Signal Processing using DIGITS

Hanbyul Yang
Solutions Architect
NVIDIA Corporation

DEEP LEARNING INSTITUTE

DLI Mission

Helping people solve challenging problems using AI and deep learning.

- Developers, data scientists and engineers
- Self-driving cars, healthcare and robotics
- Training, optimizing, and deploying deep neural networks

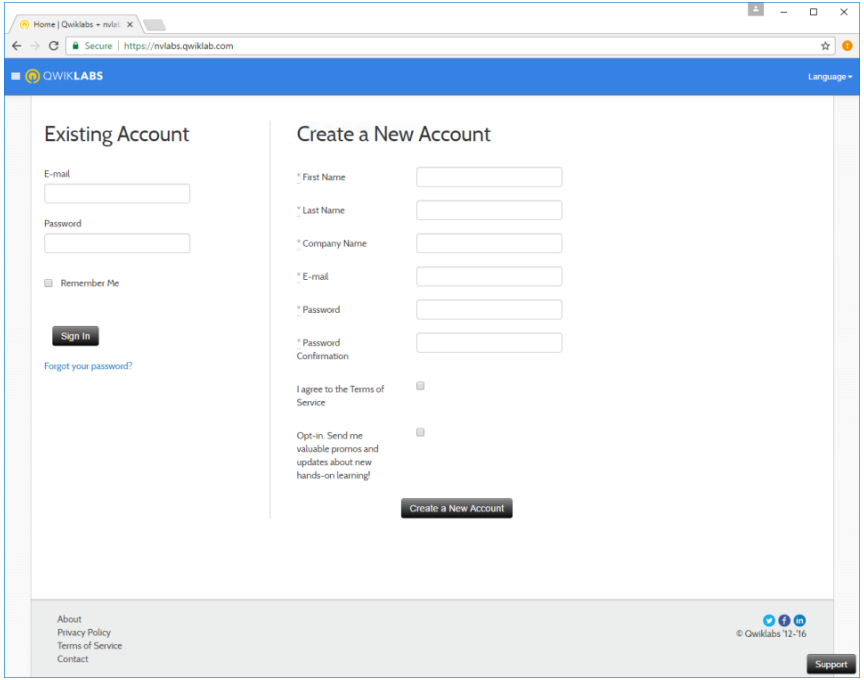
TOPICS

- Lab Perspective
- Signal detection
- Lab
  - Discussion / Overview
  - Launching the Lab Environment
  - Lab Review

NAVIGATING TO QWIKLABS

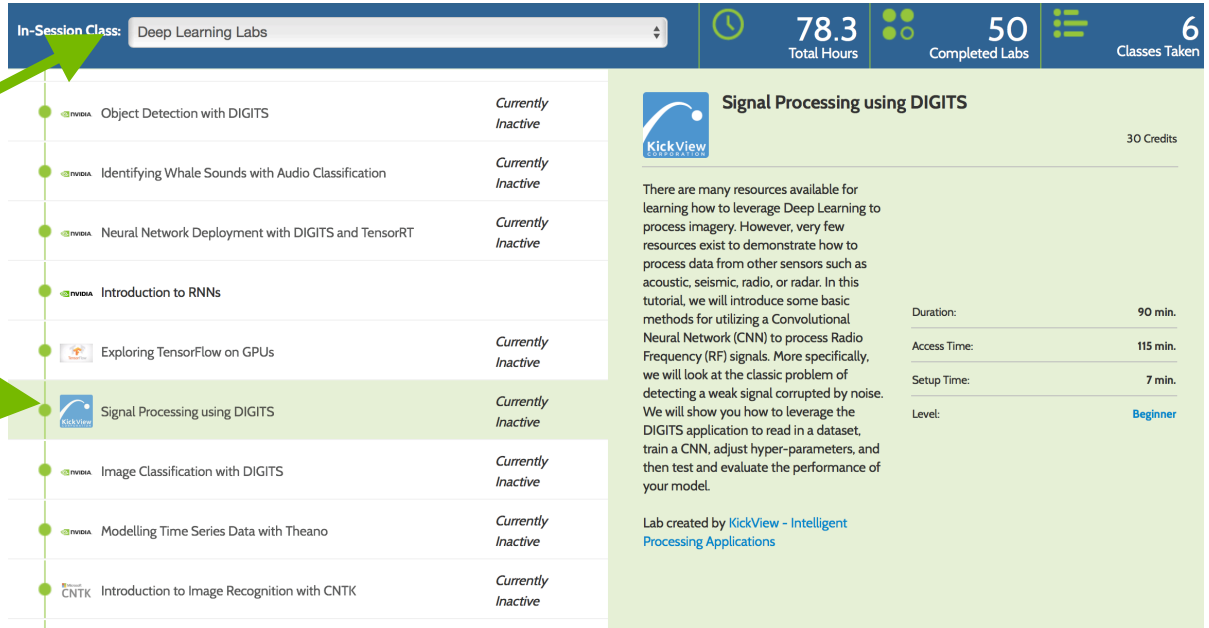
1. Navigate to: https://nvlabs.qwiklab.com

1. Login or create a new account



ACCESSING LAB ENVIRONMENT

- Select the event specific In-Session Class in the upper left
- Click the “Signal Processing using DIGITS” Class from the list



LAB PERSPECTIVE

WHAT THIS LAB IS

- Introduction of utilizing a convolutional neural network to process Radio Frequency (RF) signals with images.

- Hands-on exercises using DIGITS and Caffe for CNN training.

WHAT THIS LAB IS NOT

- Intro to machine learning from first principles
- Rigorous mathematical formalism of convolutional neural networks
- Survey of all the features and options of Caffe

ASSUMPTIONS

- You are familiar with convolutional neural networks (CNN)
- Helpful to have:
  - Signal processing experience
  - Caffe experience
  - Python experience

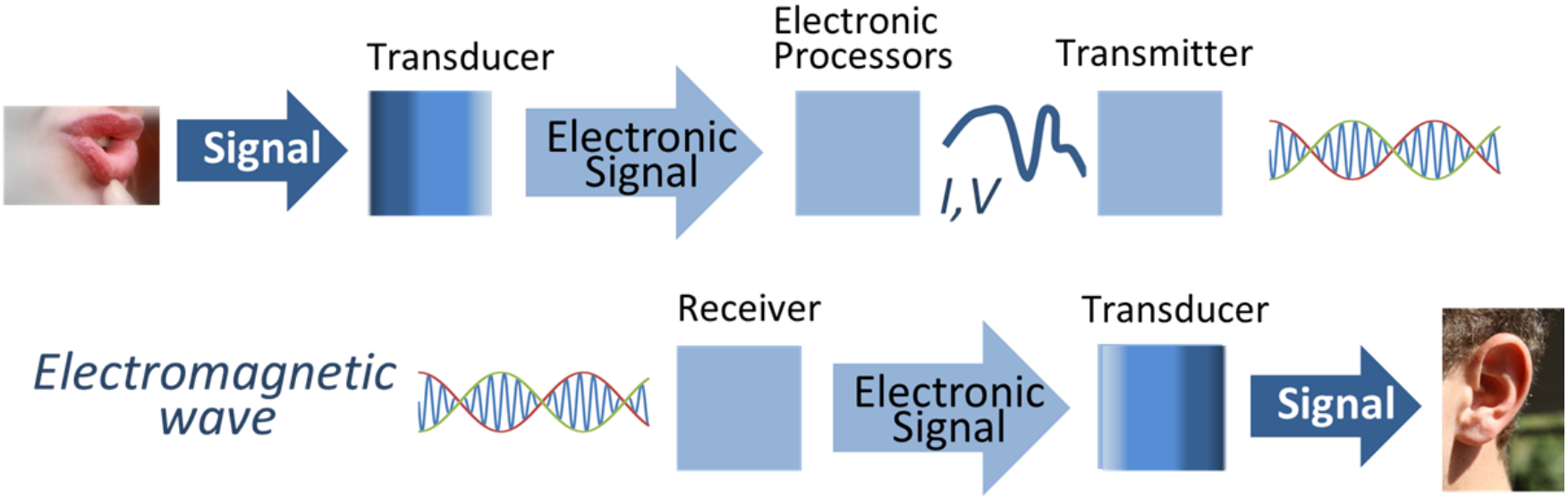
TAKE AWAYS

- You can setup your own signal processing workflow in DIGITS and adapt it to your use case
- Know where to go for more info
- Familiarity with DIGITS

INTRO TO SIGNAL DETECTION

Signal processing

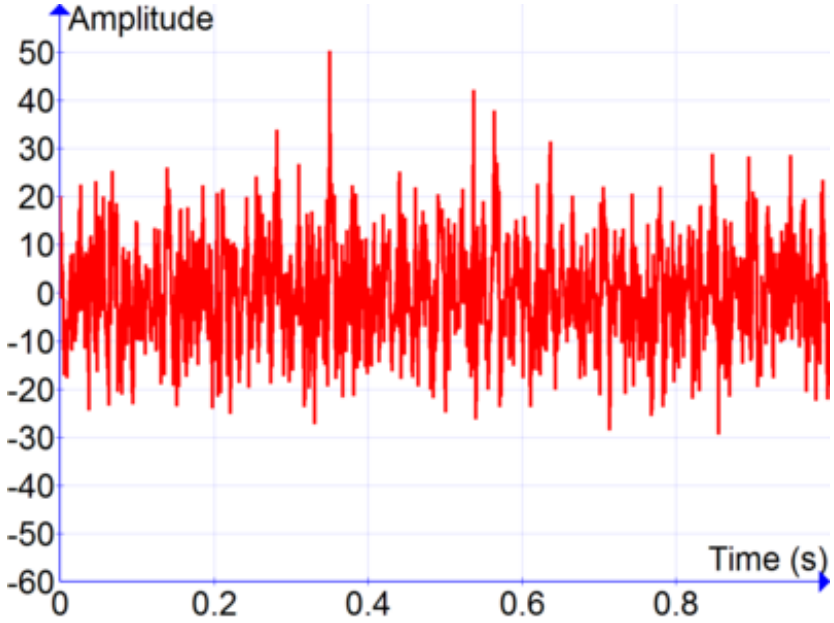
Example of audio signal processing

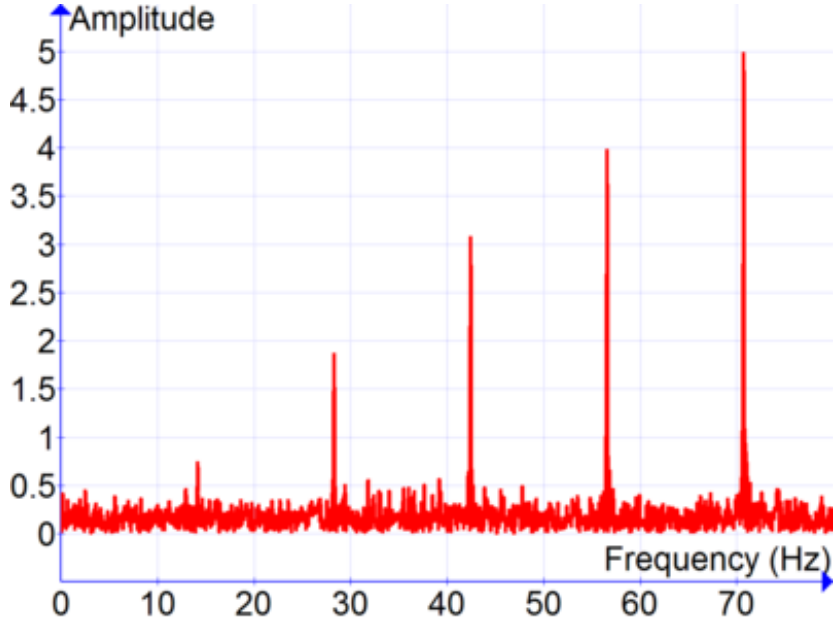


Ref : https://en.wikipedia.org/wiki/Signal_processing

Signal processing

Fourier transform





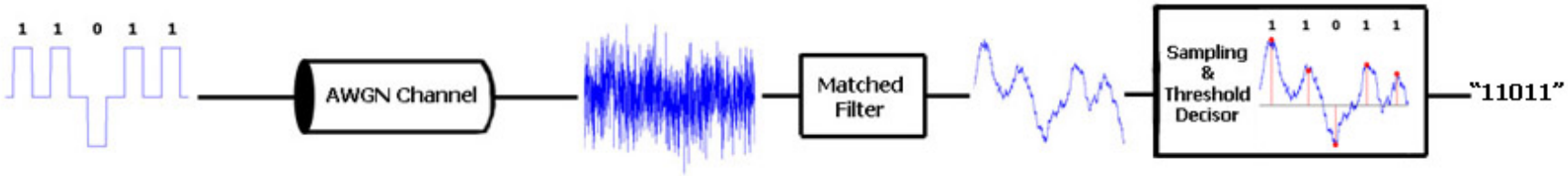
Ref : https://en.wikipedia.org/wiki/Signal_processing

INTRO TO SIGNAL DETECTION

- Detecting certain signal identifying features in signals is challenging due to noise.
- Traditional signal detection methods
  - Matched filtering
  - Correlation-based techniques

SIGNAL DETECTION

Matched filter in digital communications



Ref : https://en.wikipedia.org/wiki/Matched_filter

INTRO TO SIGNAL DETECTION

A little background information

- Assume signal is corrupted with additive white Gaussian noise (AWGN).
- Depend on signal duration, amplitude, corresponding noise process.
- Weak signals with short duration is most difficult to detect.
- More difficult If interfering signals are in same band as the signal for detection.
- Real-world signals often have frequency components that changed with time.
  - Linear Frequency-Modulated (LFM) signals

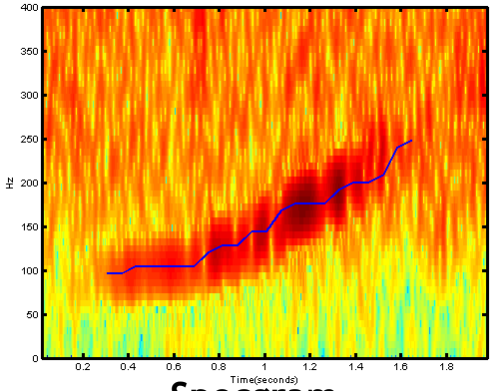
IN THIS LAB…

A little background information

- Simple approach to detecting signals in noise using a Convolutional Neural Network(CNN)
- Will utilize **spectrograms**.
  - 2D representation of signal.
    (x-axis is time, y-axis is frequency)

Waveform

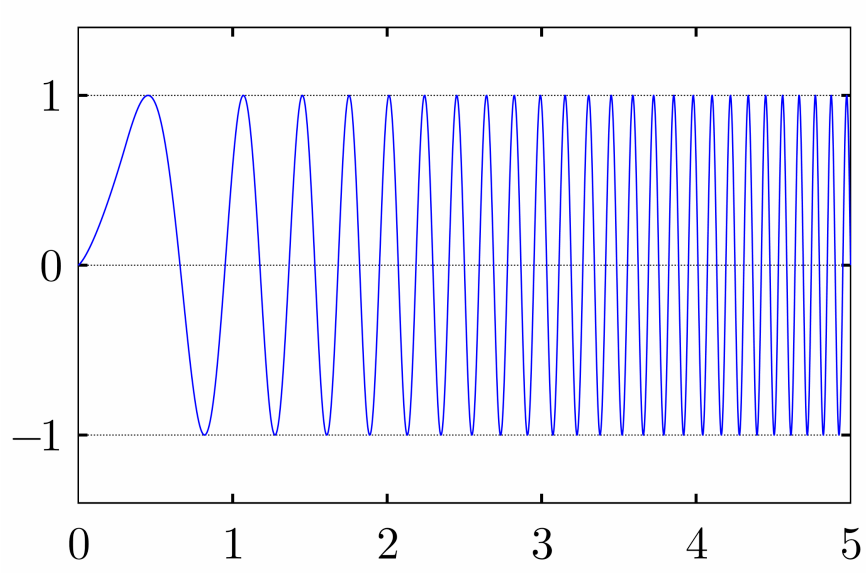
FFT



Specgram

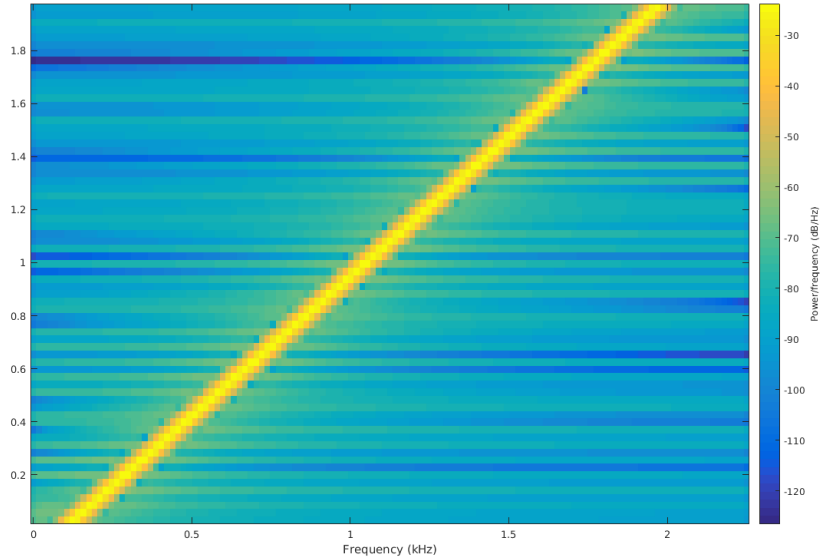
CHIRP

Linear Frequency-Modulated Signals



x-axis : time, y-axis : amplitude

$$x(t) = \sin\left[\phi_0 + 2\pi\left(f_0 t + \frac{k}{2}t^2\right)\right]$$



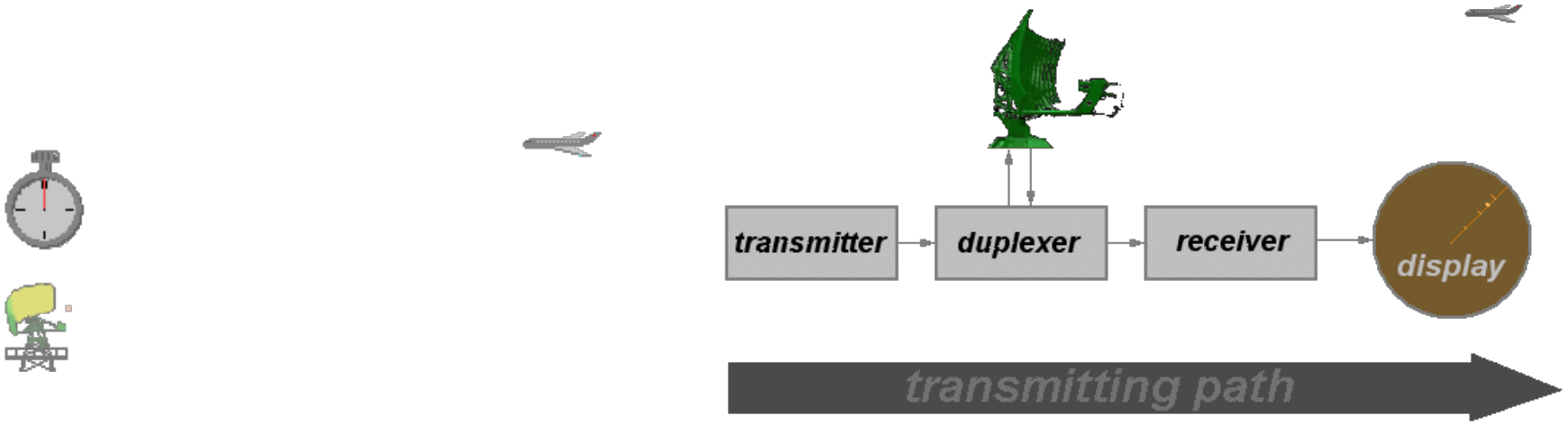
x-axis : time, y-axis : frequency

$$f(t) = f_0 + kt$$

Ref :https://en.wikipedia.org/wiki/Chirp

RADAR PRINCIPLE

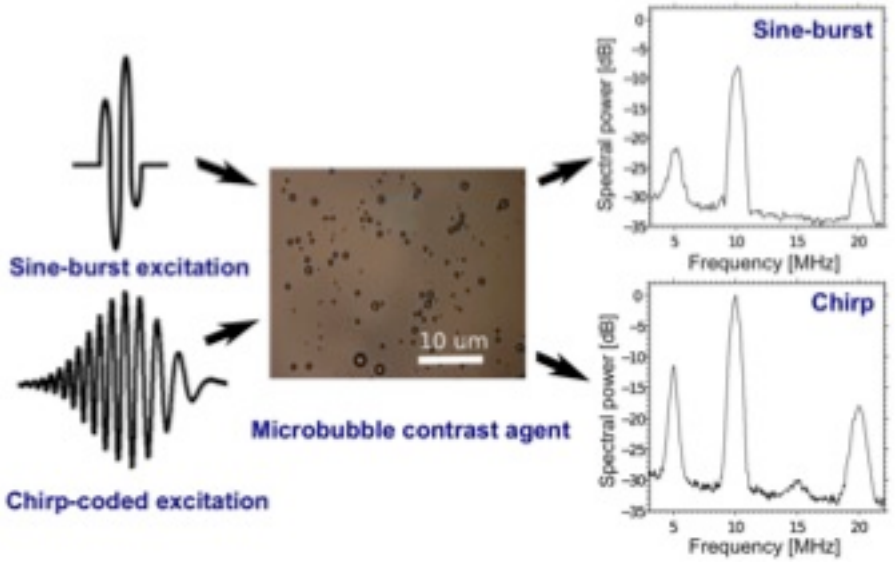
RAdio (Aim) Detecting And Ranging



Ref : http://www.radartutorial.eu/01.basics/Radar%20Principle.en.html

CHIRP EXAMPLE

Ultrasound imaging

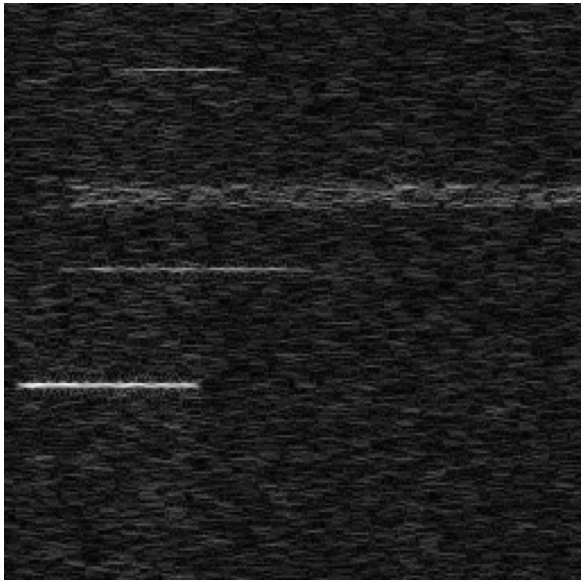


w/o chirp w/ chirp

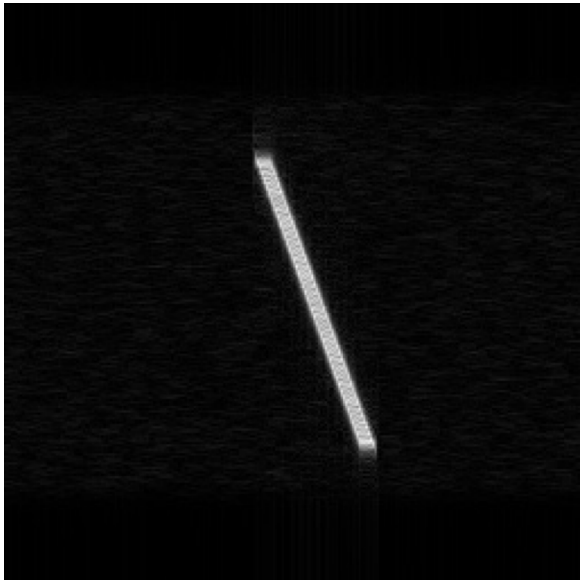
Ref : *http://www.ece.rochester.edu/projects/doyley_lab/research/hifu.html*
http://research.vuse.vanderbilt.edu/beamlab/chirp/default.html

SPECTROGRAM

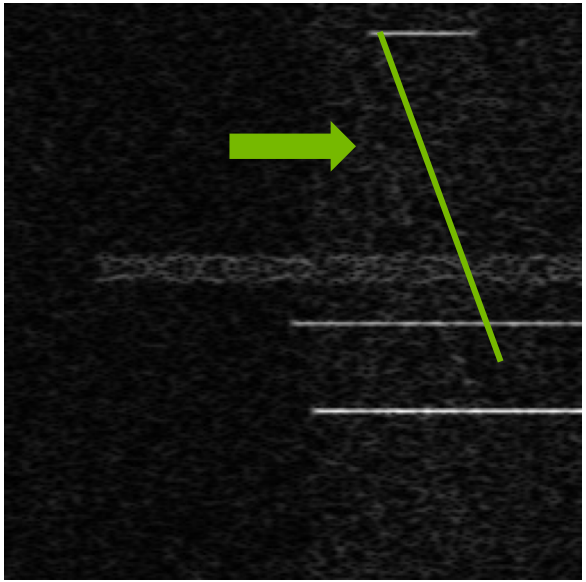
Chirp (Linear Frequency-Modulated) Signals



Multiple signals and noise
without chirp
(x-axis is time, y-axis is frequency)



High-SNR
with chirp



Weak chirp embedded
in noise with other signals
(SNR -7dB)

LAB DISCUSSION / OVERVIEW

DATA DETAILS

- Original images are 255 x 255 grayscale JPEG

- Dataset
- Train set - 10000 images (5000 for pos, 5000 for neg)
  - Training set consist of 8500 images
  - Validation set consist of 1500 images
- Test set – 2000 images (1000 for pos, 1000 for neg)
- Fine tune training set – 1000 images (500 for pos, 500 for neg)

DATA DETAILS

POS

posex_2.08db_4890.jpg
posex_2.09db_1019.jpg
posex_2.09db_1101.jpg
posex_2.09db_1575.jpg
posex_2.09db_1781.jpg

NEG

negex_1894.jpg
negex_1895.jpg
negex_1896.jpg
negex_1897.jpg
negex_1898.jpg
negex_1899.jpg

posex_**2.08db**_4890.jpg

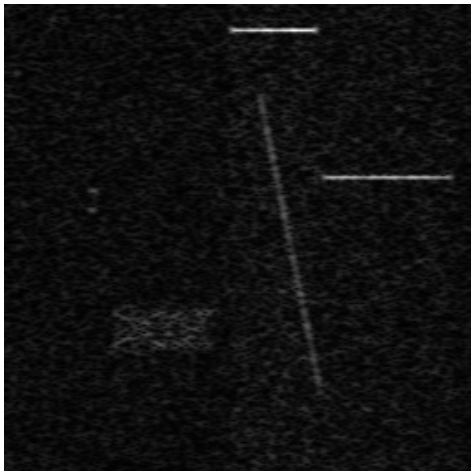
Signal-to-noise Ratio

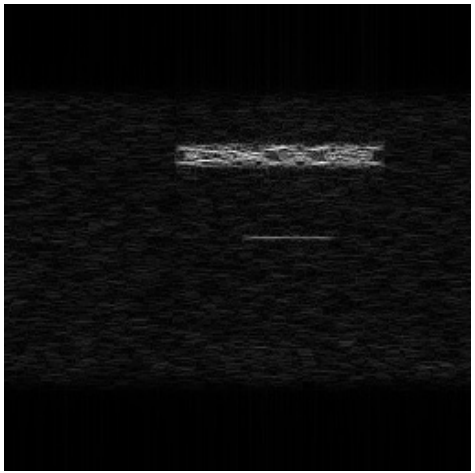
-2.08dB

TASK 1

Deep Spectral Detection: Data and Network Creation

- Two-output convolutional neural network
  - Determine whether a chirp signal is present (pos) or not (neg).
- Time-frequency spectrograms for training





TASK 1

Deep Spectral Detection: Data and Network Creation

- Visualize your model
  - AlexNet and pruned the number of fully-connected layers to 2
  - The two fully connected layers were also reduced in size (fewer neurons).
- Check your model works
  - Tuning your model and re-train
  - Quick sanity check with single image.

TASK 2

Neural Network Generalization on New Signals

- Using a test dataset
  - 1000 positive and 1000 negative examples
  - Be used to determine the generalization capability of the trained network
- SNR values (included in file name) are used in the test set in order to analyze the network's ability to discriminate low and high SNR signals
- Metrics
  - probability of detection (PD)
  - probability of false alarm (PFA)

TASK 2

Neural Network Generalization on New Signals

- Using trained network model
  - Use pre-installed trained network model.
  - (or optionally) Use your own trained network model just before.
- Using Python analysis script
  - Note: Processing may take a few minutes, so be patient for the output to display.
  - `detection_tst() , false_alarm_tst()`
  - The functions take in arguments: positive/negative image file directory, caffe model directory, pic type ('jpg' or 'png')

TASK 3

Fine tuning the network model

- Fine tuning the network model
  - Transfer learning
  - Additional training dataset with very low SNRs, -4 ~ -8 dB
- Test your fine-tuned network model with python script
  - Use pre-installed network model.
  - Or (Optionally) Make new dataset and train with additional dataset.

LAB REVIEW

CREATING DATASET

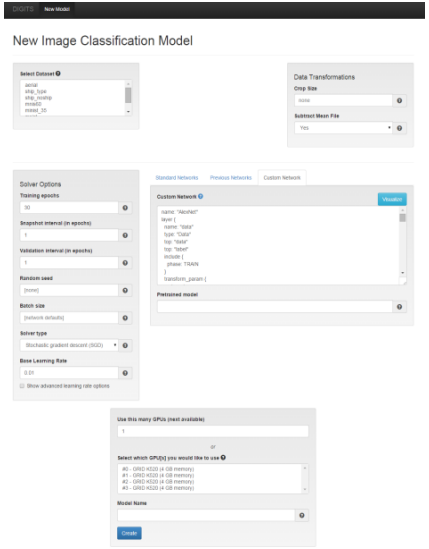
- Login : Use lower case letters.
- Dataset settings
  - Image Type : **Grayscale**
  - Image Size : 256 x 256
  - Resize Transformation : Crop
  - Training Images: **/home/ubuntu/demo/traindemo**
  - % for validation : **15**
  - Dataset Name : chirp_data_train

NVIDIA'S DIGITS

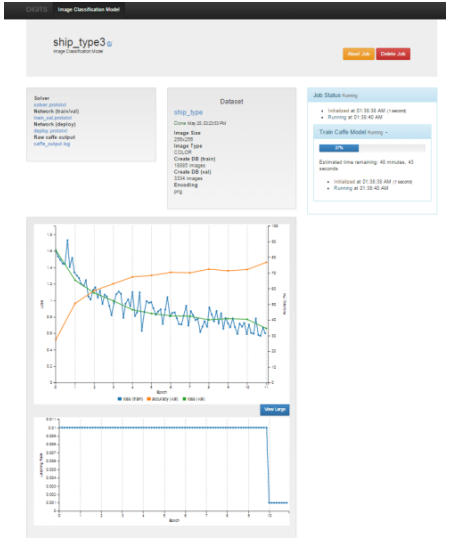
Interactive Deep Learning GPU Training System

Process Data

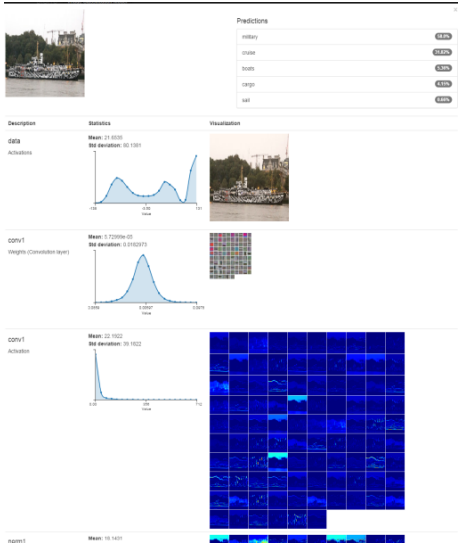
Configure DNN

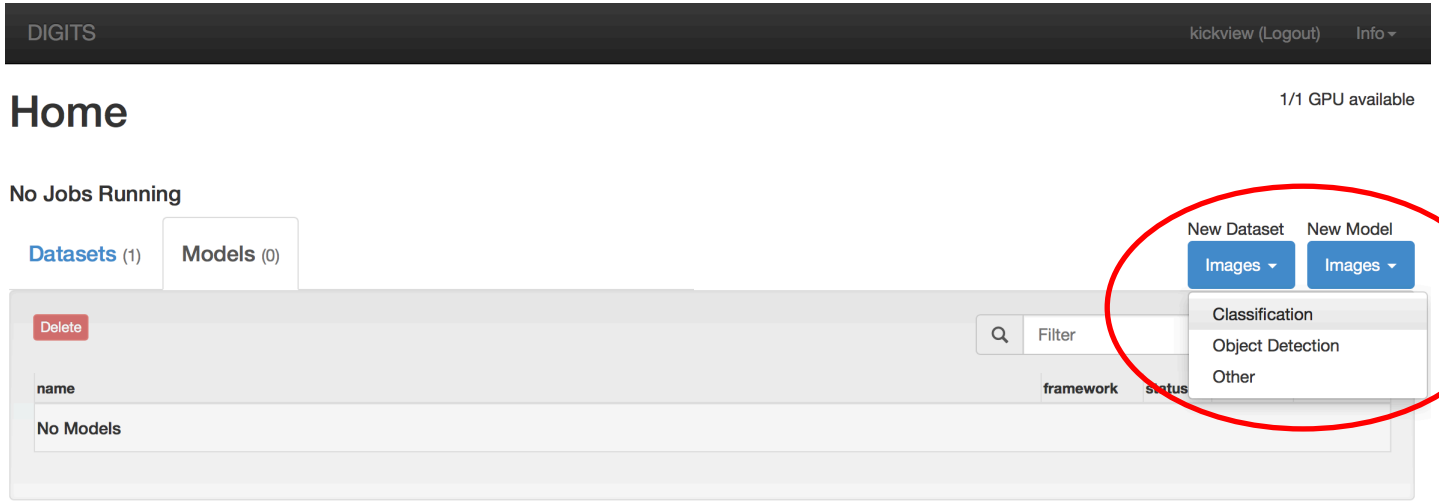


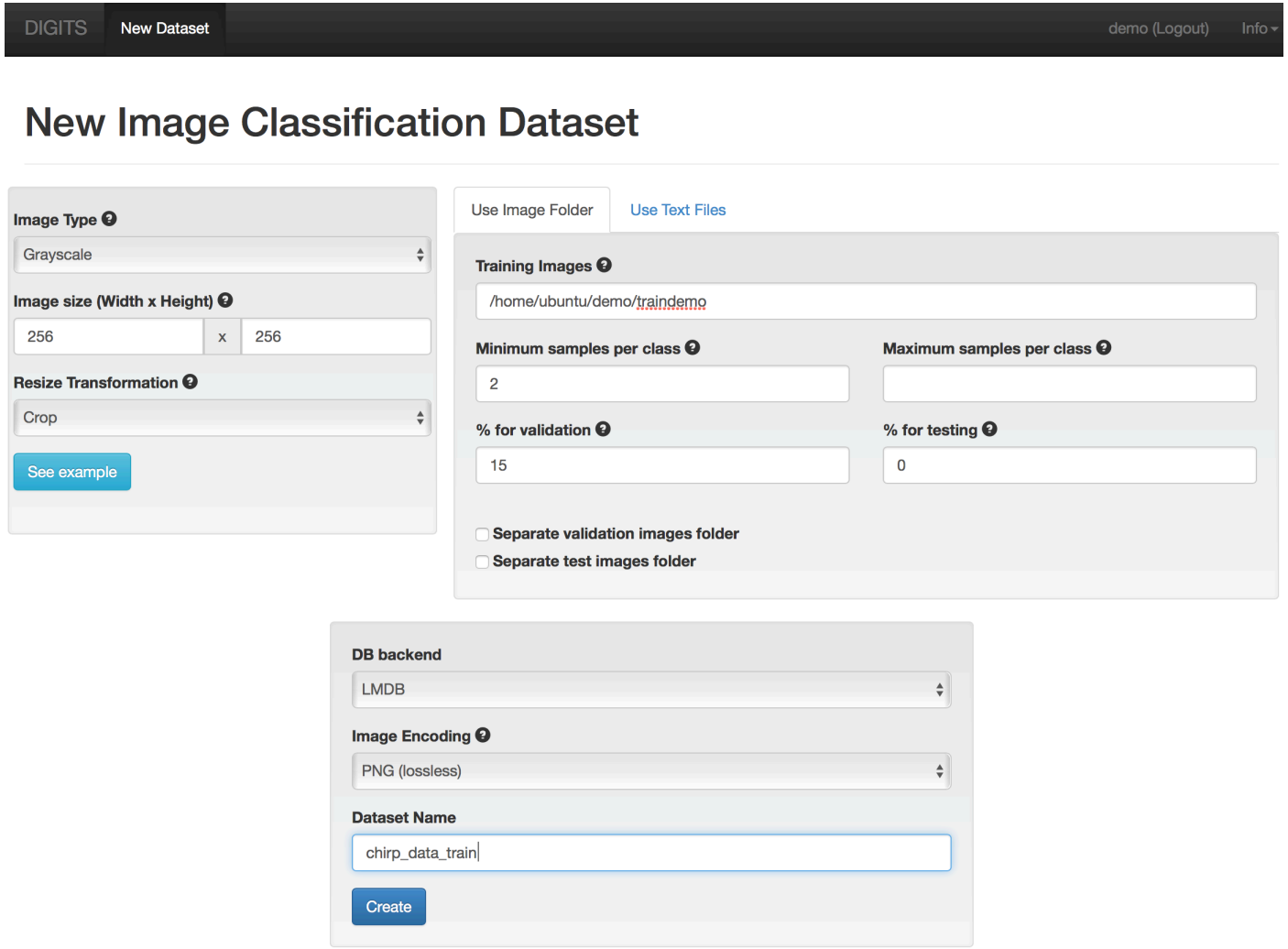
Monitor Progress



Visualization




DIGITS
kickview (Logout)
Info
Home
1/1 GPU available
No Jobs Running
Datasets (1)
Models (0)
New Dataset
New Model
Images
Images
Classification
Object Detection
Other
Delete
Filter
name
framework
No Models


DIGITS
New Dataset
demo (Logout)
Info
New Image Classification Dataset
Image Type
Grayscale
Image size (Width x Height)
256
x
256
Resize Transformation
Crop
See example
Use Image Folder
Use Text Files
Training Images
/home/ubuntu/demo/traindemo
Minimum samples per class
2
Maximum samples per class
% for validation
15
% for testing
0
Separate validation images folder
Separate test images folder
DB backend
LMDB
Image Encoding
PNG (lossless)
Dataset Name
chirp_data_train
Create

DIGITS | Image Classification Dataset | demo (Logout) | Info

chirp_data_train

Owner: demo

Clone Job | Abort Job | Delete Job

Job Information

Job Directory
/home/ubuntu/digits-4.0/digits/jobs/20160826-164208-0ab7

Image Dimensions
256x256 (Width x Height)
Image Type
Grayscale
Resize Transformation
Crop
DB Backend
lmdb
Image Encoding
png
DB Compression
none

Dataset size
0 B

Parse Folder (train/val)

Folder
/home/ubuntu/demo/traindemo

Job Status Running

- Initialized at 04:42:08 PM (1 second)
- Running at 04:42:09 PM

Parse Folder (train/val) Running

Create DB (train) Running

69%

Estimated time remaining: 7 seconds

- Initialized at 04:42:08 PM (2 seconds)
- Running at 04:42:10 PM

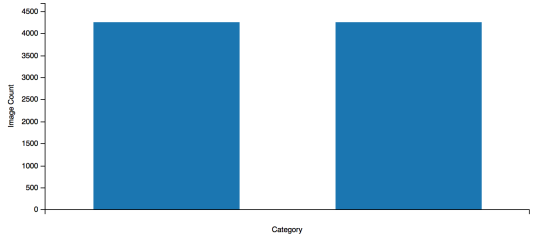
Create DB (val) Running

Notes

None

Create DB (train)

Input File (before shuffling)
train.txt
DB Creation log file
create_train_db.log



Job Directory
/home/ubuntu/digits-4.0/digits/jobs/20160826-164208-0ab7

Image Dimensions
256x256 (Width x Height)
Image Type
Grayscale
Resize Transformation
Crop
DB Backend
lmdb
Image Encoding
png
DB Compression
none

Dataset size
0 B

Folder
/home/ubuntu/demo/traindemo

- Initialized at 04:42:08 PM (1 second)
- Running at 04:42:09 PM (22 seconds)
- Done at 04:42:32 PM
(Total - 23 seconds)

Parse Folder (train/val) Done

Create DB (train) Done

Create DB (val) Done

Notes

None

Create DB (train)

Input File (before shuffling)
train.txt
DB Creation log file
create_train_db.log

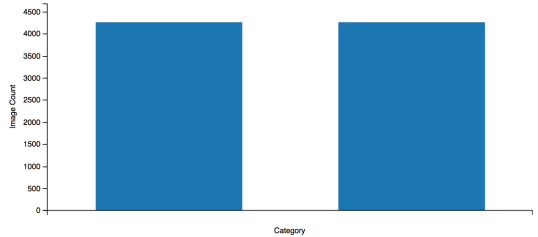
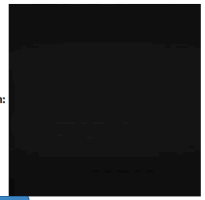


Image Mean:

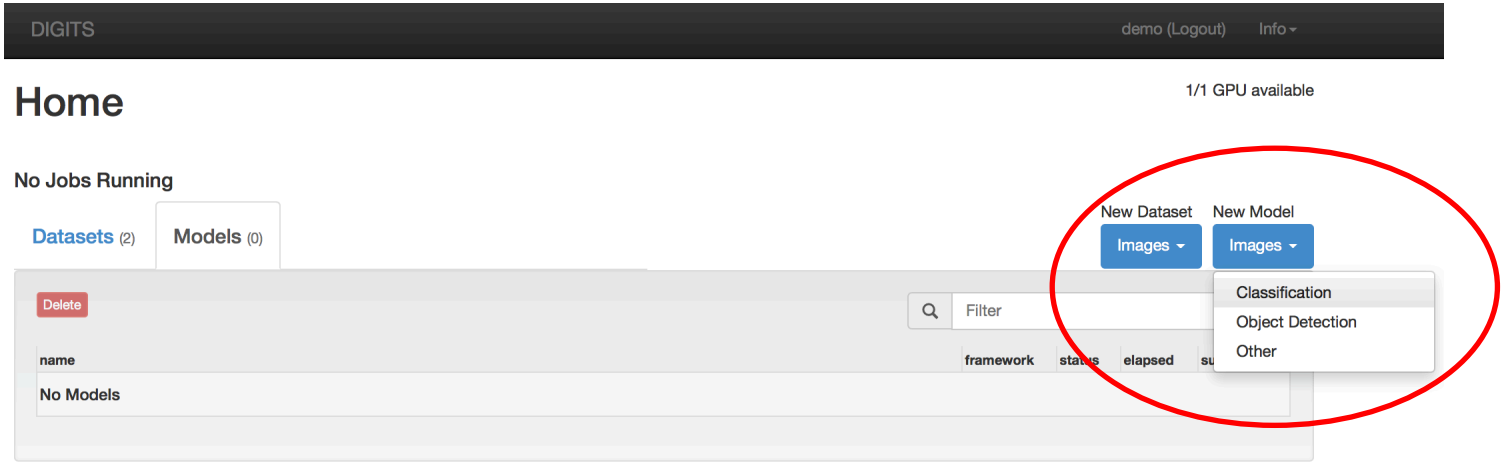


Explore the db

CREATING MODEL

- Select the “**chirp_data_train**” dataset
- Set the number of “**Training Epochs**” to **5**
- Select the solver type of “**Nesterov’s accelerated gradient (NAG)**”
  - Policy “**Exponential Decay**”
- Set the base learning rate of “**0.001**”
- Set the model to “**Custom Network**”, input chirp CNN prototxt
  - if not available, go to https://github.com/NVIDIA-Korea/DLI
- Set the name of the model to “**chirp_demo**”

NEW CNN MODEL CREATION



New Image Classification Model

Select Dataset

chirp_data_train

chirp_data_train

Done 04:38:52 PM

Image Size
256x256
Image Type
GRAYSCALE
DB backend
lmdb
Create DB (train)
8500 images
Create DB (val)
1500 images

Python Layers

Server-side file

Use client-side file

Solver Options

Training epochs
5

Snapshot interval (in epochs)
1

Validation interval (in epochs)
1

Random seed
[none]

Batch size multiples allowed
32

Batch Accumulation

Solver type
Nesterov's accelerated gradient (NAG)

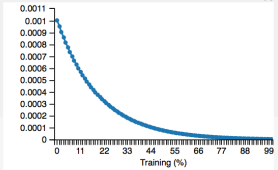
Base Learning Rate multiples allowed
0.001

Show advanced learning rate options

Policy
Exponential Decay

Gamma
0.95

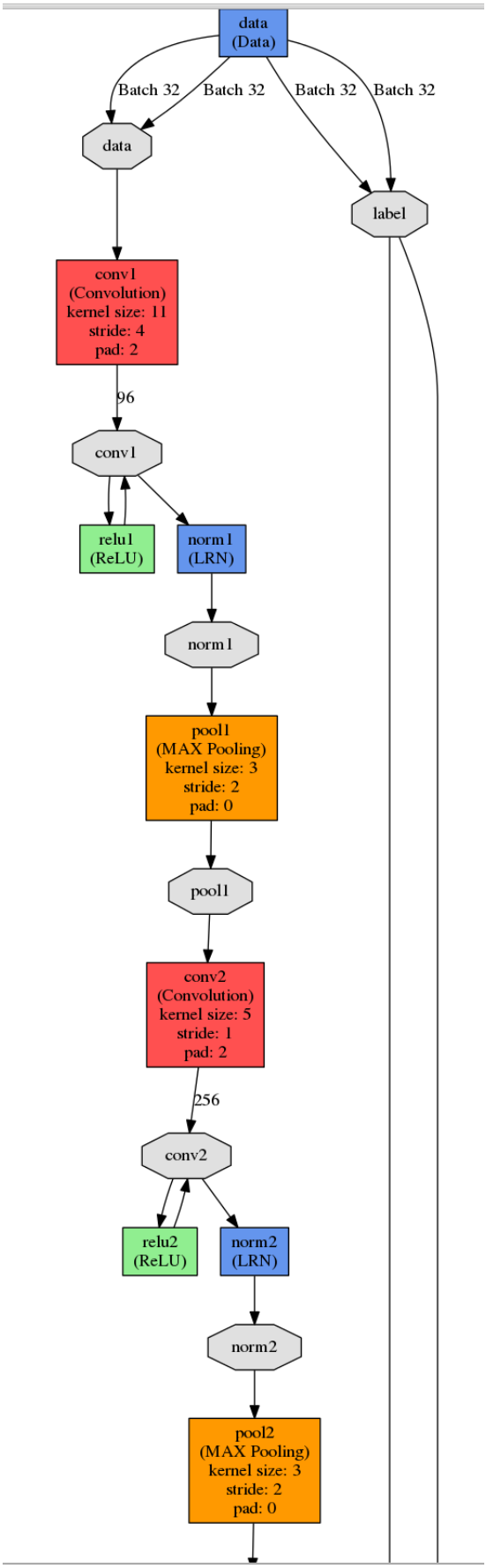
Visualize LR



Data Transformations

Crop Size
none

Subtract Mean
Image


data
(Data)
Batch 32
Batch 32
Batch 32
Batch 32
data
label
conv1
(Convolution)
kernel size: 11
stride: 4
pad: 2
96
conv1
relu1
(ReLU)
norm1
(LRN)
norm1
pool1
(MAX Pooling)
kernel size: 3
stride: 2
pad: 0
pool1
conv2
(Convolution)
kernel size: 5
stride: 1
pad: 2
256
conv2
relu2
(ReLU)
norm2
(LRN)
norm2
pool2
(MAX Pooling)
kernel size: 3
stride: 2
pad: 0

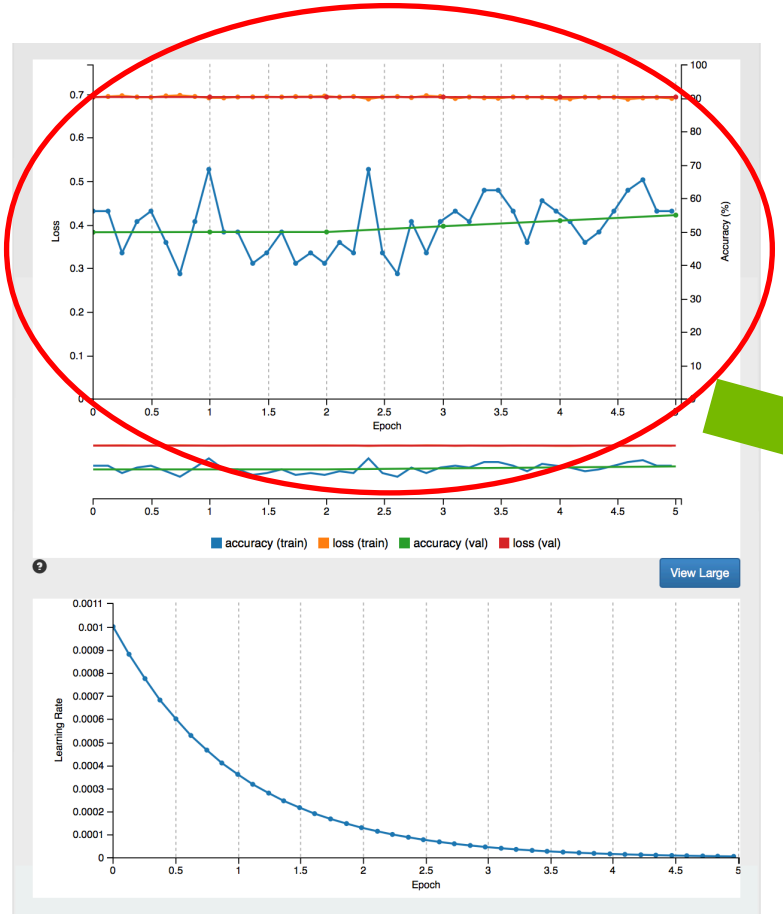


Image Classification Model

chirp_demo Done

chirp_demo Running

Notes

None

Loss does not decrease.
Accuracy does not increase.

CHANGE MODEL

- Select “Clone job” button at the top right
- Increase the learning rate to **0.0065**
- Set the number of “**Training Epochs**” to **8**

Solver Options

Training epochs

8

Snapshot interval (in epochs)

1.0

Validation interval (in epochs)

1.0

Random seed

[none]

Batch size multiples allowed

32

Batch Accumulation

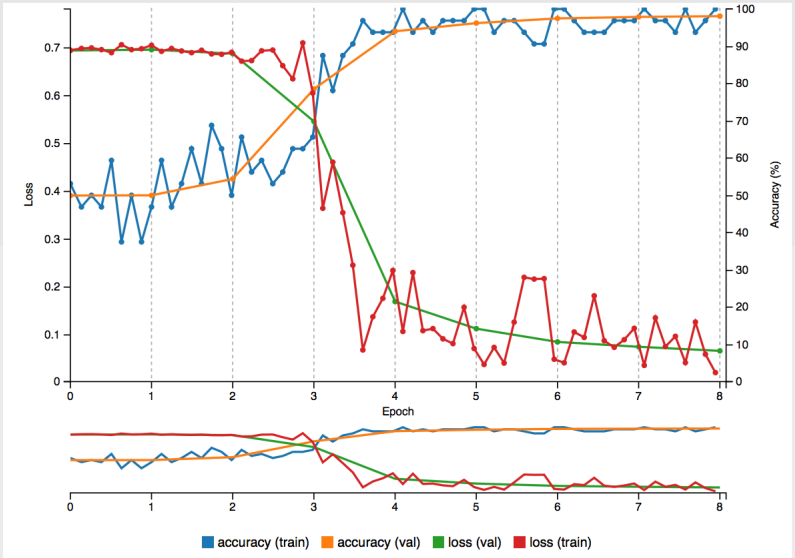
Solver type

Nesterov's accelerated gradient (NAG)

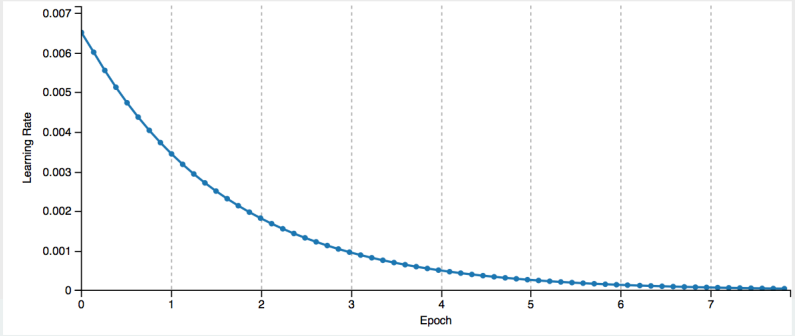
Base Learning Rate multiples allowed

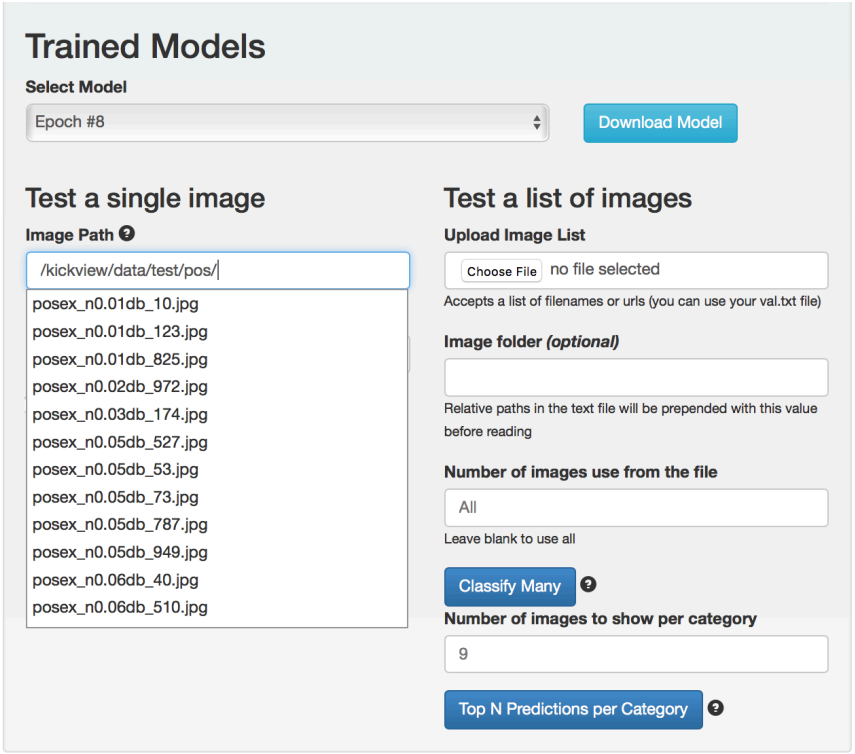
0.0065

Show advanced learning rate options



View Large





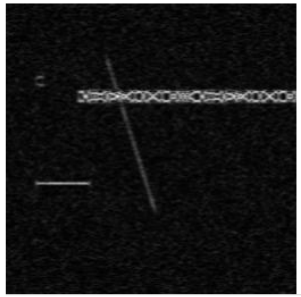
- Test a single image

- Enter the path to the file '/home/ubuntu/demo/traindemo/pos/posex_0.01db_2061.jpg‘

- Check “Show visualizations and statistics”

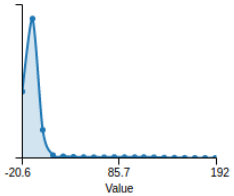
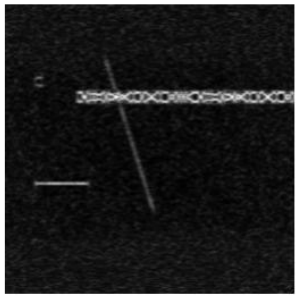
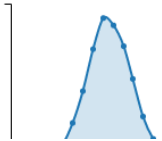
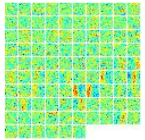
- Click “Classify One”

chirp_demo Image Classification Model



Predictions

pos	99.98%
neg	0.02%

Description	Statistics	Visualization
data Activation	**Data shape:** (1, 255, 255) **Mean:** -2.81838 **Std deviation:** 18.5414 	
conv1 Weights *(Convolution layer)* 11,712 learned parameters	**Data shape:** (96, 1, 11, 11) **Mean:** 6.97925e-05 **Std deviation:** 0.0108461 	

DOWNLOAD SAVED NETWORK MODEL

or use **pre-installed the trained network model**

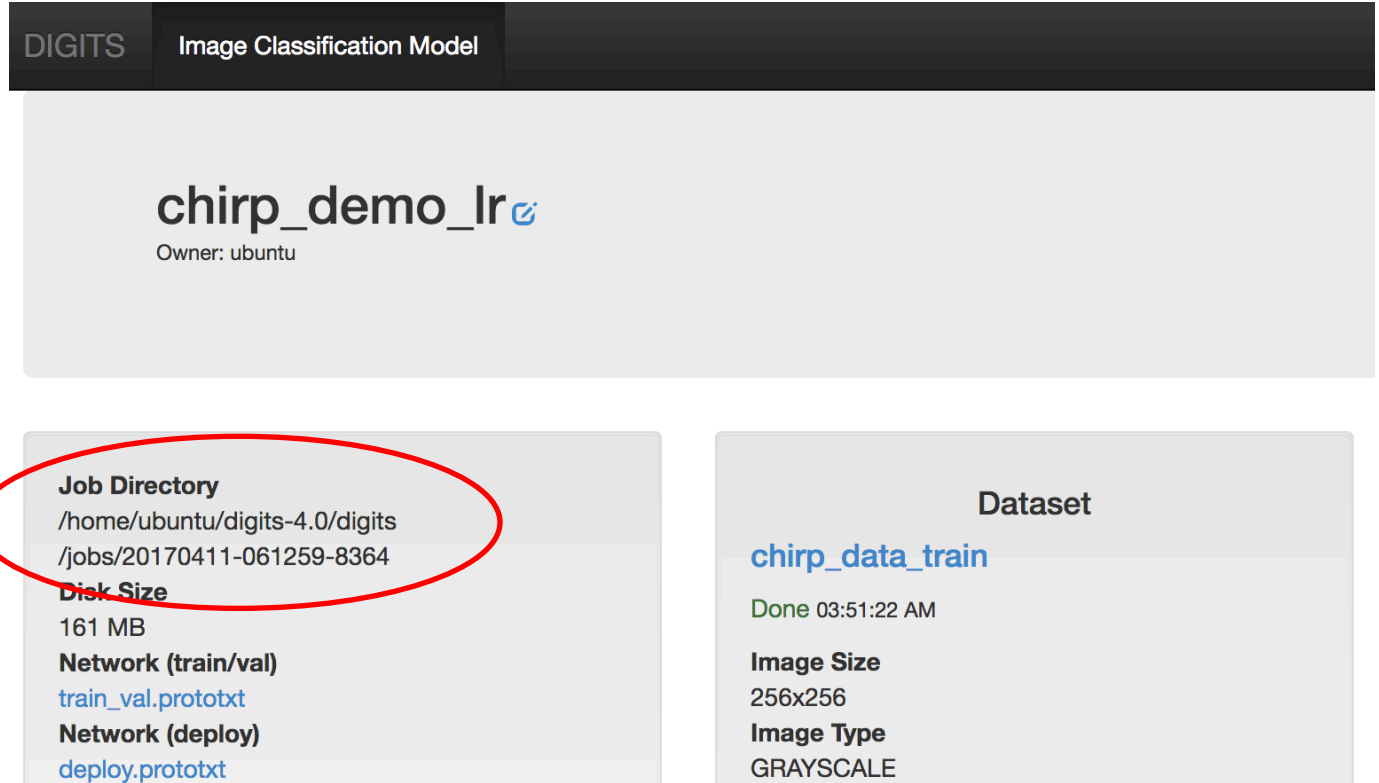
PYTHON ANALYSIS SCRIPT

- `analyze_spect_dir.py`
  - `detection_tst( pos_file_dir, model_dir, pic_type)`
  - `false_alarm_tst( neg_file_dir, model_dir, pic_type)`
  - Arguments
    - positive/negative image file directory
    - caffe model directory
    - pic type ('jpg' or 'png')

```
import analyze_spect_dir as ansp
```

USING TRAINED NETWORK MODEL

or use pre-installed the trained network model



PYTHON ANALYSIS SCRIPT

probability of detection (PD)

```
ansp.detection_tst('/home/ubuntu/demo/testdemo/pos', '/home/ubuntu/demo/model', 'jpg')
```

```
GPU id used:  0
(1, 1, 255, 255)
Network load from snapshot_iter_2660.caffemodel
Processing 1000 files from /home/ubuntu/demo/testdemo/pos
File posex_n6.94db_286.jpg triggered,  neg=0.9, pos=0.1, Miss rate: 1/1000
File posex_n3.67db_425.jpg triggered,  neg=1.0, pos=0.0, Miss rate: 2/1000
File posex_n6.8db_438.jpg triggered,  neg=1.0, pos=0.0, Miss rate: 3/1000
File posex_n5.4db_421.jpg triggered,  neg=0.8, pos=0.2, Miss rate: 4/1000
File posex_n6.97db_89.jpg triggered,  neg=0.8, pos=0.2, Miss rate: 5/1000
File posex_n6.09db_751.jpg triggered,  neg=0.5, pos=0.5, Miss rate: 6/1000
File posex_n5.29db_240.jpg triggered,  neg=0.7, pos=0.3, Miss rate: 7/1000
File posex_n6.98db_563.jpg triggered,  neg=1.0, pos=0.0, Miss rate: 8/1000
File posex_n6.63db_400.jpg triggered,  neg=0.7, pos=0.3, Miss rate: 9/1000
File posex_n6.75db_611.jpg triggered,  neg=1.0, pos=0.0, Miss rate: 10/1000
File posex_n5.84db_692.jpg triggered,  neg=0.6, pos=0.4, Miss rate: 11/1000
Final PD: 98.9%, Pmiss: 1.1%
```

PYTHON ANALYSIS SCRIPT

probability of false alarm (PFA)

```
ansp.false_alarm_tst('/home/ubuntu/demo/testdemo/neg', '/home/ubuntu/demo/model', 'jpg')
```

```
GPU id used:  0
(1, 1, 255, 255)
Network load from snapshot_iter_2660.caffemodel
Processing 1000 files from /home/ubuntu/demo/testdemo/neg
File negex_164.jpg triggered,  neg=0.4, pos=0.6, False alarm rate: 1/1000
File negex_269.jpg triggered,  neg=0.3, pos=0.7, False alarm rate: 2/1000
File negex_807.jpg triggered,  neg=0.2, pos=0.8, False alarm rate: 3/1000
                                    ⋮
File negex_914.jpg triggered,  neg=0.4, pos=0.6, False alarm rate: 20/1000
File negex_74.jpg triggered,  neg=0.5, pos=0.5, False alarm rate: 21/1000
File negex_161.jpg triggered,  neg=0.3, pos=0.7, False alarm rate: 22/1000
File negex_607.jpg triggered,  neg=0.4, pos=0.6, False alarm rate: 23/1000
File negex_664.jpg triggered,  neg=0.4, pos=0.6, False alarm rate: 24/1000
File negex_713.jpg triggered,  neg=0.1, pos=0.9, False alarm rate: 25/1000
Final PFA: 2.5%
```

FINE TUNING THE NETWORK MODEL

Python analysis script - PD

```
ansp.detection_tst('/home/ubuntu/demo/testdemo/pos', '/home/ubuntu/demo/model/finetunelow', 'jpg')
```

```
GPU id used:  0
(1, 1, 255, 255)
Network load from snapshot_iter_135.caffemodel
Processing 1000 files from /home/ubuntu/demo/testdemo/pos
File posex_n6.7db_709.jpg triggered,  neg=0.6, pos=0.4, Miss rate: 1/1000
File posex_n6.94db_286.jpg triggered,  neg=0.8, pos=0.2, Miss rate: 2/1000
File posex_n3.67db_425.jpg triggered,  neg=0.9, pos=0.1, Miss rate: 3/1000
File posex_n6.8db_438.jpg triggered,  neg=0.9, pos=0.1, Miss rate: 4/1000
File posex_n5.4db_421.jpg triggered,  neg=0.9, pos=0.1, Miss rate: 5/1000
File posex_n6.97db_89.jpg triggered,  neg=0.6, pos=0.4, Miss rate: 6/1000
File posex_n6.98db_563.jpg triggered,  neg=0.9, pos=0.1, Miss rate: 7/1000
File posex_n6.75db_611.jpg triggered,  neg=1.0, pos=0.0, Miss rate: 8/1000
Final PD: 99.2%, Pmiss: 0.8%
```

Improved! (98.9% before)

FINE TUNING THE NETWORK MODEL

Python analysis script - PFA

```
ansp.false_alarm_tst('/home/ubuntu/demo/testdemo/neg', '/home/ubuntu/demo/model/finetunelow', 'jpg')
```

```
GPU id used:  0
(1, 1, 255, 255)
Network load from snapshot_iter_135.caffemodel
Processing 1000 files from /home/ubuntu/demo/testdemo/neg
File negex_164.jpg triggered,  neg=0.5, pos=0.5, False alarm rate: 1/1000
File negex_585.jpg triggered,  neg=0.2, pos=0.8, False alarm rate: 2/1000

                               ⋮

File negex_161.jpg triggered,  neg=0.4, pos=0.6, False alarm rate: 12/1000
File negex_609.jpg triggered,  neg=0.4, pos=0.6, False alarm rate: 13/1000
File negex_664.jpg triggered,  neg=0.4, pos=0.6, False alarm rate: 14/1000
File negex_713.jpg triggered,  neg=0.1, pos=0.9, False alarm rate: 15/1000
Final PFA: 1.5%
```

Improved! (2.5% before)

FINE TUNING THE NETWORK MODEL

Optional

- make a new dataset on DIGITS for low SNR dataset. (-4dB ~ -8dB)
  - `/home/ubuntu/demo/finetunelow`
- Clone previous trained classification model on DIGITS
- In the "Pretrained model" window, use `.caffemodel` that you saved off from the previous training
- Make the learning rate much smaller (/10 or /100)
- Train on the low SNR data set for a handful of epochs.
- See if the network is able to improve.

WHAT ELSE?

- To further improve the network, we could
  - Train with more noise examples
  - Freeze the CNN layer weights during this fine tuning
    (so the network does not change too much)
- Create a detector for other signal types.
- Download this lab for later viewing
  - Your browsers File menu (not the Jupyter notebook file menu)
    save the complete web page. (This will ensure the images are copied down as well)

WHAT'S NEXT

- Use / practice what you learned
- Discuss with peers practical applications of DNN
- Reach out to NVIDIA and the Deep Learning Institute

GPU TECHNOLOGY CONFERENCE

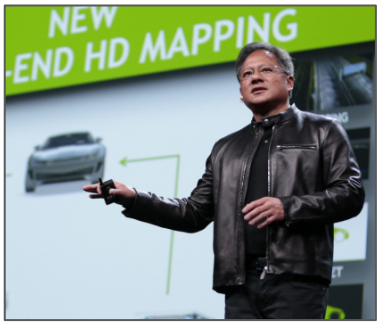
May 8 - 11, 2017 | Silicon Valley | #GTC17
www.gputechconf.com

CONNECT

Connect with technology experts from NVIDIA and other leading organizations

LEARN

Gain insight and valuable hands-on training through hundreds of sessions and research posters



DISCOVER

See how GPUs are creating amazing breakthroughs in important fields such as deep learning and AI

INNOVATE

Hear about disruptive innovations from startups

Don’t miss the world’s most important event for GPU developers
May 8 – 11, 2017 in Silicon Valley

NVIDIA | DEEP LEARNING INSTITUTE

www.nvidia.com/dli