



DEEP
LEARNING
INSTITUTE

Signal Processing using DIGITS

Hanbyul Yang
Solutions Architect
NVIDIA Corporation

A photograph of a woman with glasses, wearing a maroon long-sleeved shirt, speaking into a handheld microphone. She is gesturing with her right hand, pointing upwards. The background shows a blurred audience and a purple wall.

DEEP LEARNING INSTITUTE

DLI Mission

Helping people solve challenging problems using AI and deep learning.

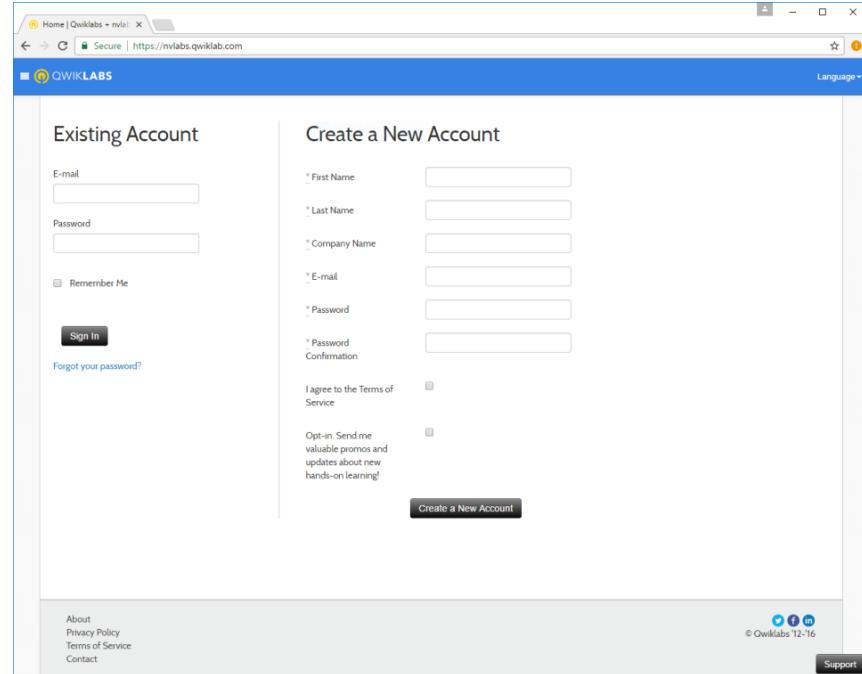
- Developers, data scientists and engineers
- Self-driving cars, healthcare and robotics
- Training, optimizing, and deploying deep neural networks

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

The screenshot shows a user interface for managing lab environments. At the top, there's a header bar with the text "In-Session Class: Deep Learning Labs". To the right of this are four metrics: "78.3 Total Hours", "50 Completed Labs", and "6 Classes Taken". Below the header is a list of available lab classes:

Class Name	Status
Object Detection with DIGITS	Currently Inactive
Identifying Whale Sounds with Audio Classification	Currently Inactive
Neural Network Deployment with DIGITS and TensorRT	Currently Inactive
Introduction to RNNs	Currently Inactive
Exploring TensorFlow on GPUs	Currently Inactive
Signal Processing using DIGITS	Currently Inactive
Image Classification with DIGITS	Currently Inactive
Modelling Time Series Data with Theano	Currently Inactive
Introduction to Image Recognition with CNTK	Currently Inactive

A green arrow points to the "In-Session Class" dropdown, and another green arrow points to the "Signal Processing using DIGITS" class in the list.

To the right of the list is a detailed view of the selected class:

Signal Processing using DIGITS
30 Credits

There are many resources available for learning how to leverage Deep Learning to process imagery. However, very few resources exist to demonstrate how to process data from other sensors such as acoustic, seismic, radio, or radar. In this tutorial, we will introduce some basic methods for utilizing a Convolutional Neural Network (CNN) to process Radio Frequency (RF) signals. More specifically, we will look at the classic problem of detecting a weak signal corrupted by noise. We will show you how to leverage the DIGITS application to read in a dataset, train a CNN, adjust hyper-parameters, and then test and evaluate the performance of your model.

Duration: 90 min.
Access Time: 115 min.
Setup Time: 7 min.
Level: Beginner

Lab created by [KickView - Intelligent Processing Applications](#)

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

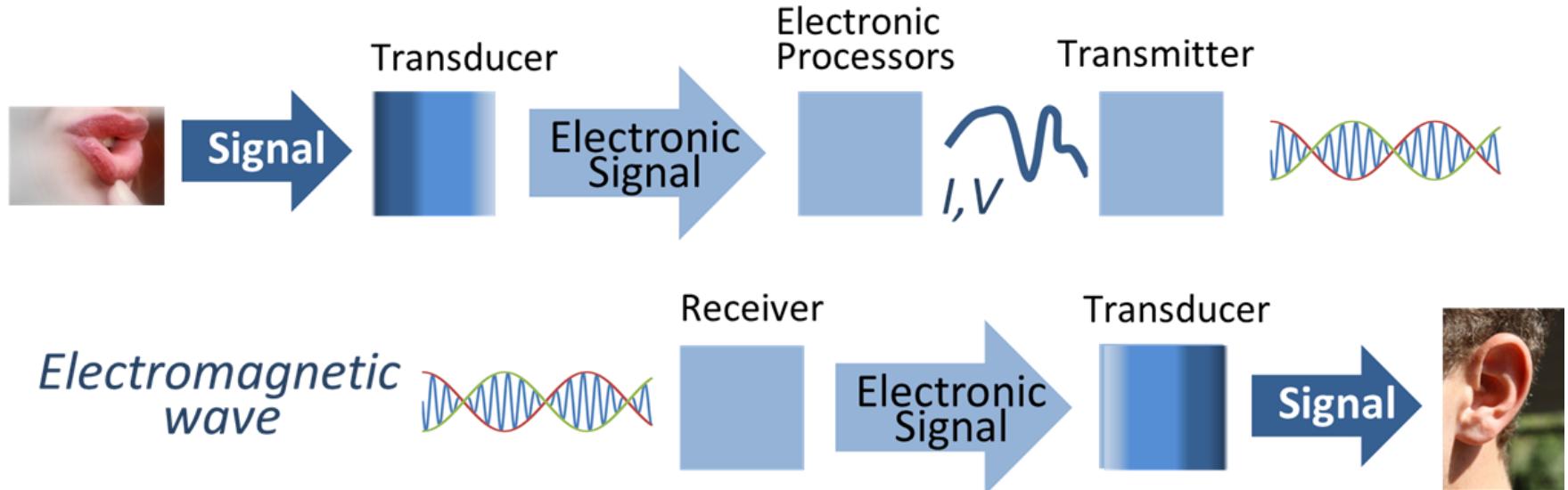
TAKE AWAYS

- 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

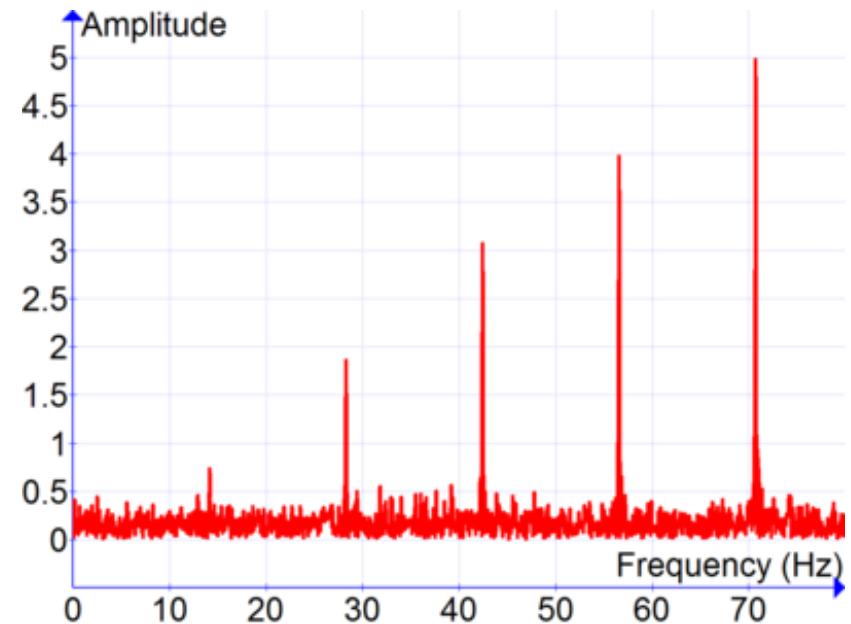
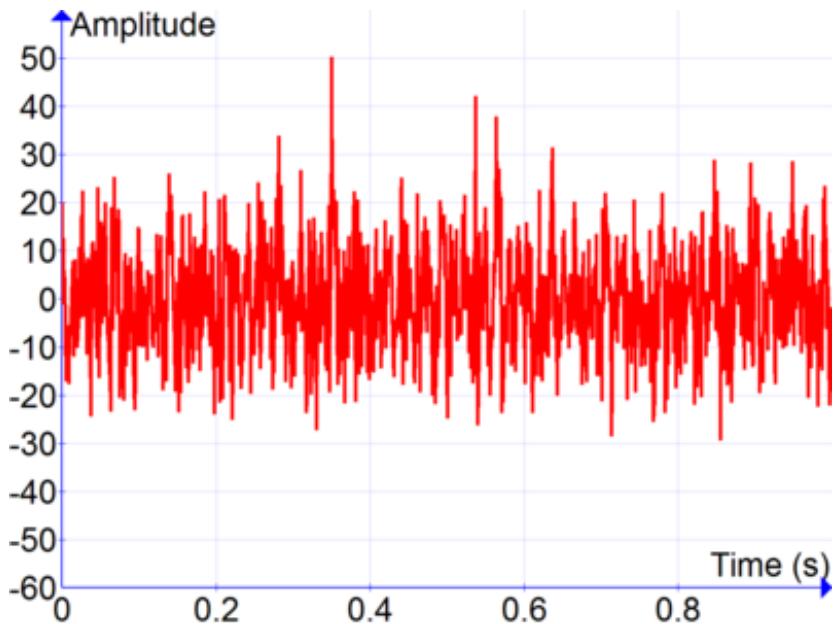
Signal processing

Example of audio signal processing



Signal processing

Fourier transform

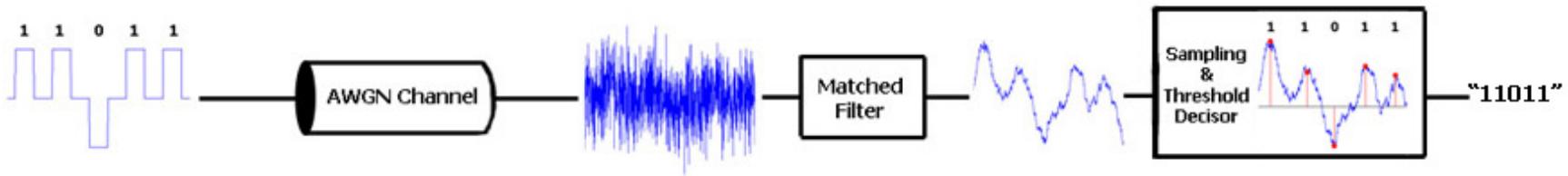


INTRO TO SIGNAL DETECTION

- 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



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...

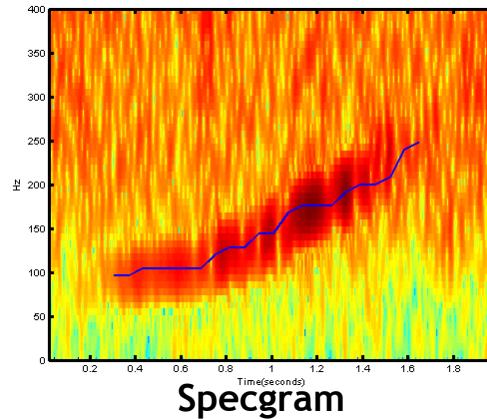
A little background information

- 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)



FFT

Waveform

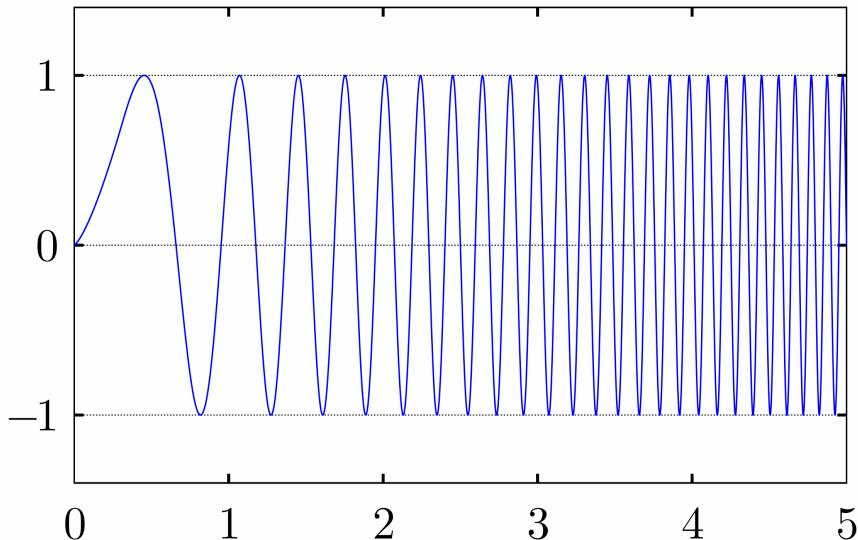


Specgram

CHIRP



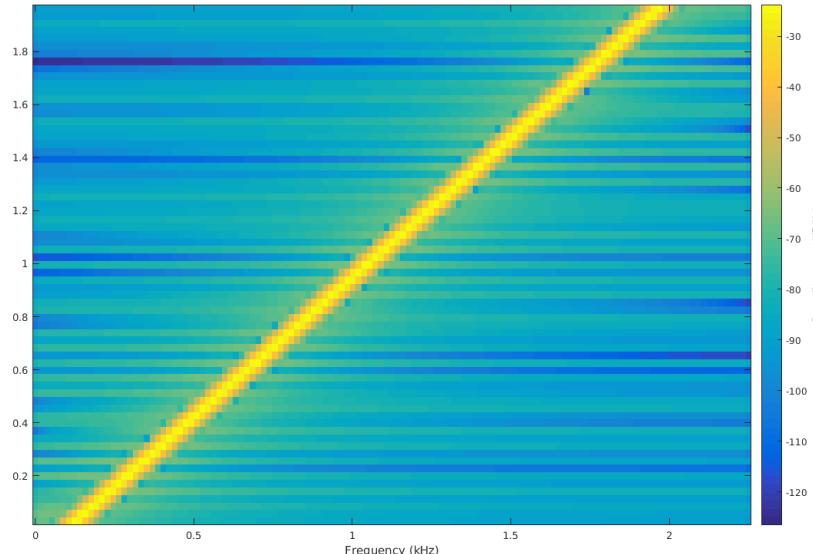
Linear Frequency-Modulated Signals



x-axis : time, y-axis : amplitude

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

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

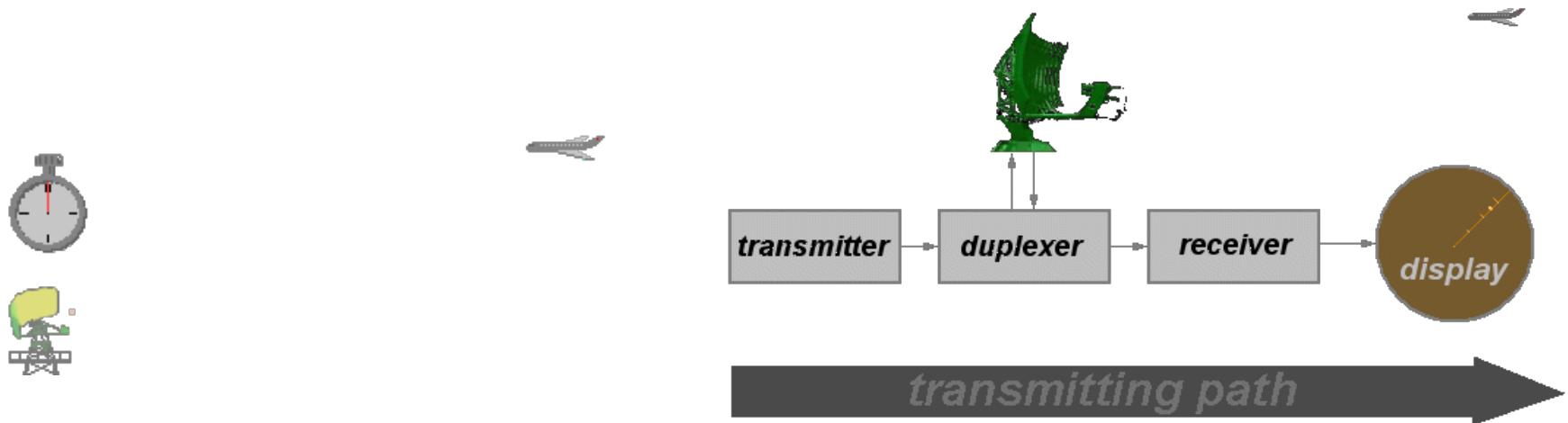


x-axis : time, y-axis : frequency

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

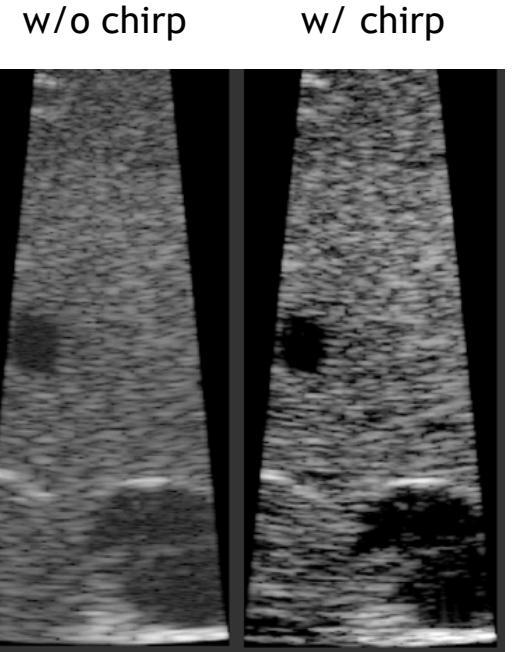
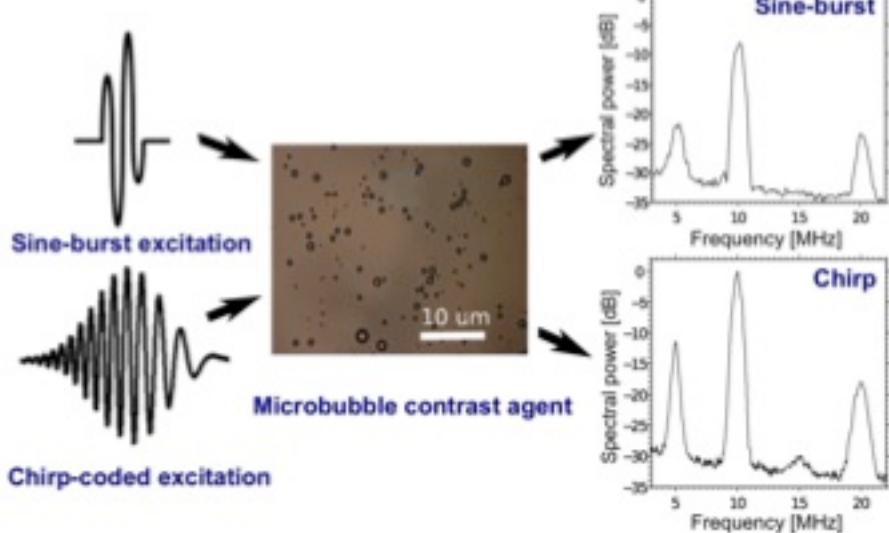
RADAR PRINCIPLE

RAdio (Aim) Detecting And Ranging



CHIRP EXAMPLE

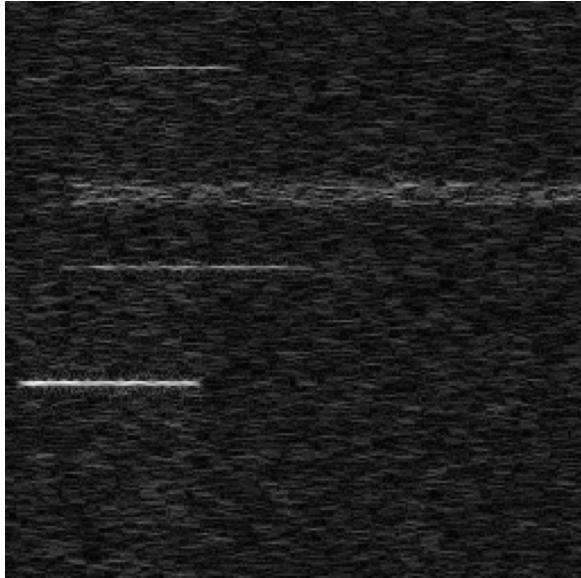
Ultrasound imaging



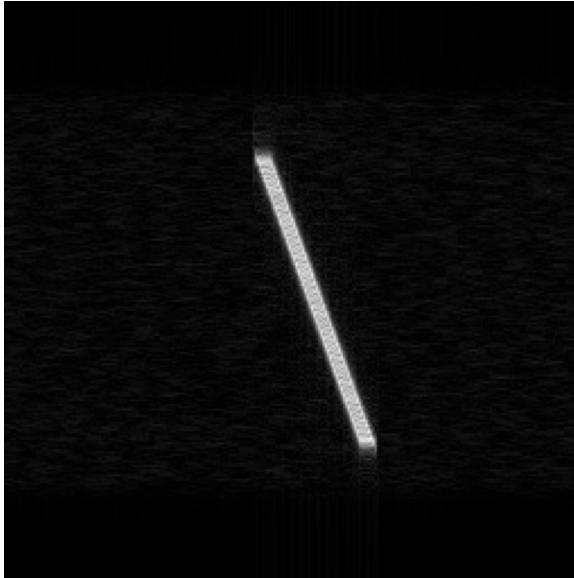
Ref : http://www.ece.rochester.edu/projects/doyley_lab/research/hifu.html
<http://research.vuse.vanderbilt.edu/beamlab/chirp/default.html>

SPECTROGRAM

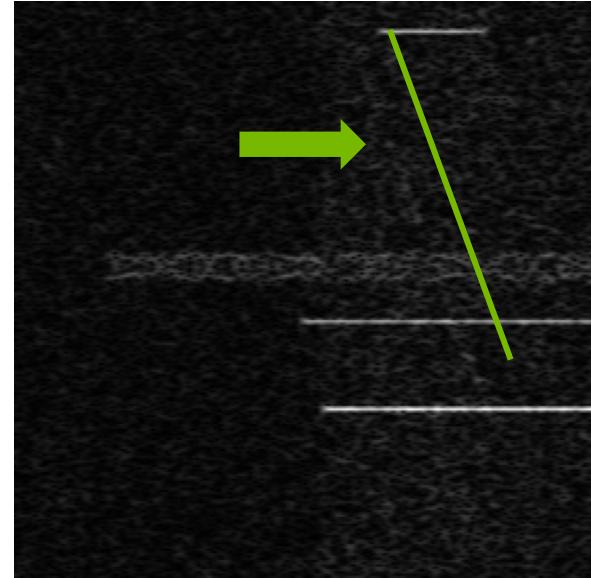
Chirp (Linear Frequency-Modulated) Signals



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



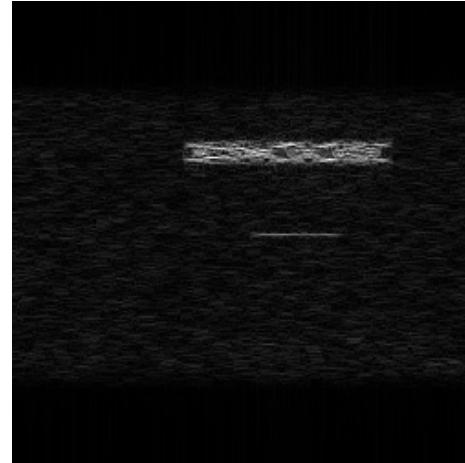
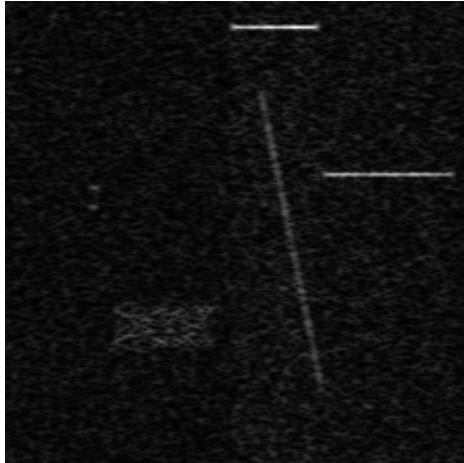
Signal-to-noise Ratio

-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

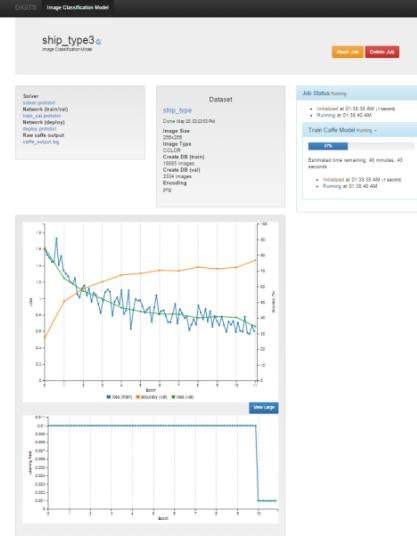
Process Data



Configure DNN

The screenshot shows the 'New Image Classification Model' configuration screen. It includes sections for 'Data Transformations' (Drop Size: 100), 'Solver Options' (Training epochs: 10, Snapshot interval: 1 epoch, Validation interval: 1 epoch, Random seed: 12345, Optimizer type: stochastic gradient descent (SGD), Base Learning Rate: 0.01), and 'Custom Network' (with a detailed JSON configuration). At the bottom, there are fields for 'GPUs' (1), 'GPUs to use' (4x NVIDIA K40), and 'Model Name' (copy1).

Monitor Progress



Visualization



Home

1/1 GPU available

No Jobs Running

Datasets (1)

Models (0)

Delete

name

No Models

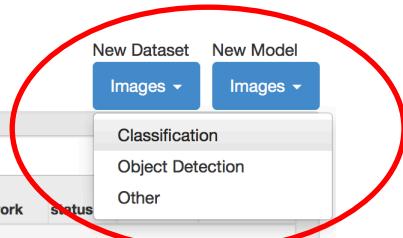


Filter

framework

status

New Dataset New Model
Images ▾ Images ▾
Classification
Object Detection
Other



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/trainedemo

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](#)

Category	Image Count
Category 1	~4200
Category 2	~4100

Image Mean: 

[Explore the db](#)

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

Folder
/home/ubuntu/demo/trainedemo

- 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 ↗

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

DIGITS

demo (Logout) Info ▾

1/1 GPU available

Home

No Jobs Running

[Datasets \(2\)](#) [Models \(0\)](#)

[Delete](#)

name

No Models

Filter

framework status elapsed sub

New Dataset Images ▾ New Model Images ▾

Classification
Object Detection
Other

A screenshot of the DIGITS web interface. At the top, there's a dark header bar with the word "DIGITS" on the left, "demo (Logout)" and "Info ▾" on the right, and "1/1 GPU available" below it. Below the header is a section titled "Home" with the sub-header "No Jobs Running". There are two buttons: "Datasets (2)" (highlighted in blue) and "Models (0)". Under "Datasets", there's a "Delete" button and a "name" input field containing "No Models". To the right of the datasets section is a search bar with a magnifying glass icon and the word "Filter", followed by several sorting columns: "framework", "status", "elapsed", and "sub". On the far right, there are two dropdown menus: "New Dataset" (set to "Images") and "New Model" (set to "Images"). A red oval highlights the "New Model" dropdown menu, which is open to show three options: "Classification", "Object Detection", and "Other".

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

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

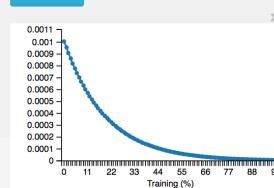
Policy

Exponential Decay

Gamma

0.95

Visualize LR

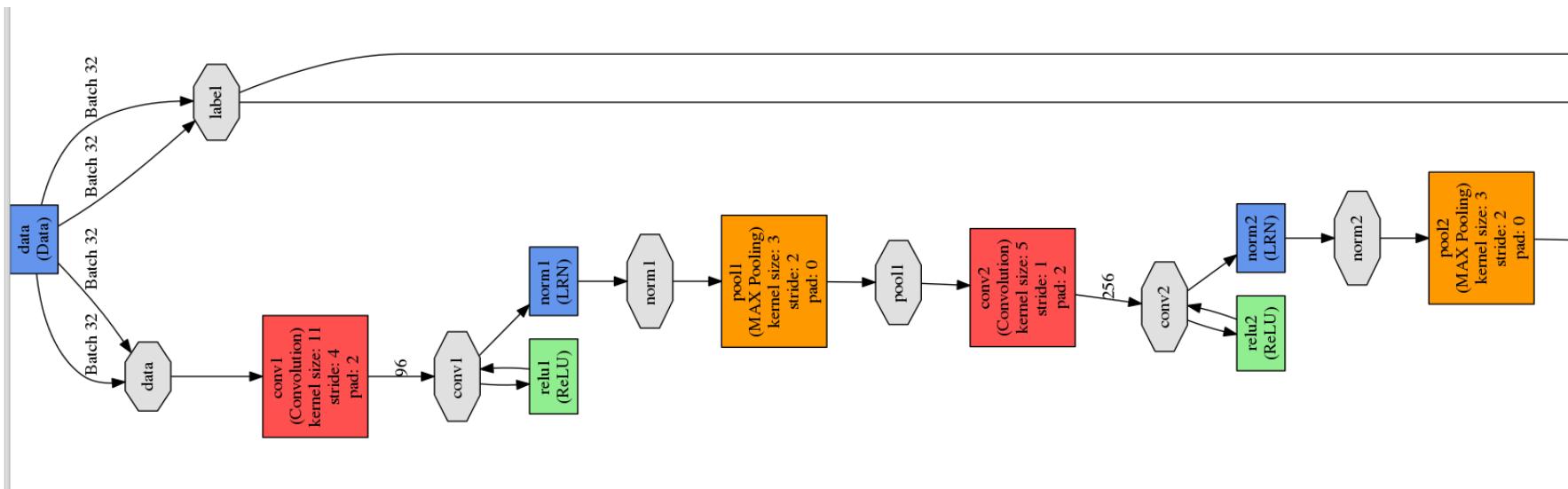
**Data Transformations** ⓘ

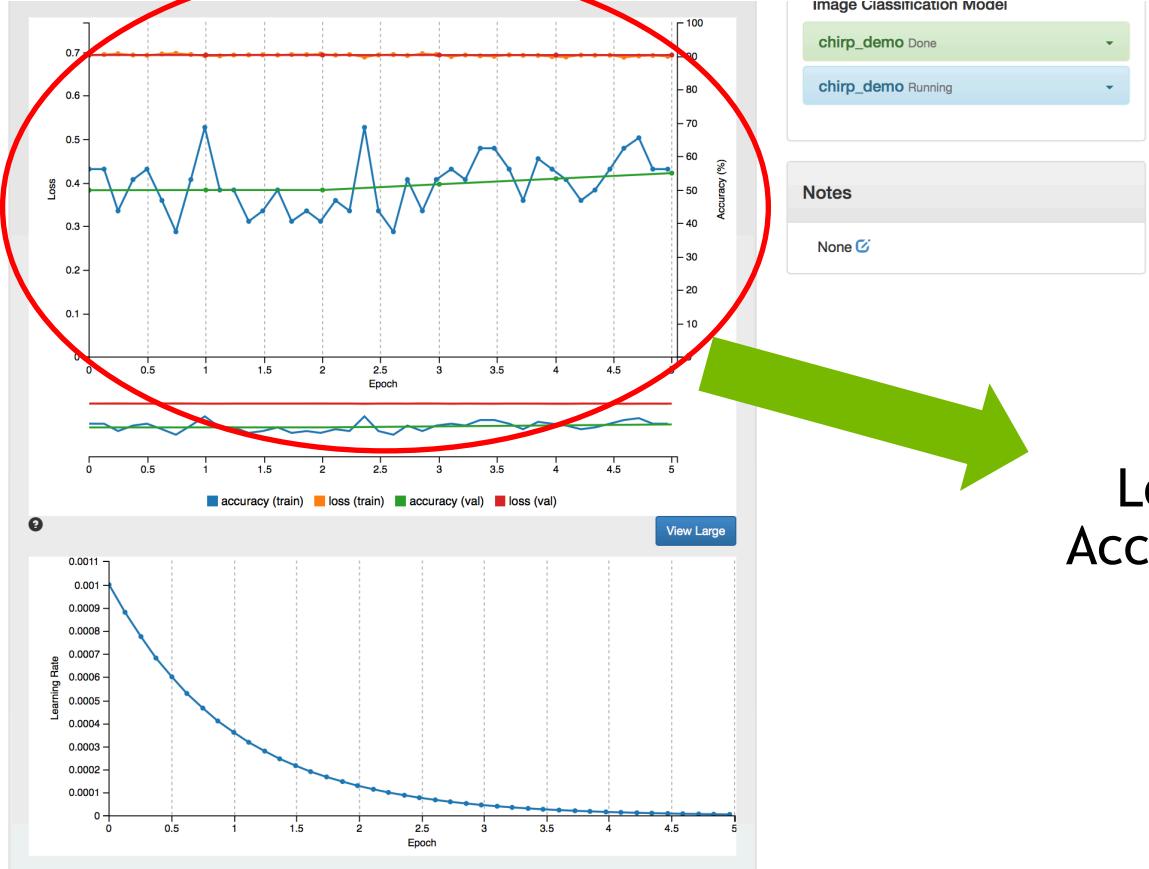
Crop Size

none

Subtract Mean

Image





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 ?

Snapshot interval (in epochs) ?

Validation interval (in epochs) ?

Random seed ?

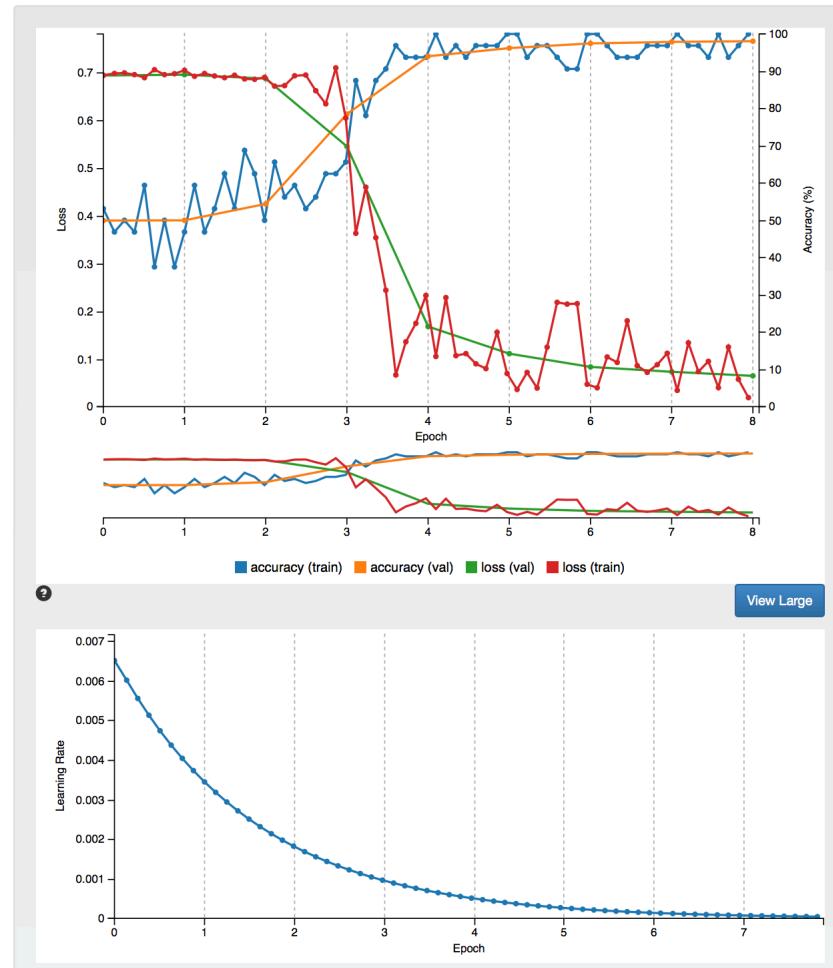
Batch size multiples allowed ?

Batch Accumulation ?

Solver type

Base Learning Rate multiples allowed ?

Show advanced learning rate options



Trained Models

Select Model

Epoch #8

Download Model

Test a single image

Image Path

```
/kickview/data/test/pos/  
posex_n0.01db_10.jpg  
posex_n0.01db_123.jpg  
posex_n0.01db_825.jpg  
posex_n0.02db_972.jpg  
posex_n0.03db_174.jpg  
posex_n0.05db_527.jpg  
posex_n0.05db_53.jpg  
posex_n0.05db_73.jpg  
posex_n0.05db_787.jpg  
posex_n0.05db_949.jpg  
posex_n0.06db_40.jpg  
posex_n0.06db_510.jpg
```

Test a list of images

Upload Image List

Choose File no file selected

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 

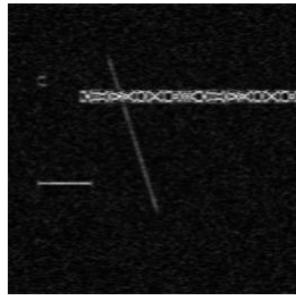
- Test a single image

- 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

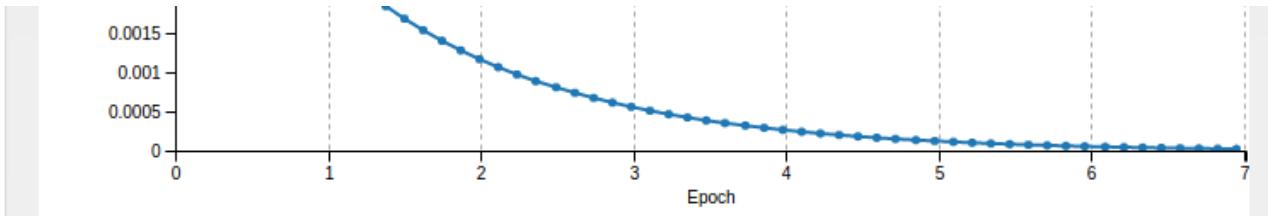


Description	Statistics	Visualization
data Activation	Data shape: (1, 255, 255) Mean: -2.81838 Std deviation: 18.5414	A histogram-like plot showing the distribution of activation values. The x-axis is labeled "Value" with ticks at -20.6, 85.7, and 192. The distribution is highly skewed, with a very sharp peak at approximately -20.6 and a much smaller secondary peak at approximately 85.7.
conv1 Weights (Convolution layer)	Data shape: (96, 1, 11, 11) Mean: 6.97925e-05 Std deviation: 0.0108461	A 11x11 grid of colored squares, likely representing the learned weights for the convolutional layer. The colors range from green to yellow and orange, indicating the magnitude of the weights.
11,712 learned parameters		A 11x11 grid of colored squares, likely representing the learned weights for the convolutional layer. The colors range from green to yellow and orange, indicating the magnitude of the weights.



DOWNLOAD SAVED NETWORK MODEL

or use pre-installed the trained network model



Trained Models

Select Model

Epoch #7

Epoch #7

Epoch #6

Epoch #5

Epoch #4

Epoch #3

Epoch #2

Epoch #1

Download Model

Number of images

List

No file chosen

Accepts a list of filenames or urls (you can use your val.txt file)

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

DIGITS Image Classification Model

chirp_demo_lr

Owner: ubuntu

Job Directory
/home/ubuntu/digits-4.0/digits
/jobs/20170411-061259-8364

Disk Size
161 MB

Network (train/val)
train_val.prototxt

Network (deploy)
deploy.prototxt

Dataset

chirp_data_train

Done 03:51:22 AM

Image Size
256x256

Image Type
GRAYSCALE

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_720.jpg triggered, neg=0.1, pos=0.9, False alarm rate: 4/1000
...
File negex_487.jpg triggered, neg=0.1, pos=0.9, False alarm rate: 19/1000
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



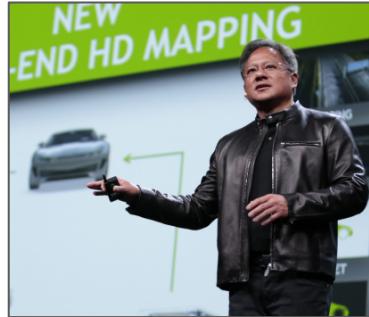
CONNECT

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



www.nvidia.com/dli

DEEP
LEARNING
INSTITUTE

