Part 2:
Caffe flow with CIFAR10

XILINX®

CIFAR-10 dataset

Secure | https://www.cs.toronto.edu/~kriz/cifar.html

Apps New Tab COMPASS Google Images Imported From IE Apps Imported Zen_Music

< Back to Alex Krizhevsky's home page

The CIFAR-10 and CIFAR-100 are labeled subsets of the 80 million tiny images dataset. They were collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

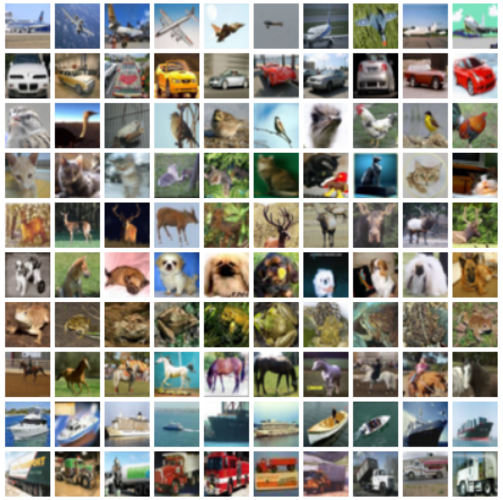
The CIFAR-10 dataset

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Here are the classes in the dataset, as well as 10 random images from each:

airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck



The classes are completely mutually exclusive. There is no overlap between automobiles and trucks. "Automobile" includes sedans, SUVs, things of that sort. "Truck" includes only big trucks. Neither includes pickup trucks.

CIFAR-10 project directories structure

```
ML/cifar10/
├── caffe
│   ├── code
│   ├── models
│   │   ├── miniGoogleNet
│   │   ├── miniVggNet
│   └── rpt
├── deephi
│   └── miniVggNet
│       ├── pruning
│       │   ├── quantiz
│       │   │   ├── data
│       │   │   ├── decent_output
│       │   │   ├── dnnc_output
│       │   │   └── rpt
│       │   └── rpt
│       ├── quantiz
│       │   ├── data
│       │   ├── decent_output
│       │   ├── dnnc_output
│       │   └── rpt
│       └── zcu102
│           ├── baseline
│           │   ├── model
│           │   ├── rpt
│           │   ├── src
│           │   └── test_images -> ../test_images
│           ├── pruned
│           │   ├── model
│           │   ├── rpt
│           │   ├── src
│           │   └── test_images -> ../test_images
│           └── test_images
├── input
│   ├── cifar10_jpg
│   │   ├── calib
│   │   ├── test
│   │   ├── train
│   │   └── val
│   └── lmdb
│       ├── train_lmdb
│       └── valid_lmdb
```

The **cifar10** project is organized in the following subdirectories:

1) **`caffe/code`** contains all the Python2.7 scripts

2) **`caffe/models`** contain only the prototxt files for solver, training and deploy of the 2 CNNs

3) **`caffe/rpt`** contains some logfiles as a reference

4) **`deephi`** contains the files for quantization of baseline CNN (**`quantiz`**) or of pruned CNN (**`pruning`**) plus the files for ZCU102 run time execution (respectively **`zcu102/baseline`**, **`zcu102/pruned`**)

5) **`input`** contains the JPEG images and lmdb databases for Caffe phases of training and validation and JPEG images for testing top-1 accuracy, plus the JPEG images for DeePhi’ calibration during the quantization phase

XILINX

Python 2.7 scripts

- `1_write_cifar10_images.py`
  - download the dataset from `keras.datasets` (you must have keras/tensorflow installed into your python virtual env, it is needed only to run this script)
  - store it in JPEG format on the folder `input/cifar10_jpg` (`test`, `train`, `val`, `calib`)
  - To be executed only once forever (folder `calib` is needed only for Quantization with DeePhi)
- `2a_create_lmdb.py`
  - It creates the LMDB databases `input/lmdb/train_lmdb` and `input/lmdb/valid_lmdb` for the training step
  - To be executed only once forever
- `2b_compute_mean.py`
  - It computes the mean values for the `train_lmdb` database in `input/mean.binaryproto`
  - To be executed only once forever
- `3_read_lmdb.py`
  - Just to debug the first 2 scripts

Python 2.7 scripts

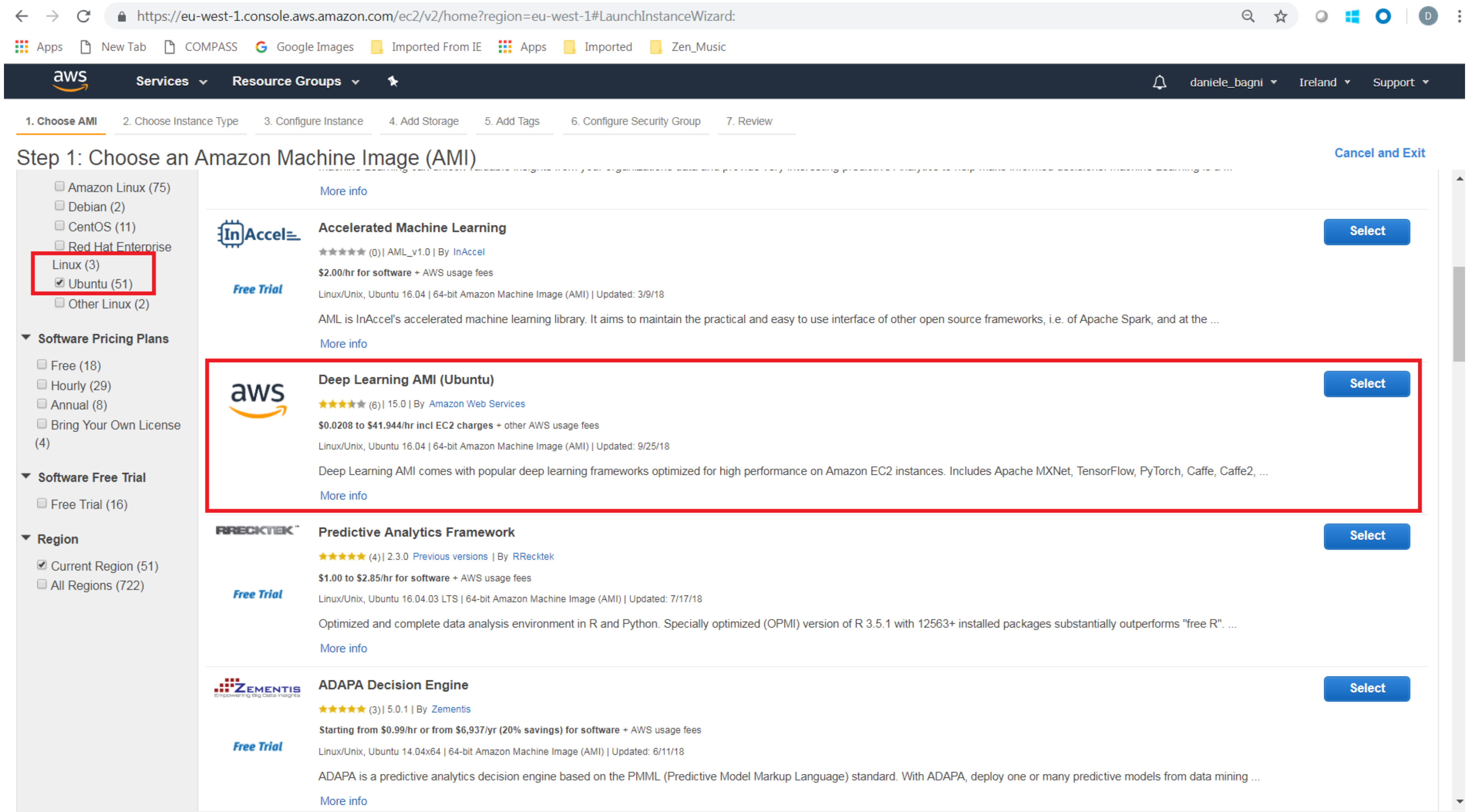
- **`4_training.py`**
  - To launch the training process in Caffe, given solver and CNN description prototxt files
  - To be used for any trial of training
- **`5_plot_learning_curve.py + plot_training_log.py`**
  - To be launched at the end of the training to plot the learning curves of accuracy and lost (in different ways)
- **`6_make_predictions.py`**
  - To be launched at the end of the training to measure the prediction accuracy achieved by the CNN just trained. You need to have scikit library installed
- **`check_dpu_runtime_accuracy.py`**
  - To be launched only once the CNN will be running on the ZCU102 board, to compute the effective top-1 accuracy of DeePhi DPU at runtime
- All those scripts can be orchestrated in the shell script called **`caffe_flow_CNN.sh`** (see next page), and normally the first 4 scripts are commented there

caffe_flow_CNN.sh script

```
#!/bin/sh
CAFFE_ROOT=/caffe/BVLC1v0-Caffe
CAFFE_TOOLS_DIR=$CAFFE_ROOT/distribute
WORK_DIR=/home/ML/cifar10/caffe #working dir

MOD_NUM=3   # model number
NUMIT=40000 # number of iterations
NET=miniVggNet

# ##########################################################################################
# SCRIPTS 1 2 3 (DATABASE AND MEAN VALUES)
echo "DATABASE: training and validation in LMDB, test in JPG and MEAN values"

# load the database from keras and write it as JPEG images
python $WORK_DIR/code/1_write_cifar10_images.py

#create LMDB databases -training (50K), validation (9K), test (1K) images - and compute mean values
python $WORK_DIR/code/2a_create_lmdb.py
python $WORK_DIR/code/2b_compute_mean.py

#check goodness of LMDB databases (just for debug: you can skip it)
python $WORK_DIR/code/3_read_lmdb.py

 ##########################################################################################
# SCRIPT 4  (SOLVER AND TRAINING AND LEARNING CURVE)
echo "TRAINING. Remember that: <Epoch_index = floor((iteration_index * batch_size) / (# data_samples))>"

python $WORK_DIR/code/4_training.py -s ./models/$NET/m$MOD_NUM/solver_$MOD_NUM\_$NET.prototxt -l ./models/$NET/m$MOD_NUM/logfile_$MOD_NUM\_$NET.log

# print image of CNN architecture
echo "PRINT CNN BLOCK DIAGRAM"
python $CAFFE_TOOLS_DIR/python/draw_net.py $WORK_DIR/models/$NET/m$MOD_NUM/train_val_$MOD_NUM\_$NET.prototxt $WORK_DIR/models/$NET/m$MOD_NUM/bd_$MOD_NUM\_$NET.png

# ##########################################################################################
# SCRIPT 5: plot the learning curve
echo "PLOT LEARNING CURVERS"
python $WORK_DIR/code/5_plot_learning_curve.py $WORK_DIR/models/$NET/m$MOD_NUM/logfile_$MOD_NUM\_$NET.log $WORK_DIR/models/$NET/m$MOD_NUM/plt_train_val_$MOD_NUM\_$NET.png

# ##########################################################################################
# SCRIPT 6 (PREDICTION)
echo "COMPUTE PREDICTIONS"
python $WORK_DIR/code/6_make_predictions.py -d $WORK_DIR/models/$NET/m$MOD_NUM/deploy_$MOD_NUM\_$NET.prototxt \
       -w $WORK_DIR/models/$NET/m$MOD_NUM/snapshot_$MOD_NUM\_$NET\__iter_$NUMIT.caffemodel 2>&1 | tee $WORK_DIR/models/$NET/m$MOD_NUM/predictions_$MOD_NUM\_$NET.txt
```

AWS AMI for ML

Select **Deep Learning AMI (Ubuntu)**



AWS EC2 instance

Launch Instance | Connect | Actions

Filter by tags and attributes or search by keyword

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP
	i-0f0652a6b93847cd7	p2.xlarge	eu-west-1b	pending	Initializing	None		-

Instance: i-0f0652a6b93847cd7 Private IP: 172.31.4.119

Description | Status Checks | Monitoring | Tags

Instance ID	i-0f0652a6b93847cd7	Public DNS (IPv4)	-
Instance state	pending	IPv4 Public IP	-
Instance type	p2.xlarge	IPv6 IPs	-
Elastic IPs		Private DNS	ip-172-31-4-119.eu-west-1.compute.internal
Availability zone	eu-west-1b	Private IPs	172.31.4.119
Security groups	Deep Learning AMI -Ubuntu--15-0-AutogenByAWSMP-. view inbound rules. view outbound rules	Secondary private IPs	
Scheduled events	-	VPC ID	vpc-5127a337
AMI ID	Deep Learning AMI (Ubuntu) Version 15.0 (ami-028a25f96614a39e9)	Subnet ID	subnet-1bdd087d
Platform	-	Network interfaces	eth0
IAM role	-	Source/dest. check	True

XILINX

Logging in into AWS

```
ec2-user@ip-172-31-10-57:~
[ec2-user@ip-172-31-10-57 ~]$
[ec2-user@ip-172-31-10-57 ~]$ cd
[ec2-user@ip-172-31-10-57 ~]$ ls -l
total 2800
drwxrwxr-x 24 ec2-user ec2-user    4096 19 lug 19.19 anaconda3
drwxrwxr-x  8 ec2-user ec2-user    4096 19 lug 18.10 examples
drwxrwxr-x  3 ec2-user ec2-user    4096  4 ott 06.40 ML
-rw-rw-r--  1 ec2-user ec2-user 2839191 14 lug 02.18 Nvidia_Cloud_EULA.pdf
-rw-rw-r--  1 ec2-user ec2-user    2792 12 ago 13.50 README
drwxrwxr-x 10 ec2-user ec2-user    4096 19 lug 20.02 src
drwxrwxr-x  5 ec2-user ec2-user    4096 19 lug 18.10 tutorials
[ec2-user@ip-172-31-10-57 ~]$ ls -l ML
total 73984
drwxrwxr-x 5 ec2-user ec2-user     4096 10 set 16.41 cats-vs-dogs
-rw-rw-r-- 1 ec2-user ec2-user 75751719  4 ott 06.40 ML-CIFAR10-tutorial-master.zip
[ec2-user@ip-172-31-10-57 ~]$ ls -l src/caffe_python_2/
total 124
drwxrwxr-x 14 ec2-user ec2-user  4096 19 lug 20.02 build
-rw-rw-r--  1 ec2-user ec2-user  1180 19 lug 20.01 caffe.cloc
drwxrwxr-x  5 ec2-user ec2-user  4096 19 lug 20.01 cmake
-rw-rw-r--  1 ec2-user ec2-user  4197 19 lug 20.01 CMakeLists.txt
-rw-rw-r--  1 ec2-user ec2-user  1917 19 lug 20.01 CONTRIBUTING.md
-rw-rw-r--  1 ec2-user ec2-user   620 19 lug 20.01 CONTRIBUTORS.md
drwxrwxr-x  5 ec2-user ec2-user  4096 19 lug 20.01 data
drwxrwxr-x  4 ec2-user ec2-user  4096 19 lug 20.01 docker
drwxrwxr-x  6 ec2-user ec2-user  4096 19 lug 20.01 docs
drwxrwxr-x 15 ec2-user ec2-user  4096 19 lug 20.01 examples
drwxrwxr-x  3 ec2-user ec2-user  4096 19 lug 20.01 include
-rw-rw-r--  1 ec2-user ec2-user   210 19 lug 20.01 INSTALL.md
-rw-rw-r--  1 ec2-user ec2-user  2092 19 lug 20.01 LICENSE
-rw-rw-r--  1 ec2-user ec2-user 24041 19 lug 20.01 Makefile
-rw-rw-r--  1 ec2-user ec2-user  4634 19 lug 20.01 Makefile.config
-rw-rw-r--  1 ec2-user ec2-user  4631 19 lug 20.01 Makefile.config.example
drwxrwxr-x  5 ec2-user ec2-user  4096 19 lug 20.01 matlab
drwxrwxr-x  7 ec2-user ec2-user  4096 19 lug 20.01 models
drwxrwxr-x  3 ec2-user ec2-user  4096 19 lug 20.01 python
-rw-rw-r--  1 ec2-user ec2-user  2130 19 lug 20.01 README.md
drwxrwxr-x  3 ec2-user ec2-user  4096 19 lug 20.01 scripts
drwxrwxr-x  4 ec2-user ec2-user  4096 19 lug 20.01 src
drwxrwxr-x  3 ec2-user ec2-user  4096 19 lug 20.01 tools
```

It is assumed that:

1) you have created a folder named ~/**ML**

2) You have unzipped the **ML-CIFAR10-tutorial-master.zip**

3) You have renamed it simply **cifar10**

4) At the end of above steps the folder **~/ML/cifar10** exists

XILINX

WARNING: Caffe with Python2.7 (`caffe_p27`) on AWS

> Caffe on AWS has a slightly different installation structure, being based on **`anaconda`**. In particular:
  - folder **`distribute`** does not exists and is replaced by folder **`build/install`**
  - **`caffe.bin`** executable is just named **`caffe`** s in previous page
  - **`compute_image_mean.bin`** has not been compiled with OpenCV: you should recompile the whole Caffe on AWS, do not do that, too much time consuming! Instead, just copy `mean.binaryproto` I have provided in the archive to your AWS site

WARNING: Tensorflow/Python2.7 (`tensorflow_p27`) on AWS

> Furthermore, from **tensorflow_p27** environment (**`source activate tensorflow_p27`**) I also had to run the following commands (the first one only once forever)

```
>> conda install keras
>> export LD_LIBRARY_PATH=~/src/caffe_python_2/build/install/lib64/:$LD_LIBRARY_PATH
>> export PYTHONPATH=/usr/local/lib/python2.7/dist-packages/:$PYTHONPATH
```

> **ALL THOSE SETTINGS ARE CAPTURED IN THE SCRIPT set_aws_ML_env_cuda9.sh (have a look at it before launching, read the comments!)**

```
>> cd ~/ML/cifar10/aws_scripts
>> source set_aws_ML_env_cuda9.sh 2>&1 | tee ../rpt/logfile_set_aws_ML_env.txt
```


XILINX.

aws_caffe_flow_miniVggNet.sh script

```
#!/bin/sh
CAFFE_ROOT=/caffe/BVLC1v0-Caffe
CAFFE_TOOLS_DIR=$CAFFE_ROOT/distribute
WORK_DIR=/home/ML/cifar10/caffe #working dir
MOD_NUM=3   # model number
NUMIT=40000 # number of iterations
NET=miniVggNet

# ###########################################################################################
# SCRIPTS 1 2 3 (DATABASE AND MEAN VALUES)

# go to TensorFlow environment, load the database from Keras and write it as JPEG images
source activate tensorflow_p27
export PYTHONPATH=/usr/local/lib/python2.7/dist-packages/:/home/ubuntu/anaconda3/envs/caffe_p27/lib/python2.7/site-packages/:$PYTHONPATH #needed for opencv
echo "DATABASE: training and validation in LMDB, test in JPG and MEAN values"
python $WORK_DIR/code/1_write_cifar10_images.py #--pathname /home/danieleb/ML/cifar10/input/cifar10_jpg
export PYTHONPATH=/home/ubuntu/src/cntk/bindings/python
source deactivate tensorflow_p27

# go to Caffe environment
source activate caffe_p27

#create LMDB databases -training (50K), validation (9K), test (1K) images - and compute mean values
python $WORK_DIR/code/2a_create_lmdb.py
# DO NOT RUN BELOW COMMAND AS mean.binaryproto is already available! Is is here only for reference
##python $WORK_DIR/code/2b_compute_mean.py

#check goodness of LMDB databases (just for debug: you can skip it)
python $WORK_DIR/code/3_read_lmdb.py

# ###########################################################################################
# SCRIPT 4  (SOLVER AND TRAINING AND LEARNING CURVE)
echo "TRAINING. Remember that: <Epoch_index = floor((iteration_index * batch_size) / (# data_samples))>"
python $WORK_DIR/code/4_training.py -s ./models/$NET/m$MOD_NUM/solver_$MOD_NUM\_$NET.prototxt -l ./models/$NET/m$MOD_NUM/logfile_$MOD_NUM\_$NET.log

# print image of CNN architecture
echo "PRINT CNN BLOCK DIAGRAM"
python $CAFFE_TOOLS_DIR/python/draw_net.py $WORK_DIR/models/$NET/m$MOD_NUM/train_val_$MOD_NUM\_$NET.prototxt $WORK_DIR/models/$NET/m$MOD_NUM/bd_$MOD_NUM\_$NET.png

# ###########################################################################################
# SCRIPT 5: plot the learning curve
echo "PLOT LEARNING CURVERS"
python ./code/5_plot_learning_curve.py $WORK_DIR/models/$NET/m$MOD_NUM/logfile_$MOD_NUM\_$NET.log $WORK_DIR/models/$NET/m$MOD_NUM/plt_train_val_$MOD_NUM\_$NET.png
```