

# Image Detection with Lazarus

maXbox Starter87 – Build with CAI

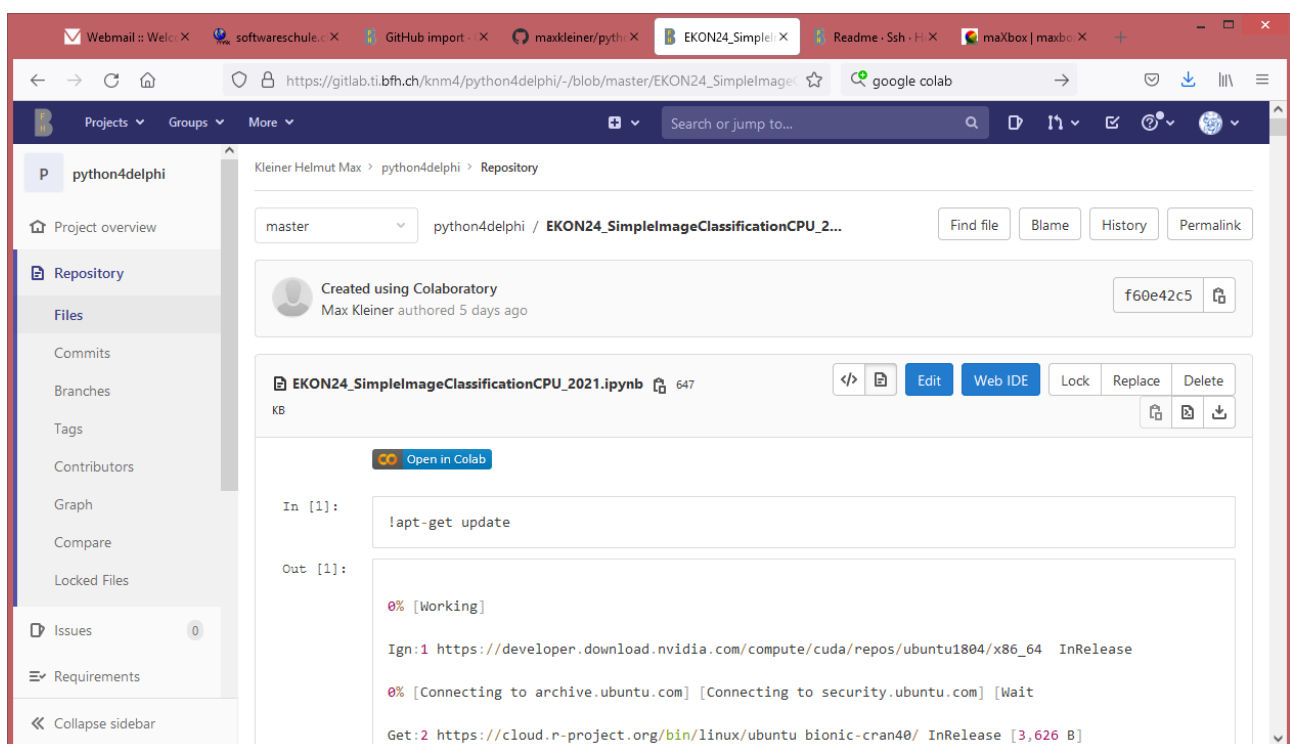
Try finally begin. – Max

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](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 use also this service. The Bern University of Applied Sciences (BFH<sup>1</sup>) 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.



1 <https://www.bfh.ch/en/>

\_PIC: 87\_gitlab\_probh.png

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:

## 1. *!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 interesting command will be:

## 2. *!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 ...
```

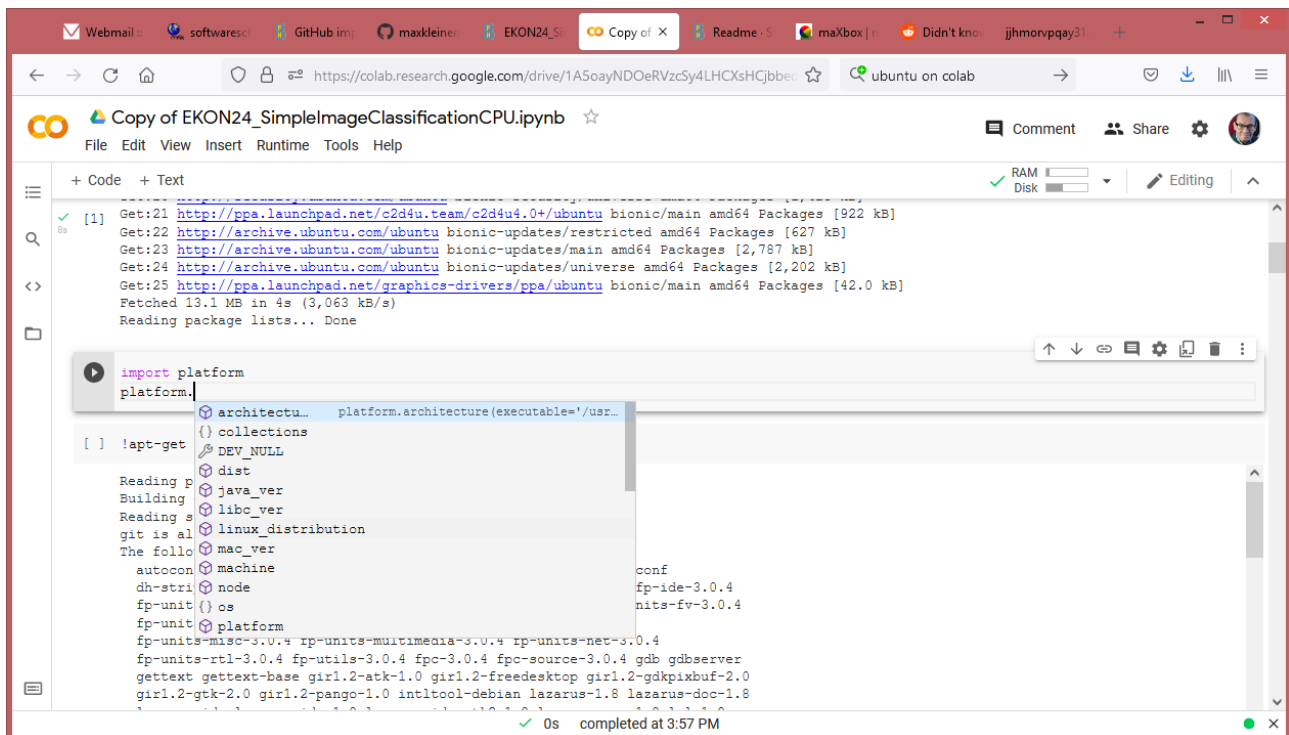
So the last entries of install fpc will be:

```
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!



\_PIC: 87\_colab\_python.png

### 3. `!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.
```

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<sup>2</sup>.

<https://sourceforge.net/projects/cai/files/>

<sup>2</sup> Described in Blaise - Machine Learning with CAI V.87/88

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:

```
4. !svn checkout https://svn.code.sf.net/p/lazarus-
ccr/svn/components/multithreadprocs mtprocs
5. !lazbuild mtprocs/multithreadproclaz.lpk
6. !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
```

Now we build the project:

```
7. !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:

```
8. 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 9 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
```

## 10. `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 also 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:

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>

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:

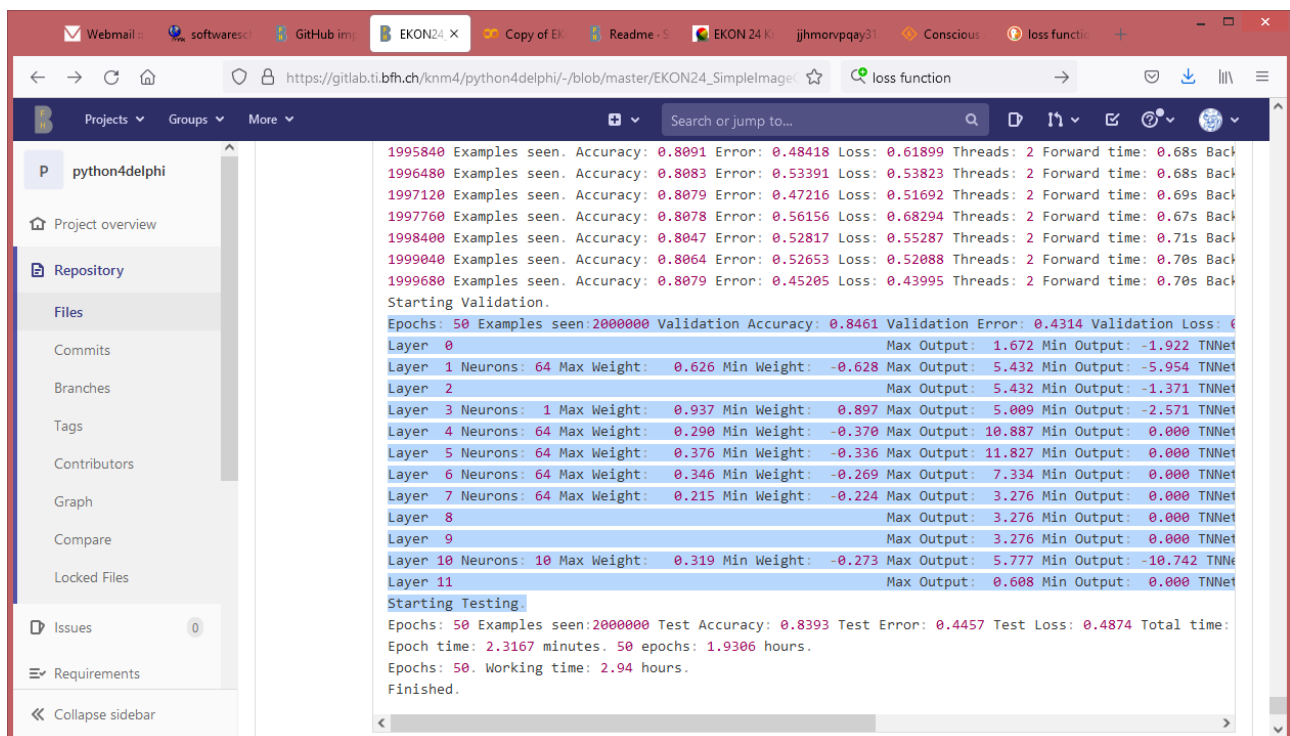
Starting Testing.

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.



```
1995840 Examples seen. Accuracy: 0.8091 Error: 0.48418 Loss: 0.61899 Threads: 2 Forward time: 0.68s Back
1996480 Examples seen. Accuracy: 0.8083 Error: 0.53391 Loss: 0.53823 Threads: 2 Forward time: 0.68s Back
1997120 Examples seen. Accuracy: 0.8079 Error: 0.47216 Loss: 0.51692 Threads: 2 Forward time: 0.69s Back
1997760 Examples seen. Accuracy: 0.8078 Error: 0.56156 Loss: 0.68294 Threads: 2 Forward time: 0.67s Back
1998400 Examples seen. Accuracy: 0.8047 Error: 0.52817 Loss: 0.55287 Threads: 2 Forward time: 0.71s Back
1999040 Examples seen. Accuracy: 0.8064 Error: 0.52653 Loss: 0.52088 Threads: 2 Forward time: 0.70s Back
1999680 Examples seen. Accuracy: 0.8079 Error: 0.45205 Loss: 0.43995 Threads: 2 Forward time: 0.70s Back
Starting Validation.
Epochs: 50 Examples seen:2000000 Validation Accuracy: 0.8461 Validation Error: 0.4314 Validation Loss: 0.4874
Layer 0 Max Output: 1.672 Min Output: -1.922 TNNet
Layer 1 Neurons: 64 Max Weight: 0.626 Min Weight: -0.628 Max Output: 5.432 Min Output: -5.954 TNNet
Layer 2 Max Output: 5.432 Min Output: -1.371 TNNet
Layer 3 Neurons: 1 Max Weight: 0.937 Min Weight: 0.897 Max Output: 5.009 Min Output: -2.571 TNNet
Layer 4 Neurons: 64 Max Weight: 0.290 Min Weight: -0.370 Max Output: 10.887 Min Output: 0.000 TNNet
Layer 5 Neurons: 64 Max Weight: 0.376 Min Weight: -0.336 Max Output: 11.827 Min Output: 0.000 TNNet
Layer 6 Neurons: 64 Max Weight: 0.346 Min Weight: -0.269 Max Output: 7.334 Min Output: 0.000 TNNet
Layer 7 Neurons: 64 Max Weight: 0.215 Min Weight: -0.224 Max Output: 3.276 Min Output: 0.000 TNNet
Layer 8 Max Output: 3.276 Min Output: 0.000 TNNet
Layer 9 Max Output: 3.276 Min Output: 0.000 TNNet
Layer 10 Neurons: 10 Max Weight: 0.319 Min Weight: -0.273 Max Output: 5.777 Min Output: -10.742 TNNet
Layer 11 Max Output: 0.608 Min Output: 0.000 TNNet
Starting Testing
Epochs: 50 Examples seen:2000000 Test Accuracy: 0.8393 Test Error: 0.4457 Test Loss: 0.4874 Total time:
Epoch time: 2.3167 minutes. 50 epochs: 1.9306 hours.
Epochs: 50. Working time: 2.94 hours.
Finished.
```

\_PIC: 87\_colab\_trainresults.png

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` cause colab increments files with a postfix in the file name: `SimpleImageClassifier-65.csv`

Then 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>

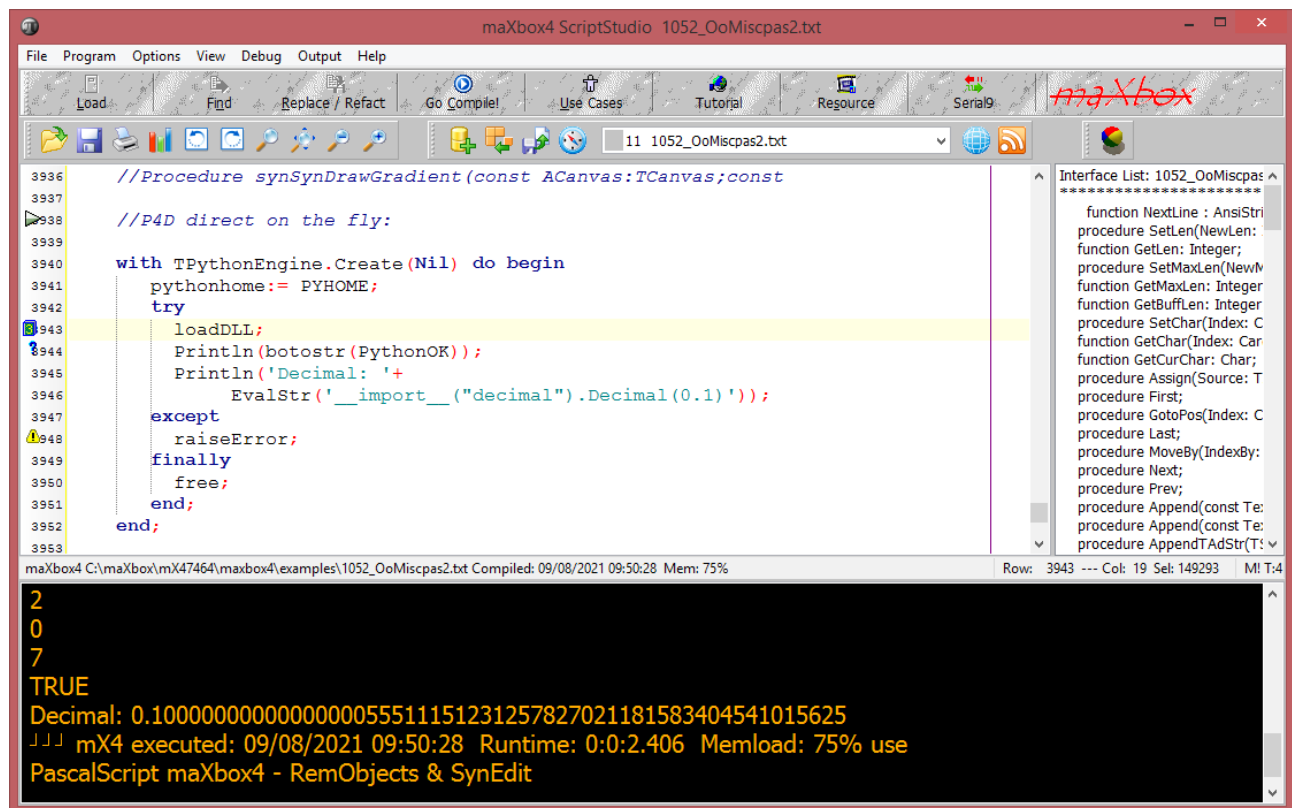
At last a minimal configuration called "Pyonfly" with a colab platform tester. The minimal configuration depends on your Python-installation and the *UseLastKnownVersion* property in *TDynamicDll* and if something went wrong you got a *raiseError* Py exception:

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

```

finally
    free;
end;
end;

```



\_PIC: 1052\_pyonthefly.png

## 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/multithreadproclaz.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')
```

## References:

Docs: [https://maxbox4.wordpress.com/blog/http://www.softwareschule.ch/download/maxbox\\_starter86\\_3.pdf](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](https://colab.research.google.com/github/maxkleiner/maXbox/blob/master/EKON24_SimpleImageClassificationCPU.ipynb)

## Appendix Colab Log Details:

!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

```
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:
```

```
Setting up dh-strip-nondeterminism (0.040-1.1~build1) ...
Processing triggers for libc-bin (2.27-3ubuntu1.3) ...
/sbin/ldconfig.real: /usr/local/lib/python3.7/dist-
```

```
packages/ideep4py/lib/libmkldnn.so.0 is not a symbolic link
.....
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) ...
```

### 3. `!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.
```

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

#### mtprocs/examples

```
A  mtprocs/examples/parallellloop1.lpr
A  mtprocs/examples/parallellloop_nested1.lpi
A  mtprocs/examples/parallellloop_nested1.lpr
A  mtprocs/examples/recursivemtp1.lpr
A  mtprocs/examples/simplemtp1.lpr
A  mtprocs/examples/parallellloop1.lpi
A  mtprocs/examples/recursivemtp1.lpi
A  mtprocs/examples/simplemtp1.lpi
A  mtprocs/examples/testmtp1.lpi
A  mtprocs/examples/testmtp1.lpr
A  mtprocs/Readme.txt
A  mtprocs/mtprocs.pas
A  mtprocs/mtpcpu.pas
A  mtprocs/multithreadproclaz.lpk
A  mtprocs/mtutils.pas
A  mtprocs/multithreadproclaz.pas
Checked out revision 8093.
```

### 5. `!lazbuild mtprocs/multithreadproclaz.lpk`

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

```
7. !lazbuild neural-api/examples/SimpleImageClassifier/SimpleImageClassifier.lpi
Hint: (lazarus) [RunTool] /usr/bin/fpc "-iWTOTP"
Hint: (lazarus) [RunTool] /usr/bin/fpc "-va" "compilertest.pas"
TProject.DoLoadStateFile Statefile not found: /content/neural-api/bin/x86_64-
linux/units/SimpleImageClassifier.compiled
Hint: (lazarus) [RunTool] /usr/bin/fpc "-iWTOTP" "-Px86_64" "-Tlinux"
Hint: (lazarus) [RunTool] /usr/bin/fpc "-va" "-Px86_64" "-Tlinux"
"compilertest.pas"
```

```
Hint: (11031) End of reading config file /etc/fpc.cfg
Free Pascal Compiler version 3.0.4+dfsg-1ubuntu2 [2018/08/29] for x86_64
Copyright (c) 1993-2017 by Florian Klaempfl and others
(1002) Target OS: Linux for x86-64
```

```
(3104) Compiling SimpleImageClassifier.lpr
(3104) Compiling /content/neural-api/neural/neuralnetwork.pas
(3104) Compiling /content/neural-api/neural/neuralvolume.pas
```

8. `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*
```

9. `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

10. `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/
```

11. `!tar -xvf ./file.tar`

```
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. `copy check`

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

Copying files to current folder

13. `Start Running`

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

## RUNNING!

Creating Neural Network...

Layers: 12

Neurons:331

Weights:162498 Sum: -19.536575

## File name is: SimpleImageClassifier-64

Learning rate:0.001000 L2 decay:0.000010 Inertia:0.900000 Batch size:64 Step size:64 Staircase ephocs:10

Training images: 40000

Validation images: 10000

Test images: 10000

## File name is: SimpleImageClassifier-64

Learning rate:0.001000 L2 decay:0.000010 Inertia:0.900000 Batch size:64 Step size:64 Staircase ephocs:10

Training images: 40000

Validation images: 10000

Test images: 10000

## RUNNING!

Creating Neural Network...

Layers: 12

Neurons:331

Weights:162498 Sum: -19.536575

Layer 0 Neurons: 0 Weights: 0 TNNetInput(32,32,3,0,0) Output:32,32,3  
Learning Rate:0.0100 Inertia:0.90 Weight Sum: 0.0000 Branches:1  
Layer 1 Neurons: 64 Weights: 4800 TNNetConvolutionLinear(64,5,2,1,1)  
Output:32,32,64 Learning Rate:0.0100 Inertia:0.90 Weight Sum: 9.3083 Parent:0  
Branches:1  
Layer 2 Neurons: 0 Weights: 0 TNNetMaxPool(4,4,0,0,0) Output:8,8,64  
Learning Rate:0.0100 Inertia:0.90 Weight Sum: 0.0000 Parent:1 Branches:1  
Layer 3 Neurons: 1 Weights: 2 TNNetMovingStdNormalization(0,0,0,0,0)  
Output:8,8,64 Learning Rate:0.0100 Inertia:0.90 Weight Sum: 1.0000 Parent:2  
Branches:1  
Layer 4 Neurons: 64 Weights: 36864 TNNetConvolutionReLU(64,3,1,1,1)  
Output:8,8,64 Learning Rate:0.0100 Inertia:0.90 Weight Sum:-16.7340 Parent:3  
Branches:1  
Layer 5 Neurons: 64 Weights: 36864 TNNetConvolutionReLU(64,3,1,1,1)  
Output:8,8,64 Learning Rate:0.0100 Inertia:0.90 Weight Sum: -2.0621 Parent:4  
Branches:1  
Layer 6 Neurons: 64 Weights: 36864 TNNetConvolutionReLU(64,3,1,1,1)  
Output:8,8,64 Learning Rate:0.0100 Inertia:0.90 Weight Sum: -3.9453 Parent:5  
Branches:1  
Layer 7 Neurons: 64 Weights: 36864 TNNetConvolutionReLU(64,3,1,1,1)  
Output:8,8,64 Learning Rate:0.0100 Inertia:0.90 Weight Sum: 3.3115 Parent:6  
Branches:1  
Layer 8 Neurons: 0 Weights: 0 TNNetDropout(2,1,0,0,0) Output:8,8,64  
Learning Rate:0.0100 Inertia:0.90 Weight Sum: 0.0000 Parent:7 Branches:1  
Layer 9 Neurons: 0 Weights: 0 TNNetMaxPool(2,2,0,0,0) Output:4,4,64  
Learning Rate:0.0100 Inertia:0.90 Weight Sum: 0.0000 Parent:8 Branches:1  
Layer 10 Neurons: 10 Weights: 10240 TNNetFullConnectLinear(10,1,1,0,0)  
Output:10,1,1 Learning Rate:0.0100 Inertia:0.90 Weight Sum:-10.4149 Parent:9  
Branches:1  
Layer 11 Neurons: 0 Weights: 0 TNNetSoftMax(0,0,0,0,0) Output:10,1,1  
Learning Rate:0.0100 Inertia:0.90 Weight Sum: 0.0000 Parent:10 Branches:0  
Loading 10K images from file "data\_batch\_1.bin" ... GLOBAL MIN MAX -2.0000

```

1.9844 -2.0000 1.9844 -2.0000 1.9844 Done.
Loading 10K images from file "data_batch_2.bin" ... GLOBAL MIN MAX -2.0000
1.9844 -2.0000 1.9844 -2.0000 1.9844 Done.
Loading 10K images from file "data_batch_3.bin" ... GLOBAL MIN MAX -2.0000
1.9844 -2.0000 1.9844 -2.0000 1.9844 Done.
Loading 10K images from file "data_batch_4.bin" ... GLOBAL MIN MAX -2.0000
1.9844 -2.0000 1.9844 -2.0000 1.9844 Done.
Loading 10K images from file "data_batch_5.bin" ... GLOBAL MIN MAX -2.0000
1.9844 -2.0000 1.9844 -2.0000 1.9844 Done.
Loading 10K images from file "test_batch.bin" ... GLOBAL MIN MAX -2.0000
1.9844 -2.0000 1.9844 -2.0000 1.9844 Done.
File name is: SimpleImageClassifier-64
Learning rate:0.001000 L2 decay:0.000010 Inertia:0.900000 Batch size:64 Step
size:64 Staircase epochs:10
Training images: 40000
Validation images: 10000
Test images: 10000
Computing...
640 Examples seen. Accuracy: 0.1481 Error: 1.79544 Loss: 2.29493 Threads: 2
Forward time: 0.69s Backward time: 0.52s Step time: 2.55s
1280 Examples seen. Accuracy: 0.1479 Error: 1.77985 Loss: 2.23596 Threads: 2
Forward time: 0.67s Backward time: 0.52s Step time: 2.50s
1920 Examples seen. Accuracy: 0.1498 Error: 1.75829 Loss: 2.17276 Threads: 2
Forward time: 0.71s Backward time: 0.53s Step time: 2.50s
2560 Examples seen. Accuracy: 0.1550 Error: 1.72284 Loss: 2.10358 Threads: 2
Forward time: 0.67s Backward time: 0.53s Step time: 2.50s
3200 Examples seen. Accuracy: 0.1607 Error: 1.71363 Loss: 2.12127 Threads: 2
Forward time: 0.67s Backward time: 0.53s Step time: 2.54s
3840 Examples seen. Accuracy: 0.1660 Error: 1.72590 Loss: 2.05349 Threads: 2
Forward time: 0.69s Backward time: 0.50s Step time: 2.51s
4480 Examples seen. Accuracy: 0.1770 Error: 1.63273 Loss: 1.86535 Threads: 2
Forward time: 0.68s Backward time: 0.51s Step time: 2.49s
5120 Examples seen. Accuracy: 0.1856 Error: 1.63568 Loss: 1.95574 Threads: 2
Forward time: 0.70s Backward time: 0.52s Step time: 2.50s
5760 Examples seen. Accuracy: 0.1979 Error: 1.62821 Loss: 1.88185 Threads: 2
Forward time: 0.66s Backward time: 0.53s Step time: 2.51s
6400 Examples seen. Accuracy: 0.2075 Error: 1.66402 Loss: 2.07736 Threads: 2
Forward time: 0.68s Backward time: 0.52s Step time: 2.50s
7040 Examples seen. Accuracy: 0.2177 Error: 1.64277 Loss: 2.04907 Threads: 2
Forward time: 0.68s Backward time: 0.52s Step time: 2.51s
7680 Examples seen. Accuracy: 0.2264 Error: 1.59409 Loss: 1.79512 Threads: 2
Forward time: 0.70s Backward time: 0.53s Step time: 2.50s
8320 Examples seen. Accuracy: 0.2360 Error: 1.54550 Loss: 1.85149 Threads: 2
Forward time: 0.69s Backward time: 0.53s Step time: 2.52s
8960 Examples seen. Accuracy: 0.2456 Error: 1.56664 Loss: 1.77805 Threads: 2
Forward time: 0.70s Backward time: 0.54s Step time: 2.55s
9600 Examples seen. Accuracy: 0.2541 Error: 1.59044 Loss: 1.93252 Threads: 2
Forward time: 0.69s Backward time: 0.54s Step time: 2.51s
10240 Examples seen. Accuracy: 0.2597 Error: 1.58854 Loss: 1.81173 Threads: 2
Forward time: 0.68s Backward time: 0.54s Step time: 2.50s
10880 Examples seen. Accuracy: 0.2660 Error: 1.49855 Loss: 1.64290 Threads: 2
Forward time: 0.67s Backward time: 0.51s Step time: 2.49s
11520 Examples seen. Accuracy: 0.2734 Error: 1.47110 Loss: 1.53985 Threads: 2
Forward time: 0.68s Backward time: 0.53s Step time: 2.49s
12160 Examples seen. Accuracy: 0.2793 Error: 1.43010 Loss: 1.51361 Threads: 2
Forward time: 0.66s Backward time: 0.51s Step time: 2.49s
12800 Examples seen. Accuracy: 0.2864 Error: 1.54338 Loss: 1.81172 Threads: 2
Forward time: 0.67s Backward time: 0.52s Step time: 2.48s
13440 Examples seen. Accuracy: 0.2888 Error: 1.56118 Loss: 1.84152 Threads: 2
Forward time: 0.66s Backward time: 0.52s Step time: 2.52s
14080 Examples seen. Accuracy: 0.2940 Error: 1.45340 Loss: 1.62768 Threads: 2

```

Forward time: 0.71s Backward time: 0.52s Step time: 2.50s  
14720 Examples seen. Accuracy: 0.2992 Error: 1.53040 Loss: 1.71362 Threads: 2  
Forward time: 0.70s Backward time: 0.53s Step time: 2.47s  
15360 Examples seen. Accuracy: 0.3066 Error: 1.53963 Loss: 1.90071 Threads: 2  
Forward time: 0.68s Backward time: 0.52s Step time: 2.45s  
16000 Examples seen. Accuracy: 0.3135 Error: 1.46878 Loss: 1.61237 Threads: 2  
Forward time: 0.71s Backward time: 0.56s Step time: 2.49s  
16640 Examples seen. Accuracy: 0.3198 Error: 1.51176 Loss: 1.79796 Threads: 2  
Forward time: 0.67s Backward time: 0.50s Step time: 2.46s  
17280 Examples seen. Accuracy: 0.3235 Error: 1.44523 Loss: 1.63515 Threads: 2  
Forward time: 0.67s Backward time: 0.49s Step time: 2.46s  
17920 Examples seen. Accuracy: 0.3291 Error: 1.46046 Loss: 1.47104 Threads: 2  
Forward time: 0.68s Backward time: 0.50s Step time: 2.43s  
18560 Examples seen. Accuracy: 0.3359 Error: 1.48456 Loss: 1.75611 Threads: 2  
Forward time: 0.72s Backward time: 0.56s Step time: 2.48s  
19200 Examples seen. Accuracy: 0.3392 Error: 1.43991 Loss: 1.63127 Threads: 2  
Forward time: 0.69s Backward time: 0.55s Step time: 2.47s  
19840 Examples seen. Accuracy: 0.3407 Error: 1.46214 Loss: 1.58897 Threads: 2  
Forward time: 0.67s Backward time: 0.51s Step time: 2.50s  
20480 Examples seen. Accuracy: 0.3448 Error: 1.47581 Loss: 1.80389 Threads: 2  
Forward time: 0.66s Backward time: 0.52s Step time: 2.46s  
21120 Examples seen. Accuracy: 0.3535 Error: 1.51542 Loss: 1.71892 Threads: 2  
Forward time: 0.66s Backward time: 0.53s Step time: 2.45s  
21760 Examples seen. Accuracy: 0.3606 Error: 1.40996 Loss: 1.61076 Threads: 2  
Forward time: 0.66s Backward time: 0.51s Step time: 2.46s  
22400 Examples seen. Accuracy: 0.3626 Error: 1.57226 Loss: 2.04786 Threads: 2  
Forward time: 0.70s Backward time: 0.52s Step time: 2.46s  
23040 Examples seen. Accuracy: 0.3684 Error: 1.39514 Loss: 1.45001 Threads: 2  
Forward time: 0.66s Backward time: 0.49s Step time: 2.44s  
23680 Examples seen. Accuracy: 0.3742 Error: 1.36091 Loss: 1.61692 Threads: 2  
Forward time: 0.68s Backward time: 0.51s Step time: 2.44s  
24320 Examples seen. Accuracy: 0.3760 Error: 1.46164 Loss: 1.61883 Threads: 2  
Forward time: 0.68s Backward time: 0.53s Step time: 2.47s  
24960 Examples seen. Accuracy: 0.3802 Error: 1.38044 Loss: 1.39986 Threads: 2  
Forward time: 0.68s Backward time: 0.52s Step time: 2.50s  
25600 Examples seen. Accuracy: 0.3861 Error: 1.51735 Loss: 1.78022 Threads: 2  
Forward time: 0.68s Backward time: 0.51s Step time: 2.48s  
26240 Examples seen. Accuracy: 0.3902 Error: 1.41811 Loss: 1.61700 Threads: 2  
Forward time: 0.69s Backward time: 0.49s Step time: 2.49s  
26880 Examples seen. Accuracy: 0.3887 Error: 1.46899 Loss: 1.56543 Threads: 2  
Forward time: 0.66s Backward time: 0.53s Step time: 2.48s  
27520 Examples seen. Accuracy: 0.3936 Error: 1.31012 Loss: 1.30139 Threads: 2  
Forward time: 0.68s Backward time: 0.51s Step time: 2.46s  
28160 Examples seen. Accuracy: 0.3983 Error: 1.39935 Loss: 1.62048 Threads: 2  
Forward time: 0.70s Backward time: 0.52s Step time: 2.46s  
28800 Examples seen. Accuracy: 0.3986 Error: 1.41875 Loss: 1.52329 Threads: 2  
Forward time: 0.67s Backward time: 0.50s Step time: 2.44s  
29440 Examples seen. Accuracy: 0.4022 Error: 1.39563 Loss: 1.52202 Threads: 2  
Forward time: 0.66s Backward time: 0.51s Step time: 2.47s  
30080 Examples seen. Accuracy: 0.4017 Error: 1.45439 Loss: 1.65365 Threads: 2  
Forward time: 0.68s Backward time: 0.50s Step time: 2.44s  
30720 Examples seen. Accuracy: 0.4051 Error: 1.33340 Loss: 1.45610 Threads: 2  
Forward time: 0.67s Backward time: 0.50s Step time: 2.42s  
31360 Examples seen. Accuracy: 0.4062 Error: 1.43259 Loss: 1.59719 Threads: 2  
Forward time: 0.69s Backward time: 0.51s Step time: 2.43s  
32000 Examples seen. Accuracy: 0.4073 Error: 1.41820 Loss: 1.82291 Threads: 2  
Forward time: 0.66s Backward time: 0.50s Step time: 2.47s  
32640 Examples seen. Accuracy: 0.4096 Error: 1.47982 Loss: 1.65160 Threads: 2  
Forward time: 0.67s Backward time: 0.50s Step time: 2.45s  
33280 Examples seen. Accuracy: 0.4113 Error: 1.38456 Loss: 1.50167 Threads: 2  
Forward time: 0.68s Backward time: 0.49s Step time: 2.44s

33920 Examples seen. Accuracy: 0.4124 Error: 1.37090 Loss: 1.55271 Threads: 2  
Forward time: 0.66s Backward time: 0.50s Step time: 2.44s  
34560 Examples seen. Accuracy: 0.4161 Error: 1.43860 Loss: 1.73516 Threads: 2  
Forward time: 0.67s Backward time: 0.50s Step time: 2.44s  
35200 Examples seen. Accuracy: 0.4171 Error: 1.39582 Loss: 1.64415 Threads: 2  
Forward time: 0.68s Backward time: 0.49s Step time: 2.45s  
35840 Examples seen. Accuracy: 0.4200 Error: 1.40152 Loss: 1.58708 Threads: 2  
Forward time: 0.71s Backward time: 0.50s Step time: 2.45s  
36480 Examples seen. Accuracy: 0.4234 Error: 1.33571 Loss: 1.57454 Threads: 2  
Forward time: 0.69s Backward time: 0.54s Step time: 2.53s  
37120 Examples seen. Accuracy: 0.4301 Error: 1.20996 Loss: 1.25736 Threads: 2  
Forward time: 0.67s Backward time: 0.53s Step time: 2.52s  
37760 Examples seen. Accuracy: 0.4323 Error: 1.44967 Loss: 1.83971 Threads: 2  
Forward time: 0.69s Backward time: 0.50s Step time: 2.53s  
38400 Examples seen. Accuracy: 0.4334 Error: 1.32567 Loss: 1.47977 Threads: 2  
Forward time: 0.67s Backward time: 0.51s Step time: 2.46s  
39040 Examples seen. Accuracy: 0.4319 Error: 1.34760 Loss: 1.38344 Threads: 2  
Forward time: 0.68s Backward time: 0.52s Step time: 2.47s  
39680 Examples seen. Accuracy: 0.4334 Error: 1.32686 Loss: 1.47318 Threads: 2  
Forward time: 0.68s Backward time: 0.50s Step time: 2.47s  
Starting Validation.  
VALIDATION RECORD! Saving NN at SimpleImageClassifier-64.nn  
Epochs: 1 Examples seen:40000 Validation Accuracy: 0.5163 Validation Error:  
1.2367 Validation Loss: 1.3394 Total time: 3.66min  
Epoch time: 2.5750 minutes. 50 epochs: 2.1458 hours.  
Epochs: 1. Working time: 0.06 hours.  
40640 Examples seen. Accuracy: 0.4402 Error: 1.39607 Loss: 1.68056 Threads: 2  
Forward time: 0.66s Backward time: 0.51s Step time: 2.45s  
41280 Examples seen. Accuracy: 0.4421 Error: 1.25289 Loss: 1.23783 Threads: 2  
Forward time: 0.72s Backward time: 0.52s Step time: 2.52s  
41920 Examples seen. Accuracy: 0.4458 Error: 1.22554 Loss: 1.16512 Threads: 2  
Forward time: 0.68s Backward time: 0.51s Step time: 2.56s  
42560 Examples seen. Accuracy: 0.4467 Error: 1.33534 Loss: 1.68178 Threads: 2  
Forward time: 0.69s Backward time: 0.53s Step time: 2.49s  
43200 Examples seen. Accuracy: 0.4503 Error: 1.36218 Loss: 1.59631 Threads: 2  
Forward time: 0.66s Backward time: 0.51s Step time: 2.46s  
43840 Examples seen. Accuracy: 0.4508 Error: 1.39591 Loss: 1.59698 Threads: 2  
Forward time: 0.66s Backward time: 0.53s Step time: 2.47s  
44480 Examples seen. Accuracy: 0.4512 Error: 1.35992 Loss: 1.53007 Threads: 2  
Forward time: 0.67s Backward time: 0.50s Step time: 2.48s  
45120 Examples seen. Accuracy: 0.4494 Error: 1.32980 Loss: 1.38933 Threads: 2  
Forward time: 0.70s Backward time: 0.50s Step time: 2.45s  
45760 Examples seen. Accuracy: 0.4512 Error: 1.24250 Loss: 1.40123 Threads: 2  
Forward time: 0.67s Backward time: 0.49s Step time: 2.47s  
46400 Examples seen. Accuracy: 0.4528 Error: 1.41063 Loss: 1.56648 Threads: 2  
Forward time: 0.65s Backward time: 0.51s Step time: 2.43s  
47040 Examples seen. Accuracy: 0.4542 Error: 1.34692 Loss: 1.58015 Threads: 2  
Forward time: 0.74s Backward time: 0.52s Step time: 2.42s  
47680 Examples seen. Accuracy: 0.4582 Error: 1.25918 Loss: 1.29102 Threads: 2  
Forward time: 0.68s Backward time: 0.47s Step time: 2.42s  
48320 Examples seen. Accuracy: 0.4607 Error: 1.31420 Loss: 1.48015 Threads: 2  
Forward time: 0.67s Backward time: 0.50s Step time: 2.42s  
48960 Examples seen. Accuracy: 0.4616 Error: 1.35348 Loss: 1.57047 Threads: 2  
Forward time: 0.68s Backward time: 0.49s Step time: 2.43s  
49600 Examples seen. Accuracy: 0.4612 Error: 1.22624 Loss: 1.37146 Threads: 2  
Forward time: 0.69s Backward time: 0.50s Step time: 2.46s  
50240 Examples seen. Accuracy: 0.4595 Error: 1.40065 Loss: 1.56835 Threads: 2  
Forward time: 0.72s Backward time: 0.49s Step time: 2.49s  
50880 Examples seen. Accuracy: 0.4617 Error: 1.36675 Loss: 1.55090 Threads: 2  
Forward time: 0.68s Backward time: 0.49s Step time: 2.43s  
51520 Examples seen. Accuracy: 0.4609 Error: 1.34522 Loss: 1.38099 Threads: 2



Forward time: 0.69s Backward time: 0.52s Step time: 2.44s  
52160 Examples seen. Accuracy: 0.4633 Error: 1.28366 Loss: 1.41689 Threads: 2  
Forward time: 0.66s Backward time: 0.49s Step time: 2.42s  
52800 Examples seen. Accuracy: 0.4679 Error: 1.21513 Loss: 1.42658 Threads: 2  
Forward time: 0.68s Backward time: 0.52s Step time: 2.45s  
53440 Examples seen. Accuracy: 0.4708 Error: 1.30849 Loss: 1.53340 Threads: 2  
Forward time: 0.69s Backward time: 0.49s Step time: 2.43s  
54080 Examples seen. Accuracy: 0.4732 Error: 1.40573 Loss: 1.46094 Threads: 2  
Forward time: 0.67s Backward time: 0.47s Step time: 2.42s  
54720 Examples seen. Accuracy: 0.4775 Error: 1.23261 Loss: 1.33425 Threads: 2  
Forward time: 0.67s Backward time: 0.49s Step time: 2.42s  
55360 Examples seen. Accuracy: 0.4789 Error: 1.20426 Loss: 1.25272 Threads: 2  
Forward time: 0.70s Backward time: 0.51s Step time: 2.44s  
56000 Examples seen. Accuracy: 0.4824 Error: 1.17741 Loss: 1.18861 Threads: 2  
Forward time: 0.67s Backward time: 0.49s Step time: 2.41s  
56640 Examples seen. Accuracy: 0.4804 Error: 1.31370 Loss: 1.48129 Threads: 2  
Forward time: 0.70s Backward time: 0.56s Step time: 2.47s  
57280 Examples seen. Accuracy: 0.4816 Error: 1.34587 Loss: 1.58014 Threads: 2  
Forward time: 0.67s Backward time: 0.51s Step time: 2.47s  
57920 Examples seen. Accuracy: 0.4836 Error: 1.21501 Loss: 1.32031 Threads: 2  
Forward time: 0.66s Backward time: 0.48s Step time: 2.46s  
58560 Examples seen. Accuracy: 0.4855 Error: 1.25863 Loss: 1.42719 Threads: 2  
Forward time: 0.69s Backward time: 0.51s Step time: 2.43s  
59200 Examples seen. Accuracy: 0.4864 Error: 1.25754 Loss: 1.30167 Threads: 2  
Forward time: 0.67s Backward time: 0.48s Step time: 2.41s  
59840 Examples seen. Accuracy: 0.4902 Error: 1.18845 Loss: 1.25104 Threads: 2  
Forward time: 0.66s Backward time: 0.49s Step time: 2.40s  
60480 Examples seen. Accuracy: 0.4888 Error: 1.22724 Loss: 1.26680 Threads: 2  
Forward time: 0.69s Backward time: 0.46s Step time: 2.40s  
61120 Examples seen. Accuracy: 0.4858 Error: 1.32534 Loss: 1.66756 Threads: 2  
Forward time: 0.67s Backward time: 0.48s Step time: 2.41s  
61760 Examples seen. Accuracy: 0.4867 Error: 1.23665 Loss: 1.31761 Threads: 2  
Forward time: 0.68s Backward time: 0.51s Step time: 2.40s  
62400 Examples seen. Accuracy: 0.4891 Error: 1.37480 Loss: 1.56838 Threads: 2  
Forward time: 0.70s Backward time: 0.51s Step time: 2.45s  
63040 Examples seen. Accuracy: 0.4882 Error: 1.32497 Loss: 1.52067 Threads: 2  
Forward time: 0.68s Backward time: 0.48s Step time: 2.43s  
63680 Examples seen. Accuracy: 0.4885 Error: 1.23105 Loss: 1.33264 Threads: 2  
Forward time: 0.69s Backward time: 0.51s Step time: 2.44s  
64320 Examples seen. Accuracy: 0.4874 Error: 1.29117 Loss: 1.41264 Threads: 2  
Forward time: 0.67s Backward time: 0.50s Step time: 2.44s  
64960 Examples seen. Accuracy: 0.4905 Error: 1.04727 Loss: 1.08052 Threads: 2  
Forward time: 0.67s Backward time: 0.51s Step time: 2.43s  
65600 Examples seen. Accuracy: 0.4925 Error: 1.16781 Loss: 1.18518 Threads: 2  
Forward time: 0.66s Backward time: 0.48s Step time: 2.42s  
66240 Examples seen. Accuracy: 0.4946 Error: 1.32698 Loss: 1.36193 Threads: 2  
Forward time: 0.66s Backward time: 0.48s Step time: 2.41s  
66880 Examples seen. Accuracy: 0.4948 Error: 1.21861 Loss: 1.39665 Threads: 2  
Forward time: 0.70s Backward time: 0.48s Step time: 2.39s  
67520 Examples seen. Accuracy: 0.4989 Error: 1.13003 Loss: 1.22982 Threads: 2  
Forward time: 0.68s Backward time: 0.47s Step time: 2.38s  
68160 Examples seen. Accuracy: 0.4998 Error: 1.10655 Loss: 1.17619 Threads: 2  
Forward time: 0.66s Backward time: 0.48s Step time: 2.38s  
68800 Examples seen. Accuracy: 0.5027 Error: 1.11410 Loss: 1.23831 Threads: 2  
Forward time: 0.68s Backward time: 0.51s Step time: 2.43s  
69440 Examples seen. Accuracy: 0.5021 Error: 1.20098 Loss: 1.40219 Threads: 2  
Forward time: 0.68s Backward time: 0.49s Step time: 2.42s  
70080 Examples seen. Accuracy: 0.5025 Error: 1.28556 Loss: 1.60134 Threads: 2  
Forward time: 0.66s Backward time: 0.47s Step time: 2.39s  
70720 Examples seen. Accuracy: 0.5037 Error: 1.32673 Loss: 1.49878 Threads: 2  
Forward time: 0.71s Backward time: 0.47s Step time: 2.37s

71360 Examples seen. Accuracy: 0.5057 Error: 1.17321 Loss: 1.19003 Threads: 2  
 Forward time: 0.65s Backward time: 0.49s Step time: 2.39s  
 72000 Examples seen. Accuracy: 0.5046 Error: 1.21618 Loss: 1.37591 Threads: 2  
 Forward time: 0.66s Backward time: 0.47s Step time: 2.39s  
 72640 Examples seen. Accuracy: 0.5064 Error: 1.17879 Loss: 1.23261 Threads: 2  
 Forward time: 0.70s Backward time: 0.49s Step time: 2.38s  
 73280 Examples seen. Accuracy: 0.5059 Error: 1.12521 Loss: 1.21161 Threads: 2  
 Forward time: 0.67s Backward time: 0.48s Step time: 2.38s  
 73920 Examples seen. Accuracy: 0.5061 Error: 1.21973 Loss: 1.37913 Threads: 2  
 Forward time: 0.66s Backward time: 0.47s Step time: 2.37s  
 74560 Examples seen. Accuracy: 0.5034 Error: 1.34306 Loss: 1.40960 Threads: 2  
 Forward time: 0.69s Backward time: 0.48s Step time: 2.40s  
 75200 Examples seen. Accuracy: 0.5054 Error: 1.28294 Loss: 1.44882 Threads: 2  
 Forward time: 0.68s Backward time: 0.46s Step time: 2.40s  
 75840 Examples seen. Accuracy: 0.5065 Error: 1.37193 Loss: 1.64460 Threads: 2  
 Forward time: 0.68s Backward time: 0.50s Step time: 2.40s  
 76480 Examples seen. Accuracy: 0.5065 Error: 1.34023 Loss: 1.53348 Threads: 2  
 Forward time: 0.68s Backward time: 0.47s Step time: 2.46s  
 77120 Examples seen. Accuracy: 0.5073 Error: 1.26684 Loss: 1.33574 Threads: 2  
 Forward time: 0.65s Backward time: 0.50s Step time: 2.43s  
 77760 Examples seen. Accuracy: 0.5115 Error: 1.19319 Loss: 1.29400 Threads: 2  
 Forward time: 0.70s Backward time: 0.49s Step time: 2.40s  
 78400 Examples seen. Accuracy: 0.5139 Error: 1.18306 Loss: 1.60150 Threads: 2  
 Forward time: 0.68s Backward time: 0.46s Step time: 2.43s  
 79040 Examples seen. Accuracy: 0.5180 Error: 1.28231 Loss: 1.38482 Threads: 2  
 Forward time: 0.68s Backward time: 0.49s Step time: 2.39s  
 79680 Examples seen. Accuracy: 0.5198 Error: 1.23234 Loss: 1.17779 Threads: 2  
 Forward time: 0.68s Backward time: 0.47s Step time: 2.42s  
 Starting Validation.  
 VALIDATION RECORD! Saving NN at SimpleImageClassifier-64.nn  
 Epochs: 2 Examples seen:80000 Validation Accuracy: 0.5280 Validation Error:  
 1.2106 Validation Loss: 1.2998 Total time: 7.23min  
 Layer 0 Max Output: 1.672  
 Min Output: -1.922 TNNetInput 32,32,3 Times: 0.00s 0.00s  
 Layer 1 Neurons: 64 Max Weight: 0.241 Min Weight: -0.235 Max Output: 3.830  
 Min Output: -3.955 TNNetConvolutionLinear 32,32,64 Times: 30.20s 0.07s Parent:0  
 Layer 2 Max Output: 3.830  
 Min Output: -1.498 TNNetMaxPool 8,8,64 Times: 7.00s 0.00s Parent:1  
 Layer 3 Neurons: 1 Max Weight: 0.902 Min Weight: 0.586 Max Output: 3.596  
 Min Output: -2.310 TNNetMovingStdNormalization 8,8,64 Times: 0.15s 0.00s  
 Parent:2  
 Layer 4 Neurons: 64 Max Weight: 0.105 Min Weight: -0.104 Max Output: 4.570  
 Min Output: 0.000 TNNetConvolutionReLU 8,8,64 Times: 6.05s 0.04s Parent:3  
 Layer 5 Neurons: 64 Max Weight: 0.112 Min Weight: -0.114 Max Output: 3.343  
 Min Output: 0.000 TNNetConvolutionReLU 8,8,64 Times: 6.01s 0.03s Parent:4  
 Layer 6 Neurons: 64 Max Weight: 0.117 Min Weight: -0.102 Max Output: 2.087  
 Min Output: 0.000 TNNetConvolutionReLU 8,8,64 Times: 5.96s 0.04s Parent:5  
 Layer 7 Neurons: 64 Max Weight: 0.124 Min Weight: -0.108 Max Output: 1.748  
 Min Output: 0.000 TNNetConvolutionReLU 8,8,64 Times: 6.08s 0.01s Parent:6  
 Layer 8 Max Output: 1.748  
 Min Output: 0.000 TNNetDropout 8,8,64 Times: 0.02s 0.00s Parent:7  
 Layer 9 Max Output: 1.748  
 Min Output: 0.000 TNNetMaxPool 4,4,64 Times: 0.43s 0.00s Parent:8  
 Layer 10 Neurons: 10 Max Weight: 0.184 Min Weight: -0.146 Max Output: 2.421  
 Min Output: -2.251 TNNetFullConnectLinear 10,1,1 Times: 0.08s 0.00s Parent:9  
 Layer 11 Max Output: 0.578  
 Min Output: 0.005 TNNetSoftMax 10,1,1 Times: 0.00s 0.00s Parent:10  
 Epoch time: 2.5208 minutes. 50 epochs: 2.1007 hours.  
 Epochs: 2. Working time: 0.12 hours.  
 80640 Examples seen. Accuracy: 0.5217 Error: 1.25977 Loss: 1.32582 Threads: 2  
 Forward time: 0.67s Backward time: 0.51s Step time: 2.42s

81280 Examples seen. Accuracy: 0.5215 Error: 1.23738 Loss: 1.46194 Threads: 2  
Forward time: 0.74s Backward time: 0.53s Step time: 2.50s  
81920 Examples seen. Accuracy: 0.5240 Error: 1.36221 Loss: 1.65249 Threads: 2  
Forward time: 0.72s Backward time: 0.54s Step time: 2.64s  
82560 Examples seen. Accuracy: 0.5249 Error: 1.08699 Loss: 1.05420 Threads: 2  
Forward time: 0.70s Backward time: 0.47s Step time: 2.50s  
83200 Examples seen. Accuracy: 0.5256 Error: 1.22711 Loss: 1.34938 Threads: 2  
Forward time: 0.72s Backward time: 0.49s Step time: 2.44s  
83840 Examples seen. Accuracy: 0.5265 Error: 1.22419 Loss: 1.33143 Threads: 2  
Forward time: 0.69s Backward time: 0.48s Step time: 2.45s  
84480 Examples seen. Accuracy: 0.5255 Error: 1.28435 Loss: 1.42781 Threads: 2  
Forward time: 0.66s Backward time: 0.49s Step time: 2.44s  
85120 Examples seen. Accuracy: 0.5273 Error: 1.11672 Loss: 1.16786 Threads: 2  
Forward time: 0.70s Backward time: 0.51s Step time: 2.46s  
85760 Examples seen. Accuracy: 0.5265 Error: 1.11195 Loss: 1.16223 Threads: 2  
Forward time: 0.70s Backward time: 0.51s Step time: 2.49s  
86400 Examples seen. Accuracy: 0.5271 Error: 1.18559 Loss: 1.20379 Threads: 2  
Forward time: 0.70s Backward time: 0.47s Step time: 2.49s  
87040 Examples seen. Accuracy: 0.5304 Error: 1.06645 Loss: 1.06041 Threads: 2  
Forward time: 0.67s Backward time: 0.49s Step time: 2.44s  
87680 Examples seen. Accuracy: 0.5291 Error: 1.06728 Loss: 1.12536 Threads: 2  
Forward time: 0.67s Backward time: 0.50s Step time: 2.44s  
88320 Examples seen. Accuracy: 0.5284 Error: 1.36197 Loss: 1.49824 Threads: 2  
Forward time: 0.69s Backward time: 0.48s Step time: 2.46s  
88960 Examples seen. Accuracy: 0.5286 Error: 1.33446 Loss: 1.53771 Threads: 2  
Forward time: 0.69s Backward time: 0.50s Step time: 2.44s  
89600 Examples seen. Accuracy: 0.5299 Error: 1.20436 Loss: 1.29304 Threads: 2  
Forward time: 0.71s Backward time: 0.52s Step time: 2.49s  
90240 Examples seen. Accuracy: 0.5322 Error: 1.13818 Loss: 1.20439 Threads: 2  
Forward time: 0.69s Backward time: 0.50s Step time: 2.48s  
90880 Examples seen. Accuracy: 0.5321 Error: 1.18145 Loss: 1.34056 Threads: 2  
Forward time: 0.70s Backward time: 0.50s Step time: 2.46s  
91520 Examples seen. Accuracy: 0.5338 Error: 1.18489 Loss: 1.25516 Threads: 2  
Forward time: 0.67s Backward time: 0.46s Step time: 2.39s  
92160 Examples seen. Accuracy: 0.5332 Error: 1.06375 Loss: 1.23520 Threads: 2  
Forward time: 0.68s Backward time: 0.48s Step time: 2.38s  
92800 Examples seen. Accuracy: 0.5353 Error: 1.20202 Loss: 1.41078 Threads: 2  
Forward time: 0.74s Backward time: 0.51s Step time: 2.40s  
93440 Examples seen. Accuracy: 0.5402 Error: 1.10855 Loss: 1.23545 Threads: 2  
Forward time: 0.67s Backward time: 0.48s Step time: 2.40s  
94080 Examples seen. Accuracy: 0.5418 Error: 1.23072 Loss: 1.32038 Threads: 2  
Forward time: 0.70s Backward time: 0.51s Step time: 2.40s  
94720 Examples seen. Accuracy: 0.5421 Error: 1.19074 Loss: 1.50945 Threads: 2  
Forward time: 0.68s Backward time: 0.50s Step time: 2.45s  
95360 Examples seen. Accuracy: 0.5426 Error: 1.26900 Loss: 1.37056 Threads: 2  
Forward time: 0.68s Backward time: 0.50s Step time: 2.42s  
96000 Examples seen. Accuracy: 0.5407 Error: 1.26377 Loss: 1.49441 Threads: 2  
Forward time: 0.67s Backward time: 0.47s Step time: 2.42s  
96640 Examples seen. Accuracy: 0.5392 Error: 1.28611 Loss: 1.47267 Threads: 2  
Forward time: 0.68s Backward time: 0.49s Step time: 2.40s  
97280 Examples seen. Accuracy: 0.5416 Error: 1.18893 Loss: 1.39908 Threads: 2  
Forward time: 0.69s Backward time: 0.50s Step time: 2.43s  
97920 Examples seen. Accuracy: 0.5405 Error: 1.18357 Loss: 1.29477 Threads: 2  
Forward time: 0.69s Backward time: 0.49s Step time: 2.40s  
98560 Examples seen. Accuracy: 0.5421 Error: 1.12833 Loss: 1.16271 Threads: 2  
Forward time: 0.66s Backward time: 0.49s Step time: 2.39s  
99200 Examples seen. Accuracy: 0.5448 Error: 1.18661 Loss: 1.23381 Threads: 2  
Forward time: 0.67s Backward time: 0.47s Step time: 2.38s  
99840 Examples seen. Accuracy: 0.5451 Error: 1.13116 Loss: 1.27461 Threads: 2  
Forward time: 0.70s Backward time: 0.48s Step time: 2.40s  
100480 Examples seen. Accuracy: 0.5438 Error: 1.19824 Loss: 1.33744 Threads: 2

Forward time: 0.67s Backward time: 0.50s Step time: 2.42s  
101120 Examples seen. Accuracy: 0.5435 Error: 1.26597 Loss: 1.40315 Threads: 2  
Forward time: 0.68s Backward time: 0.46s Step time: 2.43s  
101760 Examples seen. Accuracy: 0.5447 Error: 1.14179 Loss: 1.22289 Threads: 2  
Forward time: 0.71s Backward time: 0.50s Step time: 2.42s  
102400 Examples seen. Accuracy: 0.5426 Error: 1.16172 Loss: 1.28683 Threads: 2  
Forward time: 0.68s Backward time: 0.49s Step time: 2.45s  
103040 Examples seen. Accuracy: 0.5464 Error: 1.12014 Loss: 1.14475 Threads: 2  
Forward time: 0.68s Backward time: 0.47s Step time: 2.46s  
103680 Examples seen. Accuracy: 0.5469 Error: 1.19354 Loss: 1.35633 Threads: 2  
Forward time: 0.70s Backward time: 0.51s Step time: 2.40s  
104320 Examples seen. Accuracy: 0.5503 Error: 1.09195 Loss: 1.17948 Threads: 2  
Forward time: 0.67s Backward time: 0.46s Step time: 2.38s  
104960 Examples seen. Accuracy: 0.5537 Error: 1.13195 Loss: 1.23216 Threads: 2  
Forward time: 0.70s Backward time: 0.48s Step time: 2.39s  
105600 Examples seen. Accuracy: 0.5542 Error: 1.17438 Loss: 1.25003 Threads: 2  
Forward time: 0.65s Backward time: 0.47s Step time: 2.38s  
106240 Examples seen. Accuracy: 0.5558 Error: 1.06134 Loss: 1.15410 Threads: 2  
Forward time: 0.68s Backward time: 0.49s Step time: 2.38s  
106880 Examples seen. Accuracy: 0.5553 Error: 1.14725 Loss: 1.23581 Threads: 2  
Forward time: 0.72s Backward time: 0.47s Step time: 2.45s  
107520 Examples seen. Accuracy: 0.5565 Error: 1.02141 Loss: 1.04104 Threads: 2  
Forward time: 0.66s Backward time: 0.46s Step time: 2.35s  
108160 Examples seen. Accuracy: 0.5562 Error: 1.09003 Loss: 1.10040 Threads: 2  
Forward time: 0.66s Backward time: 0.47s Step time: 2.35s  
108800 Examples seen. Accuracy: 0.5555 Error: 1.19926 Loss: 1.41436 Threads: 2  
Forward time: 0.68s Backward time: 0.49s Step time: 2.38s  
109440 Examples seen. Accuracy: 0.5555 Error: 1.13296 Loss: 1.32973 Threads: 2  
Forward time: 0.68s Backward time: 0.44s Step time: 2.38s  
110080 Examples seen. Accuracy: 0.5564 Error: 1.19110 Loss: 1.43251 Threads: 2  
Forward time: 0.68s Backward time: 0.44s Step time: 2.35s  
110720 Examples seen. Accuracy: 0.5594 Error: 1.14849 Loss: 1.27873 Threads: 2  
Forward time: 0.69s Backward time: 0.52s Step time: 2.38s  
111360 Examples seen. Accuracy: 0.5613 Error: 1.11265 Loss: 1.26440 Threads: 2  
Forward time: 0.67s Backward time: 0.47s Step time: 2.42s  
112000 Examples seen. Accuracy: 0.5588 Error: 1.05700 Loss: 1.13769 Threads: 2  
Forward time: 0.67s Backward time: 0.47s Step time: 2.39s  
112640 Examples seen. Accuracy: 0.5605 Error: 1.04210 Loss: 1.11334 Threads: 2  
Forward time: 0.67s Backward time: 0.47s Step time: 2.36s  
113280 Examples seen. Accuracy: 0.5577 Error: 1.08367 Loss: 1.09816 Threads: 2  
Forward time: 0.66s Backward time: 0.45s Step time: 2.34s  
113920 Examples seen. Accuracy: 0.5577 Error: 1.10999 Loss: 1.16392 Threads: 2  
Forward time: 0.68s Backward time: 0.45s Step time: 2.35s  
114560 Examples seen. Accuracy: 0.5576 Error: 1.12588 Loss: 1.15112 Threads: 2  
Forward time: 0.68s Backward time: 0.45s Step time: 2.35s  
115200 Examples seen. Accuracy: 0.5564 Error: 1.14632 Loss: 1.30112 Threads: 2  
Forward time: 0.67s Backward time: 0.45s Step time: 2.38s  
115840 Examples seen. Accuracy: 0.5592 Error: 1.13579 Loss: 1.28841 Threads: 2  
Forward time: 0.66s Backward time: 0.48s Step time: 2.35s  
116480 Examples seen. Accuracy: 0.5598 Error: 1.14585 Loss: 1.20761 Threads: 2  
Forward time: 0.65s Backward time: 0.48s Step time: 2.35s  
117120 Examples seen. Accuracy: 0.5603 Error: 1.03519 Loss: 1.08286 Threads: 2  
Forward time: 0.67s Backward time: 0.45s Step time: 2.39s  
117760 Examples seen. Accuracy: 0.5624 Error: 1.03469 Loss: 1.14592 Threads: 2  
Forward time: 0.65s Backward time: 0.48s Step time: 2.37s  
118400 Examples seen. Accuracy: 0.5621 Error: 1.12518 Loss: 1.22586 Threads: 2  
Forward time: 0.69s Backward time: 0.44s Step time: 2.35s  
119040 Examples seen. Accuracy: 0.5648 Error: 1.06442 Loss: 1.03027 Threads: 2  
Forward time: 0.66s Backward time: 0.46s Step time: 2.36s  
119680 Examples seen. Accuracy: 0.5632 Error: 1.14389 Loss: 1.19306 Threads: 2  
Forward time: 0.71s Backward time: 0.44s Step time: 2.34s

Starting Validation.  
 VALIDATION RECORD! Saving NN at SimpleImageClassifier-64.nn  
 Epochs: 3 Examples seen:120000 Validation Accuracy: 0.5486 Validation Error: 1.1838 Validation Loss: 1.2516 Total time: 10.79min  
 Epoch time: 2.4354 minutes. 50 epochs: 2.0295 hours.  
 Epochs: 3. Working time: 0.18 hours.  
 120640 Examples seen. Accuracy: 0.5678 Error: 1.15427 Loss: 1.31939 Threads: 2  
 Forward time: 0.67s Backward time: 0.47s Step time: 2.36s  
 121280 Examples seen. Accuracy: 0.5715 Error: 1.17294 Loss: 1.40640

**VALIDATION RECORD! Saving NN at SimpleImageClassifier-64.nn**  
 Epochs: 20 - Examples seen:800000 Validation Accuracy: 0.8077 Validation Error: 0.5409 Validation Loss: 0.5646 Total time: 69.72min  
 Layer 0

Starting Testing. Epochs: 20 Examples seen:800000 Test Accuracy: 0.8013 Test Error: 0.5539 Test Loss: 0.5934 Total time: 70.75min  
 Epoch time: 2.2958 minutes. 50 epochs: 1.9132 hours.  
 Epochs: 20. Working time: 1.18 hours.  
 Learning rate set to: 0.00082

**Layer 11** **Max Output: 0.516**  
**Min Output: 0.000 TNNetSoftMax 10,1,1 Times: 0.00s 0.00s Parent:10**

Starting Testing.

Layer 10 Neurons: 10 Max Weight: 0.319 Min Weight: -0.273 Max Output: 5.777  
 Min Output: -10.742 TNNetFullConnectLinear 10,1,1 Times: 0.13s 0.00s Parent:9

Layer 11 Max Output: 0.608  
 Min Output: 0.000 TNNetSoftMax 10,1,1 Times: 0.00s 0.00s Parent:10  
 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.

14. `from google.colab import files`

`!ls -l`  
**total 348464 total 348464**

```
-rw-r--r-- 1 root root      61 Sep 23 08:51 batches.meta.txt
drwxr-xr-x 2 2156 1103    4096 Jun  4  2009 cifar-10-batches-bin
-rw-r--r-- 1 root root 30730000 Sep 23 08:51 data_batch_1.bin
-rw-r--r-- 1 root root 30730000 Sep 23 08:51 data_batch_2.bin
-rw-r--r-- 1 root root 30730000 Sep 23 08:51 data_batch_3.bin
-rw-r--r-- 1 root root 30730000 Sep 23 08:51 data_batch_4.bin
-rw-r--r-- 1 root root 30730000 Sep 23 08:51 data_batch_5.bin
-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
-rw-r--r-- 1 root root     88 Sep 23 08:51 readme.html
drwxr-xr-x 1 root root    4096 Sep 16 13:40 sample_data
-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
-rw-r--r-- 1 root root 30730000 Sep 23 08:51 test_batch.bin
```

```
15. files.download('SimpleImageClassifier.nn')
```

```
16. files.download('SimpleImageClassifier.csv')
```

FileNotFoundError Traceback (most recent call last)

```
<ipython-input-16-dc2343cca3de> in <module>()
----> 1 files.download('SimpleImageClassifier.csv')
/usr/local/lib/python3.7/dist-packages/google/colab/files.py in
download(filename)
    141     raise OSError(msg)
    142     else:
--> 143     raise FileNotFoundError(msg) # pylint: disable=undefined-variable
    144
    145     comm_manager = _IPython.get_ipython().kernel.comm_manager
FileNotFoundError: Cannot find file: SimpleImageClassifier.csv
```

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

epoch	training accuracy	training loss	training error	validation accuracy
	validation loss	validation error	learning rate	time
	test loss	test error		test accuracy
1	0.4353	1.5563	1.4157	0.5163
2	0.5212	1.225	1.1536	0.528

```
15. files.download('SimpleImageClassifier.nn')
```

```
if (ValidationRate > ValidationRecord) then begin
    ValidationRecord := ValidationRate;
    FMessageProc('VALIDATION RECORD! Saving NN at '+fileName);
    FavgWeight.SaveToFile(fileName);
end;
```

```
--
```

```
1)TNNetInput:32;32;3;0;0;0;0;0;0#0)TNNetConvolutionLinear:64;5;2;1;1;0;0;0#1)TNNetMax
Pool:4;4;0;0;0;0;0;0#2)TNNetMovingStdNormalization:0;0;0;0;0;0;0;0#3)TNNetConvoluti
onReLU:64;3;1;1;1;0;0;0#4)TNNetConvolutionReLU:64;3;1;1;1;0;0;0#5)TNNetConvolutionR
eLU:64;3;1;1;1;0;0;0#6).....
```