official Conda User Guide: https://docs.conda.io/projects/conda/en/latest/user-guide/
AWS Deep Learning AMI Homepage: https://aws.amazon.com/machine-learning/amis/
Developer Guide and Release Notes: https://docs.aws.amazon.com/dlami/latest/devguide/what-is-dlami.html
Support: https://forums.aws.amazon.com/forum.jspa?forumID=263
For a fully managed experience, check out Amazon SageMaker at https://aws.amazon.com/sagemaker
When using INF1 type instances, please update regularly using the instructions at: https://github.com/aws/aws-neuron-sdk/tree/master/release-notes
=============================================================================
ec2-user@ip-172-31-95-202 ~]$ jupyter notebook
[I 21:25:11.849 NotebookApp] Using EnvironmentKernelSpecManager...
[I 21:25:11.850 NotebookApp] Started periodic updates of the kernel list (every 3 minutes).
[I 21:25:13.382 NotebookApp] Writing notebook server cookie secret to /home/ec2-user/.local/share/jupyter/runtime/notebook_cookie_secret
[I 21:25:21.779 NotebookApp] Loading IPython parallel extension
[I 21:25:22.388 NotebookApp] JupyterLab extension loaded from /home/ec2-user/anaconda3/lib/python3.7/site-packages/jupyterlab
[I 21:25:22.388 NotebookApp] JupyterLab application directory is /home/ec2-user/anaconda3/share/jupyter/lab
[I 21:25:26.898 NotebookApp] [nb_conda] enabled
[I 21:25:26.899 NotebookApp] Serving notebooks from local directory: /home/ec2-user
[I 21:25:26.899 NotebookApp] The Jupyter Notebook is running at:
[I 21:25:26.899 NotebookApp] http://localhost:8888/?token=01e92d7caca3c46c0b5940e0609a7c7633f1e764965dc929
[I 21:25:26.899 NotebookApp]  or http://127.0.0.1:8888/?token=01e92d7caca3c46c0b5940e0609a7c7633f1e764965dc929
[I 21:25:26.899 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 21:25:26.904 NotebookApp] No web browser found: could not locate runnable browser.
[C 21:25:26.904 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///home/ec2-user/.local/share/jupyter/runtime/nbserver-18141-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=01e92d7caca3c46c0b5940e0609a7c7633f1e764965dc929
     or http://127.0.0.1:8888/?token=01e92d7caca3c46c0b5940e0609a7c7633f1e764965dc929
[I 21:25:26.904 NotebookApp] Starting initial scan of virtual environments...
```

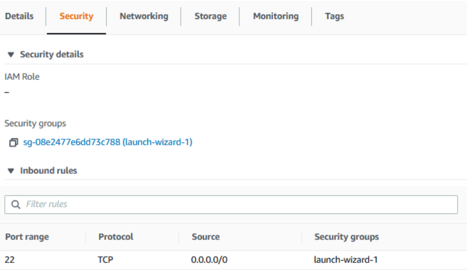
- Jupyter Notebook
  - Use a browser with http://localhost:8888
  - Use forwarding ssh connection (-L) to connect to localhost although we actually connect to the Amazon AMI instance
  - E.g., ssh -L 127.0.0.1:8888:127.0.0.1:8888 -i ~/Desktop/morris-key-pair-3.ppk ec2-user@ec2-54-92-173-254.compute-1.amazonaws.com
  - It appears we work local, but indeed we work remotely in the cloud

[9] Jupyter



Select items to perform actions on them.

Name	Last Modified	File size
□ anaconda3	19 days ago	
□ examples	19 days ago	
□ src	21 days ago	
□ tools	21 days ago	
□ tutorials	19 days ago	
□ LICENSE	2 months ago	25.6 kB
□ Nvidia_Cloud_EULA.pdf	21 days ago	2.84 MB
□ README	19 days ago	3.39 kB



▼ Security details

IAM Role
–

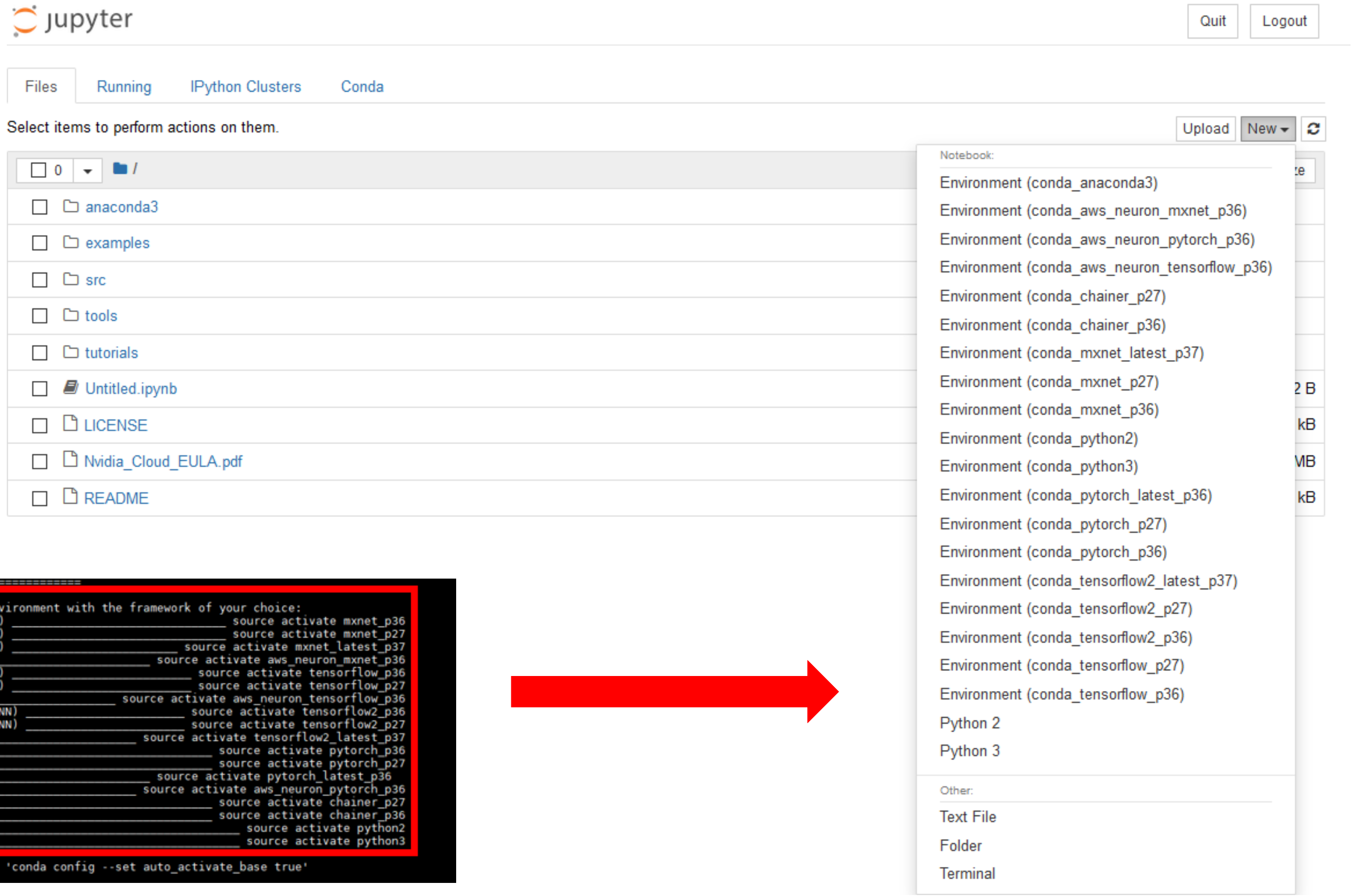
Security groups
sg-08e2477e6dd73c788 (launch-wizard-1)

▼ Inbound rules

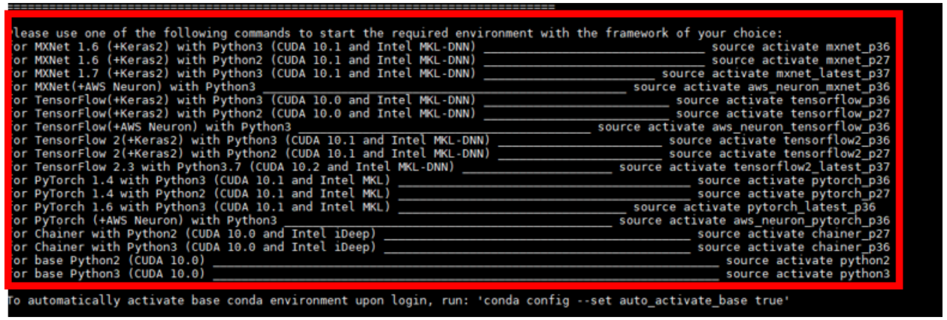
Port range	Protocol	Source	Security groups
22	TCP	0.0.0.0/0	launch-wizard-1

Using Jupyter with a Kernel & Machine & Deep Learning Software Configuration

[9] Jupyter

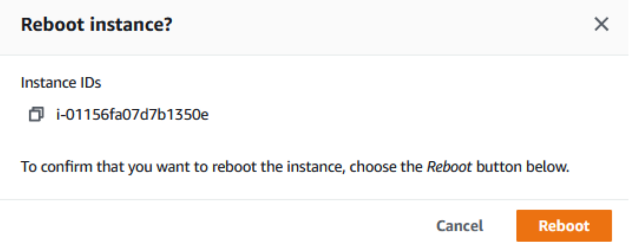


jupyter
Quit
Logout
Files
Running
IPython Clusters
Conda
Select items to perform actions on them.
Upload
New ▾
□ 0 ▾ /
□ anaconda3
□ examples
□ src
□ tools
□ tutorials
□ Untitled.ipynb
□ LICENSE
□ Nvidia_Cloud_EULA.pdf
□ README
Notebook:
Environment (conda_anaconda3)
Environment (conda_aws_neuron_mxnet_p36)
Environment (conda_aws_neuron_pytorch_p36)
Environment (conda_aws_neuron_tensorflow_p36)
Environment (conda_chainer_p27)
Environment (conda_chainer_p36)
Environment (conda_mxnet_latest_p37)
Environment (conda_mxnet_p27)
Environment (conda_mxnet_p36)
Environment (conda_python2)
Environment (conda_python3)
Environment (conda_pytorch_latest_p36)
Environment (conda_pytorch_p27)
Environment (conda_pytorch_p36)
Environment (conda_tensorflow2_latest_p37)
Environment (conda_tensorflow2_p27)
Environment (conda_tensorflow2_p36)
Environment (conda_tensorflow_p27)
Environment (conda_tensorflow_p36)
Python 2
Python 3
Other:
Text File
Folder
Terminal



```
==========================================================================
lease use one of the following commands to start the required environment with the framework of your choice:
or MXNet 1.6 (+Keras2) with Python3 (CUDA 10.1 and Intel MKL-DNN) _______________________________ source activate mxnet_p36
or MXNet 1.6 (+Keras2) with Python2 (CUDA 10.1 and Intel MKL-DNN) _______________________________ source activate mxnet_p27
or MXNet 1.7 (+Keras2) with Python3 (CUDA 10.1 and Intel MKL-DNN) _________________________ source activate mxnet_latest_p37
or MXNet(+AWS Neuron) with Python3 ___________________________________________________ source activate aws_neuron_mxnet_p36
or TensorFlow(+Keras2) with Python3 (CUDA 10.0 and Intel MKL-DNN) __________________________ source activate tensorflow_p36
or TensorFlow(+Keras2) with Python2 (CUDA 10.0 and Intel MKL-DNN) __________________________ source activate tensorflow_p27
or TensorFlow(+AWS Neuron) with Python3 _________________________________________ source activate aws_neuron_tensorflow_p36
or TensorFlow 2(+Keras2) with Python3 (CUDA 10.1 and Intel MKL-DNN) _______________________ source activate tensorflow2_p36
or TensorFlow 2(+Keras2) with Python2 (CUDA 10.1 and Intel MKL-DNN) _______________________ source activate tensorflow2_p27
or TensorFlow 2.3 with Python3.7 (CUDA 10.2 and Intel MKL-DNN) _____________________ source activate tensorflow2_latest_p37
or PyTorch 1.4 with Python3 (CUDA 10.1 and Intel MKL) ______________________________________ source activate pytorch_p36
or PyTorch 1.4 with Python2 (CUDA 10.1 and Intel MKL) ______________________________________ source activate pytorch_p27
or PyTorch 1.6 with Python3 (CUDA 10.1 and Intel MKL) _______________________________ source activate pytorch_latest_p36
or PyTorch (+AWS Neuron) with Python3 ______________________________________________ source activate aws_neuron_pytorch_p36
or Chainer with Python2 (CUDA 10.0 and Intel iDeep) ________________________________________ source activate chainer_p27
or Chainer with Python3 (CUDA 10.0 and Intel iDeep) ________________________________________ source activate chainer_p36
or base Python2 (CUDA 10.0) _____________________________________________________________________ source activate python2
or base Python3 (CUDA 10.0) _____________________________________________________________________ source activate python3
To automatically activate base conda environment upon login, run: 'conda config --set auto_activate_base true'
```

Lessons Learned – Dead Environments in the Cloud & Reboot Cloud Instance

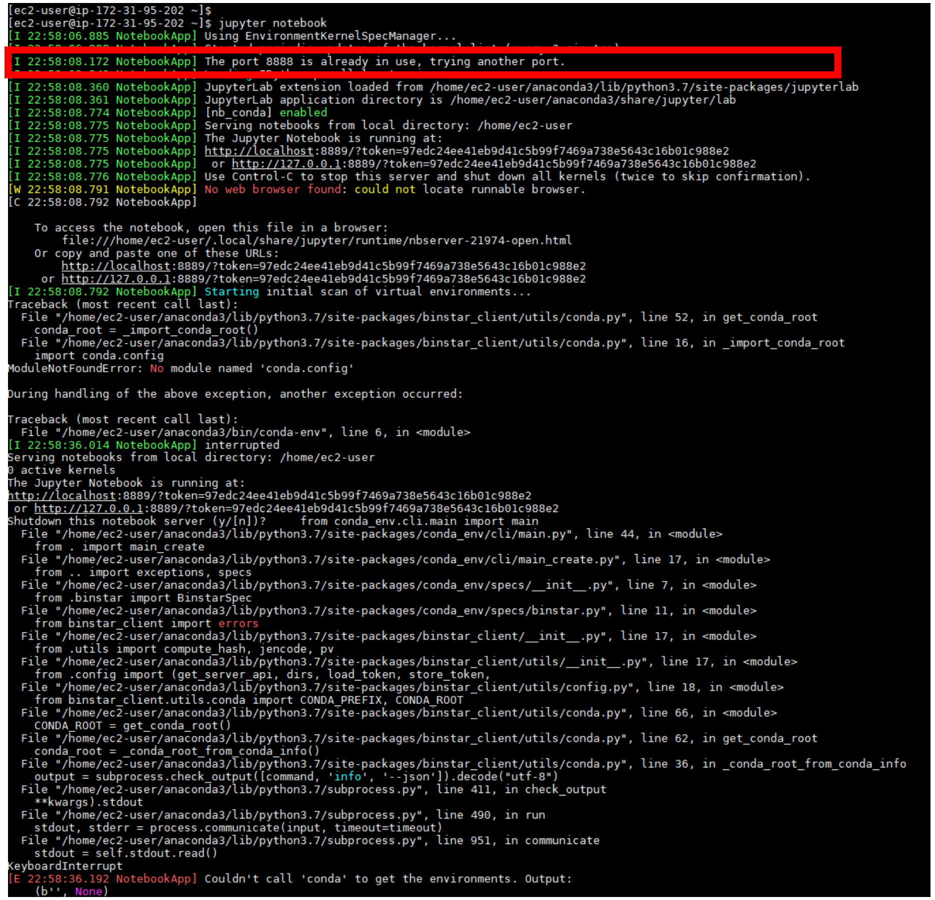


Reboot instance?

Instance IDs

i-01156fa07d7b1350e

To confirm that you want to reboot the instance, choose the *Reboot* button below.



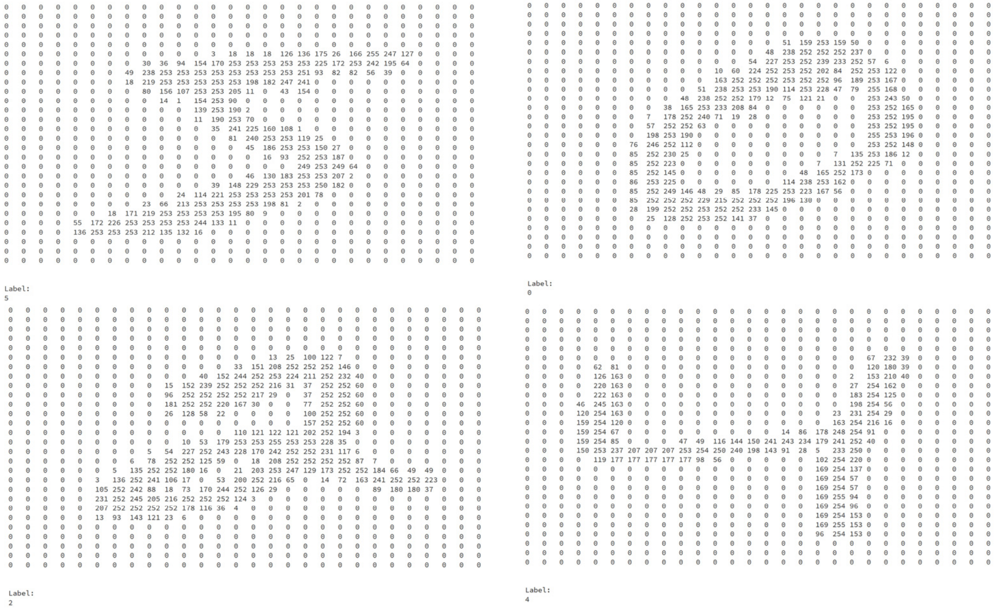
```
[ec2-user@ip-172-31-95-202 ~]$
[ec2-user@ip-172-31-95-202 ~]$ jupyter notebook
[I 22:58:06.885 NotebookApp] Using EnvironmentKernelSpecManager...
[I 22:58:08.172 NotebookApp] The port 8888 is already in use, trying another port.
[I 22:58:08.360 NotebookApp] JupyterLab extension loaded from /home/ec2-user/anaconda3/lib/python3.7/site-packages/jupyterlab
[I 22:58:08.361 NotebookApp] JupyterLab application directory is /home/ec2-user/anaconda3/share/jupyter/lab
[I 22:58:08.774 NotebookApp] [nb_conda] enabled
[I 22:58:08.775 NotebookApp] Serving notebooks from local directory: /home/ec2-user
[I 22:58:08.775 NotebookApp] The Jupyter Notebook is running at:
[I 22:58:08.775 NotebookApp] http://localhost:8889/?token=97edc24ee41eb9d41c5b99f7469a738e5643c16b01c988e2
[I 22:58:08.775 NotebookApp]  or http://127.0.0.1:8889/?token=97edc24ee41eb9d41c5b99f7469a738e5643c16b01c988e2
[I 22:58:08.776 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 22:58:08.791 NotebookApp] No web browser found: could not locate runnable browser.
[C 22:58:08.792 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///home/ec2-user/.local/share/jupyter/runtime/nbserver-21974-open.html
    Or copy and paste one of these URLs:
        http://localhost:8889/?token=97edc24ee41eb9d41c5b99f7469a738e5643c16b01c988e2
     or http://127.0.0.1:8889/?token=97edc24ee41eb9d41c5b99f7469a738e5643c16b01c988e2
[I 22:58:08.792 NotebookApp] Starting initial scan of virtual environments...
Traceback (most recent call last):
  File "/home/ec2-user/anaconda3/lib/python3.7/site-packages/binstar_client/utils/conda.py", line 52, in get_conda_root
    conda_root = _import_conda_root()
  File "/home/ec2-user/anaconda3/lib/python3.7/site-packages/binstar_client/utils/conda.py", line 16, in _import_conda_root
    import conda.config
ModuleNotFoundError: No module named 'conda.config'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/home/ec2-user/anaconda3/bin/conda-env", line 6, in <module>
[I 22:58:36.014 NotebookApp] interrupted
Serving notebooks from local directory: /home/ec2-user
0 active kernels
The Jupyter Notebook is running at:
http://localhost:8889/?token=97edc24ee41eb9d41c5b99f7469a738e5643c16b01c988e2
 or http://127.0.0.1:8889/?token=97edc24ee41eb9d41c5b99f7469a738e5643c16b01c988e2
Shutdown this notebook server (y/[n])?     from conda_env.cli.main import main
  File "/home/ec2-user/anaconda3/lib/python3.7/site-packages/conda_env/cli/main.py", line 44, in <module>
    from . import main_create
  File "/home/ec2-user/anaconda3/lib/python3.7/site-packages/conda_env/cli/main_create.py", line 17, in <module>
    from .. import exceptions, specs
  File "/home/ec2-user/anaconda3/lib/python3.7/site-packages/conda_env/specs/__init__.py", line 7, in <module>
    from .binstar import BinstarSpec
  File "/home/ec2-user/anaconda3/lib/python3.7/site-packages/conda_env/specs/binstar.py", line 11, in <module>
    from binstar_client import errors
  File "/home/ec2-user/anaconda3/lib/python3.7/site-packages/binstar_client/__init__.py", line 17, in <module>
    from .utils import compute_hash, jencode, pv
  File "/home/ec2-user/anaconda3/lib/python3.7/site-packages/binstar_client/utils/__init__.py", line 17, in <module>
    from .config import (get_server_api, dirs, load_token, store_token,
  File "/home/ec2-user/anaconda3/lib/python3.7/site-packages/binstar_client/utils/config.py", line 18, in <module>
    from binstar_client.utils.conda import CONDA_PREFIX, CONDA_ROOT
  File "/home/ec2-user/anaconda3/lib/python3.7/site-packages/binstar_client/utils/conda.py", line 66, in <module>
    CONDA_ROOT = get_conda_root()
  File "/home/ec2-user/anaconda3/lib/python3.7/site-packages/binstar_client/utils/conda.py", line 62, in get_conda_root
    conda_root = _conda_root_from_conda_info()
  File "/home/ec2-user/anaconda3/lib/python3.7/site-packages/binstar_client/utils/conda.py", line 36, in _conda_root_from_conda_info
    output = subprocess.check_output([command, 'info', '--json']).decode("utf-8")
  File "/home/ec2-user/anaconda3/lib/python3.7/subprocess.py", line 411, in check_output
    **kwargs).stdout
  File "/home/ec2-user/anaconda3/lib/python3.7/subprocess.py", line 490, in run
    stdout, stderr = process.communicate(input, timeout=timeout)
  File "/home/ec2-user/anaconda3/lib/python3.7/subprocess.py", line 951, in communicate
    stdout = self.stdout.read()
KeyboardInterrupt
[E 22:58:36.192 NotebookApp] Couldn't call 'conda' to get the environments. Output:
    (b'', None)
```

- **Despite the fact that Clouds are often stable and production ready they show sometimes still faults and errors that are partly also related to the way of using them, e.g., not terminating properly the Jupyter environment**
- **Because Clouds run remotely on computing systems they can still continue to run even if the local Laptop has no SSH connection open nor is there an active browser window, i.e. remember that this still costs money even if you do not actively use the cloud resources**

MNIST Dataset – Training/Testing Datasets & One Character Encoding

- **Different phases in machine learning**
- **Training phases is a hypothesis search**
- **Testing phase checks if we are on the right track once the hypothesis is clear**
- **Validation phane for model selection (set fixed parameters and set model types)**

- Work on two disjoint datasets
  - One for training only (i.e. training set)
  - One for testing only (i.e. test set)
  - Exact seperation is rule of thumb per use case (e.g. 10 % training, 90% test)
  - Practice: If you get a dataset take immediately test data away (‘throw it into the corner and forget about it during modelling‘)
  - Once we learned from training data it has an ‘optimistic bias‘
  - Usually start by exploring the dataset and its format & labels

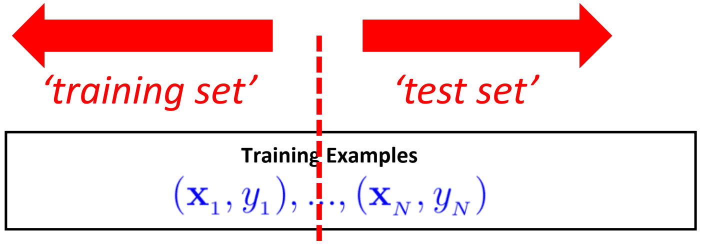


```
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   3   18  18  18  126 136 175 26  166 255 247 127 0   0   0   0
0   0   0   0   0   0   0   0   30  36  94  154 170 253 253 253 253 253 225 172 253 242 195 64  0   0   0   0
0   0   0   0   0   0   0   49  238 253 253 253 253 253 253 253 253 251 93  82  82  56  39  0   0   0   0   0
0   0   0   0   0   0   0   18  219 253 253 253 253 253 198 182 247 241 0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   80  156 107 253 253 205 11  0   43  154 0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   14  1   154 253 90  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   139 253 190 2   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   11  190 253 70  0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   35  241 225 160 108 1   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   81  240 253 253 119 25  0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   45  186 253 253 150 27  0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   16  93  252 253 187 0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   249 253 249 64  0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   46  130 183 253 253 207 2   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   39  148 229 253 253 253 250 182 0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   24  114 221 253 253 253 253 201 78  0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   23  66  213 253 253 253 253 198 81  2   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   18  171 219 253 253 253 253 195 80  9   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   55  172 226 253 253 253 253 244 133 11  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   136 253 253 253 212 135 132 16  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

Label:
5
```

```
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   51  159 253 159 50  0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   48  238 252 252 252 237 0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   54  227 253 252 239 233 252 57  6   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   10  60  224 252 253 252 202 84  252 253 122 0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   163 252 252 252 253 252 252 96  189 253 167 0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   51  238 253 253 190 114 253 228 47  79  255 168 0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   48  238 252 252 179 12  75  121 21  0   0   253 243 50  0   0   0   0   0
0   0   0   0   0   0   0   0   38  165 253 233 208 84  0   0   0   0   0   0   253 252 165 0   0   0   0   0
0   0   0   0   0   0   0   7   178 252 240 71  19  28  0   0   0   0   0   0   253 252 195 0   0   0   0   0
0   0   0   0   0   0   0   57  252 252 63  0   0   0   0   0   0   0   0   0   253 252 195 0   0   0   0   0
0   0   0   0   0   0   0   198 253 190 0   0   0   0   0   0   0   0   0   0   255 253 196 0   0   0   0   0
0   0   0   0   0   0   76  246 252 112 0   0   0   0   0   0   0   0   0   0   253 252 148 0   0   0   0   0
0   0   0   0   0   0   85  252 230 25  0   0   0   0   0   0   0   0   7   135 253 186 12  0   0   0   0   0
0   0   0   0   0   0   85  252 223 0   0   0   0   0   0   0   0   7   131 252 225 71  0   0   0   0   0   0
0   0   0   0   0   0   85  252 145 0   0   0   0   0   0   0   48  165 252 173 0   0   0   0   0   0   0   0
0   0   0   0   0   0   86  253 225 0   0   0   0   0   0   114 238 253 162 0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   85  252 249 146 48  29  85  178 225 253 223 167 56  0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   85  252 252 252 229 215 252 252 252 196 130 0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   28  199 252 252 253 252 252 233 145 0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   25  128 252 253 252 141 37  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

Label:
0
```

```
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   13  25  100 122 7   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   33  151 208 252 252 252 146 0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   40  152 244 252 253 224 211 252 232 40  0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   15  152 239 252 252 252 216 31  37  252 252 60  0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   96  252 252 252 252 217 29  0   37  252 252 60  0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   181 252 252 220 167 30  0   0   77  252 252 60  0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   26  128 58  22  0   0   0   0   100 252 252 60  0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   157 252 252 60  0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   110 121 122 121 202 252 194 3   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   10  53  179 253 253 255 253 253 228 35  0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   5   54  227 252 243 228 170 242 252 252 231 117 6   0   0   0   0   0   0   0
0   0   0   0   0   0   0   6   78  252 252 125 59  0   18  208 252 252 252 252 87  7   0   0   0   0   0   0
0   0   0   0   0   0   5   135 252 252 180 16  0   21  203 253 247 129 173 252 252 184 66  49  49  0   0   0
0   0   0   0   0   3   136 252 241 106 17  0   53  200 252 216 65  0   14  72  163 241 252 252 223 0   0   0
0   0   0   0   0   105 252 242 88  18  73  170 244 252 126 29  0   0   0   0   0   89  180 180 37  0   0   0
0   0   0   0   0   231 252 245 205 216 252 252 252 124 3   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   207 252 252 252 252 178 116 36  4   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   13  93  143 121 23  6   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

Label:
2
```

```
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   67  232 39  0   0   0   0   0
0   0   0   0   62  81  0   0   0   0   0   0   0   0   0   0   0   0   0   0   120 180 39  0   0   0   0   0
0   0   0   0   126 163 0   0   0   0   0   0   0   0   0   0   0   0   0   2   153 210 40  0   0   0   0   0
0   0   0   0   220 163 0   0   0   0   0   0   0   0   0   0   0   0   0   27  254 162 0   0   0   0   0   0
0   0   0   0   222 163 0   0   0   0   0   0   0   0   0   0   0   0   0   183 254 125 0   0   0   0   0   0
0   0   0   46  245 163 0   0   0   0   0   0   0   0   0   0   0   0   0   198 254 56  0   0   0   0   0   0
0   0   0   120 254 163 0   0   0   0   0   0   0   0   0   0   0   0   23  231 254 29  0   0   0   0   0   0
0   0   0   159 254 120 0   0   0   0   0   0   0   0   0   0   0   163 254 216 16  0   0   0   0   0   0   0
0   0   0   159 254 67  0   0   0   0   0   0   0   0   0   14  86  178 248 254 91  0   0   0   0   0   0   0
0   0   0   159 254 85  0   0   0   47  49  116 144 150 241 243 234 179 241 252 40  0   0   0   0   0   0   0
0   0   0   150 253 237 207 207 207 253 254 250 240 198 143 91  28  5   233 250 0   0   0   0   0   0   0   0
0   0   0   0   119 177 177 177 177 177 98  56  0   0   0   0   0   102 254 220 0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   169 254 137 0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   169 254 57  0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   169 254 57  0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   169 255 94  0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   169 254 96  0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   169 254 153 0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   169 255 153 0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   96  254 153 0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

Label:
4
```



‘training set’ ‘test set’

Training Examples

$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

(historical records, groundtruth data, examples)

MNIST Dataset – Data Exploration Script Training Data – Revisited

```python
import numpy as np
from keras.datasets import mnist
```

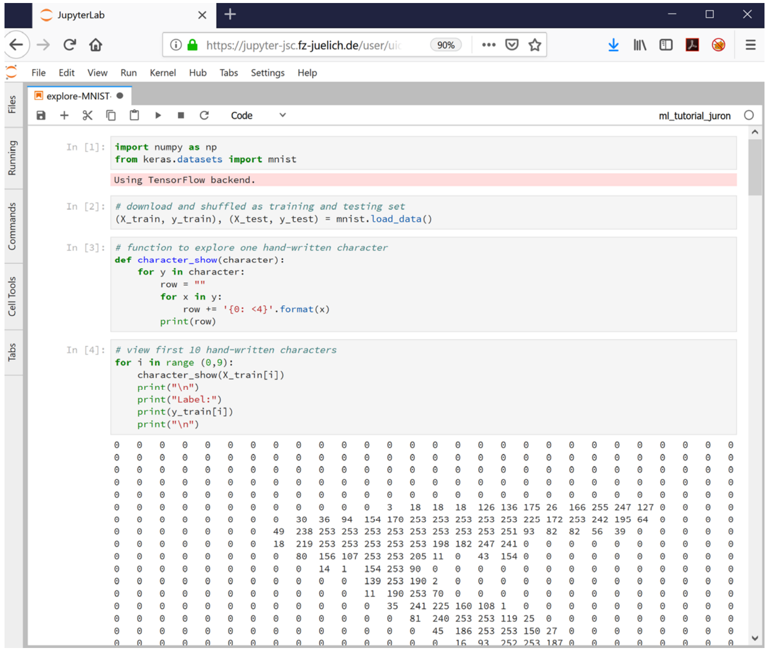
```python
# download and shuffled as training and testing set
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```python
# function to explore one hand-written character
def character_show(character):
    for y in character:
        row = ""
        for x in y:
            row += '{0: <4}'.format(x)
        print(row)
```

```python
# view first 10 hand-written characters
for i in range (0,9):
    character_show(X_train[i])
    print("\n")
    print("Label:")
    print(y_train[i])
    print("\n")
```

- Loading MNIST training datasets (X) with labels (Y) stored in a binary numpy format
- Format is 28 x 28 pixel values with grey level from 0 (white background) to 255 (black foreground)

- Small helper function that prints row-wise one ‘hand-written‘ character with the grey levels stored in training dataset
- Should reveal the nature of the number (aka label)

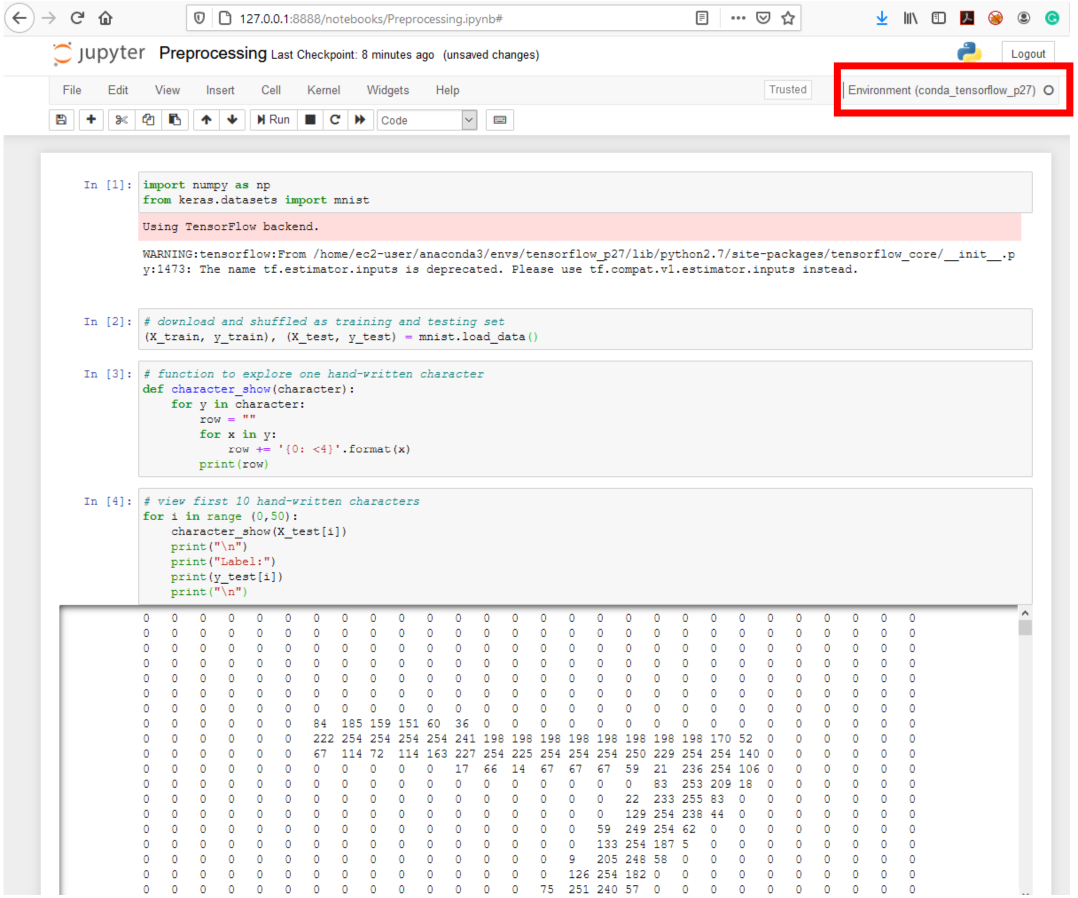

JupyterLab
https://jupyter-jsc.fz-juelich.de/user/ui
90%
File Edit View Run Kernel Hub Tabs Settings Help
Files
Running
Commands
Cell Tools
Tabs
explore-MNIST
Code
ml_tutorial_juron

```
In [1]: import numpy as np
        from keras.datasets import mnist

Using TensorFlow backend.

In [2]: # download and shuffled as training and testing set
        (X_train, y_train), (X_test, y_test) = mnist.load_data()

In [3]: # function to explore one hand-written character
        def character_show(character):
            for y in character:
                row = ""
                for x in y:
                    row += '{0: <4}'.format(x)
                print(row)

In [4]: # view first 10 hand-written characters
        for i in range (0,9):
            character_show(X_train[i])
            print("\n")
            print("Label:")
            print(y_train[i])
            print("\n")

0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   3   18  18  18  126 136 175 26  166 255 247 127 0   0   0   0
0   0   0   0   0   0   0   0   30  36  94  154 170 253 253 253 253 253 225 172 253 242 195 64  0   0   0   0
0   0   0   0   0   0   0   49  238 253 253 253 253 253 253 253 253 251 93  82  82  56  39  0   0   0   0   0
0   0   0   0   0   0   0   18  219 253 253 253 253 253 198 182 247 241 0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   80  156 107 253 253 205 11  0   43  154 0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   14  1   154 253 90  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   139 253 190 2   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   11  190 253 70  0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   35  241 225 160 108 1   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   81  240 253 253 119 25  0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   45  186 253 253 150 27  0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   16  93  252 253 187 0   0   0   0   0   0   0   0
```

[1] Jupyter @ JSC

- Example: loop of the training dataset (e.g. first 10 characters as shown here)
- At each loop interval the ‘hand-written‘ character (X) is printed in ‘matrix notation‘ & label (Y)

Data Inspection using Keras Dataset MNIST with Visualization in Jupyter


127.0.0.1:8888/notebooks/Preprocessing.ipynb#
jupyter
Preprocessing Last Checkpoint: 8 minutes ago (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Environment (conda_tensorflow_p27)
Run
Code

In [1]:

```
import numpy as np
from keras.datasets import mnist
```

```
Using TensorFlow backend.
```

```
WARNING:tensorflow:From /home/ec2-user/anaconda3/envs/tensorflow_p27/lib/python2.7/site-packages/tensorflow_core/__init__.p
y:1473: The name tf.estimator.inputs is deprecated. Please use tf.compat.v1.estimator.inputs instead.
```

In [2]:

```
# download and shuffled as training and testing set
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

In [3]:

```
# function to explore one hand-written character
def character_show(character):
    for y in character:
        row = ""
        for x in y:
            row += '{0: <4}'.format(x)
        print(row)
```

In [4]:

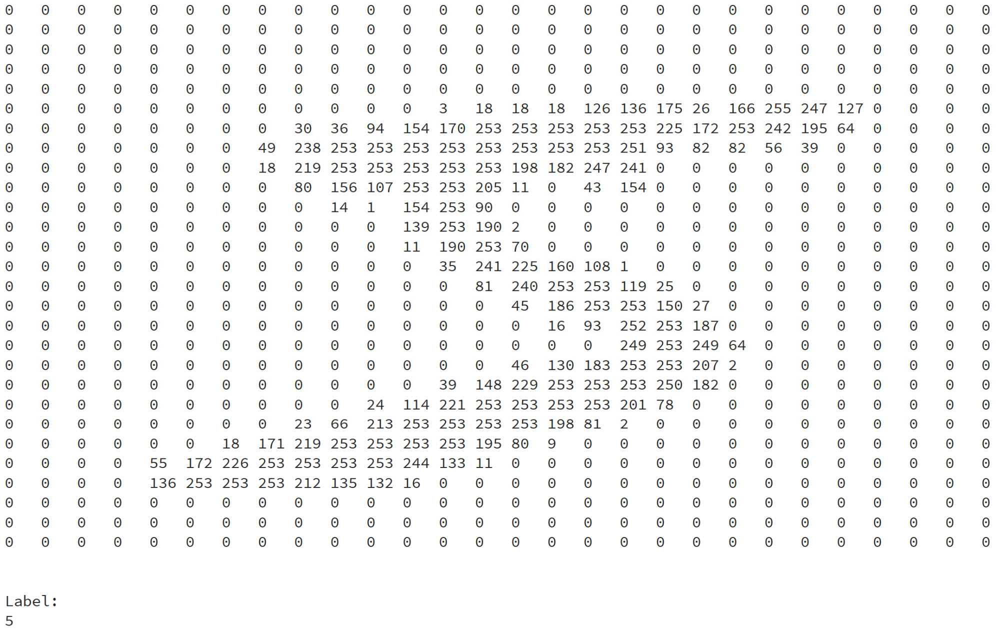
```
# view first 10 hand-written characters
for i in range (0,50):
    character_show(X_test[i])
    print("\n")
    print("Label:")
    print(y_test[i])
    print("\n")
```

```
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   84  185 159 151 60  36  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   222 254 254 254 254 241 198 198 198 198 198 198 198 198 170 52  0   0   0   0   0   0
0   0   0   0   0   0   67  114 72  114 163 227 254 225 254 254 254 250 229 254 254 140 0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   17  66  14  67  67  67  59  21  236 254 106 0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   83  253 209 18  0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   22  233 255 83  0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   129 254 238 44  0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   59  249 254 62  0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   133 254 187 5   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   9   205 248 58  0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   126 254 182 0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   75  251 240 57  0   0   0   0   0   0   0   0   0   0
```

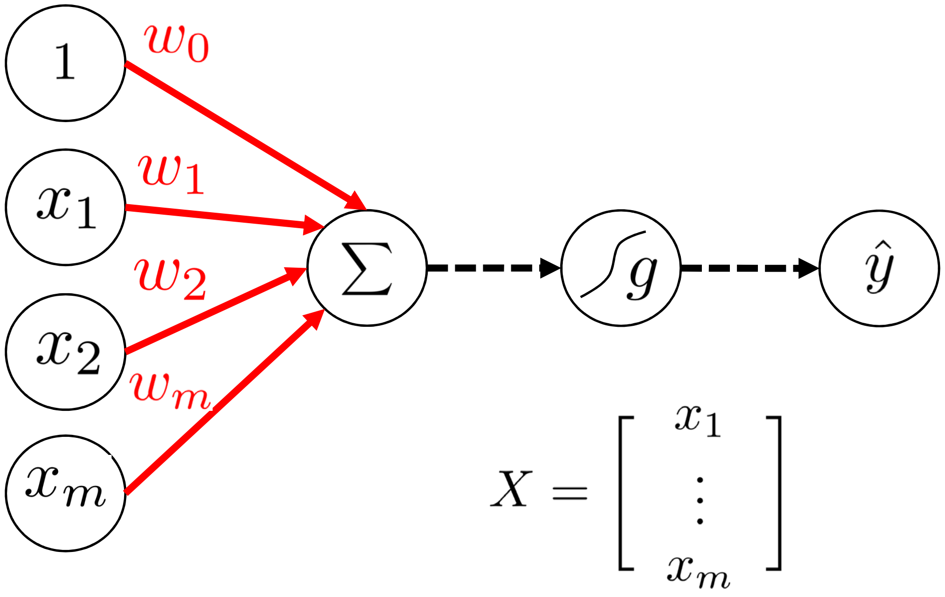
MNIST Dataset with Perceptron Learning Model – Need for Reshape

- Two dimensional dataset (28 x 28)
  - Does not fit well with input to Perceptron Model
  - Need to prepare the data even more
  - Reshape data → we need one long vector

- **Note that the reshape from two dimensional MNIST data to one long vector means that we loose the surrounding context**
- **Loosing the surrounding context is one factor why later in this lecture deep learning networks achieving essentially better performance by, e.g., keeping the surrounding context**



```
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   3   18  18  18  126 136 175 26  166 255 247 127 0   0   0   0
0   0   0   0   0   0   0   0   30  36  94  154 170 253 253 253 253 253 225 172 253 242 195 64  0   0   0   0
0   0   0   0   0   0   0   49  238 253 253 253 253 253 253 253 253 251 93  82  82  56  39  0   0   0   0   0
0   0   0   0   0   0   0   18  219 253 253 253 253 253 198 182 247 241 0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   80  156 107 253 253 205 11  0   43  154 0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   14  1   154 253 90  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   139 253 190 2   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   11  190 253 70  0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   35  241 225 160 108 1   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   81  240 253 253 119 25  0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   45  186 253 253 150 27  0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   16  93  252 253 187 0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   249 253 249 64  0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   46  130 183 253 253 207 2   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   39  148 229 253 253 253 250 182 0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   24  114 221 253 253 253 253 201 78  0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   23  66  213 253 253 253 253 198 81  2   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   18  171 219 253 253 253 253 195 80  9   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   55  172 226 253 253 253 253 244 133 11  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   136 253 253 253 212 135 132 16  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0


Label:
5
```

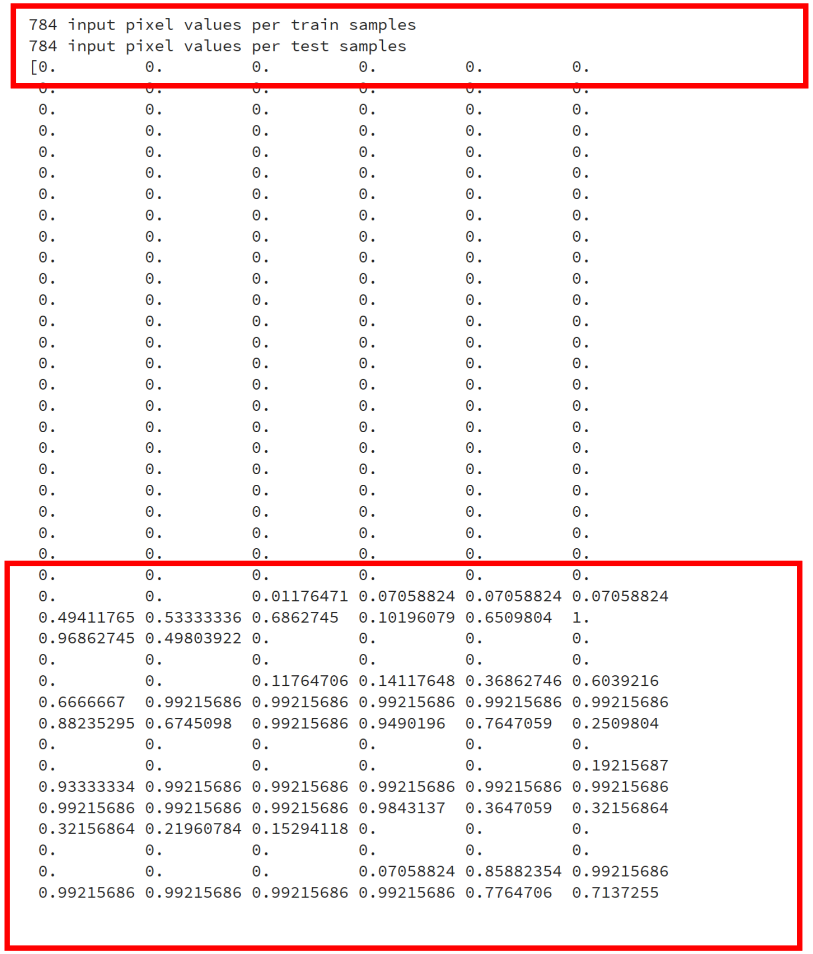


$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$

MNIST Dataset – Reshape & Normalization – Example

(one long input vector with length 784)

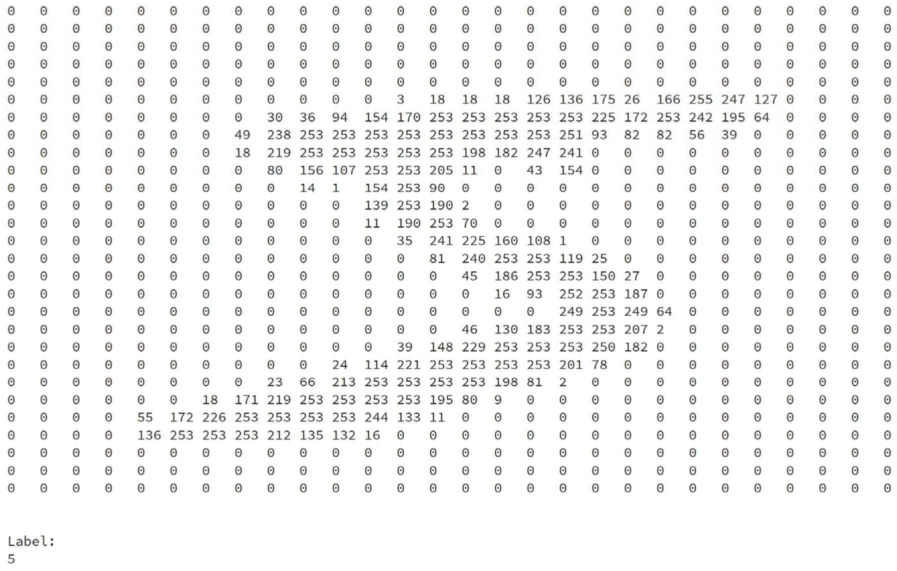
$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$



```
784 input pixel values per train samples
784 input pixel values per test samples
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.01176471 0.07058824 0.07058824 0.07058824
 0.49411765 0.53333336 0.6862745  0.10196079 0.6509804  1.
 0.96862745 0.49803922 0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.11764706 0.14117648 0.36862746 0.6039216
 0.6666667  0.99215686 0.99215686 0.99215686 0.99215686 0.99215686
 0.88235295 0.6745098  0.99215686 0.9490196  0.7647059  0.2509804
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.19215687
 0.93333334 0.99215686 0.99215686 0.99215686 0.99215686 0.99215686
 0.99215686 0.99215686 0.99215686 0.9843137  0.3647059  0.32156864
 0.32156864 0.21960784 0.15294118 0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.07058824 0.85882354 0.99215686
 0.99215686 0.99215686 0.99215686 0.99215686 0.7764706  0.7137255
```

(numbers are between 0 and 1)

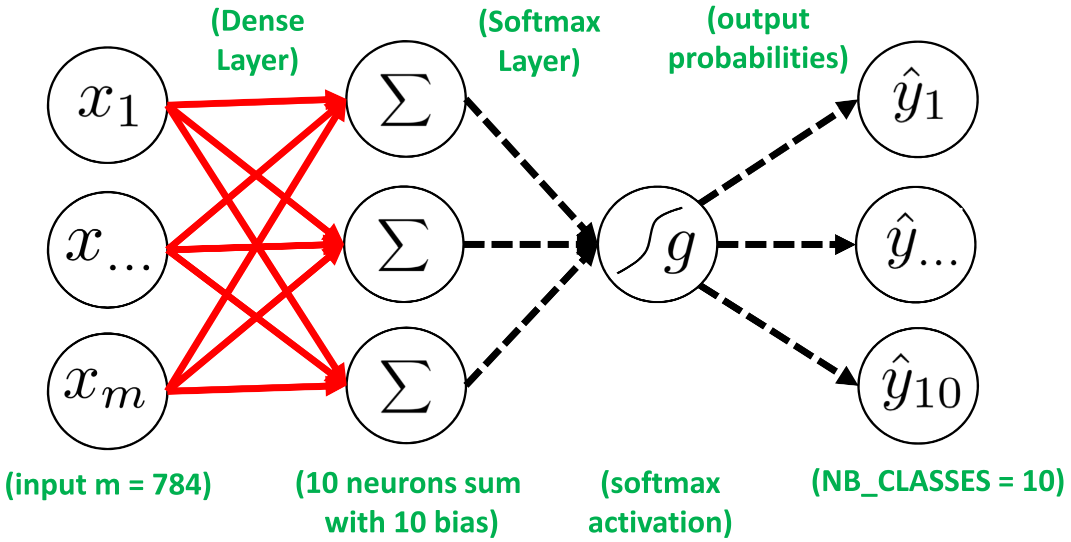
(two dimensional original input)



```
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   3   18  18  18  126 136 175 26  166 255 247 127 0   0   0   0
0   0   0   0   0   0   0   0   30  36  94  154 170 253 253 253 253 253 225 172 253 242 195 64  0   0   0   0
0   0   0   0   0   0   0   49  238 253 253 253 253 253 253 253 253 251 93  82  82  56  39  0   0   0   0   0
0   0   0   0   0   0   0   18  219 253 253 253 253 253 198 182 247 241 0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   80  156 107 253 253 205 11  0   43  154 0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   14  1   154 253 90  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   139 253 190 2   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   11  190 253 70  0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   35  241 225 160 108 1   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   81  240 253 253 119 25  0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   45  186 253 253 150 27  0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   16  93  252 253 187 0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   249 253 249 64  0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   46  130 183 253 253 207 2   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   39  148 229 253 253 253 250 182 0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   24  114 221 253 253 253 253 201 78  0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   23  66  213 253 253 253 253 198 81  2   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   18  171 219 253 253 253 253 195 80  9   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   55  172 226 253 253 253 253 244 133 11  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   136 253 253 253 212 135 132 16  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

Label:
5
```

MNIST Dataset & Multi Output Perceptron Model

- 10 Class Classification Problem
  - Use 10 Perceptrons for 10 outputs with softmax activation function (enables probabilities for 10 classes)



(Dense Layer)

(Softmax Layer)

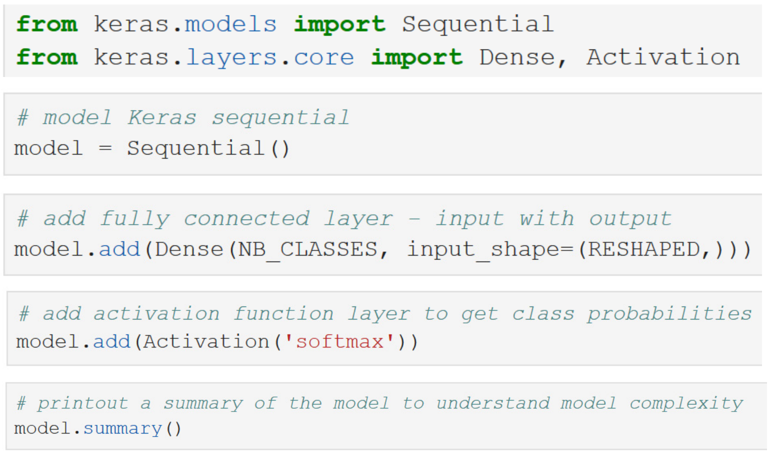
(output probabilities)

(input m = 784)

(10 neurons sum with 10 bias)

(softmax activation)

(NB_CLASSES = 10)



```
from keras.models import Sequential
from keras.layers.core import Dense, Activation
```

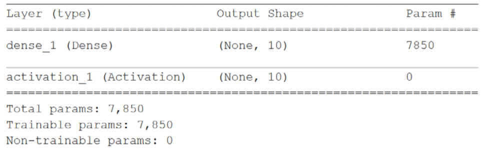
```
# model Keras sequential
model = Sequential()
```

```
# add fully connected layer - input with output
model.add(Dense(NB_CLASSES, input_shape=(RESHAPED,)))
```

```
# add activation function layer to get class probabilities
model.add(Activation('softmax'))
```

```
# printout a summary of the model to understand model complexity
model.summary()
```

(parameters = 784 * 10 + 10 bias = 7850)

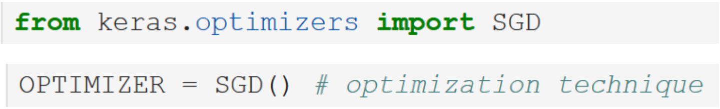


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 10)                7850
_________________________________________________________________
activation_1 (Activation)    (None, 10)                0
=================================================================
Total params: 7,850
Trainable params: 7,850
Non-trainable params: 0
```

- **Note that the output units are independent among each other in contrast to neural networks with one hidden layer**
- **The output of softmax gives class probabilities**
- **The non-linear Activation function 'softmax' represents a generalization of the sigmoid function – it squashes an n-dimensional vector of arbitrary real values into a n-dimenensional vector of real values in the range of 0 and 1 – here it aggregates 10 answers provided by the Dense layer with 10 neurons**

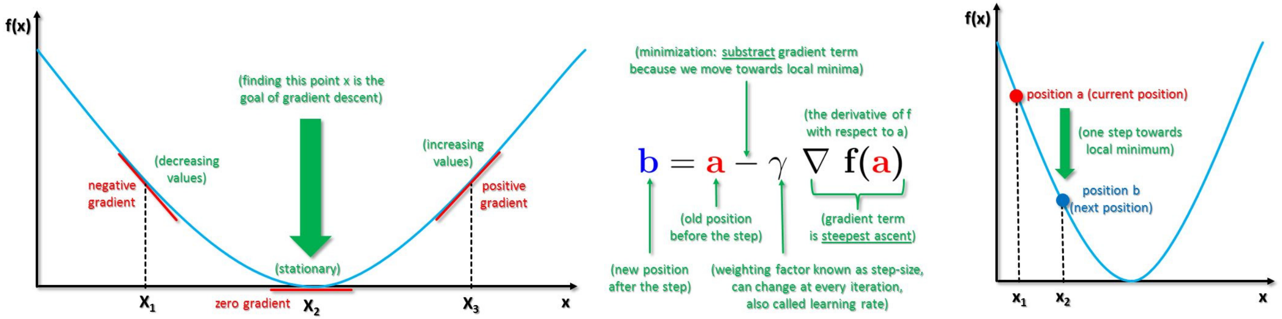
MNIST Dataset & Compile Multi Output Perceptron Model

- Compile the model
  - Optimizer as algorithm used to update weights while training the model
  - Specify loss function (i.e. objective function) that is used by the optimizer to navigate the space of weights
  - (note: process of optimization is also called loss minimization, cf. Invited lecture Gabriele Cavallaro)
  - Indicate metric for model evaluation (e.g., accuracy)
- Specify loss function
  - Compare prediction vs. given class label
  - E.g. categorical crossentropy



```
from keras.optimizers import SGD
```

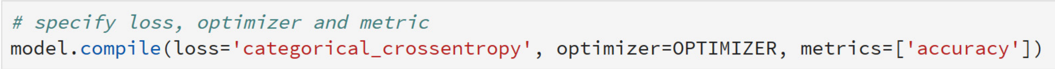
```
OPTIMIZER = SGD() # optimization technique
```


f(x)
(finding this point x is the goal of gradient descent)
(decreasing values)
negative gradient
(increasing values)
positive gradient
(stationary)
X1
zero gradient
X2
X3
x
(minimization: subtract gradient term because we move towards local minima)
(the derivative of f with respect to a)
b = a − γ ∇ f(a)
(old position before the step)
(gradient term is steepest ascent)
(new position after the step)
(weighting factor known as step-size, can change at every iteration, also called learning rate)
f(x)
position a (current position)
(one step towards local minimum)
position b (next position)
x1
x2
x

- **Compile the model to be executed by the Keras backend (e.g. TensorFlow)**
- **Optimizer Gradient Descent (GD) uses all the training samples available for a step within a iteration**
- **Optimizer Stochastic Gradient Descent (SGD) converges faster: only one training samples used per iteration**
- **Loss function is a multi-class logarithmic loss: target is *ti,j* and prediction is *pi,j***
- **Categorical crossentropy is suitable for multiclass label predictions (default with softmax)**

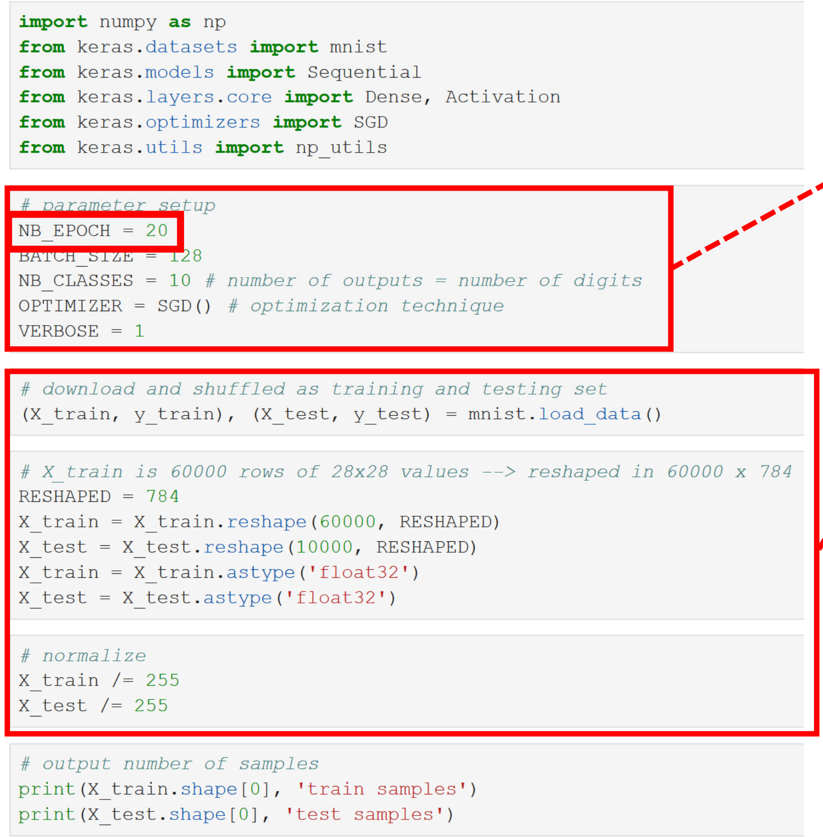
$$L_i = -\Sigma_j t_{i,j} \log(p_{i,j})$$

[10] Big Data Tips, Gradient Descent



```
# specify loss, optimizer and metric
model.compile(loss='categorical_crossentropy', optimizer=OPTIMIZER, metrics=['accuracy'])
```

Full Script: MNIST Dataset – Model Parameters & Data Normalization



```
import numpy as np
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers.core import Dense, Activation
from keras.optimizers import SGD
from keras.utils import np_utils
```

```
# parameter setup
NB_EPOCH = 20
BATCH_SIZE = 128
NB_CLASSES = 10 # number of outputs = number of digits
OPTIMIZER = SGD() # optimization technique
VERBOSE = 1
```

```
# download and shuffled as training and testing set
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

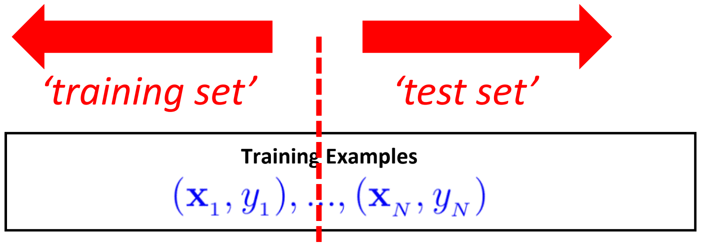
```
# X_train is 60000 rows of 28x28 values --> reshaped in 60000 x 784
RESHAPED = 784
X_train = X_train.reshape(60000, RESHAPED)
X_test = X_test.reshape(10000, RESHAPED)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
```

```
# normalize
X_train /= 255
X_test /= 255
```

```
# output number of samples
print(X_train.shape[0], 'train samples')
print(X_test.shape[0], 'test samples')
```

- **NB_CLASSES: 10 Class Problem**
- **NB_EPOCH: number of times the model is exposed to the overall training set – at each iteration the optimizer adjusts the weights so that the objective function is minimized – increasing leads to better accuracy, but also to overfitting (cf. Lecture 7)**
- **BATCH_SIZE: number of training instances taken into account before the optimizer performs a weight update to the model**
- **OPTIMIZER: Stochastic Gradient Descent (‘SGD‘) – only one training sample/iteration**

- **Data load shuffled between training and testing set in files**
- **Data preparation, e.g. X_train is 60000 samples / rows of 28 x 28 pixel values that are reshaped in 60000 x 784 including type specification (i.e. float32)**
- **Data normalization: divide by 255 – the max intensity value to obtain values in range [0,1]**

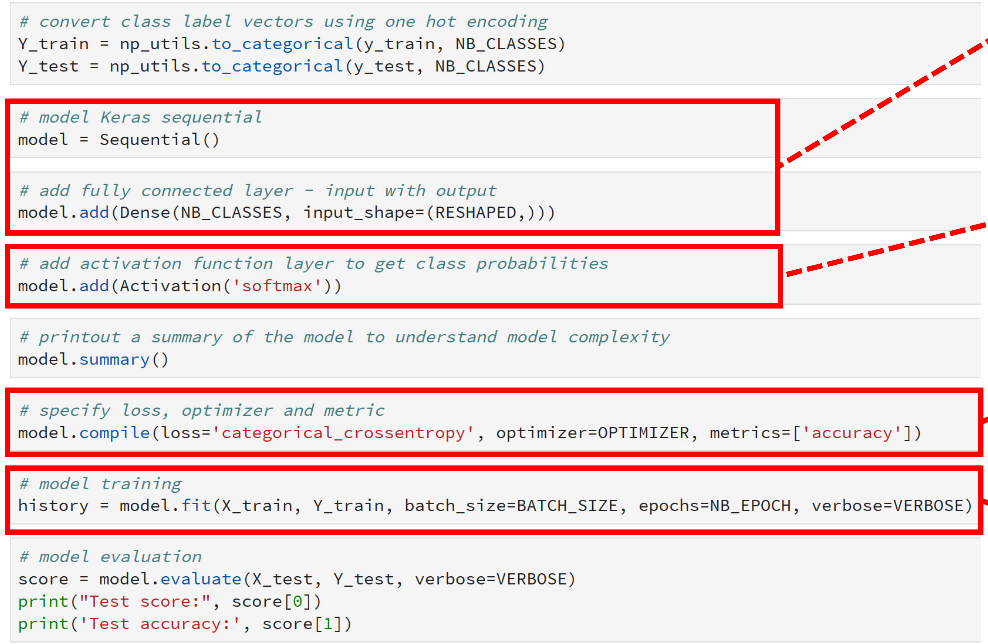


(historical records, groundtruth data, examples)

➢ **Assignment #2 will explore the change of parameters in context of changes in running time when training models on GPUs vs. CPUs**

Full Script: MNIST Dataset – Fitting a Multi Output Perceptron Model

(full script continued from previous slide)



```
# convert class label vectors using one hot encoding
Y_train = np_utils.to_categorical(y_train, NB_CLASSES)
Y_test = np_utils.to_categorical(y_test, NB_CLASSES)
```

```
# model Keras sequential
model = Sequential()
```

```
# add fully connected layer - input with output
model.add(Dense(NB_CLASSES, input_shape=(RESHAPED,)))
```

```
# add activation function layer to get class probabilities
model.add(Activation('softmax'))
```

```
# printout a summary of the model to understand model complexity
model.summary()
```

```
# specify loss, optimizer and metric
model.compile(loss='categorical_crossentropy', optimizer=OPTIMIZER, metrics=['accuracy'])
```

```
# model training
history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NB_EPOCH, verbose=VERBOSE)
```

```
# model evaluation
score = model.evaluate(X_test, Y_test, verbose=VERBOSE)
print("Test score:", score[0])
print('Test accuracy:', score[1])
```

- **The Sequential() Keras model is a linear pipeline (aka ‘a stack‘) of various neural network layers including Activation functions of different types (e.g. softmax)**
- **Dense() represents a fully connected layer used in ANNs that means that each neuron in a layer is connected to all neurons located in the previous layer**

- **The non-linear activation function ‘softmax‘ is a generalization of the sigmoid function – it squashes an n-dimensional vector of arbitrary real values into a n-dimenensional vector of real values in the range of 0 and 1 – here it aggregates 10 answers provided by the Dense layer with 10 neurons**

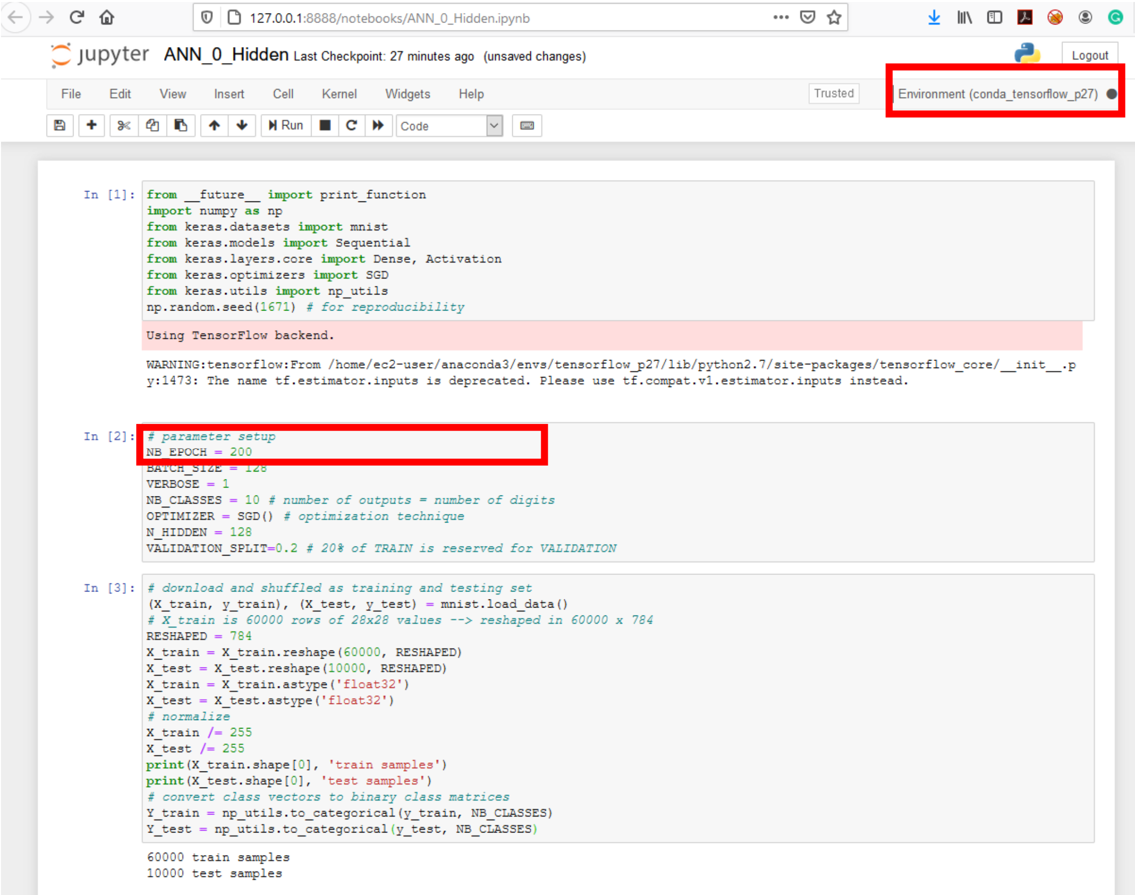
- **Loss function is a multi-class logarithmic loss: target is *ti,j* and the prediction is *pi,j***

$$L_i = -\Sigma_j t_{i,j} \log(p_{i,j})$$

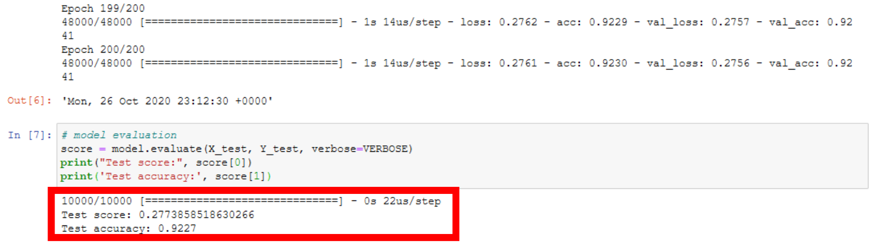
- **Train the model (‘fit‘) using selected batch & epoch sizes on training & test data**

> **Assignment #2 will explore the change of parameters in context of changes in running time when training models on GPUs vs. CPUs**

Running a Simple ANN with no hidden layers – Multi-Output-Perceptron

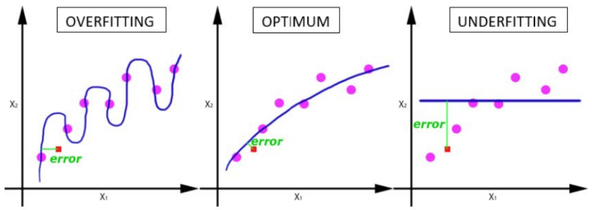

127.0.0.1:8888/notebooks/ANN_0_Hidden.ipynb
jupyter ANN_0_Hidden Last Checkpoint: 27 minutes ago (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Environment (conda_tensorflow_p27)
Run
Code

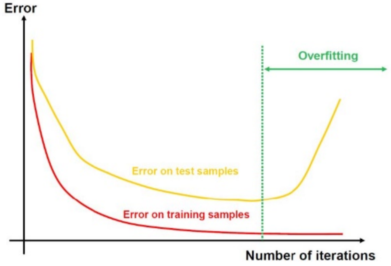
```
In [1]: from __future__ import print_function
        import numpy as np
        from keras.datasets import mnist
        from keras.models import Sequential
        from keras.layers.core import Dense, Activation
        from keras.optimizers import SGD
        from keras.utils import np_utils
        np.random.seed(1671) # for reproducibility

        Using TensorFlow backend.

        WARNING:tensorflow:From /home/ec2-user/anaconda3/envs/tensorflow_p27/lib/python2.7/site-packages/tensorflow_core/__init__.p
        y:1473: The name tf.estimator.inputs is deprecated. Please use tf.compat.v1.estimator.inputs instead.

In [2]: # parameter setup
        NB_EPOCH = 200
        BATCH_SIZE = 128
        VERBOSE = 1
        NB_CLASSES = 10 # number of outputs = number of digits
        OPTIMIZER = SGD() # optimization technique
        N_HIDDEN = 128
        VALIDATION_SPLIT=0.2 # 20% of TRAIN is reserved for VALIDATION

In [3]: # download and shuffled as training and testing set
        (X_train, y_train), (X_test, y_test) = mnist.load_data()
        # X_train is 60000 rows of 28x28 values --> reshaped in 60000 x 784
        RESHAPED = 784
        X_train = X_train.reshape(60000, RESHAPED)
        X_test = X_test.reshape(10000, RESHAPED)
        X_train = X_train.astype('float32')
        X_test = X_test.astype('float32')
        # normalize
        X_train /= 255
        X_test /= 255
        print(X_train.shape[0], 'train samples')
        print(X_test.shape[0], 'test samples')
        # convert class vectors to binary class matrices
        Y_train = np_utils.to_categorical(y_train, NB_CLASSES)
        Y_test = np_utils.to_categorical(y_test, NB_CLASSES)

        60000 train samples
        10000 test samples
```



```
        Epoch 199/200
        48000/48000 [==============================] - 1s 14us/step - loss: 0.2762 - acc: 0.9229 - val_loss: 0.2757 - val_acc: 0.92
        41
        Epoch 200/200
        48000/48000 [==============================] - 1s 14us/step - loss: 0.2761 - acc: 0.9230 - val_loss: 0.2756 - val_acc: 0.92
        41

Out[6]: 'Mon, 26 Oct 2020 23:12:30 +0000'

In [7]: # model evaluation
        score = model.evaluate(X_test, Y_test, verbose=VERBOSE)
        print("Test score:", score[0])
        print('Test accuracy:', score[1])

        10000/10000 [==============================] - 0s 22us/step
        Test score: 0.2773858518630266
        Test accuracy: 0.9227
```

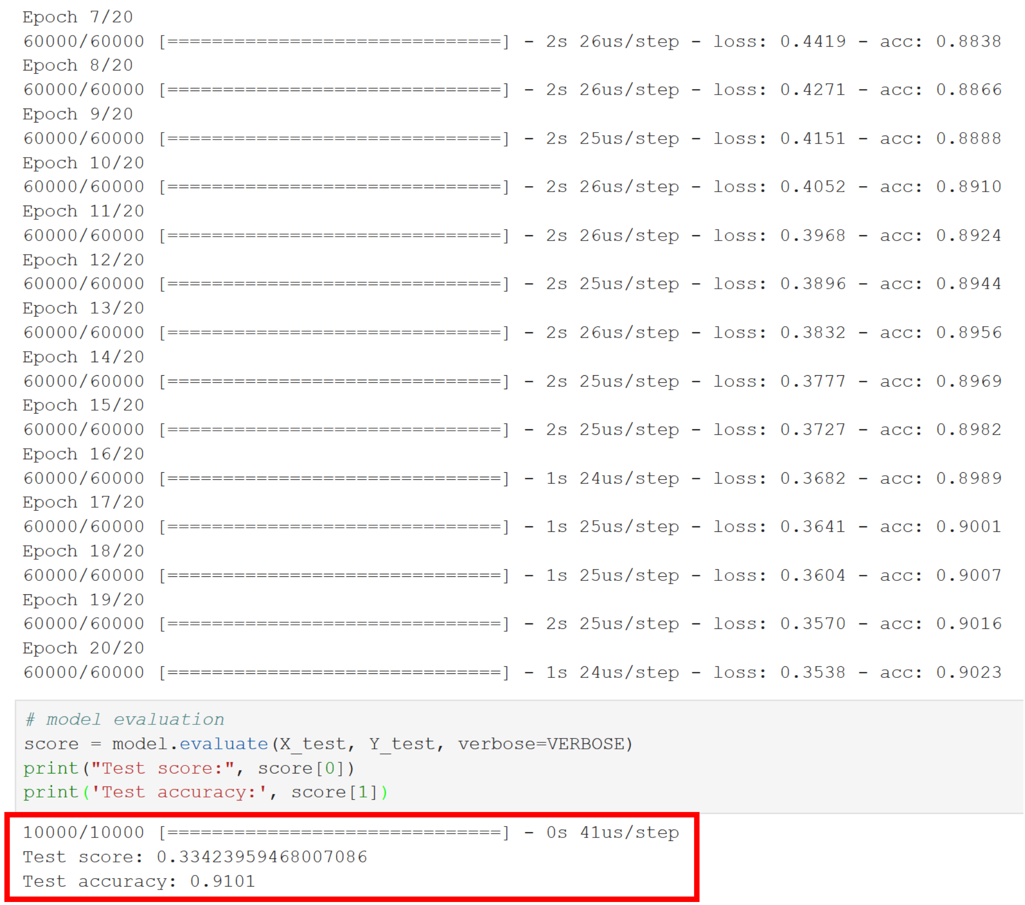
- **Note that the outcome of the training process is the result of optimization techniques like SGD that tend to vary 'a bit'**
- **Note that the outcome of the training process can be dependent on the length of training increasing accuracy to a certain point when overfitting starts**
- **Overfitting can be controlled with validation and regularization techniques that belong to advanced machine learning methods to be studied in full university machine learning course in detail**





cf. Lecture 6 and 7

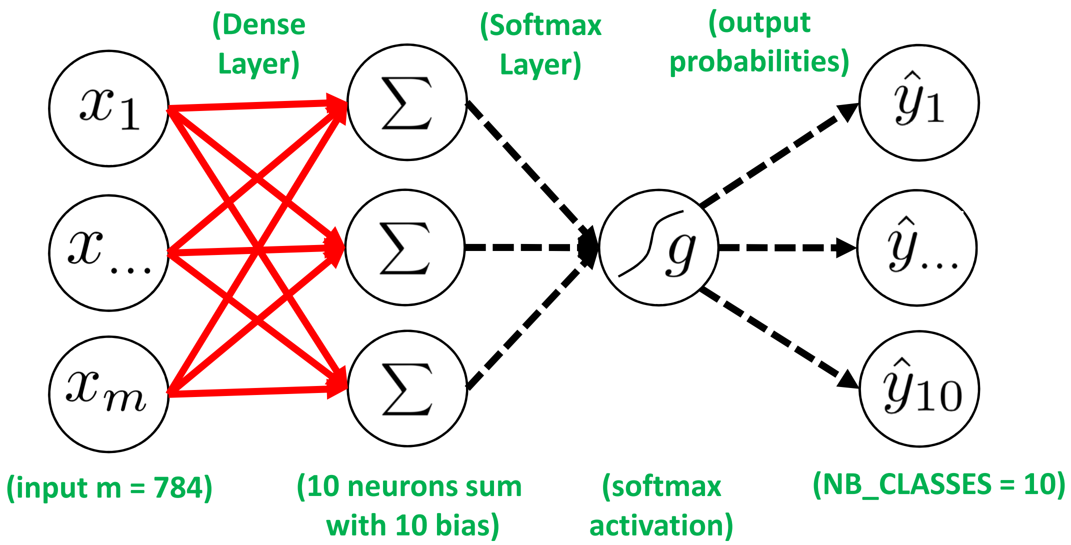
MNIST Dataset – A Multi Output Perceptron Model – Output & Evaluation



```
Epoch 7/20
60000/60000 [==============================] - 2s 26us/step - loss: 0.4419 - acc: 0.8838
Epoch 8/20
60000/60000 [==============================] - 2s 26us/step - loss: 0.4271 - acc: 0.8866
Epoch 9/20
60000/60000 [==============================] - 2s 25us/step - loss: 0.4151 - acc: 0.8888
Epoch 10/20
60000/60000 [==============================] - 2s 26us/step - loss: 0.4052 - acc: 0.8910
Epoch 11/20
60000/60000 [==============================] - 2s 26us/step - loss: 0.3968 - acc: 0.8924
Epoch 12/20
60000/60000 [==============================] - 2s 25us/step - loss: 0.3896 - acc: 0.8944
Epoch 13/20
60000/60000 [==============================] - 2s 26us/step - loss: 0.3832 - acc: 0.8956
Epoch 14/20
60000/60000 [==============================] - 2s 25us/step - loss: 0.3777 - acc: 0.8969
Epoch 15/20
60000/60000 [==============================] - 2s 25us/step - loss: 0.3727 - acc: 0.8982
Epoch 16/20
60000/60000 [==============================] - 1s 24us/step - loss: 0.3682 - acc: 0.8989
Epoch 17/20
60000/60000 [==============================] - 1s 25us/step - loss: 0.3641 - acc: 0.9001
Epoch 18/20
60000/60000 [==============================] - 1s 25us/step - loss: 0.3604 - acc: 0.9007
Epoch 19/20
60000/60000 [==============================] - 2s 25us/step - loss: 0.3570 - acc: 0.9016
Epoch 20/20
60000/60000 [==============================] - 1s 24us/step - loss: 0.3538 - acc: 0.9023
```

```python
# model evaluation
score = model.evaluate(X_test, Y_test, verbose=VERBOSE)
print("Test score:", score[0])
print('Test accuracy:', score[1])
```

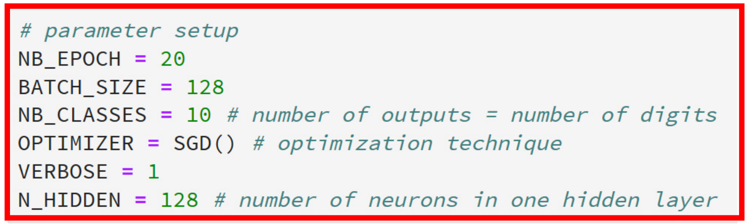
```
10000/10000 [==============================] - 0s 41us/step
Test score: 0.33423959468007086
Test accuracy: 0.9101
```



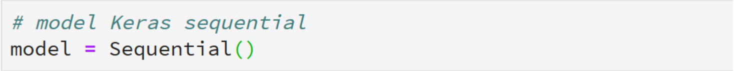
- How to improve the model design by extending the neural network topology?
- Which layers are required?
- Think about input layer need to match the data – what data we had?
- Maybe hidden layers?
- How many hidden layers?
- What activation function for which layer (e.g. maybe ReLU)?
- Think Dense layer – Keras?
- Think about final Activation as Softmay (cf. Day One) → output probability

MNIST Dataset – Add Two Hidden Layers for Artificial Neural Network (ANN)

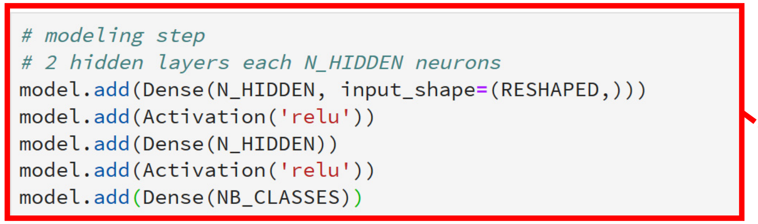
- All parameter value remain the same as before
- We add N_HIDDEN as parameter in order to set 128 neurons in one hidden layer – this number is a hyperparameter that is not directly defined and needs to be find with parameter search



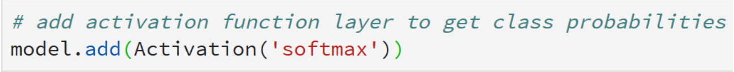
```
# parameter setup
NB_EPOCH = 20
BATCH_SIZE = 128
NB_CLASSES = 10 # number of outputs = number of digits
OPTIMIZER = SGD() # optimization technique
VERBOSE = 1
N_HIDDEN = 128 # number of neurons in one hidden layer
```



```
# model Keras sequential
model = Sequential()
```



```
# modeling step
# 2 hidden layers each N_HIDDEN neurons
model.add(Dense(N_HIDDEN, input_shape=(RESHAPED,)))
model.add(Activation('relu'))
model.add(Dense(N_HIDDEN))
model.add(Activation('relu'))
model.add(Dense(NB_CLASSES))
```

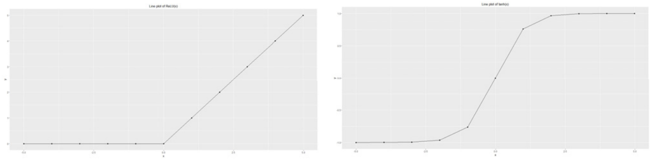


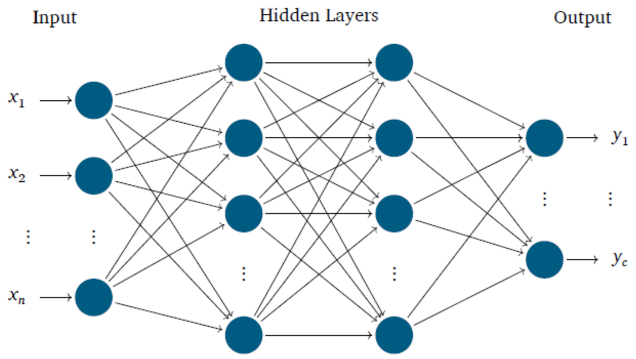
```
# add activation function layer to get class probabilities
model.add(Activation('softmax'))
```

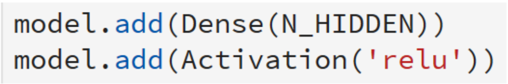
[34] big-data.tips, 'Relu Neural Network'

[35] big-data.tips, 'tanh'

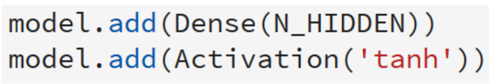
(activation functions ReLU & Tanh)







```
model.add(Dense(N_HIDDEN))
model.add(Activation('relu'))
```

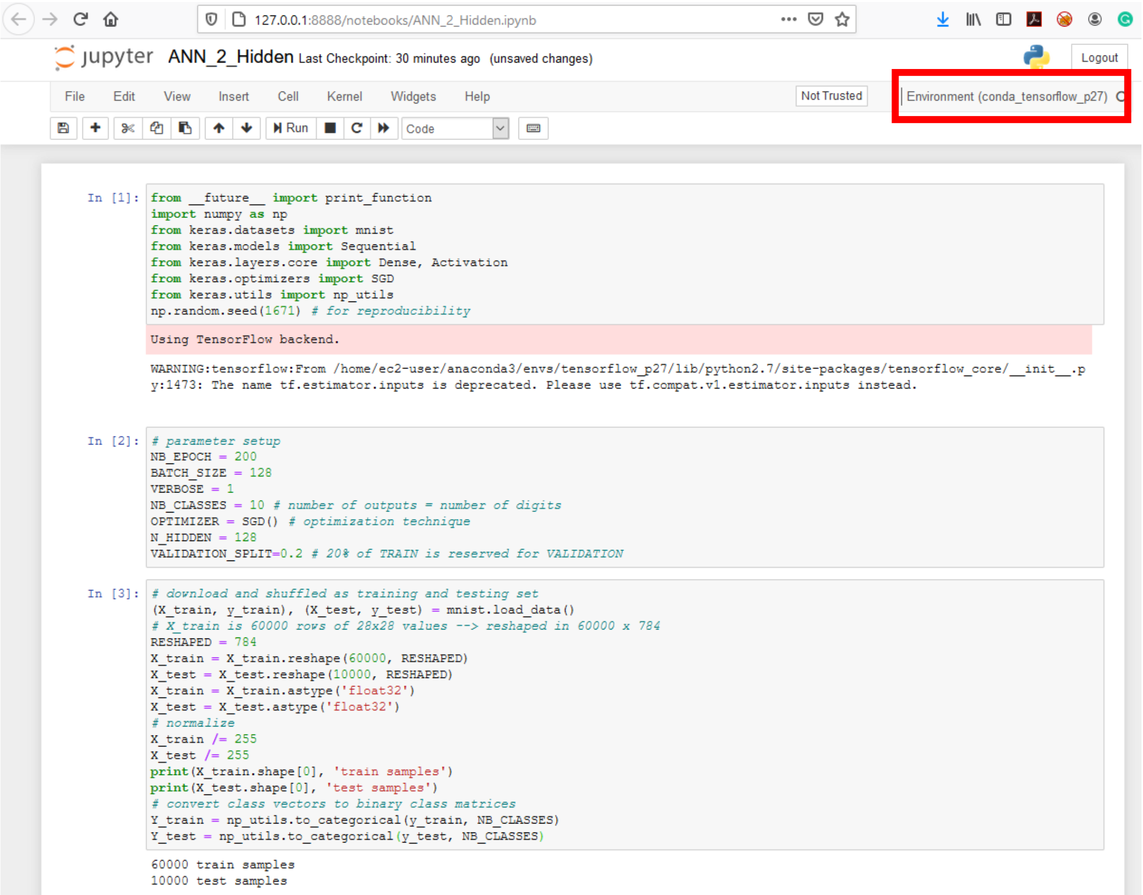


```
model.add(Dense(N_HIDDEN))
model.add(Activation('tanh'))
```

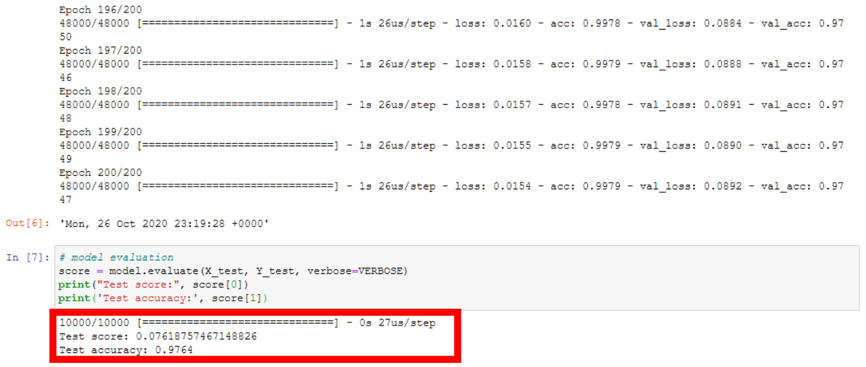
- The non-linear Activation function 'relu' represents a so-called Rectified Linear Unit (ReLU) that only recently became very popular because it generates good experimental results in ANNs and more recent deep learning models – it just returns 0 for negative values and grows linearly for only positive values
- A hidden layer in an ANN can be represented by a fully connected Dense layer in Keras by just specifying the number of hidden neurons in the hidden layer

➢ Assignment #2 will explore the change of parameters in context of changes in running time when training models on GPUs vs. CPUs

Running a Simple ANN with two hidden layers

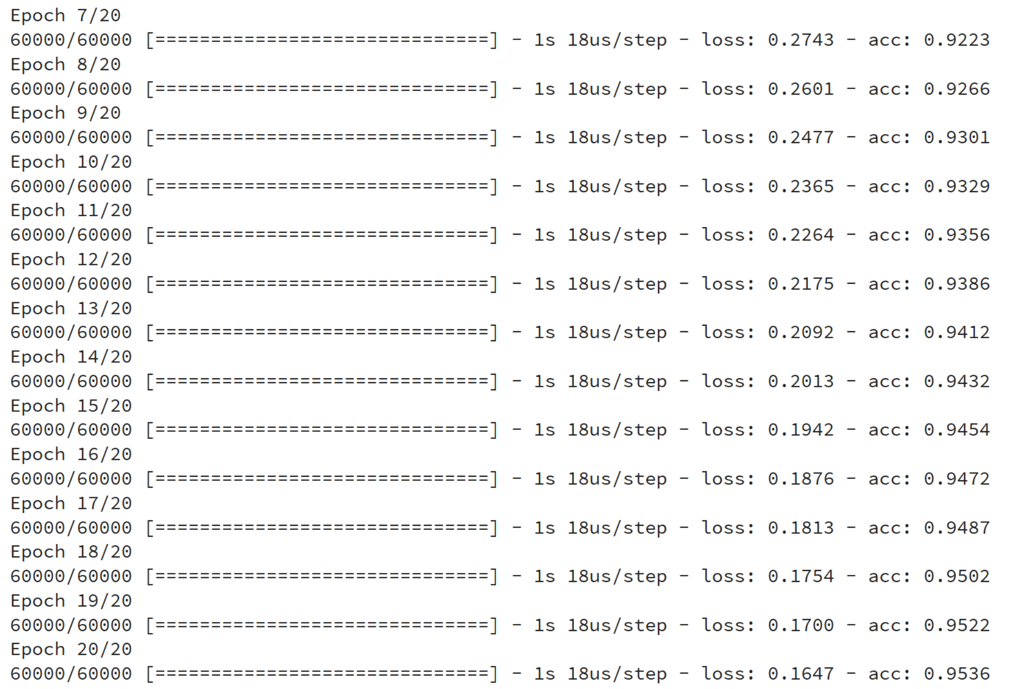

127.0.0.1:8888/notebooks/ANN_2_Hidden.ipynb
jupyter ANN_2_Hidden Last Checkpoint: 30 minutes ago (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Environment (conda_tensorflow_p27)
Run
Code

```
In [1]: from __future__ import print_function
        import numpy as np
        from keras.datasets import mnist
        from keras.models import Sequential
        from keras.layers.core import Dense, Activation
        from keras.optimizers import SGD
        from keras.utils import np_utils
        np.random.seed(1671) # for reproducibility

        Using TensorFlow backend.

        WARNING:tensorflow:From /home/ec2-user/anaconda3/envs/tensorflow_p27/lib/python2.7/site-packages/tensorflow_core/__init__.p
        y:1473: The name tf.estimator.inputs is deprecated. Please use tf.compat.v1.estimator.inputs instead.

In [2]: # parameter setup
        NB_EPOCH = 200
        BATCH_SIZE = 128
        VERBOSE = 1
        NB_CLASSES = 10 # number of outputs = number of digits
        OPTIMIZER = SGD() # optimization technique
        N_HIDDEN = 128
        VALIDATION_SPLIT=0.2 # 20% of TRAIN is reserved for VALIDATION

In [3]: # download and shuffled as training and testing set
        (X_train, y_train), (X_test, y_test) = mnist.load_data()
        # X_train is 60000 rows of 28x28 values --> reshaped in 60000 x 784
        RESHAPED = 784
        X_train = X_train.reshape(60000, RESHAPED)
        X_test = X_test.reshape(10000, RESHAPED)
        X_train = X_train.astype('float32')
        X_test = X_test.astype('float32')
        # normalize
        X_train /= 255
        X_test /= 255
        print(X_train.shape[0], 'train samples')
        print(X_test.shape[0], 'test samples')
        # convert class vectors to binary class matrices
        Y_train = np_utils.to_categorical(y_train, NB_CLASSES)
        Y_test = np_utils.to_categorical(y_test, NB_CLASSES)

        60000 train samples
        10000 test samples
```

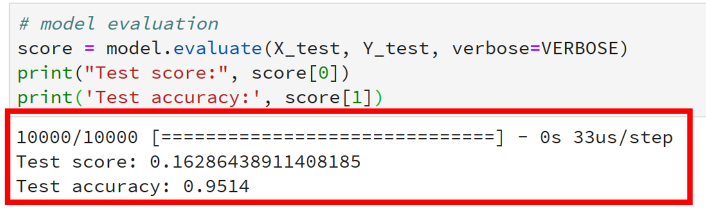


```
         Epoch 196/200
         48000/48000 [==============================] - 1s 26us/step - loss: 0.0160 - acc: 0.9978 - val_loss: 0.0884 - val_acc: 0.97
         50
         Epoch 197/200
         48000/48000 [==============================] - 1s 26us/step - loss: 0.0158 - acc: 0.9979 - val_loss: 0.0888 - val_acc: 0.97
         46
         Epoch 198/200
         48000/48000 [==============================] - 1s 26us/step - loss: 0.0157 - acc: 0.9978 - val_loss: 0.0891 - val_acc: 0.97
         48
         Epoch 199/200
         48000/48000 [==============================] - 1s 26us/step - loss: 0.0155 - acc: 0.9979 - val_loss: 0.0890 - val_acc: 0.97
         49
         Epoch 200/200
         48000/48000 [==============================] - 1s 26us/step - loss: 0.0154 - acc: 0.9979 - val_loss: 0.0892 - val_acc: 0.97
         47

Out[6]:  'Mon, 26 Oct 2020 23:19:28 +0000'

In [7]:  # model evaluation
         score = model.evaluate(X_test, Y_test, verbose=VERBOSE)
         print("Test score:", score[0])
         print('Test accuracy:', score[1])

         10000/10000 [==============================] - 0s 27us/step
         Test score: 0.07618757467148826
         Test accuracy: 0.9764
```

MNIST Dataset – ANN Model Parameters & Output Evaluation



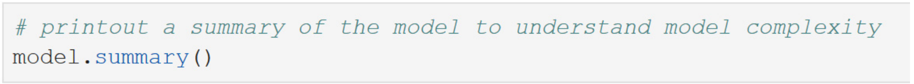
```
Epoch 7/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.2743 - acc: 0.9223
Epoch 8/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.2601 - acc: 0.9266
Epoch 9/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.2477 - acc: 0.9301
Epoch 10/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.2365 - acc: 0.9329
Epoch 11/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.2264 - acc: 0.9356
Epoch 12/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.2175 - acc: 0.9386
Epoch 13/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.2092 - acc: 0.9412
Epoch 14/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.2013 - acc: 0.9432
Epoch 15/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.1942 - acc: 0.9454
Epoch 16/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.1876 - acc: 0.9472
Epoch 17/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.1813 - acc: 0.9487
Epoch 18/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.1754 - acc: 0.9502
Epoch 19/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.1700 - acc: 0.9522
Epoch 20/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.1647 - acc: 0.9536
```



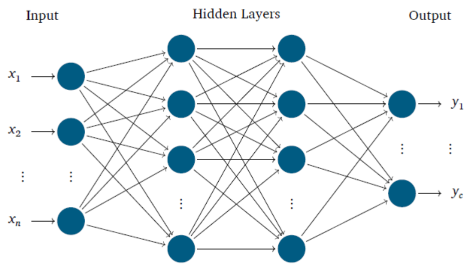
```
# model evaluation
score = model.evaluate(X_test, Y_test, verbose=VERBOSE)
print("Test score:", score[0])
print('Test accuracy:', score[1])
```

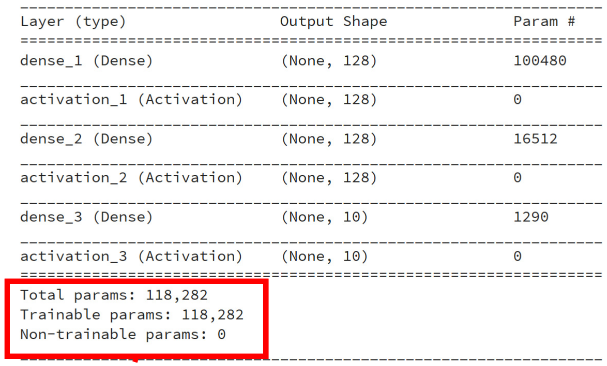
```
10000/10000 [==============================] - 0s 33us/step
Test score: 0.16286438911408185
Test accuracy: 0.9514
```

- ✓ **Multi Output Perceptron: ~91,01% (20 Epochs)**
- ✓ **ANN 2 Hidden Layers: ~95,14 % (20 Epochs)**



```
# printout a summary of the model to understand model complexity
model.summary()
```

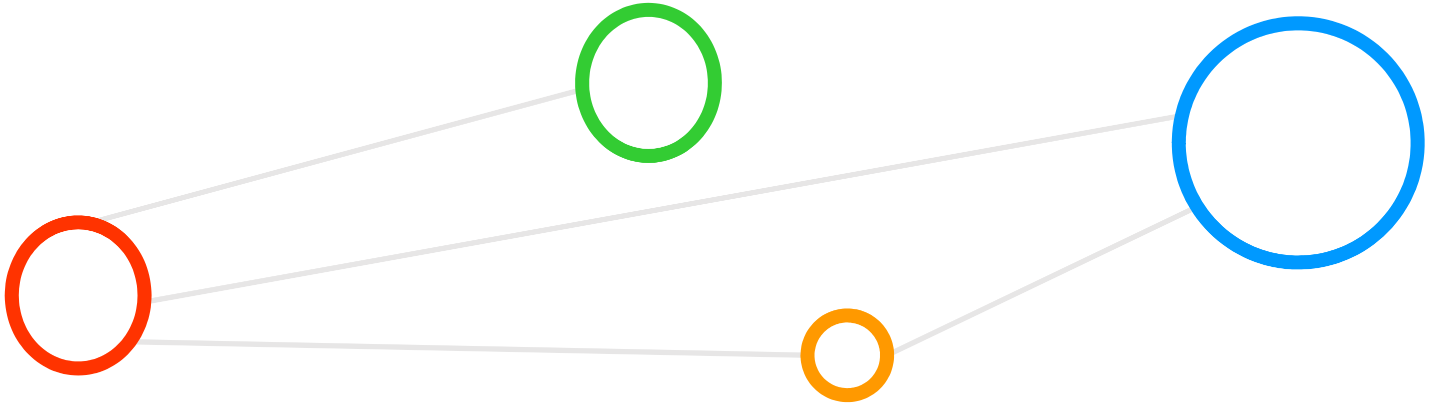




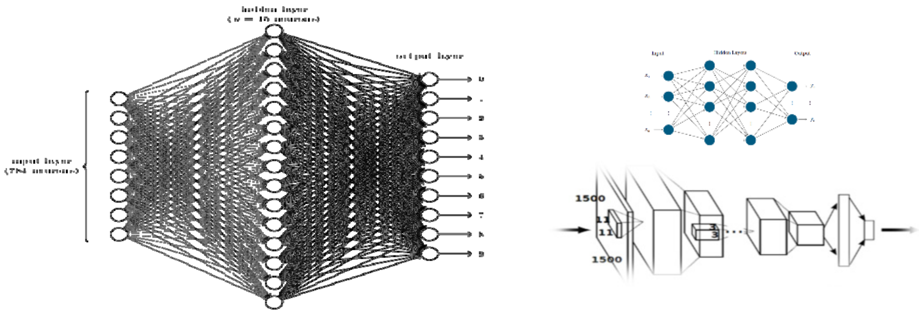
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               100480
_________________________________________________________________
activation_1 (Activation)    (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               16512
_________________________________________________________________
activation_2 (Activation)    (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                1290
_________________________________________________________________
activation_3 (Activation)    (None, 10)                0
=================================================================
Total params: 118,282
Trainable params: 118,282
Non-trainable params: 0
_________________________________________________________________
```

- **Dense Layer connects every neuron in this dense layer to the next dense layer with each of its neuron also called a fully connected network element with weights as trainiable parameters**
- **Choosing a model with different layers is a model selection that directly also influences the number of parameters (e.g. add Dense layer from Keras means new weights)**
- **Adding a layer with these new weights means much more computational complexity since each of the weights must be trained in each epoch (depending on #neurons in layer)**

Using Convolutional Neural Network (CNN) & GPUs in Clouds

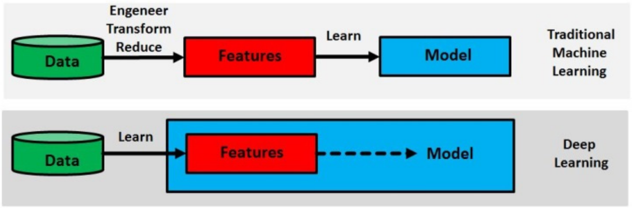


Innovative Deep Learning Techniques – Revisited (cf. Lecture 6 & 7)



[11] M. Riedel, 'Deep Learning - Using a Convolutional Neural Network', Invited YouTube Lecture, six lectures, University of Ghent, 2017

[14] Neural Network 3D Simulation



Engeneer Transform Reduce

Data → Features → Learn → Model

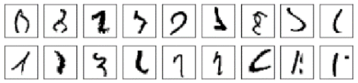
Traditional Machine Learning

Data → Learn → Features - - - -> Model

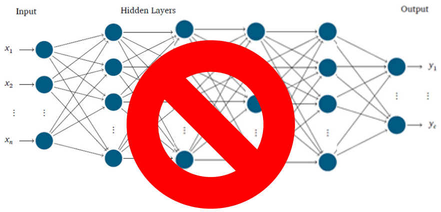
Deep Learning

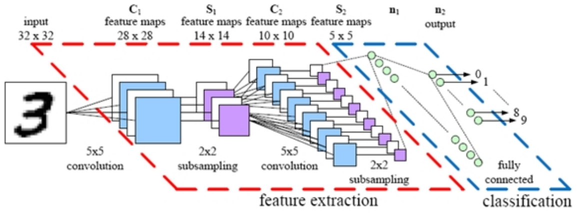
[12] M. Riedel et al., 'Introduction to Deep Learning Models', JSC Tutorial, three days, JSC, 2019

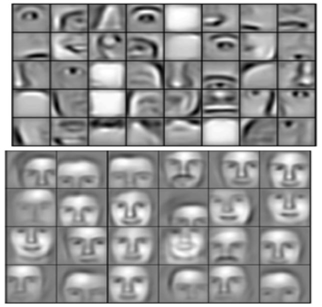
- **Innovation via specific layers and architecture types**



[15] A. Rosebrock

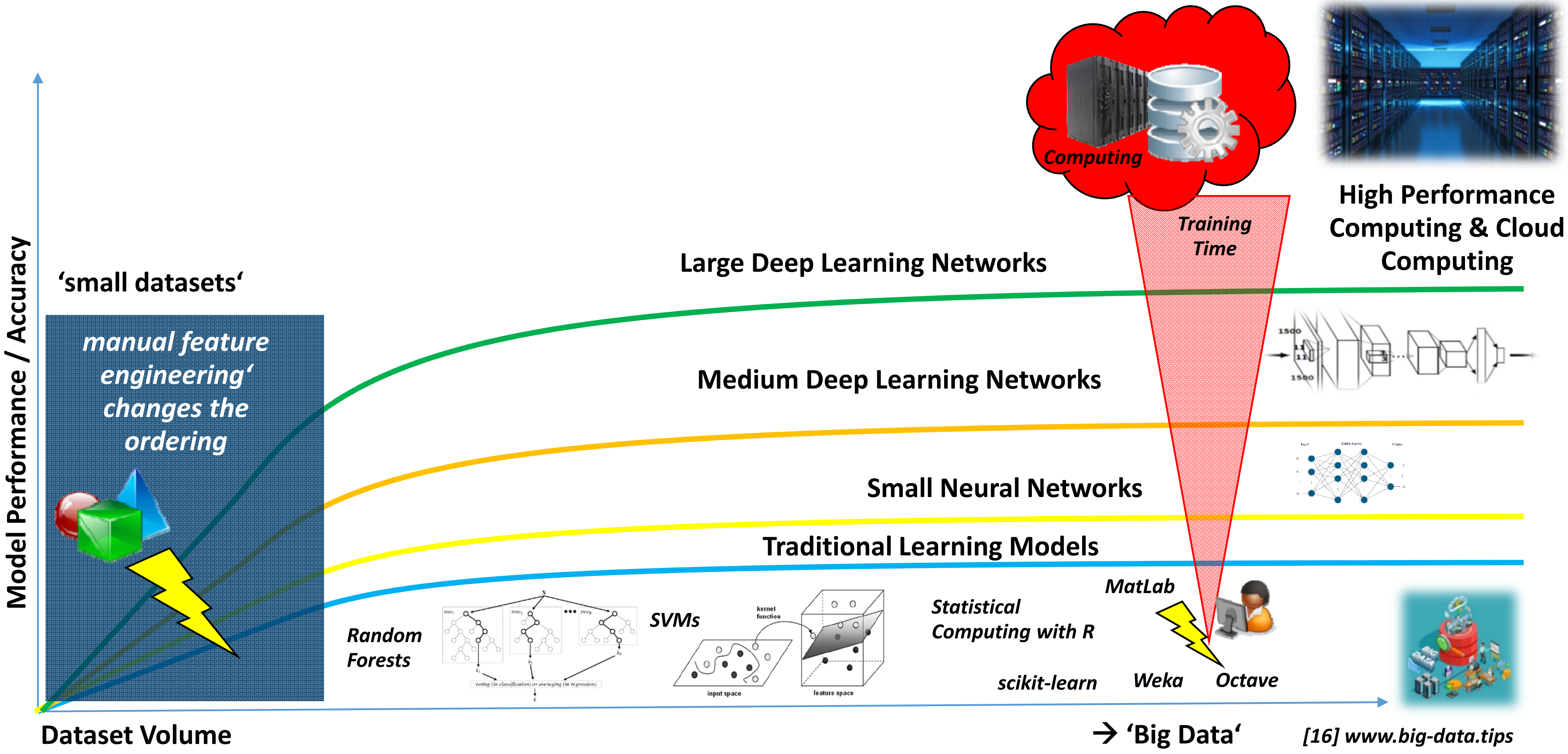






[13] H. Lee et al., 'Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical

Complex Relationships: ML & DL vs. HPC/Clouds & Big Data (cf. Lecture 0)



Computing

Training Time

High Performance Computing & Cloud Computing

Model Performance / Accuracy

'small datasets'

manual feature engineering' changes the ordering

Large Deep Learning Networks

Medium Deep Learning Networks

Small Neural Networks

Traditional Learning Models

Random Forests

SVMs

Statistical Computing with R

MatLab

scikit-learn

Weka

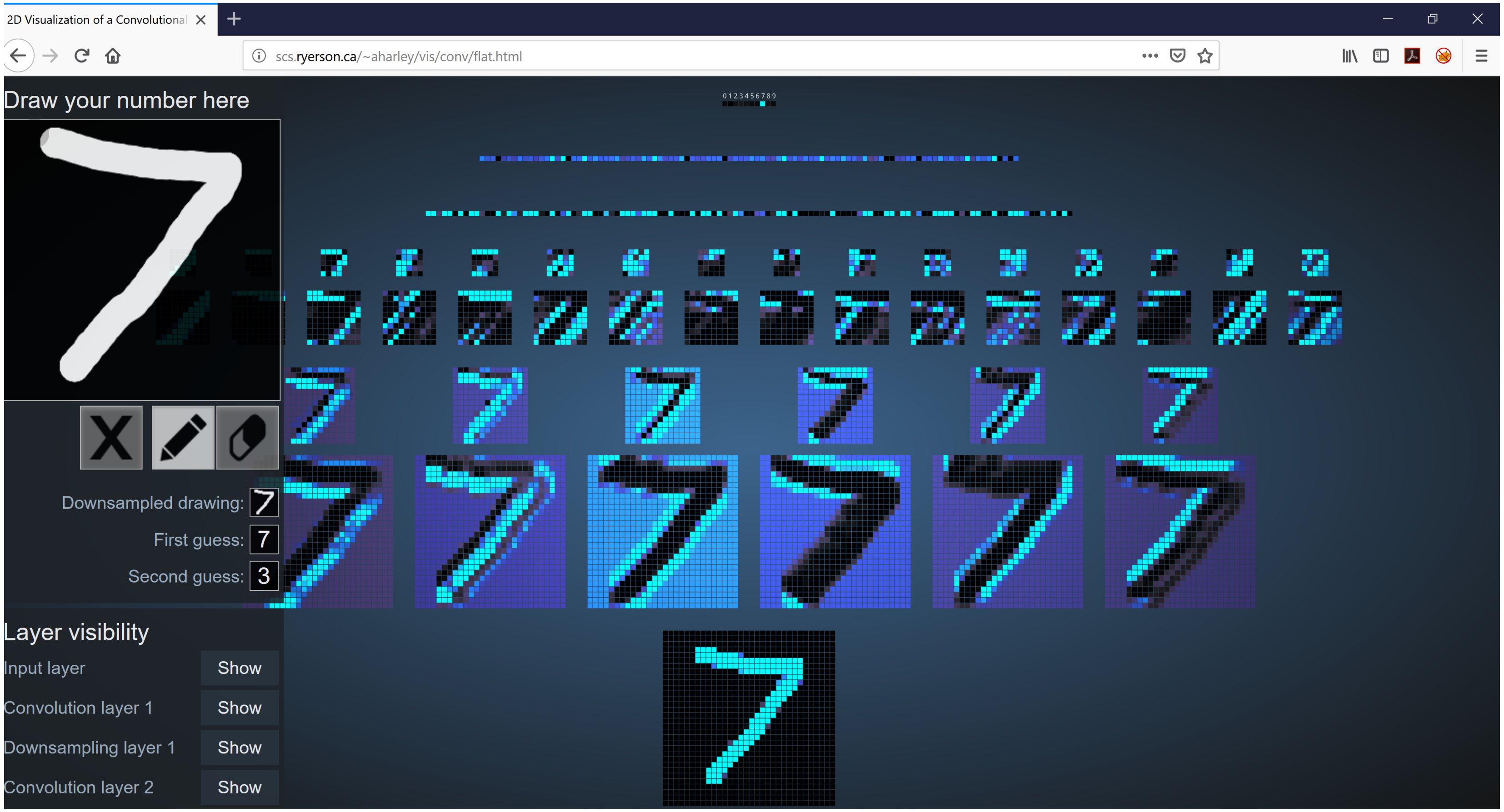
Octave

Dataset Volume

→ 'Big Data'

[16] www.big-data.tips

Understanding Feature Maps & Convolutions – Online Web Tool



2D Visualization of a Convolutional
scs.ryerson.ca/~aharley/vis/conv/flat.html
0 1 2 3 4 5 6 7 8 9

Draw your number here

Downsampled drawing: 7

First guess: 7

Second guess: 3

Layer visibility

Input layer Show

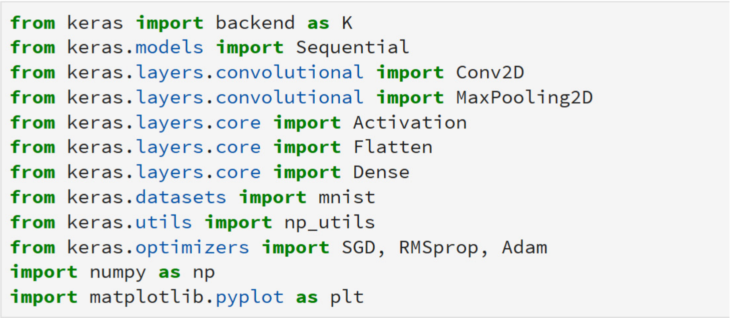
Convolution layer 1 Show

Downsampling layer 1 Show

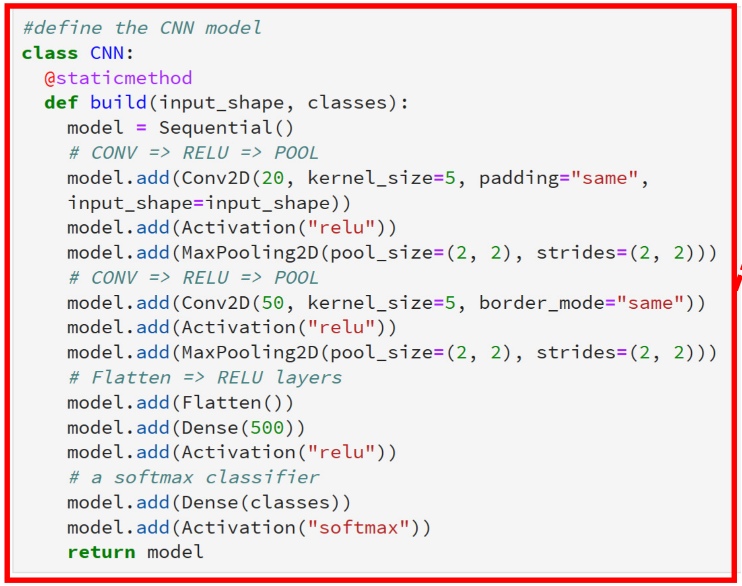
Convolution layer 2 Show

[17] Harley, A.W., An Interactive Node-Link Visualization of Convolutional Neural Networks

MNIST Dataset – Convolutional Neural Network (CNN) Model



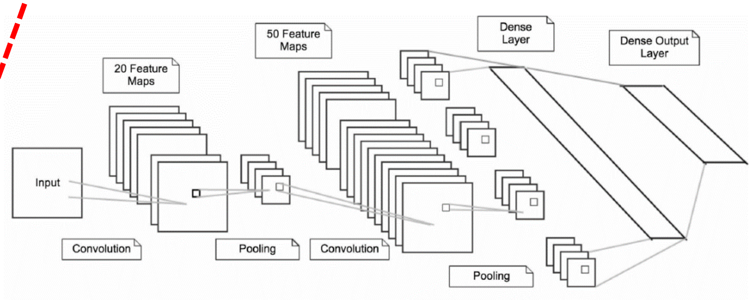
```
from keras import backend as K
from keras.models import Sequential
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation
from keras.layers.core import Flatten
from keras.layers.core import Dense
from keras.datasets import mnist
from keras.utils import np_utils
from keras.optimizers import SGD, RMSprop, Adam
import numpy as np
import matplotlib.pyplot as plt
```

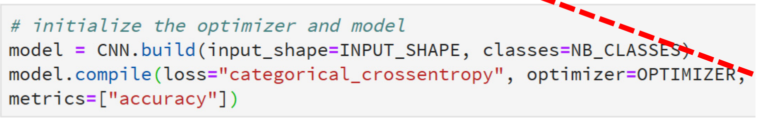


```
#define the CNN model
class CNN:
  @staticmethod
  def build(input_shape, classes):
    model = Sequential()
    # CONV => RELU => POOL
    model.add(Conv2D(20, kernel_size=5, padding="same",
    input_shape=input_shape))
    model.add(Activation("relu"))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    # CONV => RELU => POOL
    model.add(Conv2D(50, kernel_size=5, border_mode="same"))
    model.add(Activation("relu"))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    # Flatten => RELU layers
    model.add(Flatten())
    model.add(Dense(500))
    model.add(Activation("relu"))
    # a softmax classifier
    model.add(Dense(classes))
    model.add(Activation("softmax"))
    return model
```

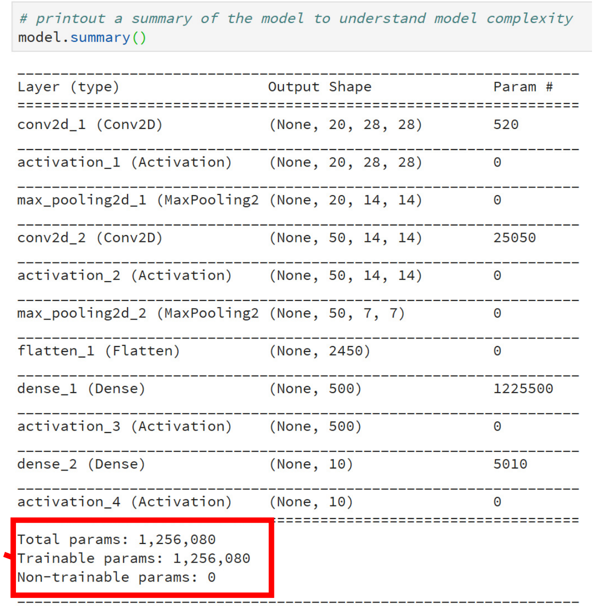
- Increasing the number of filters learned to 50 in the next layer from 20 in the first layer
- Increasing the number of filters in deeper layers is a common technique in deep learning architecture modeling
- Flattening the output as input for a Dense layer (fully connected layer)
- Fully connected / Dense layer responsible with softmay activation for classification based on learned filters and features

[18] A. Gulli et al.





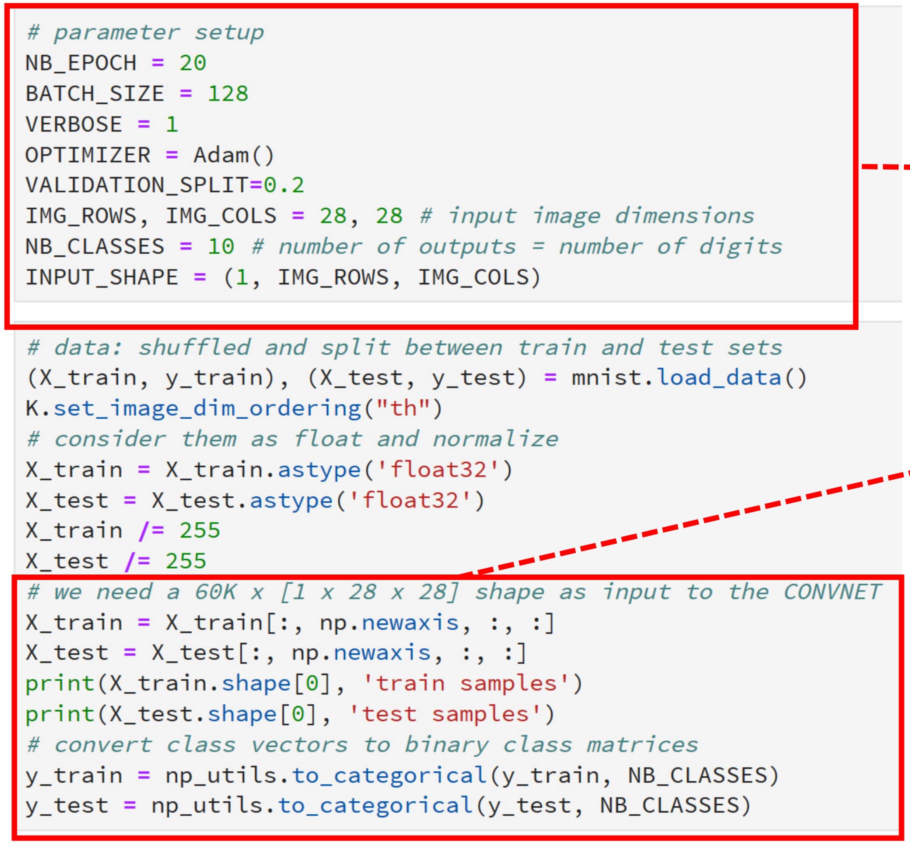
```
# initialize the optimizer and model
model = CNN.build(input_shape=INPUT_SHAPE, classes=NB_CLASSES)
model.compile(loss="categorical_crossentropy", optimizer=OPTIMIZER,
metrics=["accuracy"])
```



```
# printout a summary of the model to understand model complexity
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 20, 28, 28)        520
_________________________________________________________________
activation_1 (Activation)    (None, 20, 28, 28)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 20, 14, 14)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 50, 14, 14)        25050
_________________________________________________________________
activation_2 (Activation)    (None, 50, 14, 14)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 50, 7, 7)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2450)              0
_________________________________________________________________
dense_1 (Dense)              (None, 500)               1225500
_________________________________________________________________
activation_3 (Activation)    (None, 500)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5010
_________________________________________________________________
activation_4 (Activation)    (None, 10)                0
=================================================================
Total params: 1,256,080
Trainable params: 1,256,080
Non-trainable params: 0
_________________________________________________________________
```

MNIST Dataset – Model Parameters & 2D Input Data



```
# parameter setup
NB_EPOCH = 20
BATCH_SIZE = 128
VERBOSE = 1
OPTIMIZER = Adam()
VALIDATION_SPLIT=0.2
IMG_ROWS, IMG_COLS = 28, 28 # input image dimensions
NB_CLASSES = 10 # number of outputs = number of digits
INPUT_SHAPE = (1, IMG_ROWS, IMG_COLS)
```

```
# data: shuffled and split between train and test sets
(X_train, y_train), (X_test, y_test) = mnist.load_data()
K.set_image_dim_ordering("th")
# consider them as float and normalize
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
# we need a 60K x [1 x 28 x 28] shape as input to the CONVNET
X_train = X_train[:, np.newaxis, :, :]
X_test = X_test[:, np.newaxis, :, :]
print(X_train.shape[0], 'train samples')
print(X_test.shape[0], 'test samples')
# convert class vectors to binary class matrices
y_train = np_utils.to_categorical(y_train, NB_CLASSES)
y_test = np_utils.to_categorical(y_test, NB_CLASSES)
```

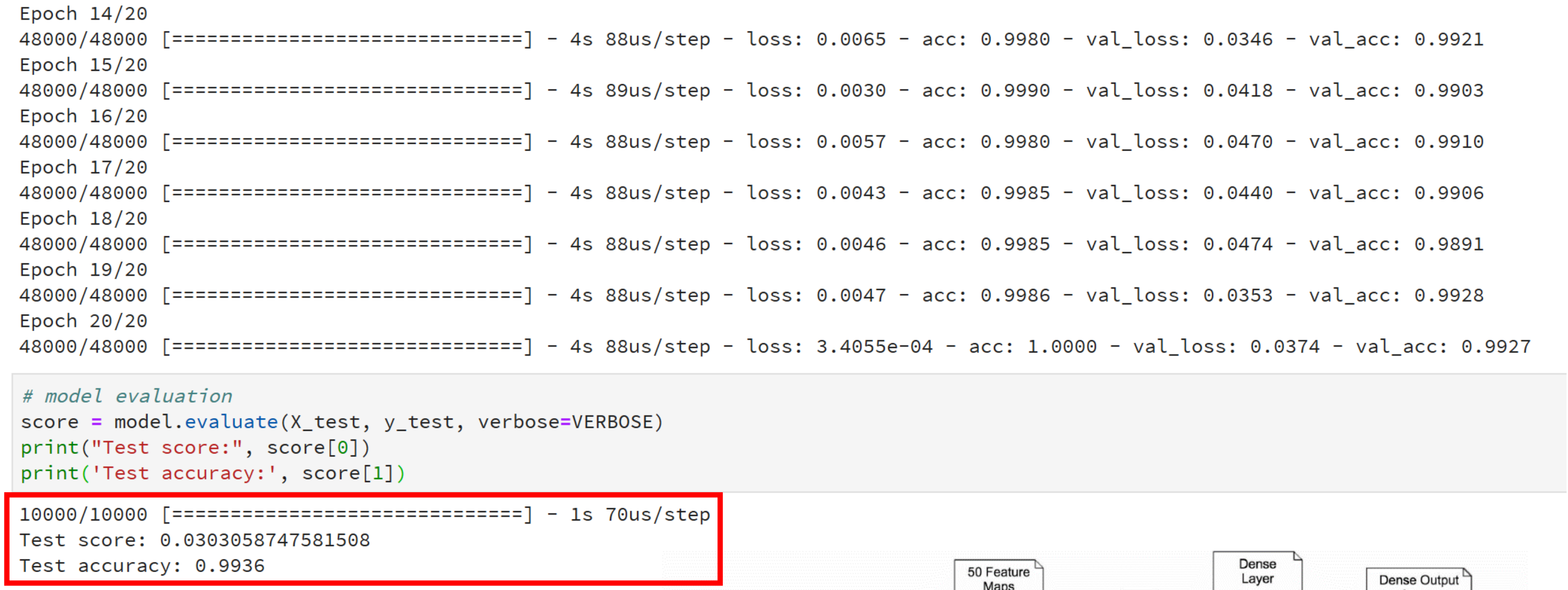
- **OPTIMIZER: Adam - advanced optimization technique that includes the concept of a momentum (a certain velocity component) in addition to the acceleration component of Stochastic Gradient Descent (SGD)**
- **Adam computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients**
- **Adam enables faster convergence at the cost of more computation and is currently recommended as the default algorithm to use (or SGD + Nesterov Momentum)**

[19] D. Kingma et al., 'Adam: A Method for Stochastic Optimization'

- **Compared to the Multi-Output Perceptron and Artificial Neural Networks (ANN) model, the input dataset remains as 2d matricew with 1 x 28 x 28 per image, including also the class vectors that are converted to binary class matrices**

➢ **Assignment #2 will explore the change of parameters in context of changes in running time when training models on GPUs vs. CPUs**

MNIST Dataset – CNN Model Output & Evaluation



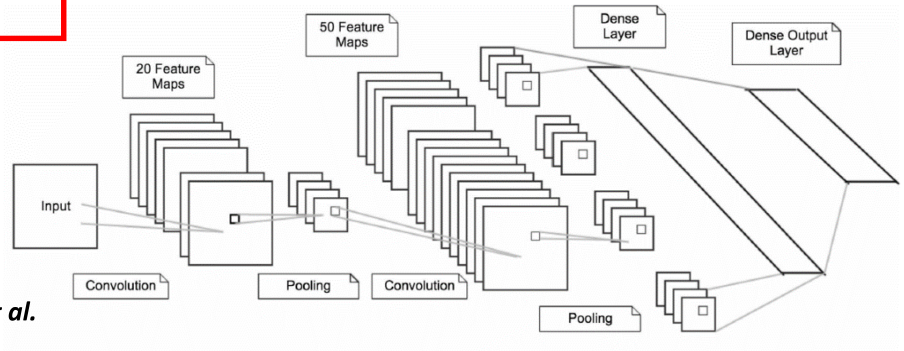
```
Epoch 14/20
48000/48000 [==============================] - 4s 88us/step - loss: 0.0065 - acc: 0.9980 - val_loss: 0.0346 - val_acc: 0.9921
Epoch 15/20
48000/48000 [==============================] - 4s 89us/step - loss: 0.0030 - acc: 0.9990 - val_loss: 0.0418 - val_acc: 0.9903
Epoch 16/20
48000/48000 [==============================] - 4s 88us/step - loss: 0.0057 - acc: 0.9980 - val_loss: 0.0470 - val_acc: 0.9910
Epoch 17/20
48000/48000 [==============================] - 4s 88us/step - loss: 0.0043 - acc: 0.9985 - val_loss: 0.0440 - val_acc: 0.9906
Epoch 18/20
48000/48000 [==============================] - 4s 88us/step - loss: 0.0046 - acc: 0.9985 - val_loss: 0.0474 - val_acc: 0.9891
Epoch 19/20
48000/48000 [==============================] - 4s 88us/step - loss: 0.0047 - acc: 0.9986 - val_loss: 0.0353 - val_acc: 0.9928
Epoch 20/20
48000/48000 [==============================] - 4s 88us/step - loss: 3.4055e-04 - acc: 1.0000 - val_loss: 0.0374 - val_acc: 0.9927
```

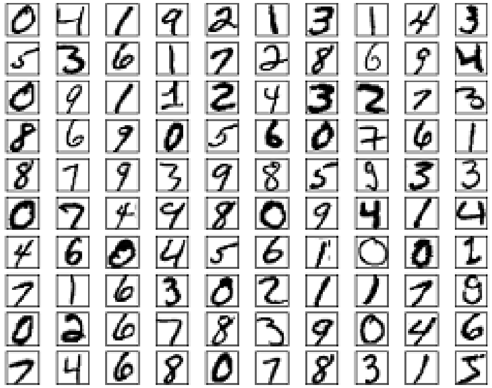
```python
# model evaluation
score = model.evaluate(X_test, y_test, verbose=VERBOSE)
print("Test score:", score[0])
print('Test accuracy:', score[1])
```

```
10000/10000 [==============================] - 1s 70us/step
Test score: 0.0303058747581508
Test accuracy: 0.9936
```

- ✓ **Multi Output Perceptron: ~91,01% (20 Epochs)**
- ✓ **ANN 2 Hidden Layers: ~95,14 % (20 Epochs)**
- ✓ **CNN Deep Learning Model: ~99,36 % (20 Epochs)**

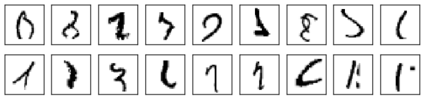
[18] A. Gulli et al.





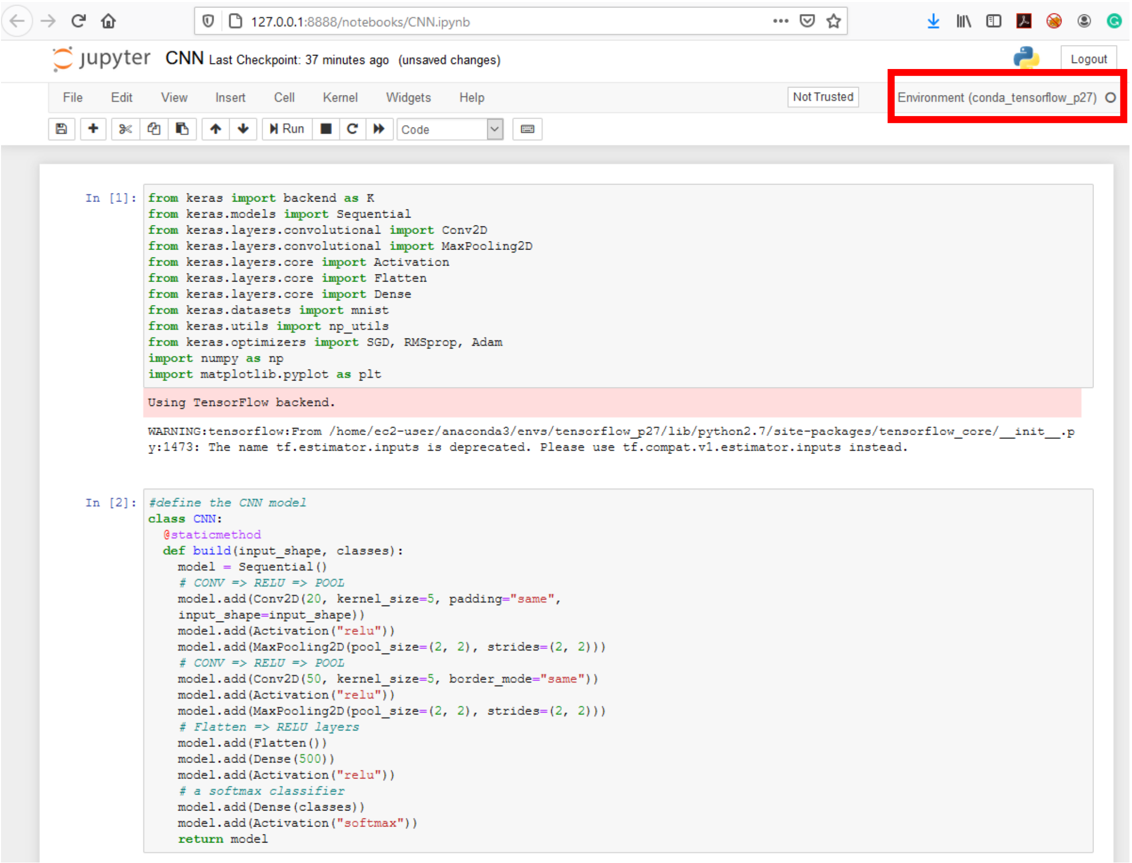
Why not 100%

?



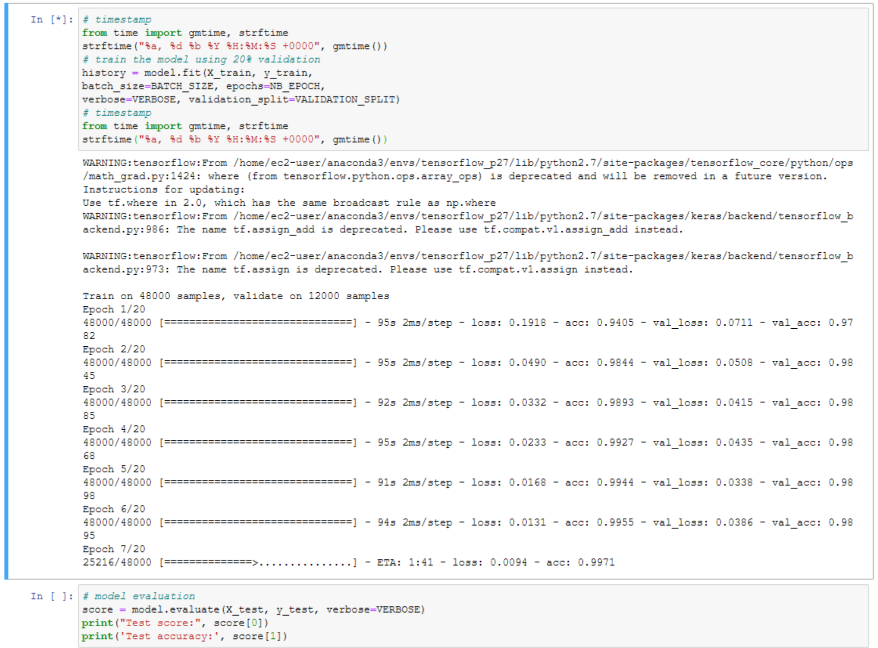
some samples even for a human unrecognizable

Running a Deep Learning Model with Convolutional Neural Network (CNN)


127.0.0.1:8888/notebooks/CNN.ipynb
jupyter
CNN Last Checkpoint: 37 minutes ago (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Run
Code

Environment (conda_tensorflow_p27)

```
In [1]: from keras import backend as K
        from keras.models import Sequential
        from keras.layers.convolutional import Conv2D
        from keras.layers.convolutional import MaxPooling2D
        from keras.layers.core import Activation
        from keras.layers.core import Flatten
        from keras.layers.core import Dense
        from keras.datasets import mnist
        from keras.utils import np_utils
        from keras.optimizers import SGD, RMSprop, Adam
        import numpy as np
        import matplotlib.pyplot as plt

        Using TensorFlow backend.

        WARNING:tensorflow:From /home/ec2-user/anaconda3/envs/tensorflow_p27/lib/python2.7/site-packages/tensorflow_core/__init__.p
        y:1473: The name tf.estimator.inputs is deprecated. Please use tf.compat.v1.estimator.inputs instead.

In [2]: #define the CNN model
        class CNN:
          @staticmethod
          def build(input_shape, classes):
            model = Sequential()
            # CONV => RELU => POOL
            model.add(Conv2D(20, kernel_size=5, padding="same",
            input_shape=input_shape))
            model.add(Activation("relu"))
            model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
            # CONV => RELU => POOL
            model.add(Conv2D(50, kernel_size=5, border_mode="same"))
            model.add(Activation("relu"))
            model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
            # Flatten => RELU layers
            model.add(Flatten())
            model.add(Dense(500))
            model.add(Activation("relu"))
            # a softmax classifier
            model.add(Dense(classes))
            model.add(Activation("softmax"))
            return model
```



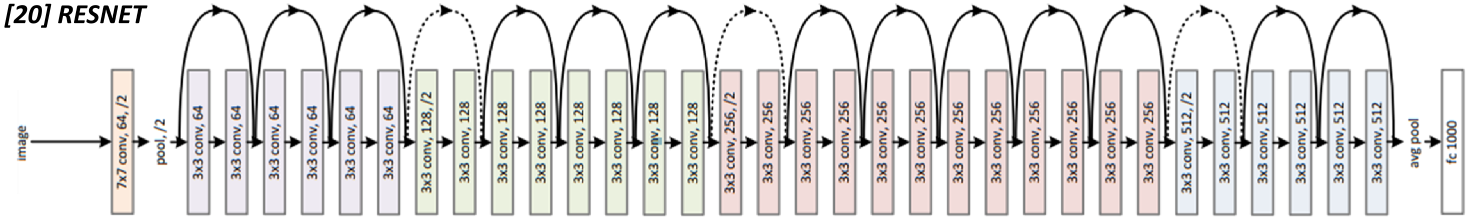
```
In [*]: # timestamp
        from time import gmtime, strftime
        strftime("%a, %d %b %Y %H:%M:%S +0000", gmtime())
        # train the model using 20% validation
        history = model.fit(X_train, y_train,
        batch_size=BATCH_SIZE, epochs=NB_EPOCH,
        verbose=VERBOSE, validation_split=VALIDATION_SPLIT)
        # timestamp
        from time import gmtime, strftime
        strftime("%a, %d %b %Y %H:%M:%S +0000", gmtime())

        WARNING:tensorflow:From /home/ec2-user/anaconda3/envs/tensorflow_p27/lib/python2.7/site-packages/tensorflow_core/python/ops
        /math_grad.py:1424: where (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
        Instructions for updating:
        Use tf.where in 2.0, which has the same broadcast rule as np.where
        WARNING:tensorflow:From /home/ec2-user/anaconda3/envs/tensorflow_p27/lib/python2.7/site-packages/keras/backend/tensorflow_b
        ackend.py:986: The name tf.assign_add is deprecated. Please use tf.compat.v1.assign_add instead.

        WARNING:tensorflow:From /home/ec2-user/anaconda3/envs/tensorflow_p27/lib/python2.7/site-packages/keras/backend/tensorflow_b
        ackend.py:973: The name tf.assign is deprecated. Please use tf.compat.v1.assign instead.

        Train on 48000 samples, validate on 12000 samples
        Epoch 1/20
        48000/48000 [==============================] - 95s 2ms/step - loss: 0.1918 - acc: 0.9405 - val_loss: 0.0711 - val_acc: 0.97
        82
        Epoch 2/20
        48000/48000 [==============================] - 95s 2ms/step - loss: 0.0490 - acc: 0.9844 - val_loss: 0.0508 - val_acc: 0.98
        45
        Epoch 3/20
        48000/48000 [==============================] - 92s 2ms/step - loss: 0.0332 - acc: 0.9893 - val_loss: 0.0415 - val_acc: 0.98
        85
        Epoch 4/20
        48000/48000 [==============================] - 95s 2ms/step - loss: 0.0233 - acc: 0.9927 - val_loss: 0.0435 - val_acc: 0.98
        68
        Epoch 5/20
        48000/48000 [==============================] - 91s 2ms/step - loss: 0.0168 - acc: 0.9944 - val_loss: 0.0338 - val_acc: 0.98
        98
        Epoch 6/20
        48000/48000 [==============================] - 94s 2ms/step - loss: 0.0131 - acc: 0.9955 - val_loss: 0.0386 - val_acc: 0.98
        95
        Epoch 7/20
        25216/48000 [==============>...............] - ETA: 1:41 - loss: 0.0094 - acc: 0.9971

In [ ]: # model evaluation
        score = model.evaluate(X_test, y_test, verbose=VERBOSE)
        print("Test score:", score[0])
        print('Test accuracy:', score[1])
```

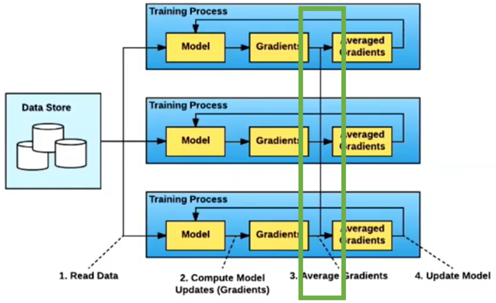
- **Using Deep Learning Techniques such as Convolutional Neural Networks (CNNs) in clouds can lead to significant improvements in accuracy, but also to significant longer run-times than traditional Artificial Neural Networks (ANNs) and are thus much more costly in clouds**
- **Using CPU resources for deep learning techniques is usually not recommended**

More Computation: Deep Learning via RESNET-50 Architecture (cf. Lecture 6 & 7)

- Application Example: Classification of land cover in scenes on remote sensing datasets
  - Very suitable for parallelization via distributed training on multi GPUs

[20] RESNET

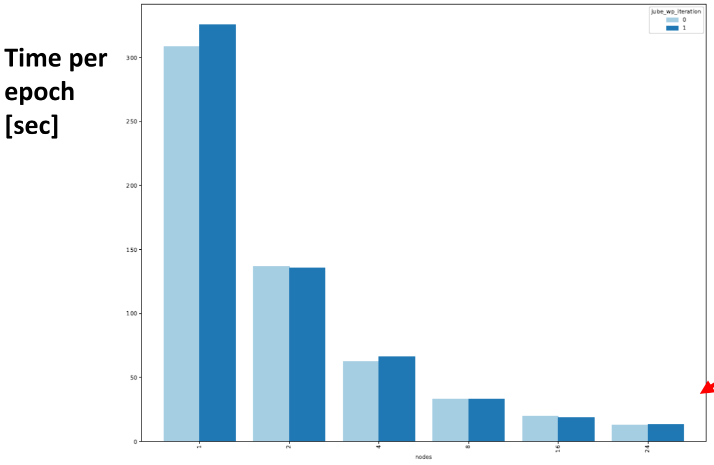




[22] Horovod

MPI_Allreduce()

- **RESNET-50 is a known neural network architecture that has established a strong baseline in terms of accuracy**
- **The computational complexity of training the RESNET-50 architecture relies in the fact that is has ~ <u>25.6 millions of trainable parameters</u>**
- **RESNET-50 still represents a good trade-off between accuracy, depth and number of parameters**
- **The setups of RESNET-50 makes it very suitable for parallelization via distributed training on multi GPUs**

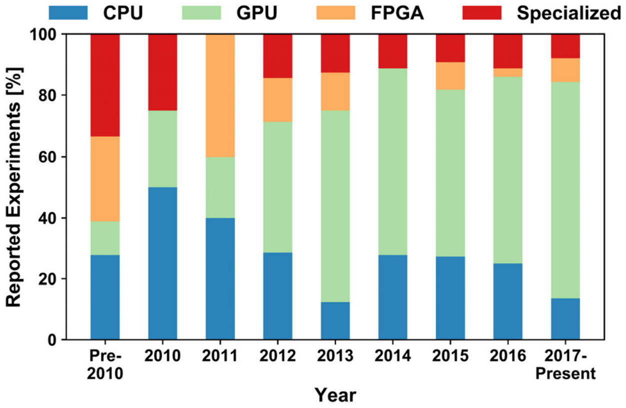


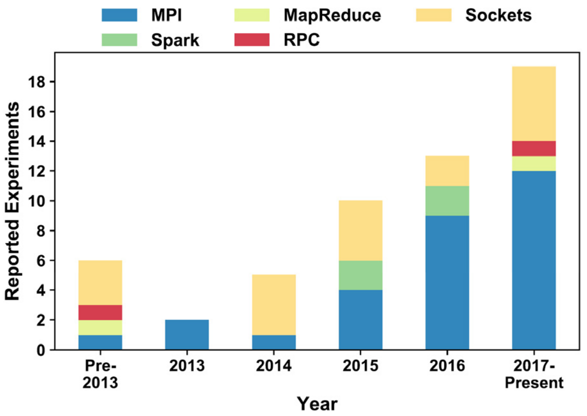
A partition of the JUWELS system has 56 compute nodes, each with 4 NVIDIA V100 GPUs (equipped with 16 GB of memory)

24 nodes x 4 GPUs = 96 GPUs

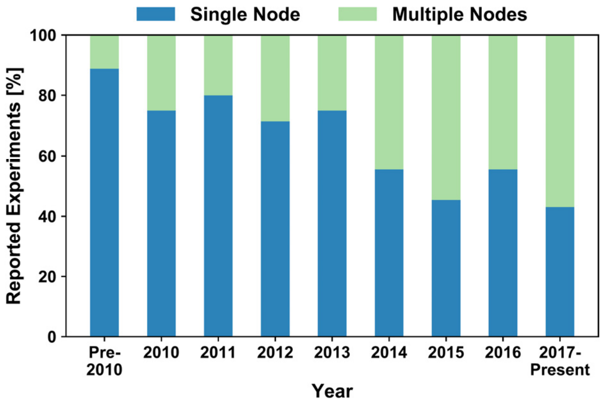
[21] R. Sedona & M. Riedel et al., 2019

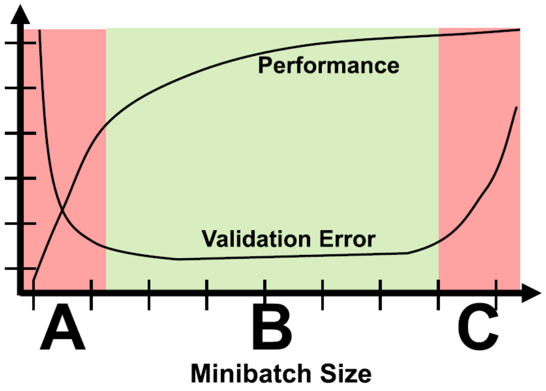
Cloud Computing & HPC using GPUs for Deep Learning – Hardware Complexity

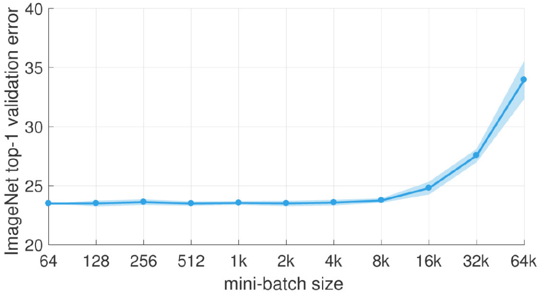




- **Facts: GPUs are mostly used today for deep learning compared to CPUs, FPGA, and specialized hardware**
- **Facts: ~55% of all users that use deep learning use it with multiple nodes instead of just a single node**
- **Facts: The communication layer MPI is mostly used as communication layer for distributed training compared to Apache Spark, Remote Procedure Calls, Apache Hadoop MapReduce, or traditional Sockets**
- **Most users use deep learning today with minibatches that are selected numbers of samples for performing the optimization (e.g. SGD on minibatches)**
- **Minibatches should be not too small to increate performance, but also not too large to increase validation error**





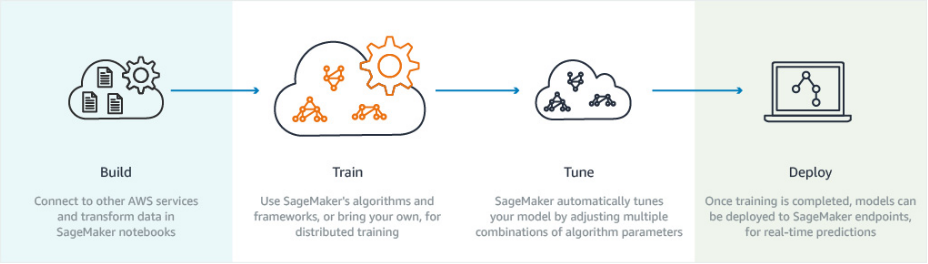


[23] T. Ben-Nun & T. Hoefler

> **Complementary High Performance Computing course offers insights into parallel programming models such as MPI & hardware impact**

AWS Amazon Sagemaker – SAAS Service to Abstract from Hardware Complexity

- AWS Cloud – Amazon Sagemaker
  - Fully managed service that enables quick & easy machine & deep learning applications
  - Avoids time-consuming manual installation of many required software frameworks
  - Builds on-top of various IAAS & PAAS services

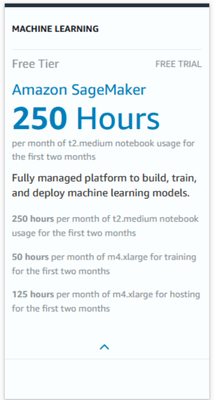


Build
Connect to other AWS services and transform data in SageMaker notebooks

Train
Use SageMaker's algorithms and frameworks, or bring your own, for distributed training

Tune
SageMaker automatically tunes your model by adjusting multiple combinations of algorithm parameters

Deploy
Once training is completed, models can be deployed to SageMaker endpoints, for real-time predictions



MACHINE LEARNING

Free Tier FREE TRIAL

Amazon SageMaker

250 Hours

per month of t2.medium notebook usage for the first two months

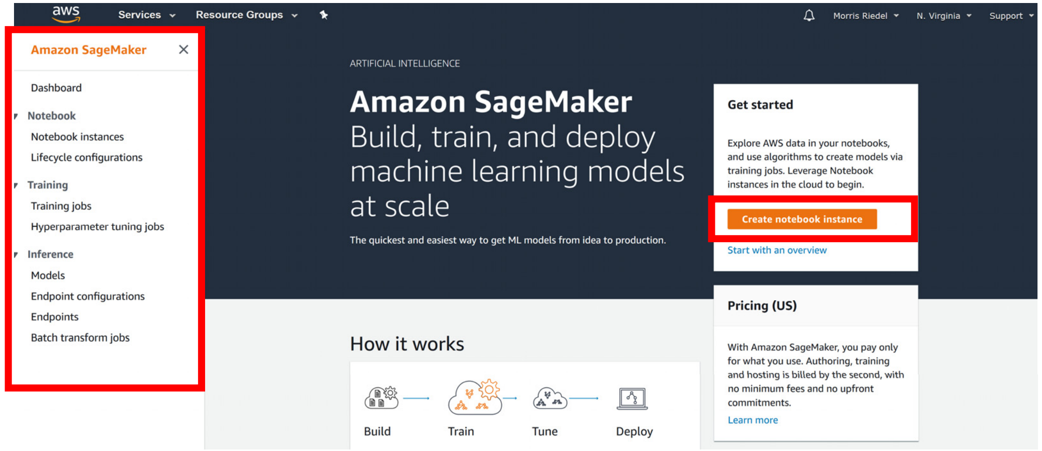
Fully managed platform to build, train, and deploy machine learning models.

250 hours per month of t2.medium notebook usage for the first two months

50 hours per month of m4.xlarge for training for the first two months

125 hours per month of m4.xlarge for hosting for the first two months

(SAAS solutions often abstracts away completely underlying resources)



ARTIFICIAL INTELLIGENCE

Amazon SageMaker
Build, train, and deploy machine learning models at scale

The quickest and easiest way to get ML models from idea to production.

Get started

Explore AWS data in your notebooks, and use algorithms to create models via training jobs. Leverage Notebook instances in the cloud to begin.

Pricing (US)

With Amazon SageMaker, you pay only for what you use. Authoring, training and hosting is billed by the second, with no minimum fees and no upfront commitments.

How it works

- **AWS Amazon Sagemaker is a SAAS oriented service that provides fully managed instances running Jupyter notebooks that include examples training & tuning various machine and deep learning models**
- **Offers Amazon SageMaker Studioas a fully integrated development environment (IDE) for machine learning in the AWS cloud**
- **SAAS services are usually not free and often require a subscription**

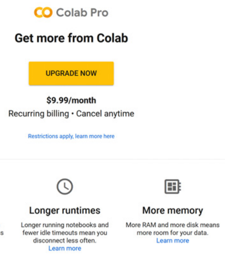
[9] Jupyter Web page

[24] AWS – Amazon Sagemaker

> **Lecture 10 provides more details about AWS Cloud services and its Software-as-a-Service (SAAS) models & other SAAS cloud services**

Using Google Colaboratory Cloud Infrastructure for Deep Learning with GPUs

- Google Colaboratory (free & pro version for 9.99 $ / month)
  - 'Colab' notebooks are Jupyter notebooks that run in the Google cloud
  - Possible to run Apache Spark via PySpark Jupyter notebooks in Colab (cf. Lecture 3)
  - Possible to train Deep Learning networks via GPUs & Jupyter notebooks in Colab
  - Highly integrated with other Google services (e.g., Google Drive for data)
  - Access to vendor-specific Tensor Processing Units (TPUs)



Colab Pro

Get more from Colab

$9.99/month

Recurring billing • Cancel anytime

Restrictions apply, learn more here

Faster GPUs

Priority access to faster GPUs and TPUs means you spend less time waiting while code is running. Learn more

Longer runtimes

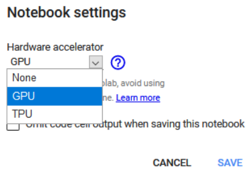
Longer running notebooks and fewer idle timeouts mean you disconnect less often. Learn more

More memory

More RAM and more disk means more room for your data. Learn more

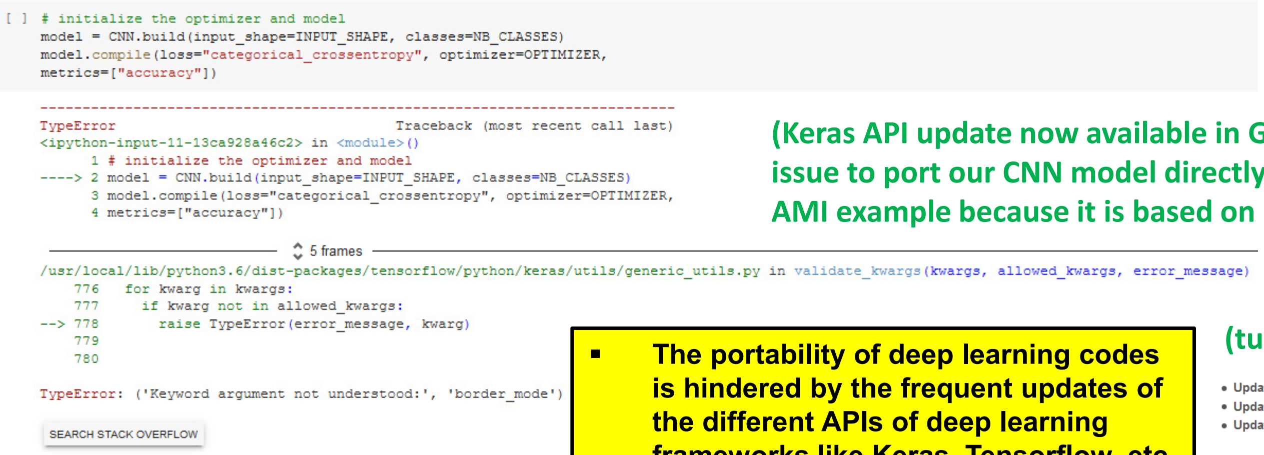
[27] Google Colaboratory

(for international students: watch out – it uses the browser language automatically)



Notebook settings

Hardware accelerator: GPU



```
[ ] # initialize the optimizer and model
    model = CNN.build(input_shape=INPUT_SHAPE, classes=NB_CLASSES)
    model.compile(loss="categorical_crossentropy", optimizer=OPTIMIZER,
    metrics=["accuracy"])

    ---------------------------------------------------------------------------
    TypeError                                 Traceback (most recent call last)
    <ipython-input-11-13ca928a46c2> in <module>()
          1 # initialize the optimizer and model
    ----> 2 model = CNN.build(input_shape=INPUT_SHAPE, classes=NB_CLASSES)
          3 model.compile(loss="categorical_crossentropy", optimizer=OPTIMIZER,
          4 metrics=["accuracy"])

    ─────────────────────────────── ⇕ 5 frames ───────────────────────────────
    /usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/utils/generic_utils.py in validate_kwargs(kwargs, allowed_kwargs, error_message)
        776   for kwarg in kwargs:
        777     if kwarg not in allowed_kwargs:
    --> 778       raise TypeError(error_message, kwarg)
        779
        780

    TypeError: ('Keyword argument not understood:', 'border_mode')
```

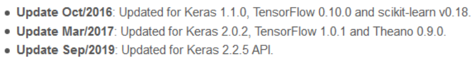
(Keras API update now available in Google 'Colab' creates an issue to port our CNN model directly from our Amazon EC2 AMI example because it is based on previous versions of Keras)

- Google Colaboratory offers 'Colab' notebooks that are implemented with Jupyter notebooks that in turn run in the Google cloud and are highly integrated with other Google cloud services such as Google Drive thus making 'Colab' notebooks easy to set up, access, and share with others

(Clouds also face this update problem)

- The portability of deep learning codes is hindered by the frequent updates of the different APIs of deep learning frameworks like Keras, Tensorflow, etc. (cf. different AWS EC2 AMI versions)

(tutorials & codes need updates)

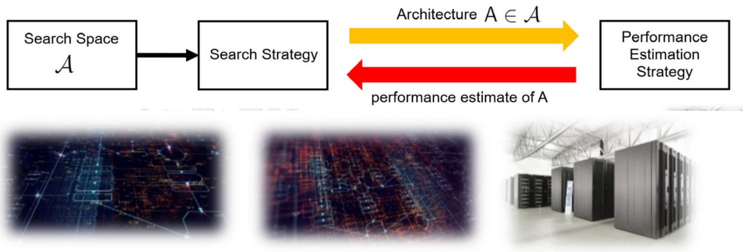


- **Update Oct/2016**: Updated for Keras 1.1.0, TensorFlow 0.10.0 and scikit-learn v0.18.
- **Update Mar/2017**: Updated for Keras 2.0.2, TensorFlow 1.0.1 and Theano 0.9.0.
- **Update Sep/2019**: Updated for Keras 2.2.5 API.

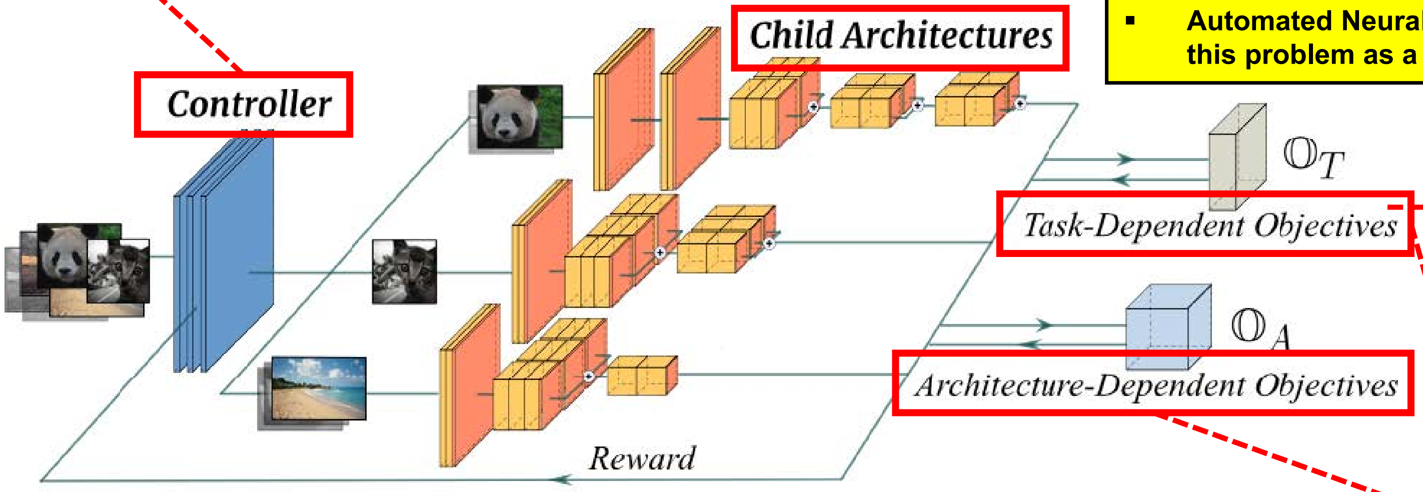
[28] Machine Learning Mastery MNIST Tutorial

Massive Requirement for Cloud Resources: Neural Architecture Search (NAS)

- Often a Recurrent Neural Network (RNN) technique that performs the agent steps


Search Space $\mathcal{A}$
Search Strategy
Architecture $A \in \mathcal{A}$
performance estimate of A
Performance Estimation Strategy

- Employed neural networks architectures are often developed manually by human experts that is time-consuming and error-prone
- Deep learning success has been accompanied by a rising demand for architecture engineering, where increasingly more complex neural architectures are designed manually
- Neural Architecture Search (NAS) methods can be categorized in (a) search space, (b) search strategy, and (c) performance estimation strategy
- Automated Neural Architecture (NAS) search methods aim to solve this problem as a process of automating Architecture engineering



[26] M. Riedel, 'NAS with Reinforcement Learning'

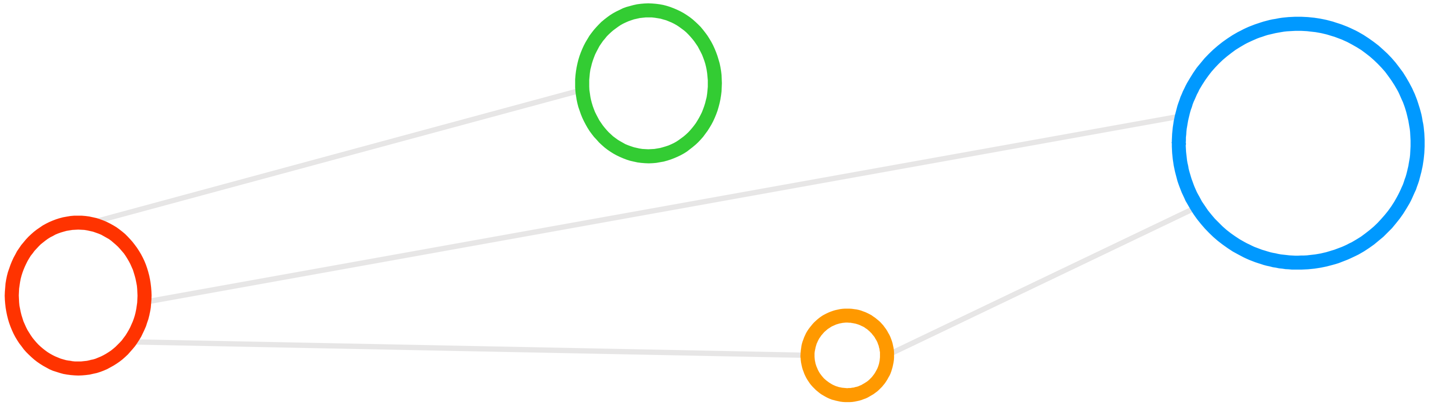
- Derived specific architectures that perform good for specific dataset samples

- E.g. what is the accuracy or error rate we obtain as metric to guide the search for specific architectures for specific dataset samples

- E.g. what is the latency of the network for a given dataset sample to guide the search for specific architectures that offer better latency by keeping accuracy(!)

[25] A.C. Cheng et al., 'InstaNAS: Instance-aware Neural Architecture Search', 2018

Lecture Bibliography



Lecture Bibliography (1)

- [1] Jupyter @ Juelich Supercomputing Centre, Online:
  https://jupyter-jsc.fz-juelich.de/
- [2] Tensorflow Deep Learning Framework, Online:
  https://www.tensorflow.org/
- [3] Keras Python Deep Learning Library, Online:
  https://keras.io/
- [4] Amazon Web Services Educate Web Page, Online:
  https://aws.amazon.com/education/awseducate/
- [5] Amazon Web Services Web Page, Online:
  https://aws.amazon.com
- [6] Key Concepts from the AWS Cloud, Online:
  https://blogs.sap.com/2010/01/28/key-concepts-from-the-aws-cloud/
- [7] SSH Client MobaXterm**,** Online:
  https://mobaxterm.mobatek.net/
- [8] Amazon Web Services EC2 On-Demand Pricing models, Online:
  https://aws.amazon.com/ec2/pricing/on-demand/
- [9] Jupyter Web page, Online:
  http://jupyter.org/
- [10] Big Data Tips, ‘Gradient Descent‘, Online:
  http://www.big-data.tips/gradient-descent
- [11] Morris Riedel, ‘Deep Learning - Using a Convolutional Neural Network‘, Invited YouTube Lecture, University of Ghent, 2017, Online:
  https://www.youtube.com/watch?v=gOL1_YIosYk&list=PLrmNhuZo9sgZUdaZ-f6OHK2yFW1kTS2qF
- [12] M. Riedel et al., ‘Introduction to Deep Learning Models‘, JSC Tutorial, three days, JSC, 2019, Online:
  http://www.morrisriedel.de/introduction-to-deep-learning-models
- [13] H. Lee et al., ‘Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations’, Online:
  http://doi.acm.org/10.1145/1553374.1553453

Lecture Bibliography (2)

- [13] H. Lee et al., ‘Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations’, Online: http://doi.acm.org/10.1145/1553374.1553453
- [14] YouTube Video, ‘Neural Network 3D Simulation‘, Online: https://www.youtube.com/watch?v=3JQ3hYko51Y
- [15] A. Rosebrock, ‘Get off the deep learning bandwagon and get some perspective‘, Online: http://www.pyimagesearch.com/2014/06/09/get-deep-learning-bandwagon-get-perspective/
- [16] Big Data Tips – Big Data Mining & Machine Learning, Online: http://www.big-data.tips/
- [17] Harley, A.W., An Interactive Node-Link Visualization of Convolutional Neural Networks, Online: http://scs.ryerson.ca/~aharley/vis/conv/flat.html
- [18] A. Gulli and S. Pal, ‘Deep Learning with Keras‘ Book, ISBN-13 9781787128422, 318 pages, Online: https://www.packtpub.com/big-data-and-business-intelligence/deep-learning-keras
- [19] D. Kingma and Jimmy Ba, ‘Adam: A Method for Stochastic Optimization‘, Online: https://arxiv.org/abs/1412.6980
- [20] Kaiming He et al., ‘Deep Residual Learning for Image Recognition’, Online: https://arxiv.org/pdf/1512.03385.pdf
- [21] Sedona, R., Cavallaro, G., Riedel, M., Benediktsson, J.A. et. al.: Remote Sensing Big Data Classification with High Performance Distributed Deep Learning, Journal of Remote Sensing, Multidisciplinary Digital Publishing Institute (MDPI), Special Issue on Analysis of Big Data in Remote Sensing, 2019, Online: https://www.researchgate.net/publication/338077024_Remote_Sensing_Big_Data_Classification_with_High_Performance_Distributed_Deep_Learning
- [22] Horovod: Uber’s Open Source Distributed Deep Learning Framework for TensorFlow, Online: https://www.slideshare.net/databricks/horovod-ubers-open-source-distributed-deep-learning-framework-for-tensorflow
- [23] T. Ben-Nun & T. Hoefler, ‘Demystifying Parallel and Distributed Deep Learning: An In-depth Concurrency Analysis‘, Online: http://doi.acm.org/10.1145/3320060
- [24] AWS Amazon Sagemaker Service, Online: https://aws.amazon.com/sagemaker

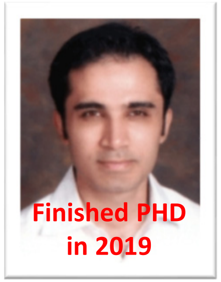
Lecture Bibliography (3)

- [25] Cheng, A.C, Lin, C.H., Juan, D.C., InstaNAS: Instance-aware Neural Architecture Search, Online: https://arxiv.org/abs/1811.10201
- [26] M. Riedel, 'Neural Architecture Search with Reinforcement Learning', Online: http://www.morrisriedel.de/neural-architecture-search-with-reinforcement-learning
- [27] Google Colaboratory, Online: https://colab.research.google.com
- [28] Machine Learning Mastery MNIST Tutorial, Online: https://machinelearningmastery.com/handwritten-digit-recognition-using-convolutional-neural-networks-python-keras/
- [29] The XOR Problem in Neural Networks, Online: https://medium.com/@jayeshbahire/the-xor-problem-in-neural-networks-50006411840b
- [30] F. Rosenblatt, 'The Perceptron - a perceiving and recognizing automaton', Report 85-460-1, Cornell Aeronautical Laboratory, 1957, Online: https://blogs.umass.edu/brain-wars/files/2016/03/rosenblatt-1957.pdf
- [31] MIT 6.S191: Introduction to Deep Learning, Online: http://introtodeeplearning.com/
- [32] An overview of gradient descent optimization algorithm, Online: http://ruder.io/optimizing-gradient-descent/index.html#visualizationofalgorithms
- [33] www.big-data.tips, 'MNIST Database', Online: http://www.big-data.tips/mnist-database
- [34] www.big-data.tips, 'Relu Neural Network‘, Online: http://www.big-data.tips/relu-neural-network
- [35] www.big-data.tips, 'tanh‘, Online: http://www.big-data.tips/tanh
- [36] www.big-data.tips,'MNIST dataset‘, Online: http://www.big-data.tips/mnist-dataset

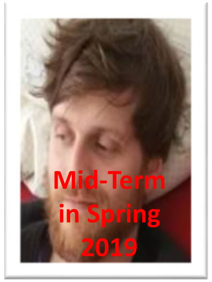
Acknowledgements – High Productivity Data Processing Research Group

PD Dr.
G. Cavallaro

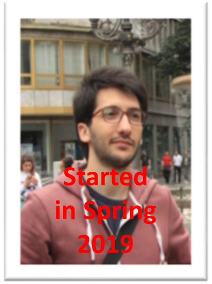
Senior PhD
Student A.S. Memon



Senior PhD
Student M.S. Memon



PhD Student
E. Erlingsson



PhD Student
S. Bakarat

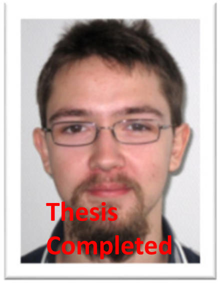
PhD Student
R. Sedona

Morris Riedel @MorrisRiedel · Feb 10

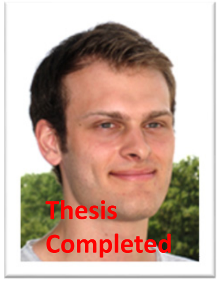
Enjoying our yearly research group dinner 'Iceland Section' to celebrate our productive collaboration of @uni_iceland @uisens @Haskoli_Islands & @fzj_jsc @fz_juelich & E.Erlingsson @ernire passed mid-term in modular supercomputing driven by @DEEPprojects - morrisriedel.de/research

You and Ernir Erlingsson

Dr. M. Goetz
(now KIT)

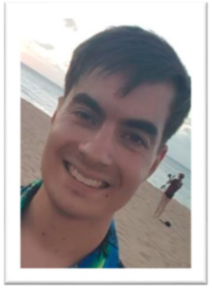


MSc M.
Richerzhagen
(now other division)



MSc
P. Glock
(now INM-1)

MSc
C. Bodenstein
(now
Soccerwatch.tv)



MSc Student
G.S. Guðmundsson
(Landsverkjun)

This research group has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 763558 (DEEP-EST EU Project)

services
measurement
within
fusion
new
funding
Policy-based
concepts
device
Analysis
forms
cross-disciplinary
resources
Centre
climate
Computer
expertise
Approaches
construction
Fusion
year
like
Computational
Test
E.g
Computing
disciplines
results
Enable
different
Cross-Disciplinary
methods
High
big
nodes
PB
community
Environment
Modelling
EU
Science
centers
Key
scientific
important
files
model
etc
DKRZ
Use
storage
basis
Infrastructure
international
simulations
ENES
technologies
formats
brain
increasing
via
EUDAT
physical
IPCC
SMAQ
engineering
References
Big
Project
Data
processing
Common
TB
Methods
MapReduce
create
computing
Using
European
databases
GB
Supercomputing
Work
images
codes
analysis
Large
Summary
data-intensive
Service
Scientific
levels
often
stored
Resources
analyze
external
DLCL
Level
use
Research
Matters
Concrete
performance
JSC
computational
datasets
research
e.g
science
HPC
Juelich
system
acquisition
compute
Landscape
management
System
Climate
modelling
AR5
Hardware
distributed
large-scale
cores
access
hundreds
Nutshell
Services
Technologies
Earth
Structure
current
network
DLCLs
Understanding
structures
directory
project
General
Germany
Simulation
Processing
Provide
NASA
Energy
systems
day
Health