DEEP LEARNING INSTITUTE

Image Classification with TensorFlow: Radiomics – 1p/19q Chromosome Status Classification Using Deep Learning

Charles Killam, LP.D.
Certified Instructor, NVIDIA Deep Learning Institute
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
- CNNs
- Keras / TensorFlow
- Lab
  - Discussion / Overview
  - Launching the Lab Environment
  - Lab Review

LAB PERSPECTIVE

PURPOSE / GOAL

Use deep learning – specifically convolutional neural networks (CNNs) - to determine the 1p/19q chromosome arms co-deleted or not co-deleted status

WHY DETERMINE 1P/19Q STATUS

Studies show that low-grade gliomas (LGG) respond better to chemotherapy and radiotherapy when 1p/19q codeletion has been detected

Studies show longer survival for patients given above scenario

May reduce the need for surgical biopsies

WHAT THIS LAB IS

Guided, hands-on exercise using Keras and TensorFlow to build a CNN to evaluate MRI images to detect the 1p/19q chromosome arms status

WHAT THIS LAB IS NOT

- Introduction to machine learning from first principles
- Explanation of LGG and other brain tumors
- Rigorous mathematical formalism of neural networks
- Survey of all the features and options of Keras / TensorFlow

ASSUMPTIONS

- You are familiar with:
  - MRIs and basics of brain tumors
  - How neural networks work – forward propagation, back propagation, etc.
- Helpful to have:
  - Elementary level understanding of programming in a language such as Python, Java, C++, etc.

TAKE AWAYS

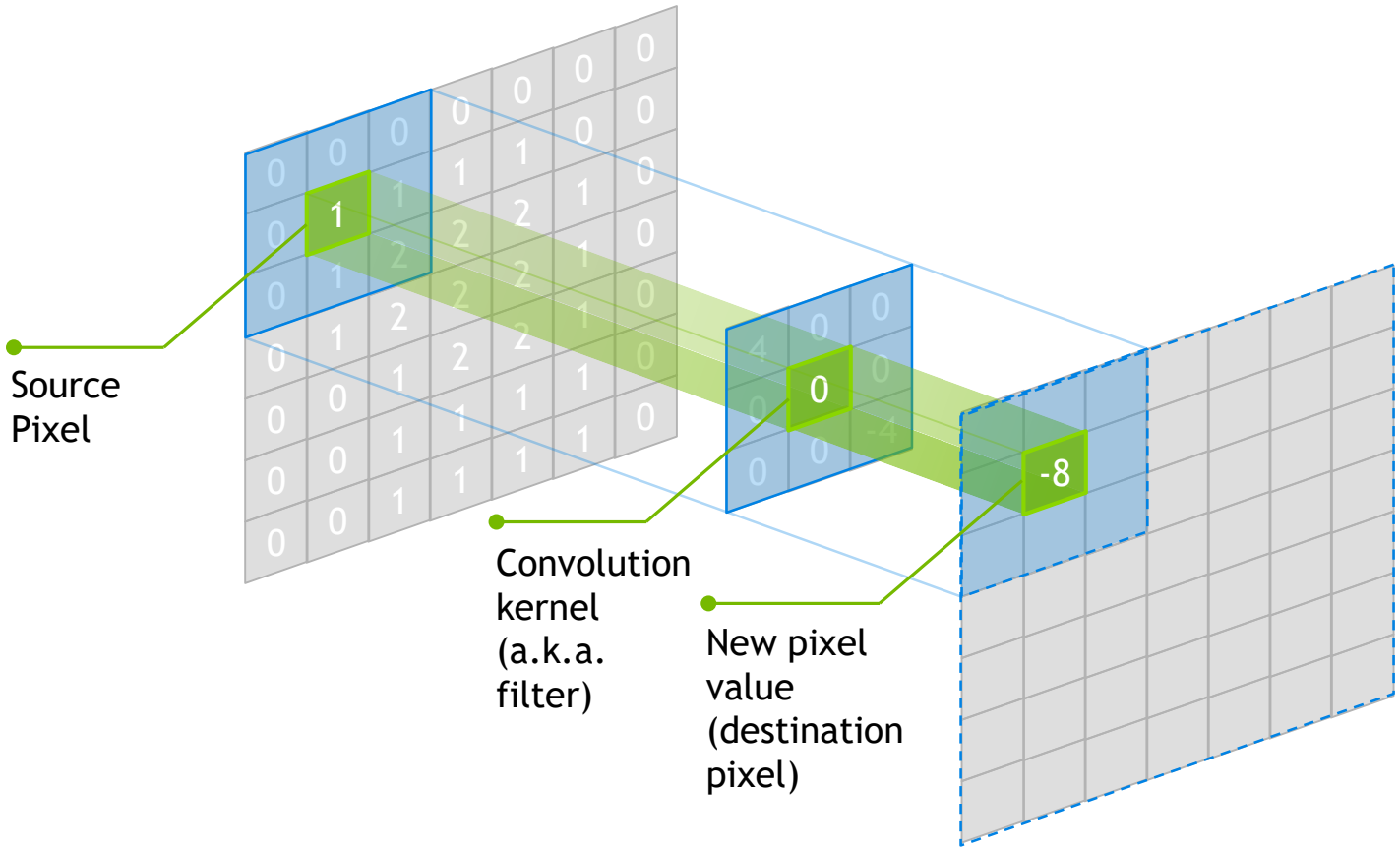
- Ability to setup your own CNN using Keras and TensorFlow
- Know where to go for more information on CNNs, Keras and TensorFlow
- Use knowledge acquired in this lab to initiate your own research in the area of Radiomics

CNNs

CONVOLUTIONAL NEURAL NETWORK

- Convolutions = kernels = filters
- Convolutions programmatically determine significant features
- Typical operations in CNN:
  - Convolution
  - Non-linearity / Activation function
  - Pooling
  - Classification

CONVOLUTIONS



- Kernel typically odd integer values
- Smaller kernel sizes may be better at identifying finer grade features

POOLING

4	-3	-7	8	9	-4	1	0	-2	-2
0	5	0	0	-1	-3	5	0	-3	-5
6	2	-1	9	7	-4	1	5	0	4
-3	8	0	3	-4	-5	0	5	1	-2
7	0	-5	6	2	0	1	7	-4	0
5	4	6	8	5	0	2	3	-3	1
2	-5	3	-2	-5	3	6	-2	5	3
0	-1	0	-4	0	-6	-8	-4	1	3

5	8	9	5	0
8	9	7	5	4
7	8	5	7	1
2	3	3	6	5

- Sliding window
  - (2 X 2 with stride of 2 in this example)
- Down-sampling technique
- MaxPooling changes negative values to 0

```
model.add(Conv2D(16, (3, 3),activation='linear',kernel_initializer='he_uniform',
                 input_shape=X_test.shape[1:]))
   model.add(LeakyReLU(alpha=.01))   # add an advanced activation
   model.add(Conv2D(16, (3, 3),activation='linear',kernel_initializer='he_uniform'))
   model.add(LeakyReLU(alpha=.01))   # add an advanced activation
   model.add(MaxPooling2D(pool_size=(2, 2)))
…
   model.add(LeakyReLU(alpha=.01))   # add an advanced activation
   model.add(Conv2D(96, (3, 3),activation='linear',kernel_initializer='he_uniform'))
   model.add(LeakyReLU(alpha=.01))   # add an advanced activation
   model.add(MaxPooling2D(pool_size=(7, 7)))
   model.add(Flatten())
   model.add(Dense(512))
   model.add(keras.layers.noise.GaussianNoise(0.3))
   model.add(LeakyReLU(alpha=.001))   # add an advanced activation
   model.add(Dropout(0.5))
   model.add(Dense(2))
   model.add(Activation('softmax'))
```

KERAS, TENSORFLOW AND MATPLOTLIB

KERAS

- Modular neural network written in Python
- Runs on TensorFlow and Theano
- Keras library allows for easy and fast prototyping
- Runs on GPUs and CPUs
- Compatible with Python 2.7 – 3.5

TENSORFLOW

Created by Google, tensorflow.org

- "Open source software library for machine intelligence"
  - Available on GitHub
- Flexibility—express your computation as a data flow graph
  - If you can express it in TF syntax you can run it
- Portability—CPUs and GPUs, workstation, server, mobile
- Language options—Python and C++
- Performance—Tuned for performance on CPUs and GPUs.
  - Assign tasks to different hardware devices.
  - Uses CUDNN

MATPLOTLIB

- Matplotlib is a Python 2D plotting library producing publication quality figures
- Matplotlib can be used in:
  - Python scripts
  - Python and IPython shell
  - Jupyter notebook
  - Web application servers
- Supports Python version 2.7 – 3.5

LAB DISCUSSION / OVERVIEW

DATA

- MRI brain scans

	Training / Validation	Test	Total
Codeleted	30	18	48
Not codeleted	130	40	170
Total	160	58	218

LAB PROCESS

1. Setup
    a. Import libraries
    b. Change channel order between frameworks (not shown in code)

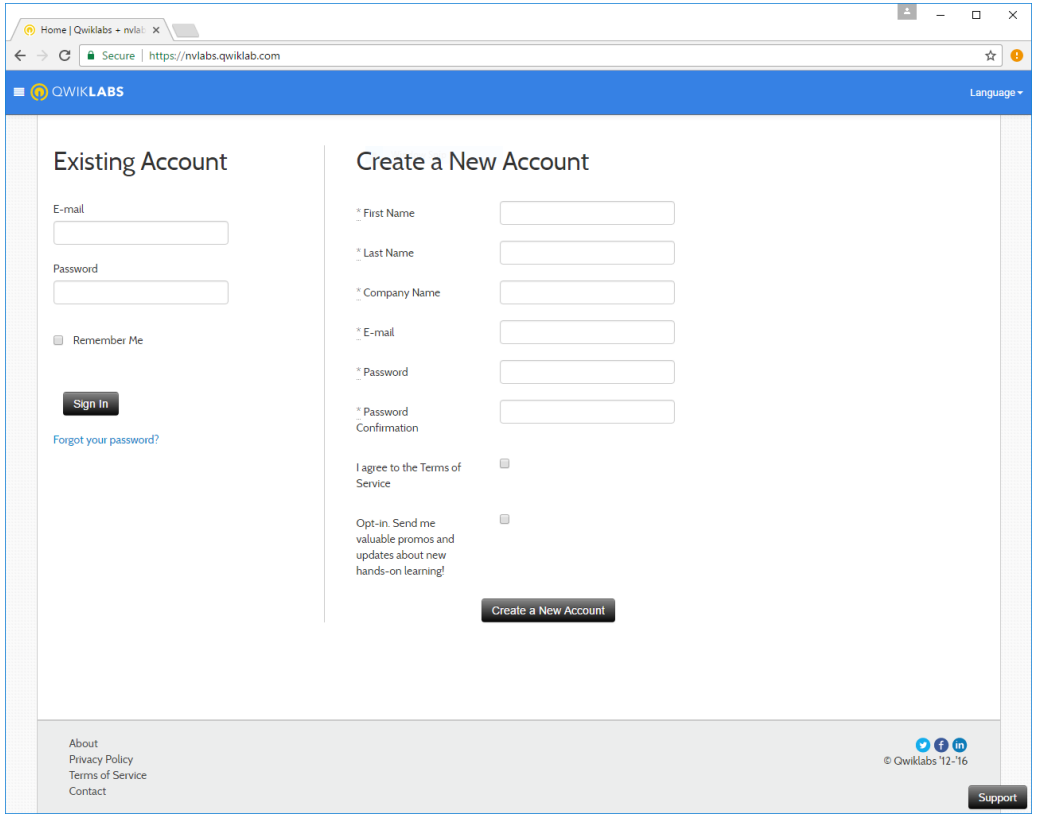
LAB PROCESS

2. Architect CNN network using Keras
3. Set hyperparameters
4. Data preparation
5. Build, evaluate and retrain model to improve performance
    - Uses five-fold cross validation
6. Run final model build for production-scenario inferencing

LAB ENVIRONMENT

NAVIGATING TO QWIKLABS

1. Navigate to: https://nvlabs.qwiklab.com
2. Login or create a new account



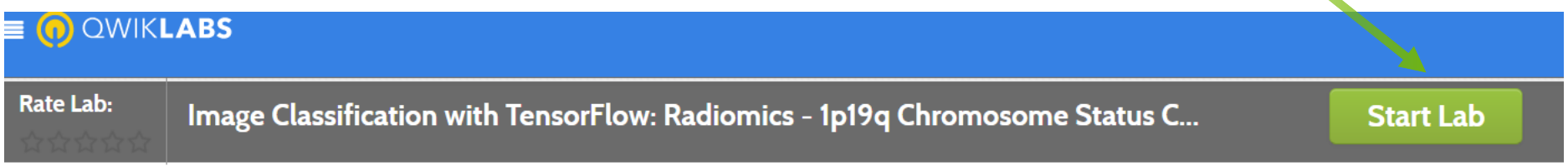
ACCESSING LAB ENVIRONMENT

Click on Radiomics – 1p19q Chromosome Status Classification

Class Labs		Duration (min.)
NVIDIA	Image Classification with DIGITS Tags: Deep Learning, Machine Learning, DIGITS	90
TensorFlow	Image Classification with TensorFlow: Radiomics - 1p19q Chromosome Status Classification with Deep Learning Tags: Machine Learning, Deep Learning, TensorFlow, self-paced, Keras	90
OpenACC	Introduction to OpenACC Directives Tags: OpenACC, self-paced, C, C++, Fortran	115

ACCESSING LAB INSTRUCTIONS

1. Click “Start Lab” to create an instance of the lab environment



LAB REVIEW

LAB REVIEW

1. Setup
    a. Import libraries
    b. Change channel order between frameworks (not shown in code)

LAB REVIEW - IMPORT LIBRARIES #1A

```
import os, sys, time, random, string, h5py
import matplotlib as mpl
import matplotlib.pyplot as plt
```

```
idgpu=0

print ('Locking GPU:')
print (idgpu)
print ('I am process:')
print (os.getpid())

#os.environ['KERAS_BACKEND'] = 'theano'
#os.environ['THEANO_FLAGS']='base_compiledir=/tmp/'+id_generator()+', mode=FAST_R

from keras.layers import Input
from keras.layers import Conv2D, MaxPooling2D, ZeroPadding2D, AveragePooling2D
from keras.layers import BatchNormalization
from keras.models import Model, Sequential
from _loadcsvdeep import load_set
from keras.callbacks import ModelCheckpoint
from keras.callbacks import LearningRateScheduler

import numpy as np

from keras.layers.advanced_activations import LeakyReLU, PReLU
from keras.layers.core import Flatten,Activation, Dense, Dropout
```

LAB REVIEW

2. Architect CNN network using Keras
3. Set hyperparameters
4. Data preparation
5. Build, evaluate and retrain model to improve performance
    - Uses five-fold cross validation
6. Run final model build for production-scenario inferencing

LAB REVIEW – ARCHITECT CNN #2

```
def cnn_model(img_rows, img_cols,img_channels):
    model = Sequential()
    model.add(Conv2D(16, (3, 3),activation='linear',kernel_initializer='he_uniform',
                     input_shape=X_test.shape[1:]))
    model.add(LeakyReLU(alpha=.01))   # add an advanced activation
    model.add(Conv2D(16, (3, 3),activation='linear',kernel_initializer='he_uniform'))
    model.add(LeakyReLU(alpha=.01))   # add an advanced activation
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(32, (3, 3),activation='linear',kernel_initializer='he_uniform'))
    model.add(LeakyReLU(alpha=.01))   # add an advanced activation
    model.add(Conv2D(32, (3, 3),activation='linear',kernel_initializer='he_uniform'))
    model.add(LeakyReLU(alpha=.01))   # add an advanced activation
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(64, (3, 3),activation='linear',kernel_initializer='he_uniform'))
    model.add(LeakyReLU(alpha=.01))   # add an advanced activation
    model.add(Conv2D(64, (3, 3),activation='linear',kernel_initializer='he_uniform'))
    model.add(LeakyReLU(alpha=.01))   # add an advanced activation
    model.add(Conv2D(96, (3, 3),activation='linear',kernel_initializer='he_uniform'))
    model.add(LeakyReLU(alpha=.01))   # add an advanced activation
    model.add(Conv2D(96, (3, 3),activation='linear',kernel_initializer='he_uniform'))
    model.add(LeakyReLU(alpha=.01))   # add an advanced activation
    model.add(MaxPooling2D(pool_size=(7, 7)))
    model.add(Flatten())
    model.add(Dense(512))
    model.add(keras.layers.noise.GaussianNoise(0.3))
    model.add(LeakyReLU(alpha=.001))   # add an advanced activation
    model.add(Dropout(0.5))
    model.add(Dense(2))
    model.add(Activation('softmax'))
```

LAB REVIEW - SET HYPERPARAMETERS #3

```
# For network monitoring
lr_reducer = ReduceLROnPlateau(monitor='val_categorical_accuracy', factor=np.sqrt(0.1), cooldown=0, patience=2, min_lr=0.5e-6)
early_stopper = EarlyStopping(monitor='val_categorical_accuracy', min_delta=0.001, patience=10)
csv_logger = CSVLogger('qp_logs.csv')

# Fit paramters
batch_size = 16 #32
nb_epoch = 40

# Input image dimensions
scan, img_rows, img_cols = 256, 256, 256
img_channels = 1 # Gray scale images

# Dataset locations
# TrainingValidation='/home/m112447/Desktop/Python_projects/DEMO_CODE/DATA/1p19q/'+'n43_PQ_Part1'+str(scan)+'.npz'
# Testing='/home/m112447/Desktop/Python_projects/DEMO_CODE/DATA/1p19q/'+'n43_PQ_Part2'+str(scan)+'.npz'
TrainingValidation = os.path.join(os.getcwd(), 'DATA', '1p19q', 'n43_PQ_Part1'+str(scan)+'.npz')
Testing = os.path.join(os.getcwd(), 'DATA', '1p19q', 'n43_PQ_Part2'+str(scan)+'.npz')
```

LAB REVIEW - DATA PREPARATION #4

```
# Data contain a third class normal --> Remove it for this demo
IND=np.nonzero(y==2)
y = np.delete(y, IND)
X = np.delete(X, IND,axis=0)
y_test=y
X_test=X

# Shuffle the training data
Y_train,X_train = shuffle(Y_train,X_train, random_state=0)
print ('Training Validation Size')
print (np.shape(X_train))
print ('Testing size')
print (np.shape(X_test))

# Normalize the data...the 10000 is set on the dataset creation process
# All the data are T2 just used MRIcron to draw a line acrross Z direction of the tumor
X_train=X_train/10000
X_test=X_test/10000
```

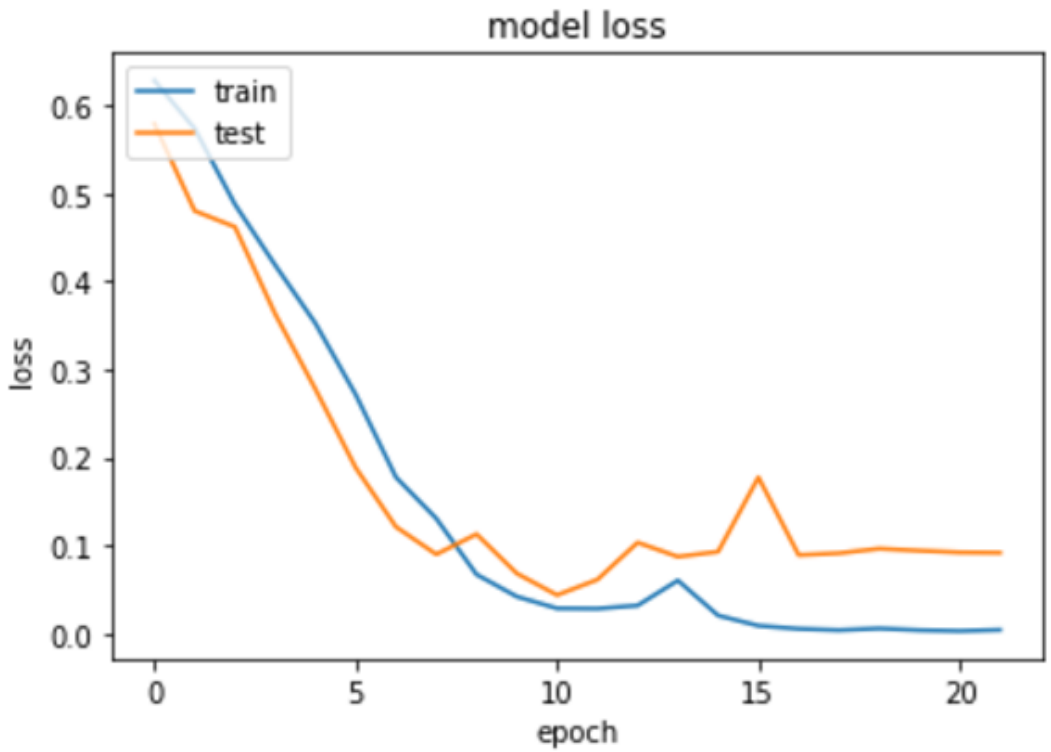
LAB REVIEW - BUILD, EVALUATE AND TRAIN MODEL #5

```
for train, test in kfold.split(X_train, Y_train):
    i+=1
    model = cnn_model(img_rows, img_cols, img_channels)
    adam=keras.optimizers.Adam(lr=0.00008)
    model.compile(loss='categorical_crossentropy',
          optimizer=adam,
          metrics=['categorical_accuracy'])

    print(model.summary())
    print ('Fold', str(i))
    print (10*'----------')
    print (10*'----------')

    best_model = ModelCheckpoint(output+'1p19q'+str(i)+'.h5', verbose=0, monitor='val_categorical_accuracy',save_best_

    history=model.fit(X_train[train], np_utils.to_categorical(Y_train[train],2),
          batch_size=batch_size,
          epochs=nb_epoch,
           validation_data=(X_train[test], np_utils.to_categorical(Y_train[test],2)),
          shuffle=True,
          callbacks=[lr_reducer, csv_logger,early_stopper,best_model],verbose=0)

    # Load best weights
    model.load_weights(output+'1p19q'+str(i)+'.h5')
```

LAB REVIEW - FINAL MODEL BUILD #6

```
model = cnn_model(img_rows, img_cols,img_channels)
adam=keras.optimizers.Adam(lr=0.00008)
model.compile(loss='categorical_crossentropy',
      optimizer=adam,
      metrics=['categorical_accuracy'])
print (10*'----------')
print (10*'----------')
best_model = ModelCheckpoint(output+'1p19q_full'+str(i)+'.h5', verbose=0, monitor='val_categorical_accuracy',save_best_only=True)
history=model.fit(X_train, np_utils.to_categorical(Y_train,2),
      batch_size=batch_size,
      epochs=nb_epoch,
        validation_split=.2,
      shuffle=True,
      callbacks=[lr_reducer, csv_logger,early_stopper,best_model],verbose=0)
f = plt.figure()
```

LAB REVIEW – FINAL MODEL BUILD #6

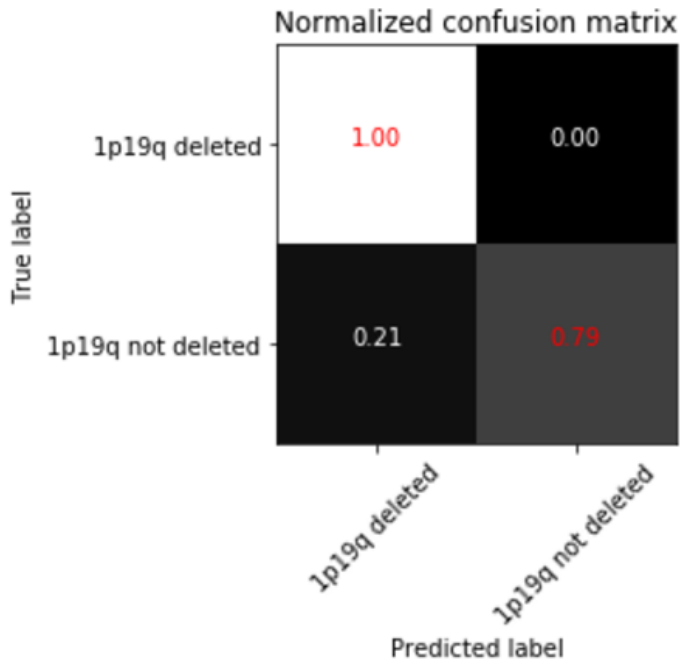


LAB REVIEW – FINAL MODEL BUILD #6

```
Testing Score
-------------
-------------
Overall Accuracy
The f1-score gives you the harmonic mean of precision and recall. The scores corresponding to every class will tell you the accu
racy of the classifier in classifying the data points in that particular class compared to all other classes.The support is the
 number of samples of the true response that lie in that class.
                   precision    recall  f1-score   support

    1p19q deleted     0.9398    1.0000    0.9690       125
1p19q not deleted     1.0000    0.7949    0.8857        39

      avg / total     0.9542    0.9512    0.9492       164
```

LAB REVIEW – FINAL MODEL BUILD #6

```
Confustion matrix: Test set
[[125   0]
 [  8  31]]
Normalized confusion matrix
[[ 1.    0.  ]
 [ 0.21  0.79]]
```



WHAT ELSE?

- Many ways to explore and possibly improve model:
  - Add additional layers to the network
  - Change the number of neurons in those layers
  - Change some of the hyperparameters in the network configuration like dropout or learning rate, etc.

WHAT’S NEXT

WHAT'S NEXT

- Use / practice what you learned
- Discuss with peers practical applications of DNN
- Reach out to NVIDIA and the Deep Learning Institute
- Attend local meetup groups
- Follow people like Andrej Karpathy and Andrew Ng

WHAT'S NEXT

TAKE SURVEY

…for the chance to win an NVIDIA SHIELD TV.

Check your email for a link.

ACCESS ONLINE LABS

Check your email for details to access more DLI training online.

ATTEND WORKSHOP

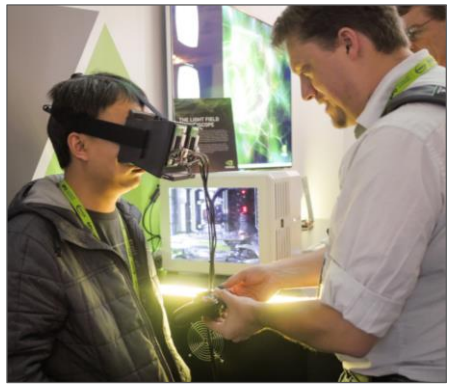
Visit www.nvidia.com/dli for workshops in your area.

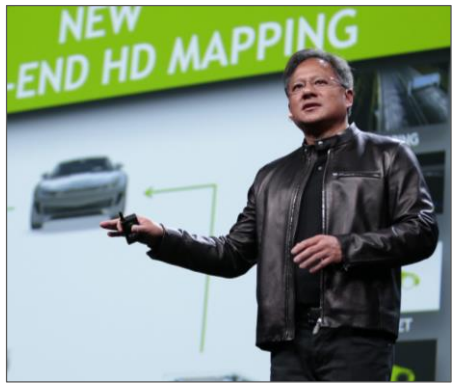
JOIN DEVELOPER PROGRAM

Visit https://developer.nvidia.com/join for more.

GPU TECHNOLOGY CONFERENCE

www.gputechconf.com





ADVANCE YOUR DEEP LEARNING TRAINING AT GTC

Don’t miss the world’s most important event for GPU developers

DEEP
LEARNING
INSTITUTE

www.nvidia.com/dli