
Training miniVggNet by NVIDIA DIGITS (Xilinx UG1335)

Author: Alex He (ahe@xilinx.com)

Date: Mar-11-2019

Xilinx publish the Edge AI Tutorials to GitHub which show how to train NN model and then deploy the model by DNNDK. The project is at <https://github.com/Xilinx/Edge-AI-Platform-Tutorials> with scripts to help you. This doc give you another way to do the model training with NVIDIA/DIGITS which has a good visualization web interface. I create a docker image for easy deploy the NDVIDIA/DIGITS w/ Xilinx DNNDK.

The docker image is here.

<https://hub.docker.com/r/alexhegit/dnndk>

Please refer to this documentation to setup the docker container and start the DIGITS.

https://github.com/alexhegit/AlexTryMachineLearning/blob/master/CIFAR10_Caffe_Tutorial_UG1335/DNNDK_w_NvidiaDigits.md

Then I suppose your DIGITS service is running and just show how to train out a miniVggNet with CIFAR10.

1. Login the DIGITS

Input <http://localhost:5000> or replace the "localhost" with the IP of container if from remote machine in the web browser.

2. Create the CIFAR10 dataset in DIGITS

The CIFAR10 Caffe Tutorial (UG1335) have some scripts to download the CIFAR10 dataset by Keras. I suppose you have run the `1_write_cifar10_images.py` which will download the original images in a structure tree.

(https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-CIFAR10-Caffe/caffe/code/1_write_cifar10_images.py)

The originals images root directory should be `/root/ML/cifar10/input/cifar10_jpg/`. There have 10 subclass folders each class in train/ and val/ like bellow.

```

root@dnndk:~/ML/cifar10/input/cifar10_jpg# tree train/ -L 1
train/
|-- airplane
|-- automobile
|-- bird
|-- cat
|-- deer
|-- dog
|-- frog
|-- horse
|-- labels.txt
|-- ship
|-- train.txt
|-- truck

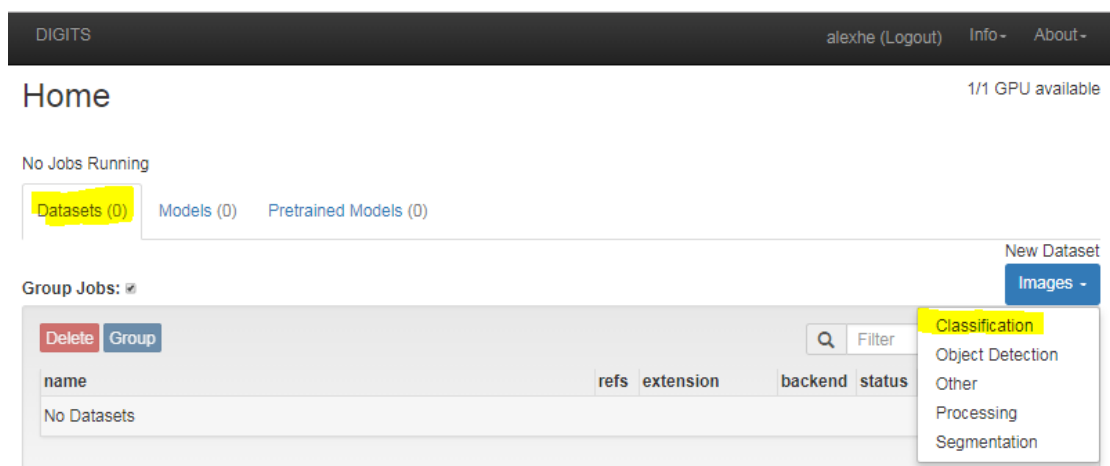
10 directories, 2 files
root@dnndk:~/ML/cifar10/input/cifar10_jpg# tree val/ -L 1
val/
|-- airplane
|-- automobile
|-- bird
|-- cat
|-- deer
|-- dog
|-- frog
|-- horse
|-- labels.txt
|-- ship
|-- test.txt
|-- truck

10 directories, 2 files

```

The DIGITS will use this structure image folder to create the training and valid dataset.

Click the “Dataset” label at top-left and Select the dataset type as “Classification”



Input the contents highlight in the UI snapshot bellow. The PATHs of Training Images and Validation Images may changed as real of your enviroments. You just need to input the root directory with subclass folder for each class in it.

New Image Classification Dataset

Image Type ⓘ

Color

Image size (Width x Height) ⓘ

32 x 32

Resize Transformation ⓘ

Squash

See example

Use Image Folder Use Text Files Use S3

Training Images ⓘ

/root/ML/cifar10/input/cifar10_jpg/train

Minimum samples per class ⓘ

Maximum samples per class ⓘ

% for validation ⓘ

25

% for testing ⓘ

0

☒ Separate validation images folder

Validation Images

/root/ML/cifar10/input/cifar10_jpg/val/

Minimum samples per class ⓘ

Maximum samples per class ⓘ

☐ Separate test images folder

DB backend

LMDB

Image Encoding ⓘ

PNG (lossless)

Group Name

Dataset Name

cifar10_32x32

Create

The dataset will be created as LMDB format.

DIGITS
Image Classification Dataset
alexhe (Logout)
Info
About

cifar10_32x32
Owner: alexhe

Clone Job
Delete Job

Job Information

Job Directory
/root/digits/digits/jobs/20190311-110230-f84c

Image Dimensions
32x32 (Width x Height)

Image Type
Color

Resize Transformation
Squash

DB Backend
Imdb

Image Encoding
png

DB Compression
none

Dataset size
0 B

Job Status Done

- Initialized at 11:02:30 AM (1 second)
- Running at 11:02:31 AM (1 minute, 6 seconds)
- Done at 11:03:37 AM (Total - 1 minute, 7 seconds)

Parse Folder (train) Done

Parse Folder (val) Done

Create DB (train) Done

Create DB (val) Done

Notes

None

Parse Folder (train)

Folder
/root/ML/cifar10/input/cifar10_jpg/train

Parse Folder (val)

Folder
/root/ML/cifar10/input/cifar10_jpg/val/

Create DB (train)

Input File (before shuffling)
train.txt

DB Creation log file
create_train_db.log

Image Mean:

Explore the db

Create DB (val)

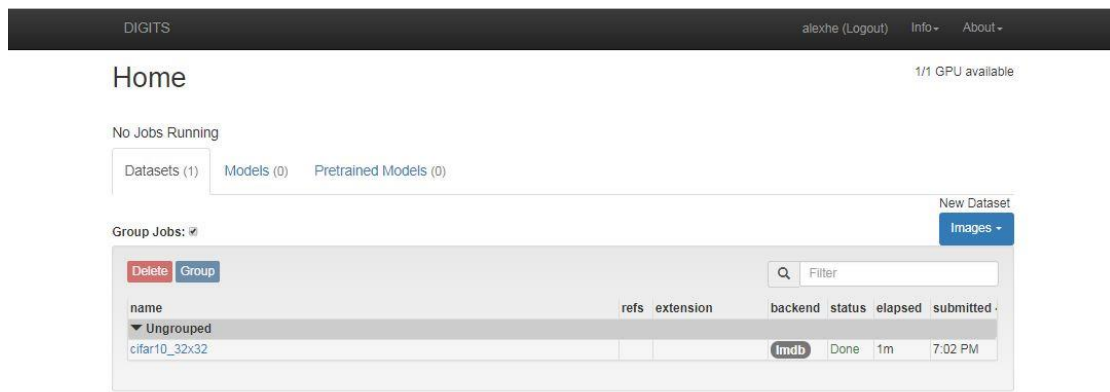
Input File (before shuffling)
val.txt

DB Creation log file
create_val_db.log

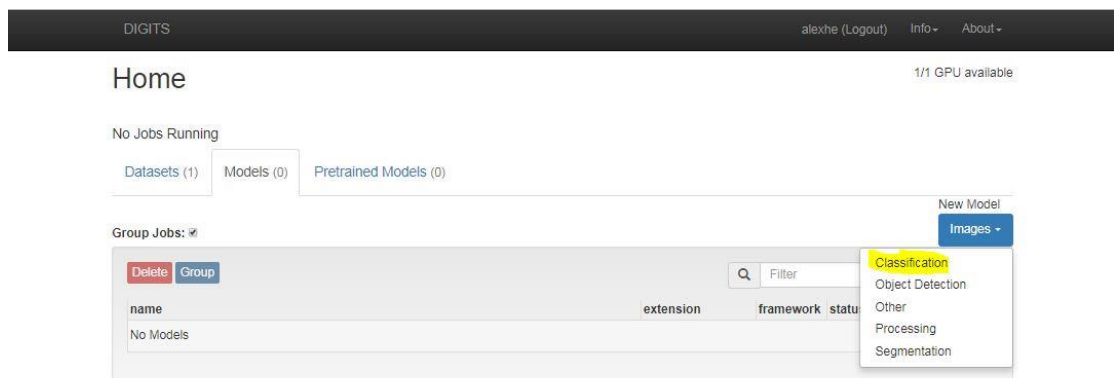
Explore the db

Now we have the cifar10_32x32 in the Datasets label.

© Copyright 2019 Xilinx



3. Create the miniVggNet model
- Click the “Models” label and chose the “Classification” type



Then

- Select the cifar10_32x32 dataset we created at previous step.
- Set the hyper-parameters of solver as you want
- Chose Custom Network
- Name the model

New Image Classification Model

Select Dataset ⓘ

cifar10_32x32

cifar10_32x32

Done 11:03:37 AM

Image Size
32x32

Image Type
COLOR

DB backend
Imdb

Create DB (train)
50000 images

Create DB (val)
9000 images

Python Layers ⓘ

Server-side file ⓘ

☐ Use client-side file

Solver Options

Training epochs ⓘ
30

Snapshot interval (in epochs) ⓘ
1

Validation interval (in epochs) ⓘ
1

Random seed ⓘ
[none]

Batch size ⓘ multiples allowed
[network defaults]

Batch Accumulation ⓘ

Solver type ⓘ
SGD (Stochastic Gradient Descent)

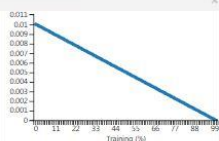
Base Learning Rate ⓘ multiples allowed
0.01

☒ Show advanced learning rate options

Policy
Polynomial Decay

Power
1

Visualize LR



Data Transformations

Subtract Mean ⓘ
Image

Crop Size ⓘ
none

Standard Networks Previous Networks Pretrained Networks **Custom Network**

Caffe Tensorflow

Custom Network ⓘ Visualize

1 |

Pretrained model(s) ⓘ

Group Name ⓘ

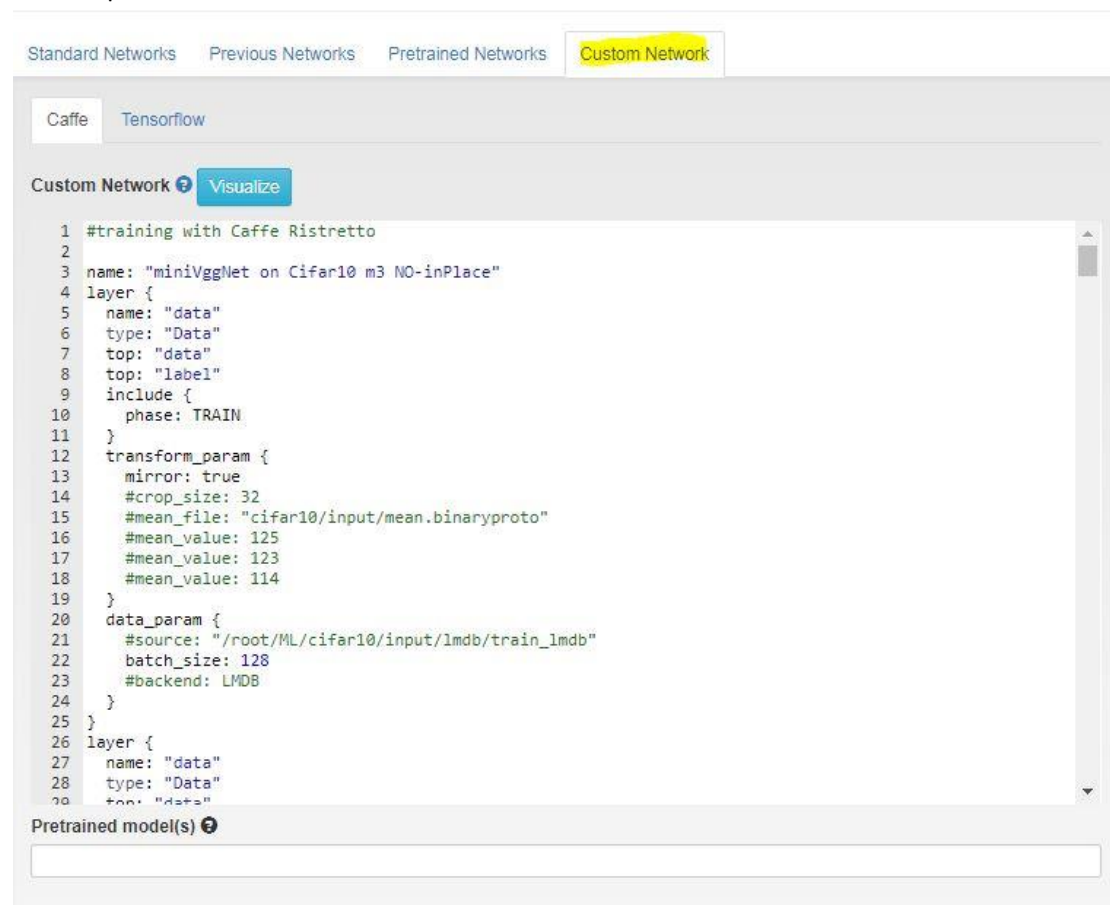
Model Name ⓘ

minVGGNet_scratch_01

Create

Define your network in prototxt format. Here is the sample of miniVggNet https://github.com/alexhegit/AlexTryMachineLearning/blob/master/CIFAR10_Caffe_Tutorial_U61335/digits_train_val_3_miniVggNet.prototxt.

You can paste the content here.



4. Train your miniVggNet

Click the "Create" button to start the training. Below is the snapshot of my training result. You can watch the accuracy/loss/learning rate/time/GPU statistics, etc of training process and results.

DIGITS
Image Classification Model
alexhe (Logout)
Info -
About -

miniVGGNet_scratch_01

Owner: alexhe

Clone Job
Delete Job

Job Directory
/root/digits/digits/jobs/20190311-111303-af39
Disk Size
0 B
Network (train/val)
train_val.prototxt
Network (deploy)
deploy.prototxt
Network (original)
original.prototxt
Solver
solver.prototxt
Raw caffe output
caffe_output.log

Dataset
cifar10_32x32
Done 11:03:37 AM
Image Size
32x32
Image Type
COLOR
DB backend
lmdb
Create DB (train)
50000 images
Create DB (val)
9000 images

Job Status Done

- Initialized at 11:13:03 AM (1 second)
- Running at 11:13:04 AM (3 minutes, 39 seconds)
- Done at 11:16:44 AM (Total - 3 minutes, 40 seconds)

Train Caffe Model Done -

Related jobs
Image Classification Dataset
cifar10_32x32 Done

Notes
None

View Large

Trained Models
Select Model
Epoch #30
Download Model
Make Pretrained Model
Publish to inference server

Test a single image
Image Path

Upload image
Browse...
☐ Show visualizations and statistics
Classify One

Test a list of images
Upload Image List
Browse...
Accepts a list of filenames or urls (you can use your val.txt file)
Image folder (optional)

Relative paths in the text file will be prepended with this value before reading
Number of images use from the file
All
Leave blank to use all
Classify Many
Number of images to show per category
9
Top N Predictions per Category

5. Test the model

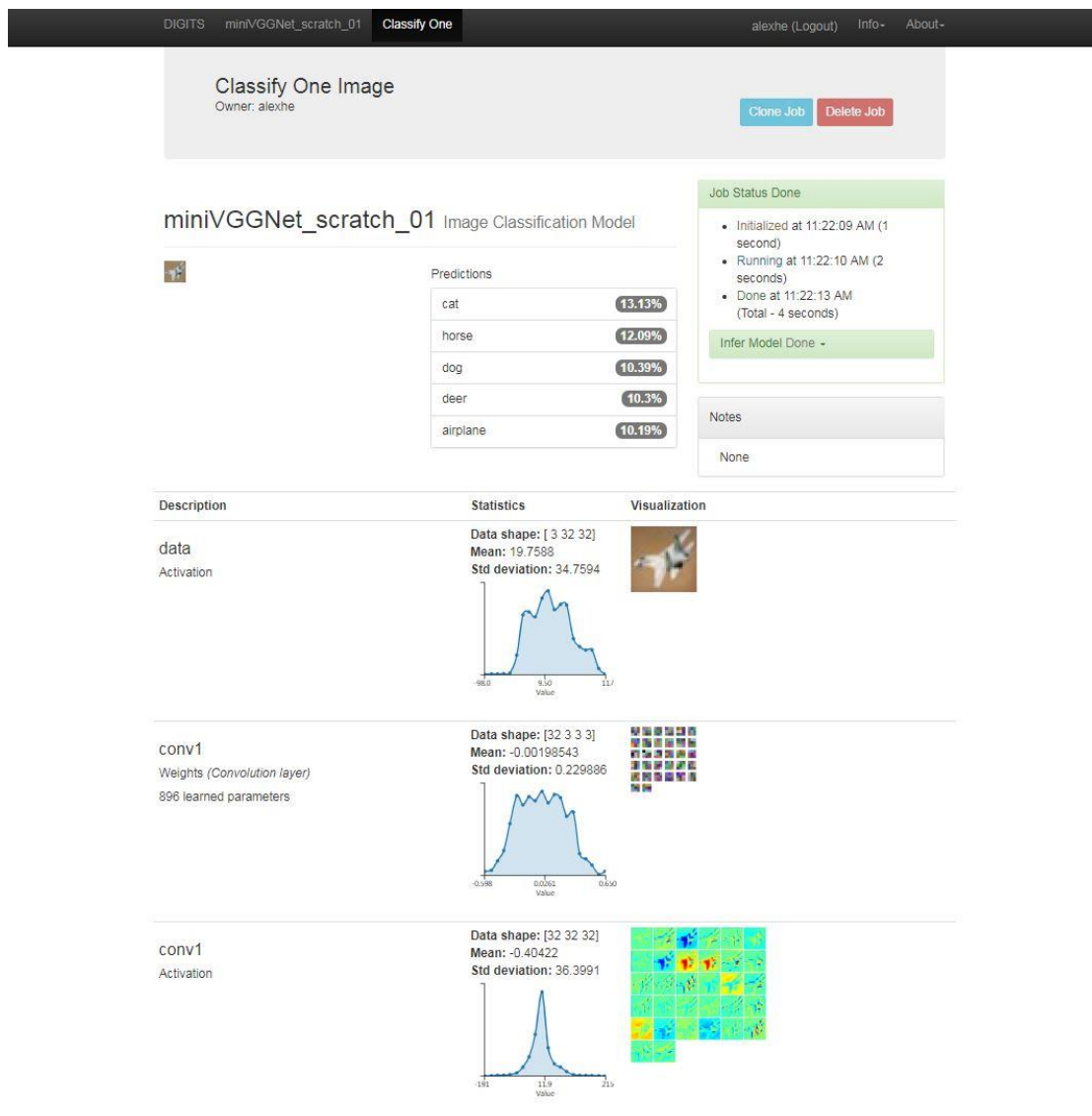
After the training fished, you can test it in the same UI in previous step. DIGITS provide two

type tests. Let me show you the “Test a single image”.

Input the Image Path and select “Show visualizations and statistics” if want.

The screenshot shows a web interface titled "Trained Models". Under the "Select Model" section, there is a dropdown menu set to "Epoch #30" and three buttons: "Download Model" (blue), "Make Pretrained Model" (green), and "Publish to inference server" (green). The interface is divided into two main columns. The left column, "Test a single image", includes an "Image Path" field with a help icon and a text input containing "/root/ML/cifar10/input/cifar10_jpg/test/ai", an "Upload image" section with a "Browse..." button, a checked checkbox for "Show visualizations and statistics" with a help icon, and a "Classify One" button. The right column, "Test a list of images", includes an "Upload Image List" section with a "Browse..." button and explanatory text, an "Image folder (optional)" field, a "Number of images use from the file" dropdown set to "All" with a note "Leave blank to use all", a "Classify Many" button with a help icon, a "Number of images to show per category" input set to "9", and a "Top N Predictions per Category" button with a help icon.

You will see the details when image data pass through the Neural-Network as well as the predictions result.



You can chose the best model snapshot to be download for DNNDK deploy process as UG1335.

