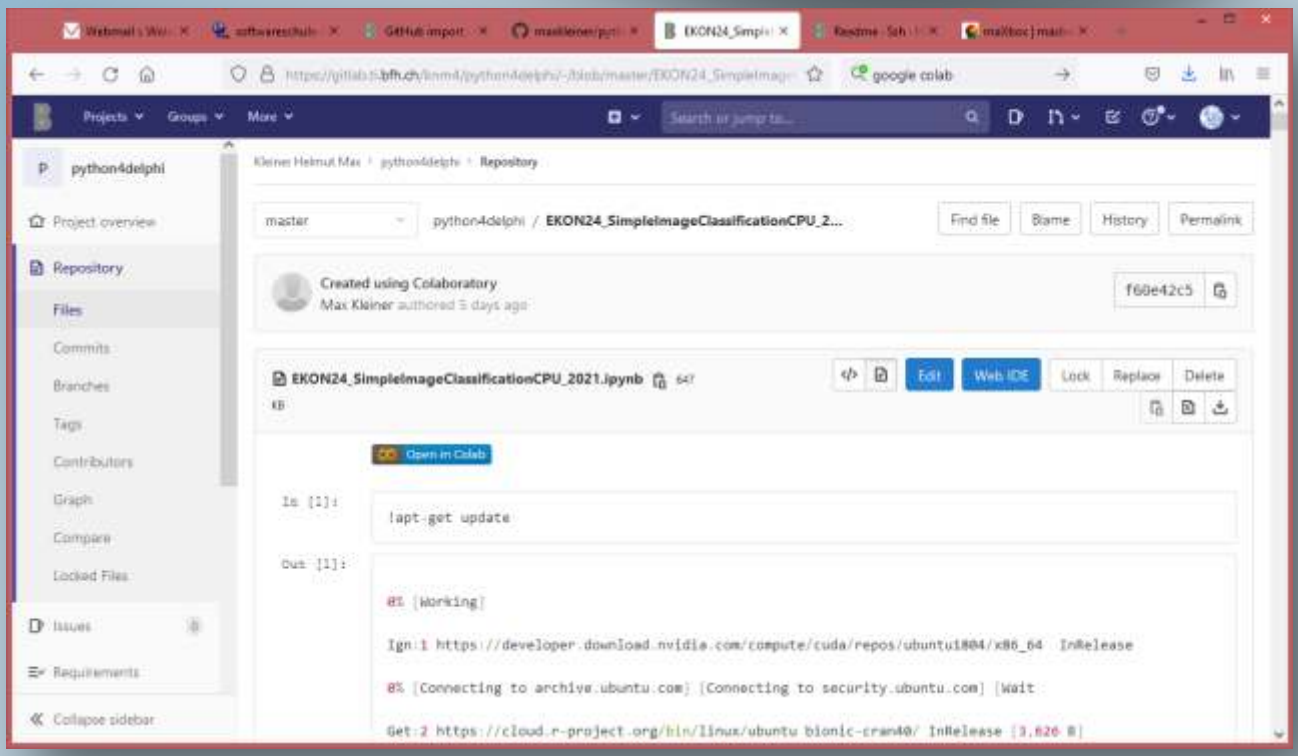
colab

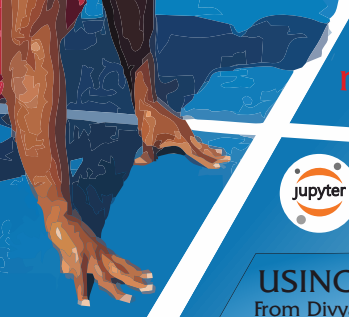
AUTHOR: MAX KLEINER `finally begin. – Max`

With the following report I show how to host and execute a deep learning project on a cloud. The cloud is hosted by google colab and enables working and testing in teams. Lazarus is also being built in colab and the deep learning network is compiled and trained too in a Jupyter notebook.

`https://gitlab.ti.bfh.ch/knm4/python4delphi/-/blob/master/EKON24_SimpleImageClassificationCPU_2021.ipynb`

So what's Colab? With Colab you can import an image dataset, train an image classifier on it, and evaluate the model, all in just a few lines of code.
Colab notebooks execute code on Google's cloud servers, meaning you can leverage the power of Google hardware, including GPUs and TPUs, regardless of the power of your machine.
We at BFH also use this service.
The Bern University of Applied Sciences (BFH) is one of the leading application-oriented universities in Switzerland.
With 31 Bachelor's and 25 Master's courses, we offer a wide range of training and further education.





jupyter colab

USING JUPYTER NOTEBOOKS AND GOOGLE COL

From Divya Singh
https://www.datasciencecentral.com/profile/DivyaSingh456

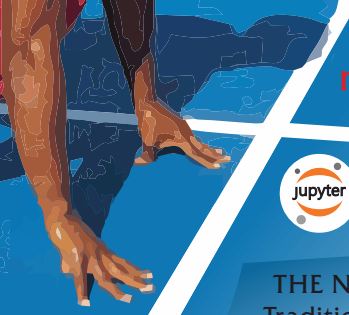
Let's start with an explanation of **Jupyter notebooks** and **Google colab.**
We try to wite code in the notebooks and focus on the basic features of notebooks.
Before diving directly into writing code, let us familiarise ourselves with writing the code notebook style!

WHAT ARE JUPYTER NOTEBOOKS?

- The **Jupyter Notebook** is an **open source web application** that you can use to create and share documents that contain live code, equations, visualizations, and text.
  Jupyter Notebook is maintained by the people at Project Jupyter.
- **Jupyter Notebooks** are a spin-off project from the **IPython** project, which used to have an **IPython Notebook** project itself.
- The name, **Jupyter,** comes from the core supported programming languages that it supports: Julia, Python, and R.
- **Jupyter** ships with the **IPYTHON** kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.
- **Jupyter** is a project aiming to **standardize interactive computing in any programming languages.** The kernel provides interactive environment that **executes user code as a server,** connected with a **frontend through sockets.**
  (A network socket is a software structure within a network node of a computer network that serves as an endpoint for sending and receiving data across the network).
- A **Jupyter Notebook** is also an **open-source web application** that allows you to create and share documents that contain live code, equations, visualizations, and describing text.

Interactive notebooks are experiencing a rise in popularity.
How do we know?
They're replacing PowerPoint in presentations, shared around organizations, and they're even taking workload away from BI (*1) suites, Jupyter, R Markdown, Apache Zeppelin, Spark Notebook and more. There are kernels/backends to multiple languages, such as Python, Julia, Scala, SQL, and others. Notebooks are typically used by data scientists for quick exploration tasks.

*B *1(Microsoft Business Intelligence (BI) is a suite of products and tools that you can use to monitor, analyze and plan your business by using scorecards, dashboards, management reporting and analytics. It contains the following tools: SQL Server Analysis Services (SSAS) SQL Server Integration Services (SSIS)). Today there are many notebooks to choose from)*



jupyter colab

THE NOTEBOOK WAY

Traditionally, notebooks have been used to document research and make results reproducible, simply by rerunning the notebook on source data. But why would one want to choose to use a notebook instead of a favorite IDE or command line?

There are many limitations in the current browser-based notebook implementations, but what they do offer is an **environment for exploration, collaboration, and visualization.** Notebooks are typically used by data scientists for quick exploration tasks. In that regard, they offer a number of advantages over any local scripts or tools. Notebooks also tend to be set up in a cluster environment, allowing the data scientist to take advantage of computational resources beyond what is available on her laptop, and operate on the full data set without having to download a local copy.

Why Jupyter Notebooks

Jupyter notebooks are particularly useful as scientific lab books when you are doing computational physics and/or lots of data analysis using computational tools. This is because, with Jupyter notebooks, you can:

- Record the code you write in a notebook as you manipulate your data. This is useful to remember what you've done, repeat it if necessary, etc.
- Graphs and other figures are rendered directly in the notebook so there's no more printing to paper, cutting and pasting as you would have with paper notebooks or copying and pasting as you would have with other electronic notebooks.
- You can have dynamic data visualizations, e.g. animations, which is simply not possible with a paper lab book.
- One can update the notebook (or parts thereof) with new data by re-running cells.
- You could also copy the cell and re-run the copy only if you want to retain a record of the previous attempt.

Google Colab

Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. With Colaboratory you can write and execute code, save and share your analyses, and access powerful computing resources, all for free from your browser. Colab let's you import an image dataset, train an image classifier on it, and evaluate the model, all in just a few lines of code.

Colab notebooks execute code on Google's cloud servers, meaning you can leverage the power of Google hardware, including GPUs and TPUs, regardless of the power of your machine.

Why Google Colab

As the name suggests, Google Colab comes with collaboration backed in the product. In fact, it is a Jupyter notebook that leverages Google Docs collaboration features. It also runs on Google servers and you don't need to install anything. Moreover, the notebooks are saved to your Google Drive account.

jupyter colab

TPU

Tensor Processing Unit (TPU) is an AI accelerator application-specific integrated circuit (ASIC) developed by Google specifically for neural network machine learning, particularly using Google's own TensorFlow software.
https://en.wikipedia.org/wiki/Tensor_Processing_Unit

LAZBUILD

Lazbuild is a command line utility to compile Lazarus projects and packages, as well as the Lazarus IDE itself. When you built Lazarus yourself you can find the lazbuild executable in the Lazarus source directory together with the lazarus executable.
When you find an error like "/bin/bash: lazbuild: command not found" then you missed !apt-get install fpc fpc-source lazarus git subversion

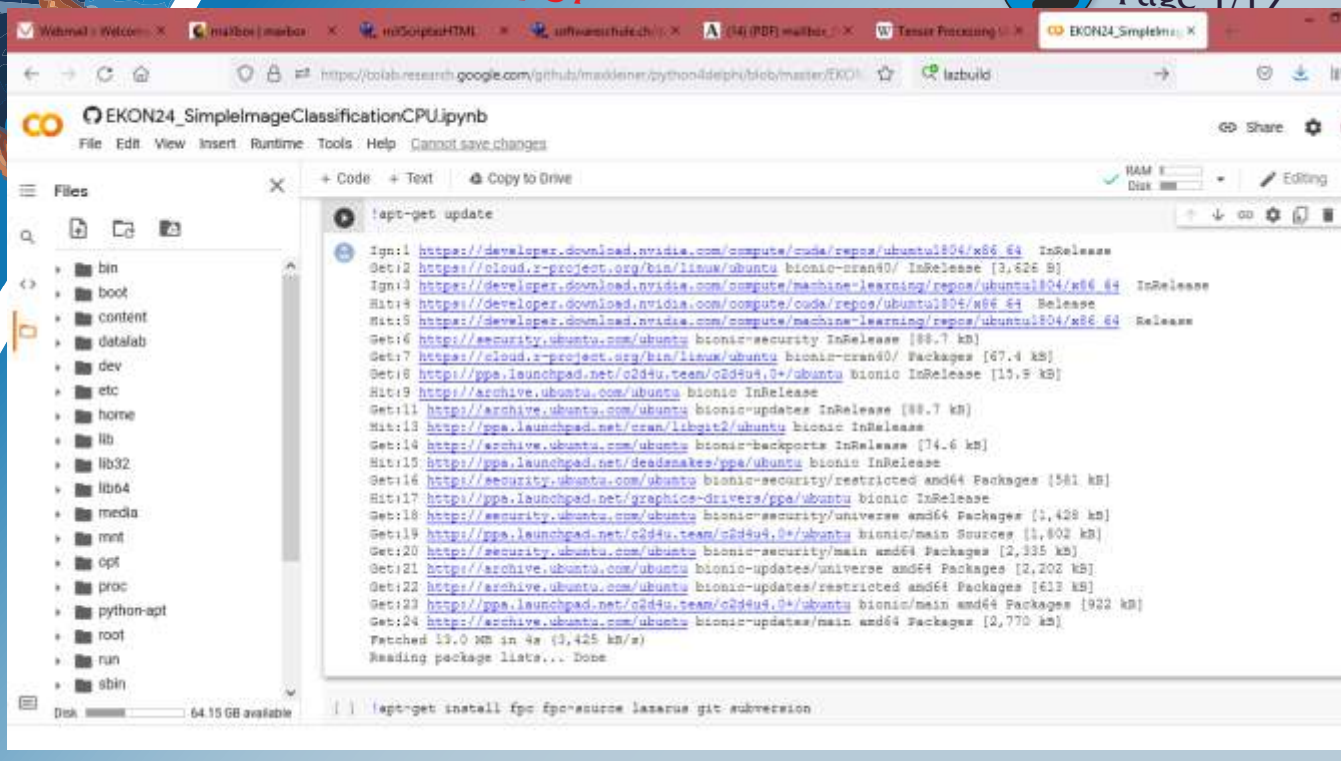
Direct Link to start:
https://colab.research.google.com/github/maxkleiner/python4delphi/blob/master/EKON24_SimpleImageClassificationCPU_2021.ipynb

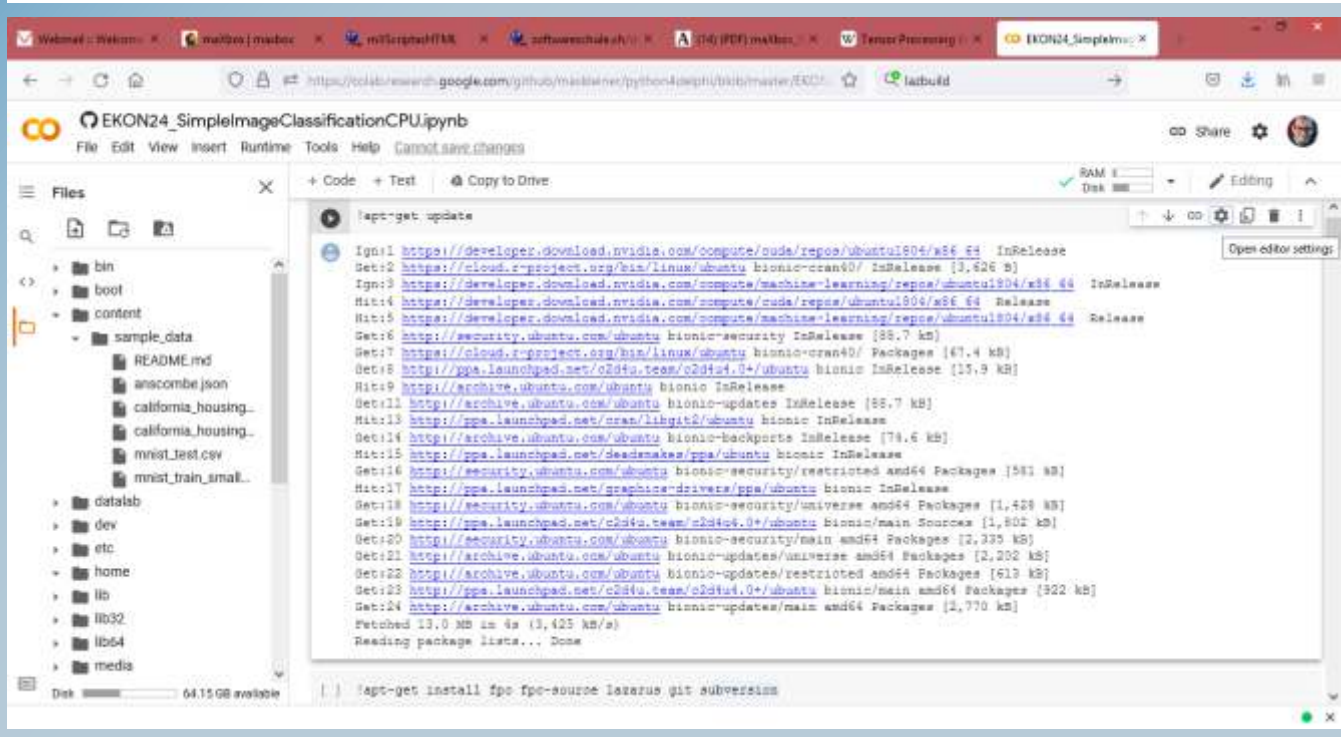
```
------------------------------------------------
/bin/bash: lazbuild: command not found
------------------------------------------------
```

```
function GetNumberOfProcessors: longint;
var
SystemInfo: TSystemInfo;
begin
GetSystemInfo(SystemInfo);
Result:= SystemInfo.dwNumberOfProcessors;
end;
```

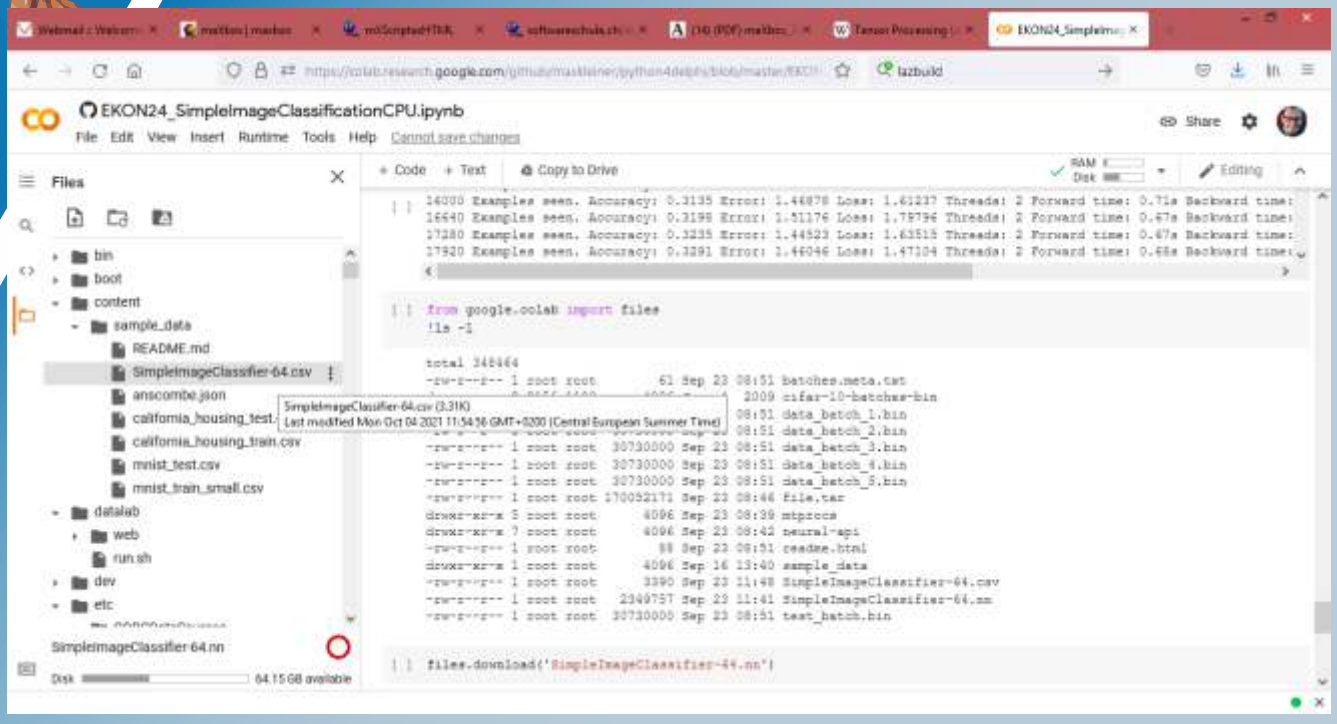
In Delphi, you print via the TPrinter object.

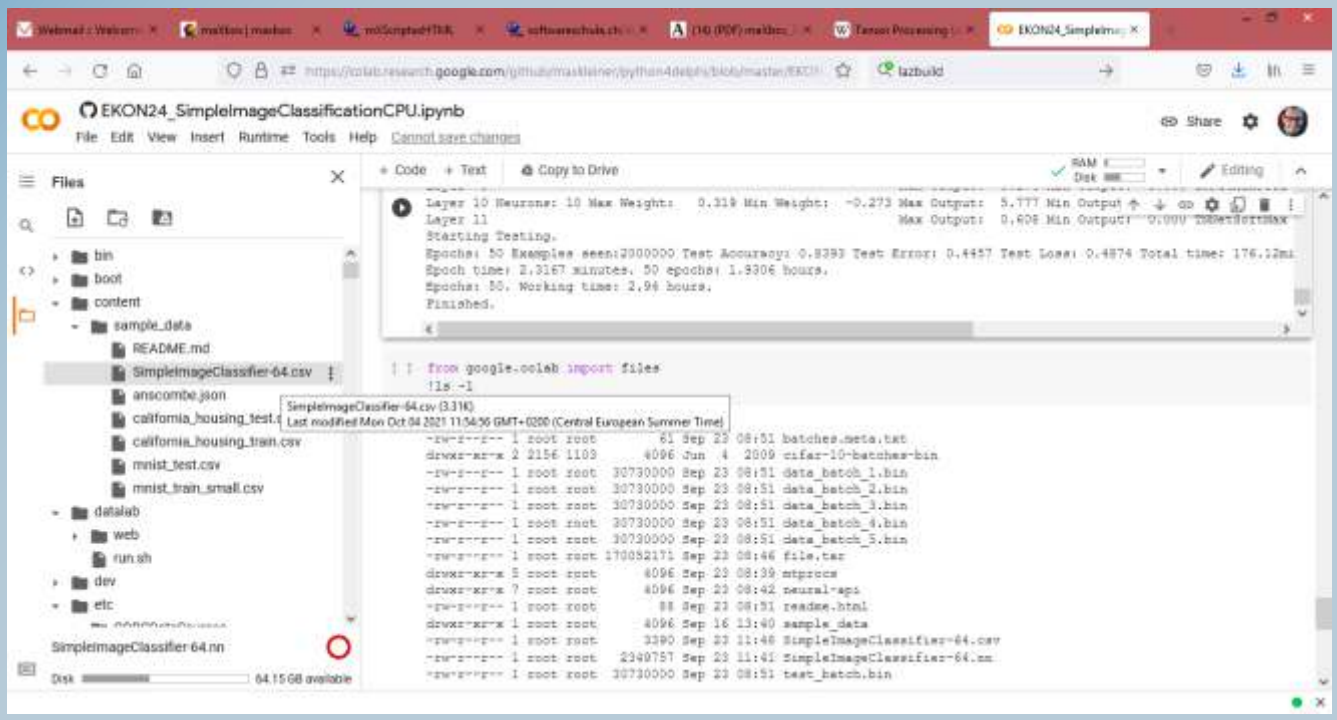
* Add printers to your uses clause
* Use the Printer function to access the global instance of Tprinter
* Printer.BeginDoc starts the print job
* Printer.EndDoc stops the print job and sends it to the printer
* Printer.NewPage forces a new page
* Printer.Canvas is used to generate the output page

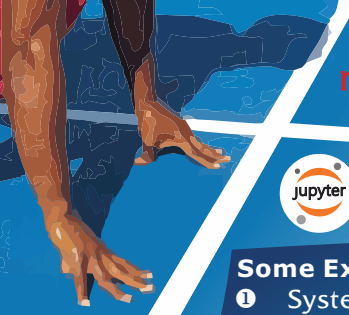

EKON24_SimpleImageClassificationCPU.ipynb
File Edit View Insert Runtime Tools Help Cannot save changes
Share
Files
+ Code + Text Copy to Drive
RAM Disk
Editing
!apt-get update
Ign:1 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64 InRelease
Get:2 https://cloud.r-project.org/bin/linux/ubuntu bionic-cran40/ InRelease [3,626 B]
Ign:3 https://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1804/x86_64 InRelease
Hit:4 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64 Release
Hit:5 https://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1804/x86_64 Release
Get:6 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:7 https://cloud.r-project.org/bin/linux/ubuntu bionic-cran40/ Packages [67.4 kB]
Get:8 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu bionic InRelease [15.9 kB]
Hit:9 http://archive.ubuntu.com/ubuntu bionic InRelease
Get:11 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Hit:13 http://ppa.launchpad.net/cran/libgit2/ubuntu bionic InRelease
Get:14 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Hit:15 http://ppa.launchpad.net/deadsnakes/ppa/ubuntu bionic InRelease
Get:16 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64 Packages [581 kB]
Hit:17 http://ppa.launchpad.net/graphics-drivers/ppa/ubuntu bionic InRelease
Get:18 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [1,428 kB]
Get:19 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu bionic/main Sources [1,802 kB]
Get:20 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [2,335 kB]
Get:21 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [2,202 kB]
Get:22 http://archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages [613 kB]
Get:23 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu bionic/main amd64 Packages [922 kB]
Get:24 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [2,770 kB]
Fetched 13.0 MB in 4s (3,425 kB/s)
Reading package lists... Done
!apt-get install fpc fpc-source lazarus git subversion
bin
boot
content
datalab
dev
etc
home
lib
lib32
lib64
media
mnt
opt
proc
python-apt
root
run
sbin
Disk 64.15 GB available


EKON24_SimpleImageClassificationCPU.ipynb
File Edit View Insert Runtime Tools Help Cannot save changes
Share
Files
+ Code + Text Copy to Drive
RAM Disk
Editing
!apt-get update
Open editor settings
Ign:1 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64 InRelease
Get:2 https://cloud.r-project.org/bin/linux/ubuntu bionic-cran40/ InRelease [3,626 B]
Ign:3 https://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1804/x86_64 InRelease
Hit:4 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64 Release
Hit:5 https://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1804/x86_64 Release
Get:6 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:7 https://cloud.r-project.org/bin/linux/ubuntu bionic-cran40/ Packages [67.4 kB]
Get:8 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu bionic InRelease [15.9 kB]
Hit:9 http://archive.ubuntu.com/ubuntu bionic InRelease
Get:11 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Hit:13 http://ppa.launchpad.net/cran/libgit2/ubuntu bionic InRelease
Get:14 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Hit:15 http://ppa.launchpad.net/deadsnakes/ppa/ubuntu bionic InRelease
Get:16 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64 Packages [581 kB]
Hit:17 http://ppa.launchpad.net/graphics-drivers/ppa/ubuntu bionic InRelease
Get:18 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [1,428 kB]
Get:19 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu bionic/main Sources [1,802 kB]
Get:20 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [2,335 kB]
Get:21 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [2,202 kB]
Get:22 http://archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages [613 kB]
Get:23 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu bionic/main amd64 Packages [922 kB]
Get:24 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [2,770 kB]
Fetched 13.0 MB in 4s (3,425 kB/s)
Reading package lists... Done
!apt-get install fpc fpc-source lazarus git subversion
bin
boot
content
sample_data
README.md
anscombe.json
california_housing...
california_housing...
mnist_test.csv
mnist_train_small...
datalab
dev
etc
home
lib
lib32
lib64
media
Disk 64.15 GB available

jupyter colab







jupyter colab

Some Extra Features

❶ System Aliases
Jupyter includes shortcuts for common operations, such as ls and others.

❷ Tab-Completion and Exploring Code
Colab provides tab completion to explore attributes of Python objects, as well as to quickly view documentation strings.

❸ Exception Formatting
Exceptions are formatted nicely in Colab outputs

❹ Rich, Interactive Outputs
Until now all of the generated outputs have been text, but they can be more interesting.

❺ Integration with Drive
Colaboratory is integrated with Google Drive. It allows you to share, comment, and collaborate on the same document with multiple people.

Differences between Google Colab and Jupyter notebooks

❶ **Infrastructure**
Google Colab runs on Google Cloud Platform (GCP). Hence it's robust, flexible

❷ **Hardware**
Google Colab recently added support for Tensor Processing Unit (TPU) apart from its existing GPU and CPU instances. So, it's a big deal for all deep learning people.

❸ **Pricing**
Despite being so good at hardware, the services provided by Google Colab are completely free. This makes it even more awesome.

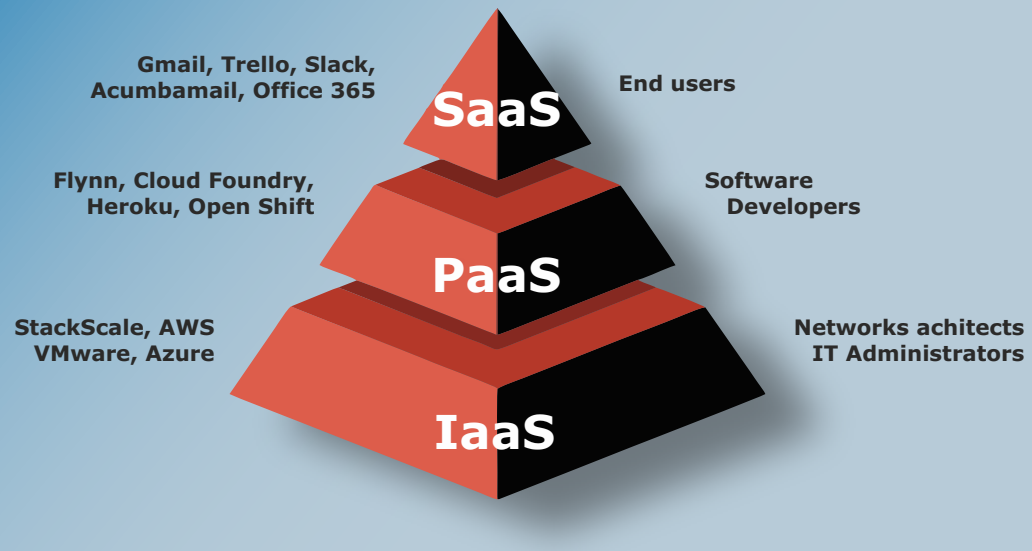
❹ **Integration with Google Drive**
Yes, this seems interesting as you can use your google drive as an interactive file system with Google Colab. This makes it easy to deal with larger files while computing your stuff.

❺ **Boon for Research and Startup Community**
Perhaps this is the only tool available in the market which provides such a good PaaS for free to users. This is overwhelmingly helpful for startups, the research ommunity and students in deep learning space

jupyter colab

Platform as a service (PaaS) is a cloud computing model where a third-party provider delivers hardware and software tools to users over the internet. Usually, these tools are needed for application development. A PaaS provider hosts the hardware and software on its own infrastructure.

Cloud service Models



jupyter colab

Now I want to show step by step (16 steps) how we can organize this Lazarus-Project and build and train an image classifier and detector. You can open the link in Colab and start with:

❶ `!apt-get update`

Update, as mentioned above, will fetch available software and update the lists while upgrade will install new versions of software installed on your computer or in our case on the colab cloud (actual software updates).
Then it goes like this:

```
0% [Working]
Ign:1
https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64
InRelease 0% [Connecting to archive.ubuntu.com]
[Connecting to security.ubuntu.com] [Wait
Get:2 https://cloud.r-project.org/bin/linux/ubuntu bionic-cran40/
InRelease [3,626 B]
```

These steps are done for you with a Jupyter notebook. A Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and describing text. So the second more inter-esting command will be:

❷ `!apt-get install fpc fpc-source lazarus git subversion`

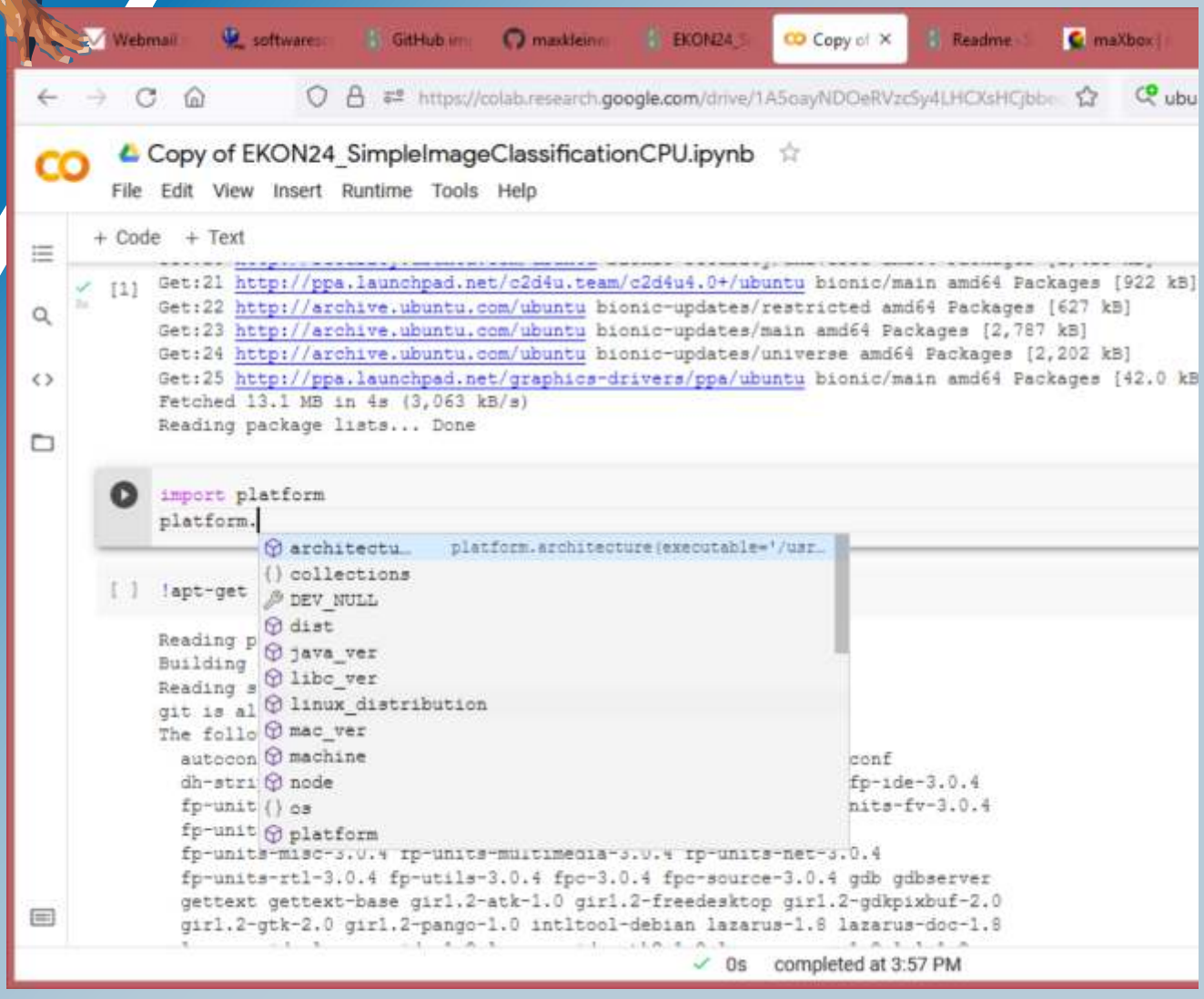
Reading package lists... Done Building dependency tree Reading state information... Done git is already the newest version (1:2.17.1-1ubuntu0.9).
The following additional packages will be installed:
autoconf automake autopoint autotools-dev debhelper dh-autoreconf dh-strip-nondeterminism file fp-compiler-3.0.4 fp-docs-3.0.4 fp-ide-3.0.4 fp-units-base-3.0.4 fp-units-db-3.0.4 fp-units-fcl-3.0.4 fp-units-fv-3.0.4 fp-units-gfx-3.0.4 fp-units-gtk2-3.0.4 fp-units-math-3.0.4 ...

So the last entries of install fpc will be:
Processing triggers for man-db (2.8.3-2ubuntu0.1) ... Processing triggers for mime-support (3.60ubuntu1) ...
Processing triggers for libvlc-bin:amd64 (3.0.8-0ubuntu18.04.1) ...

Thanks to FPC and git subversion we now can install Lazarus on a Ubuntu Bionic image machine. Ubuntu is distributed on three types of images, but we let colab to choose from. You can check your actual platform with a live python notebook script:

```
import platform
platform.platform()
>>> Linux-5.4.104+-x86_64-with-Ubuntu-18.04-bionic
```

Our next task should get the API for the neural network working on the bionic platform!




```
❸ !git clone https://github.com/joaopauloschuler/neural-api.git
Cloning into 'neural-api'...
remote: Enumerating objects: 2372, done.
remote: Counting objects: 100% (494/494), done.
remote: Compressing objects: 100% (370/370), done.
remote: Total 2372 (delta 340), reused 236 (delta 124), pack-reused 1878
Receiving objects: 100% (2372/2372), 4.58 MiB | 10.38 MiB/s, done.
Resolving deltas: 100% (1585/1585), done.
```

jupyter colab

The git clone is a git command, which creates a clone/copy of an existing repository into a new directory.
It is also used to create remote-tracking branches for each branch in the cloned repository.
It is the most common command which allows users to obtain a development copy of an existing central repository.
Good to know after the clone, a plain `git fetch` without arguments will update all the `remote-tracking branches`, and a git pull without arguments will in addition merge the remote master branch into the current master branch.
The neural-api or **CAI API (Conscious Artificial Intelligence)** is something like TensorFlow for Pascal and is a platform-independent open source library for artificial intelligence or machine learning in the field of speech recognition, image classification, OpenCL, data science and computer vision.
`https://sourceforge.net/projects/cai/files/`

It could be that you see some **Pascal dialect** but the other dialects of **Object Pascal** have always aligned themselves closely with the Delphi dialect of the language.
Free Pascal/Lazarus, Oxygene, Smart Pascal, maXbox, DWScript, PdScript, PascalABC, etc… all do them. So while there isn't an official standard of Object Pascal, the dialects stay close to each other.

Then we use checkout and Lazbuild to prepare more of the project, above all we compile a package MultiThreadProcsLaz 1.2.1 with in then end with 1215 lines compiled, 0.1 sec and 5 hints issued:

```
❹ !svn checkout https://svn.code.sf.net/p/
      lazarus-ccr/svn/components/multithreadprocs mtprocs
❺ !lazbuild mtprocs/multithreadprocslaz.lpk
❻ !ls -l neural-api/examples/SimpleImageClassifier/SimpleImageClassifier.lpi
```

Point 6 shows the project we use:

```
-rw-r--r-- 1 root root 5694 Sep 23 08:37 neural-api/
      examples/SimpleImageClassifier/SimpleImageClassifier.lpi
```

jupyter colab

Now we build the project:

```
❼ !lazbuild neural-api/examples/SimpleImageClassifier/SimpleImageClassifier.lpi
```

On that platform is the `Free Pascal Compiler version 3.0.4+dfsg-18ubuntu2 [2018/08/29] for x86_64` running. As you maybe know **lazbuild** is a command-line tool that **builds Lazarus projects and packages.** It checks also recursively all dependencies and compiles needed packages first. It uses the **Free Pascal compiler (FPC)** to compile.

OPTIONS

```
-h,--help Displays a short help message.
-B,--build-all build all files of project/package.
```

We check that **lpi-build** with:

```
❽. ls -l neural-api/bin/x86_64-linux/bin/SimpleImageClassifier
  -rwxr-xr-x 1 root root 1951024 Sep 23 08:43 neural-api/bin/x86_64-
  linux/bin/SimpleImageClassifier*
```

We can see we have execution rights rwx on the project-code in our scripts.
Next step ❾ is to get the image based data to train and test with it and step 10 checks that download:

```
import os
import urllib.request

if not os.path.isfile('cifar-10-batches-bin/data_batch_1.bin'):
  print("Downloading CIFAR-10 Files")
  url = 'https://www.cs.toronto.edu/~kriz/cifar-10-binary.tar.gz'
  urllib.request.urlretrieve(url, './file.tar')

>>> Downloading CIFAR-10 Files
```

```
❿. ls -l

total 166080
-rw-r--r-- 1 root root 170052171 Sep 23 08:46 file.tar
drwxr-xr-x 5 root root      4096 Sep 23 08:39 mtprocs/
drwxr-xr-x 7 root root      4096 Sep 23 08:42 neural-api/
drwxr-xr-x 1 root root      4096 Sep 16 13:40 sample_data/
```

It's the `file.tar` we downloaded.
We made a script that executes the whole build script in maXbox imported from a jupyter notebook. This version 4.7.5.80 from July 2021 allows us with the help of Python4Delphi and an environment with modules in site-packages to execute Py-functions.
But the most is only available in a 32-bit space as maXbox4 is still 32-bit, possible also with 64-bit Python means calling external shell(ExecuteShell) and **Python4Lazarus.**
We have to unpack those files:

jupyter colab

```
11. !tar -xvf ./file.tar
and we get:
cifar-10-batches-bin/
cifar-10-batches-bin/data_batch_1.bin
cifar-10-batches-bin/batches.meta.txt
cifar-10-batches-bin/data_batch_3.bin
cifar-10-batches-bin/data_batch_4.bin
cifar-10-batches-bin/test_batch.bin
cifar-10-batches-bin/readme.html
cifar-10-batches-bin/data_batch_5.bin
cifar-10-batches-bin/data_batch_2.bin
```

```
12. Copying files to current folder

if not os.path.isfile('./data_batch_1.bin'):
  print("Copying files to current folder")
  !cp ./cifar-10-batches-bin/* ./
```

In 12, we copy the image files to prepare for running the project of image classification training step 13

```
if os.path.isfile('./data_batch_1.bin'):
  print("RUNNING!")
  !neural-api/bin/x86_64-linux/bin/SimpleImageClassifier
```

Hurray, analyze, build, compile and deploy
of the 12 layers neural network with 331 neurons
(abcd) is running now for about 3 hours!

```
RUNNING!
Creating Neural Network...
 Layers: 12
 Neurons:331
 Weights:162498 Sum:  -19.536575
Learning rate:0.001000 L2 decay:0.000010 Batch size:64 Step size:64
File name is: SimpleImageClassifier-64
Training images: 40000 - Validation images: 10000 - Test images: 10000
Computing...
```

Imagine the accuracy goes up and the loss-function (error-rate) goes down.
The loss function is the bread and butter of modern machine learning;
it takes your algorithm from theoretical to practical and transforms neural networks
from glorified matrix multiplication into deep learning.

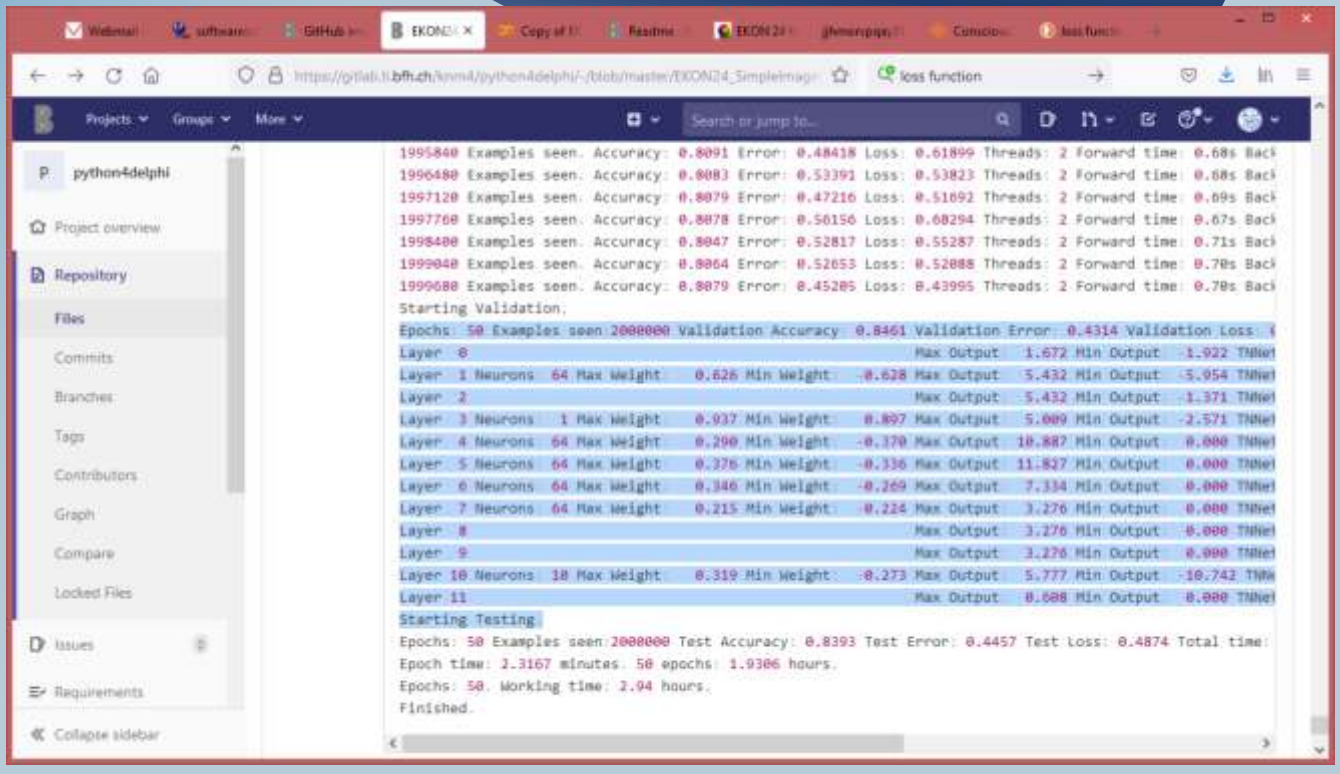
https://algorithmia.com/blog/introduction-to-loss-functions

jupyter colab

Loss functions are used in regression when finding a line of best fit by minimizing the overall loss of all the points with the prediction from the line. Loss functions are used while training perceptrons and neural networks by influencing how their weights are updated (*our result will be such a file with the trained weights!*). The larger the loss, the larger the update.

NOTE:
Loss functions are different based on a problem statement to which deep learning is being applied. The cost function is another term used interchangeably for the loss function, but it holds a more different meaning.
A loss function is for a single training example, while a cost function is an average loss over the complete train dataset (./data_batch_1.bin).

After a certain amount of working time we get:
Starting Testing.
Epochs: 50 Examples seen:2000000 Test Accuracy: 0.8393 Test Error: 0.4457
Test Loss: 0.4874 Total time: 176.12min
Epoch time: 2.3167 minutes. 50 epochs: 1.9306 hours.
Epochs: 50. Working time: 2.94 hours.
Finished.



jupyter colab

Now we
want to export the trained
result, means we get 2 files:

14. from google.colab import files !ls -l

```
-rw-r--r-- 1 root root      3390 Sep 23 11:48 SimpleImageClassifier-64.csv
-rw-r--r-- 1 root root   2349757 Sep 23 11:41 SimpleImageClassifier-64.nn
```

15. files.download('SimpleImageClassifier-64.nn')

16. files.download('SimpleImageClassifier-64.csv')

Note: Probably you get an `FileNotFoundError:`
`Cannot find file: SimpleImageClassifier.csv`
because colab increments files with a postfix in the file name:
`SimpleImageClassifier-65.csv`

In that casse you have to adjust the download command 15 and 16:

16. files.download('SimpleImageClassifier-65.csv')

So what's the context of the 2 files.
The csv is just the log of the training with the hyperparameters such as learning rate:

```
epoch  training accuracy  training loss      training error  validation accuracy
       validation loss    validation error   learning rate   time test accuracy
       test loss          test error
```

The `*.nn file` serves as a pretrained file (FAvgWeight)
to classify or predict images we trained on.
Also the CIFAR-10 classification examples with
`experiments/testcnnalgo/testcnnalgo.lpr` and a number of
CIFAR-10 classification examples are available on `/experiments`.

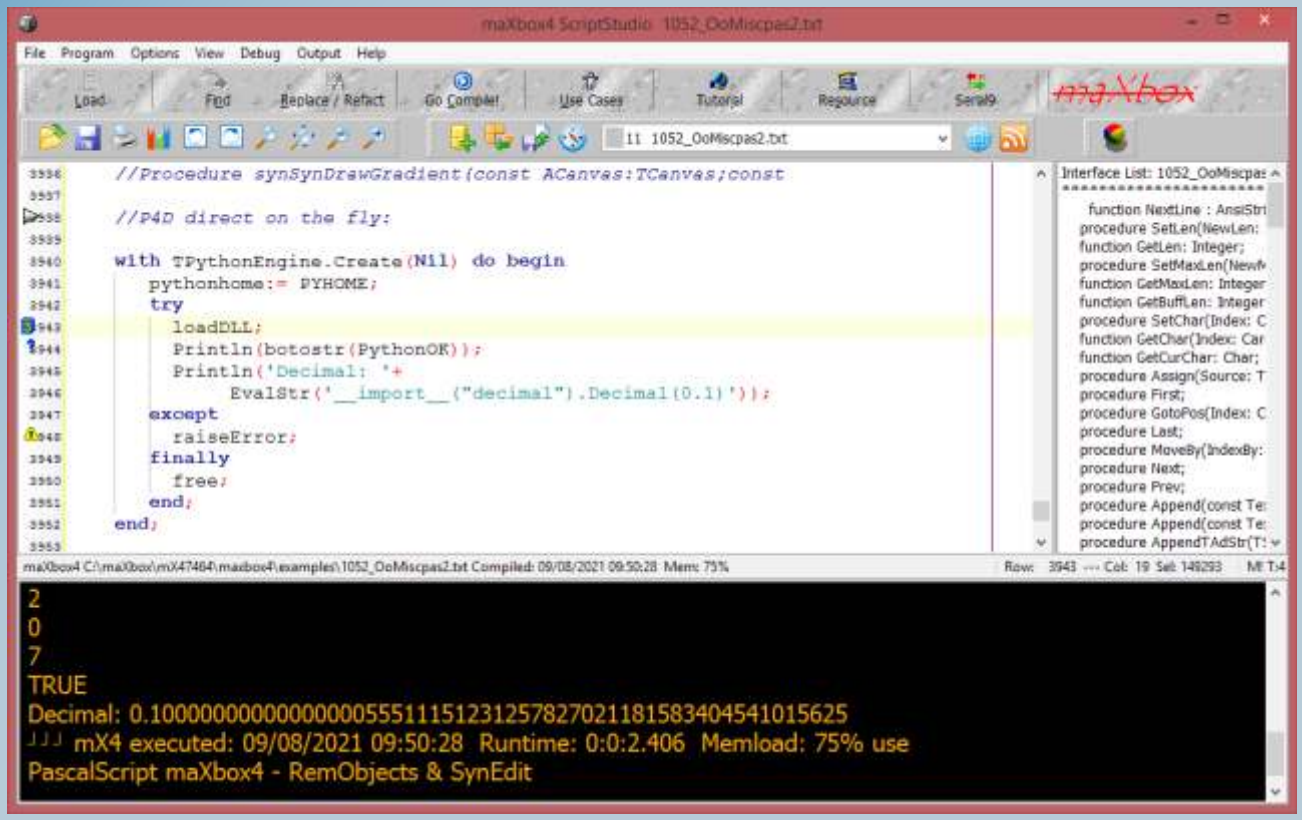
Git is a distributed version control system, which means you can work locally,
then share or "push" your changes to a server. In our case, the server is GitLab.
This fascinating option is to run the entire system as runtime virtualization
with the help of a **Jupyter notebook** running on **Ubuntu** in the cloud on
Colab or an **Colab.research** container.

We step a last step to exec a script in a script!
If we call a file or a Python command then we use
`ExecString(PYCMD): http://www.softwareschule.ch/examples/pydemo19.txt`

jupyter colab

At last a minimal configuration called **"Pyonfly"** with a colab platform tester. The minimal configuration depends on your Python-installation and the `UseLastKnownVersion` property in `TDynamicDll` and if something went wrong you got a raiseError Py exception:

```
with TPythonEngine.Create(Nil) do begin
 pythonhome:= PYHOME;
 try
  loadDLL;
  Println('Colab Platform: '+
    EvalStr('__import__("platform").platform()'));
 except
   raiseError;
 finally
  free;
 end;
end;
```



jupyter colab

EKON CAI, P4D and Colab topics

https://entwickler-konferenz.de/delphi-innovations-fundamentals/python4delphi/
https://colab.research.google.com/
https://entwickler-konferenz.de/blog/machine-learning-mit-cai/

Learn about Jupyter.org

https://jupyter.org/
https://forum.lazarus.freepascal.org/index.php?topic=38955.0

CONCLUSION SCRIPT:
Note: You will need a google account to run a predefined jupyter notebook on Colab;
the exported script of the classification:

```
# -*- coding: utf-8 -*-
"""Copy EKON_SimpleImageClassificationCPU.ipynb
Automatically generated by Colaboratory.
Original file is located at     https://colab.research.google.com/drive/1clvG2uoMGo-
_bfrJnxBJmpNTxjvnsMx9
"""

!apt-get update
!apt-get install fpc fpc-source lazarus git subversion
!git clone https://github.com/joaopauloschuler/neural-api.git

!svn checkout https://svn.code.sf.net/p/lazarus-ccr/svn/components/multithreadprocs mtprocs

!lazbuild mtprocs/multithreadprocslaz.lpk

!ls -l neural-api/examples/SimpleImageClassifier/SimpleImageClassifier.lpi

!lazbuild neural-api/examples/SimpleImageClassifier/SimpleImageClassifier.lpi

ls -l neural-api/bin/x86_64-linux/bin/SimpleImageClassifier

import os
import urllib.request

if not os.path.isfile('cifar-10-batches-bin/data_batch_1.bin'):
  print("Downloading CIFAR-10 Files")
  url = 'https://www.cs.toronto.edu/~kriz/cifar-10-binary.tar.gz'
  urllib.request.urlretrieve(url, './file.tar')

ls -l
!tar -xvf ./file.tar

if not os.path.isfile('./data_batch_1.bin'):
  print("Copying files to current folder")
  !cp ./cifar-10-batches-bin/* ./

if os.path.isfile('./data_batch_1.bin'):
  print("RUNNING!")
  !neural-api/bin/x86_64-linux/bin/SimpleImageClassifier

from google.colab import files
!ls -l

files.download('SimpleImageClassifier-66.nn')
files.download('SimpleImageClassifier-66.csv')
```

jupyter colab

References:

Docs: https://maxbox4.wordpress.com/blog/
http://www.softwareschule.ch/download/maxbox_starter86_3.pdf

Image Classification with Lazarus

https://colab.research.google.com/github/maxkleiner/maXbox/blob/master/EKON24_SimpleImageClassificationCPU.ipynb

16 SEP Blog

MACHINE LEARNING MIT CAI

This report visualizes the field of object recognition using computer vision techniques from machine learning. An image classifier from the CAI framework in Lazarus and Delphi, the so-called CIFAR-10 image classifier, is also used.

MEHR

18 SEP Blog, Interview

"DELPHI DEVELOPMENT IS STILL GOING STRONG"

Marco Cantu talkes about the current status of Delphi, how it has evolved, and what's in store for this language in the future.

MEHR

jupyter colab

SPEAKERS OF EKON 25

Stefan Glienke
Aagon

Dr. Cary Jensen
Jensen Data Systems, Inc.

Dr. Don Wibier
DevExpress

Lisa Moritz
INNOQ

Ray Konopka
Raize Software

Max Kleiner
kleiner kommunikation

Nigel Tavendale
All Things Syslog

Markus Humm
ebm-papst Group

Bruno Fierens
tmssoftware.com bvba

Arnaud Bouchez
Developer at TranquilIT - Founder of the Open Source mORMot Framework

Andrea Magni
Freelance

Jens Fudge
Archersoft Aps

Marco Cantu
Embarcadero

Serge Pilko
Softacom Ltd.

Bernd Ua
probucon Business Consulting GmbH&Co KG

Matthias Eißing
Embarcadero

Christoph Schneider
Schneider Infosystems AG

Dr. Falk W. Müller
diconium digital solutions

Dr. Annegret Junker
Allianz Deutschland AG