Fundamentals of Deep Learning for Multiple Data Types

PD Dr. Juan J. Durillo

DEEP LEARNING INSTITUTE

DLI Mission

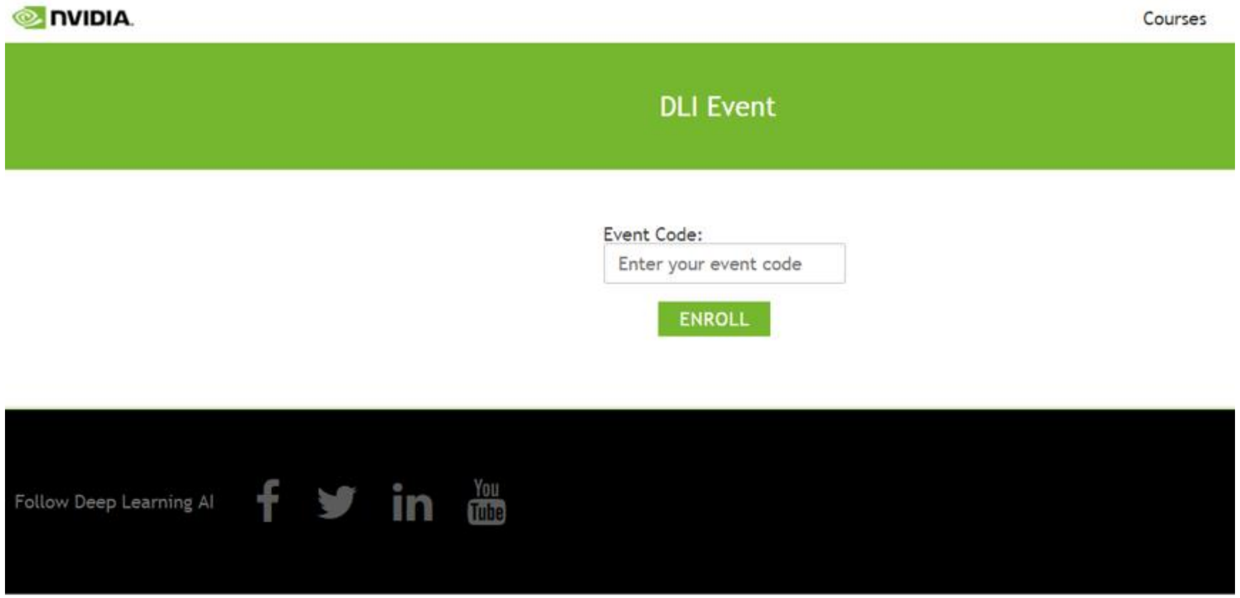
Helping people solve challenging problems using AI and deep learning.

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

NAVIGATING TO THE DLI PLATFORM

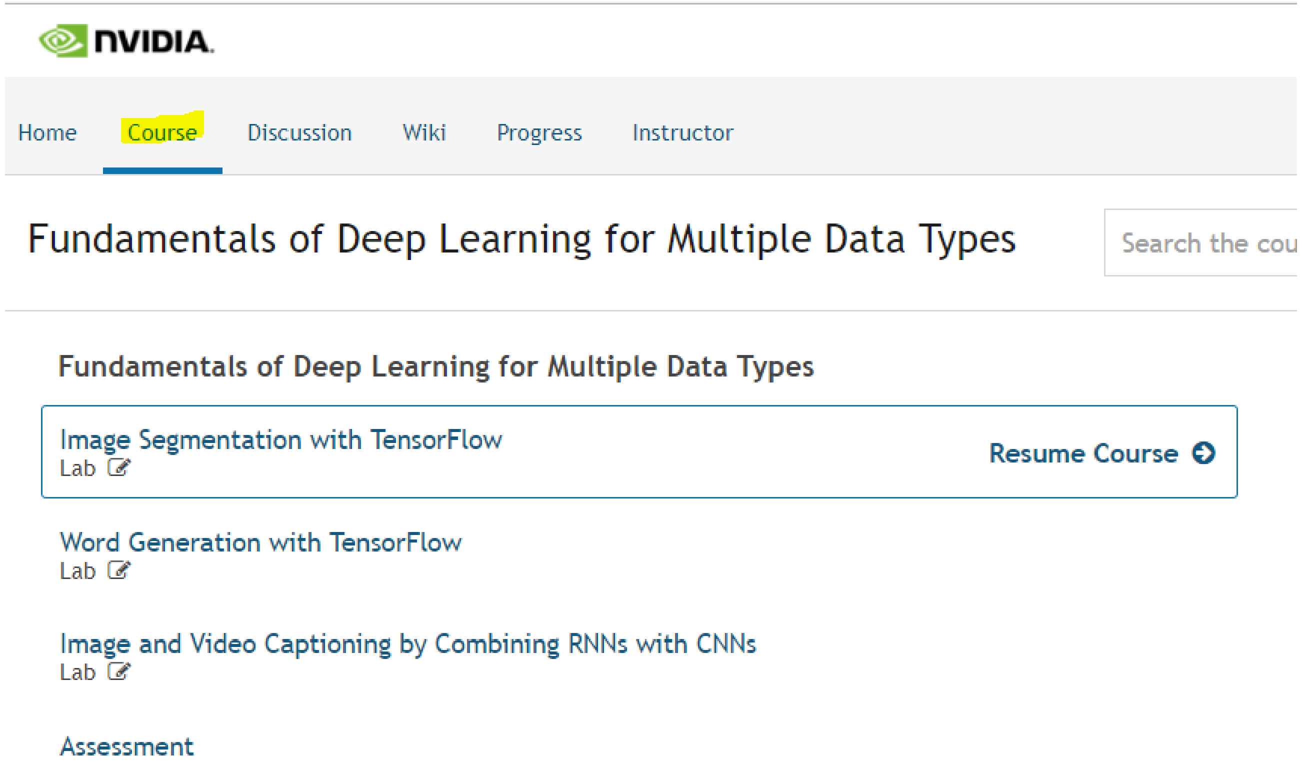
1. Navigate to: https://courses.nvidia.com/dli-event
2. Use given event code

1. “Log in with my NVIDIA Account”
2. Log in or “Create an Account”
3. Note: After confirming email address, close the newly opened tab!



NAVIGATING TO THE DLI PLATFORM

1. You have arrived when you have reached the "Course" tab
2. Open the first lab: "Image Segmentation with TensorFlow" and launch the task with the "Start" button.
3. "Start" will become "Loading" which will become "Launch"
4. Launch the task when it is ready and begin working through Task 1



Today's Project: Generating Captions

Learning Targets

- Introduce TensorFlow
- Compare Computer Vision Workflows
- Introduce Natural Language Processing
- Highlight the value of mid-network information
- Increase the diversity of solvable problems with Deep Learning

This course is not:

- A PhD in Data Science/Deep Learning/etc.
- A deep dive into the role of the GPU or CUDA
- A comparison of approaches/frameworks/technology
- A playbook to achieve state-of-the-art performance
- Simply a how-to for the workflows taught

Image Segmentation with TensorFlow

PD Dr. Juan J. Durillo
Certified Instructor, NVIDIA Deep Learning Institute

WHAT THIS LAB IS

- Discussion/Demonstration of Image Segmentation using Deep Learning
- Hands-on exercises using TensorFlow for CNN training and evaluation of Image Segmentation workflow

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 TensorFlow

ASSUMPTIONS

- You are familiar with convolutional neural networks (CNN)
- Helpful to have:
  - Image recognition experience
  - TensorFlow experience
  - Python experience

TAKE AWAYS

- You can setup your own image segmentation workflow in TensorFlow and adapt it to your use case
- Know where to go for more info
- Familiarity with TensorFlow

IMAGE SEGMENTATION

COMPUTER VISION TASKS

Image Classification

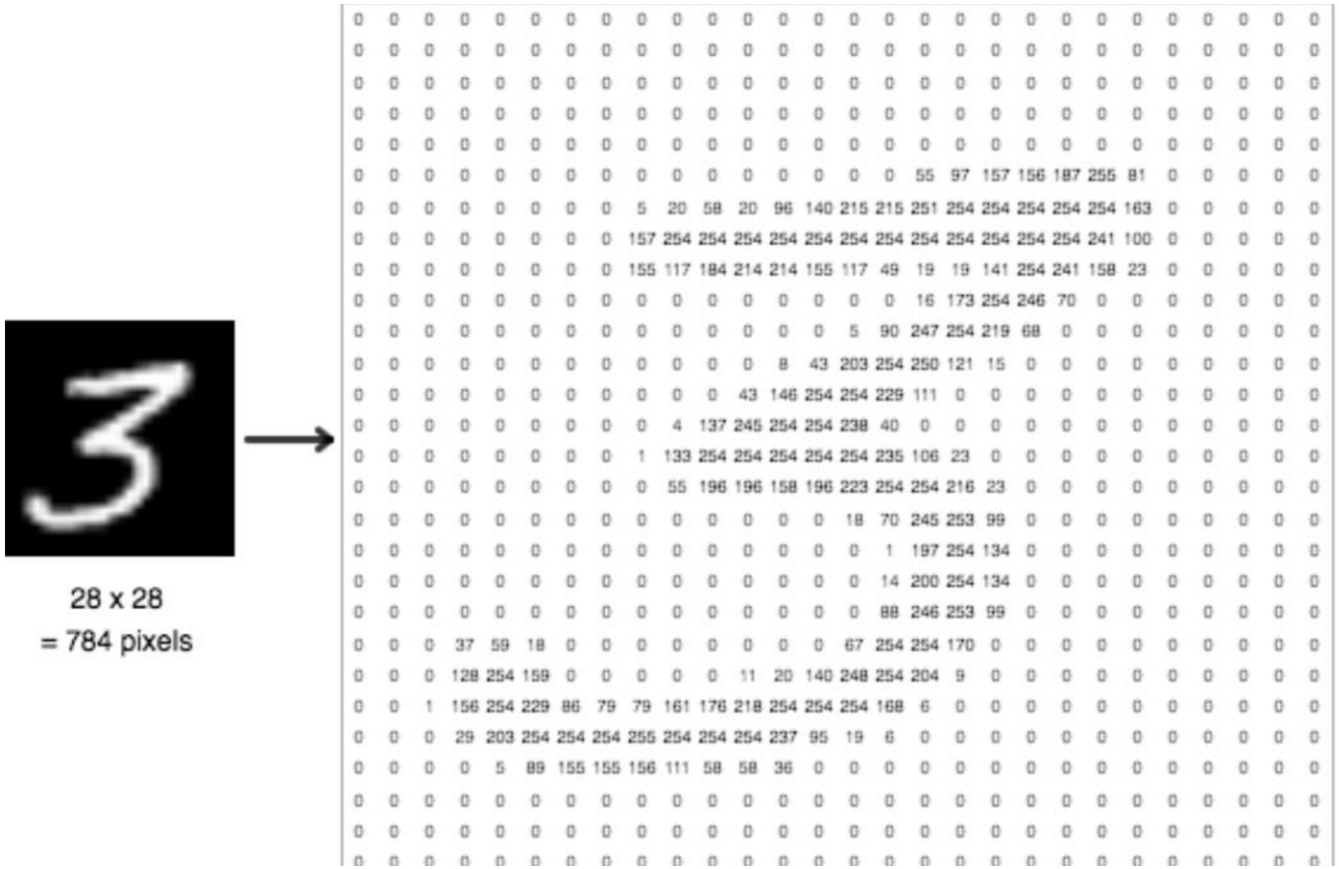
Image Classification + Localization

Object Detection

Image Segmentation

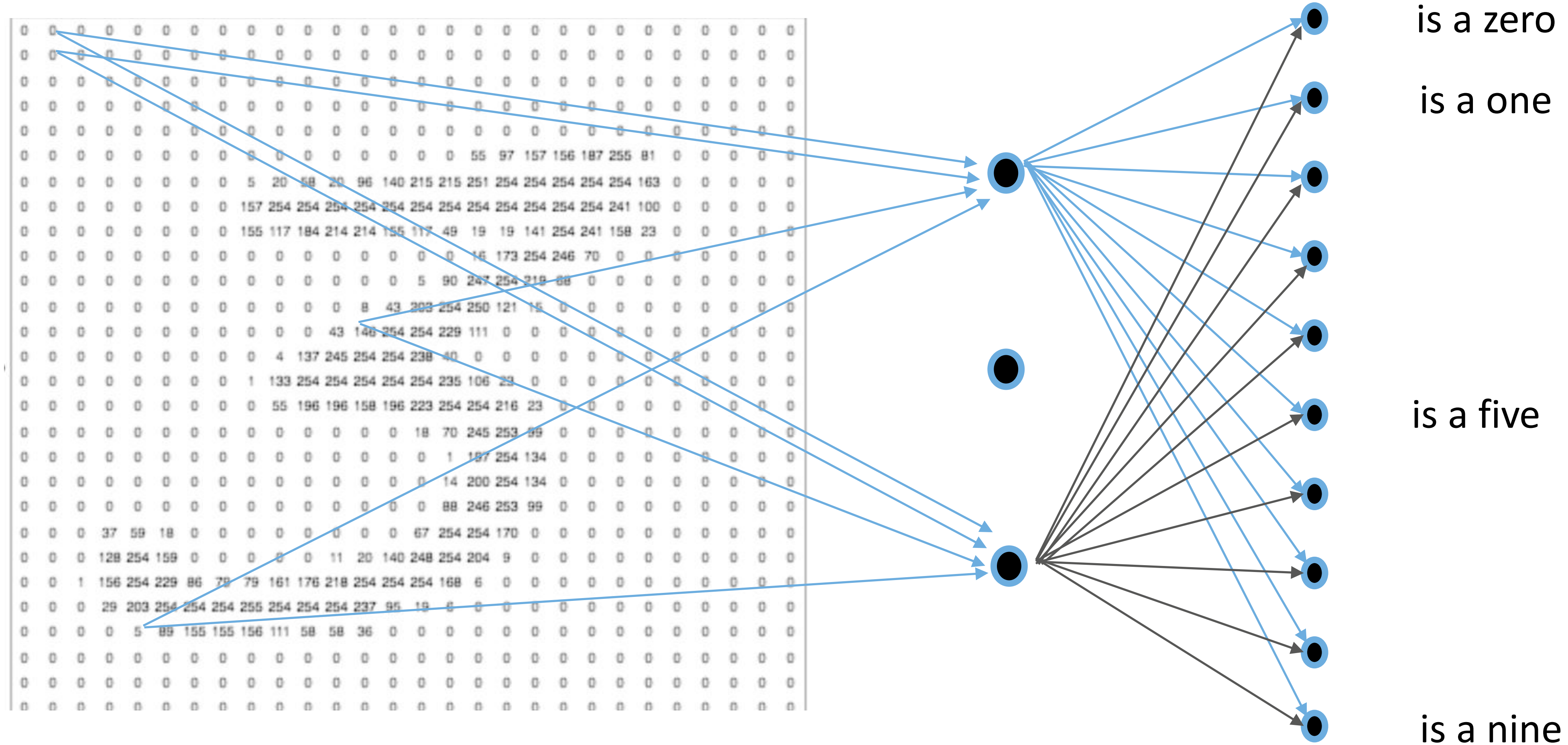
(inspired by a slide used in cs231n lecture from Stanford University)

On Image Representation



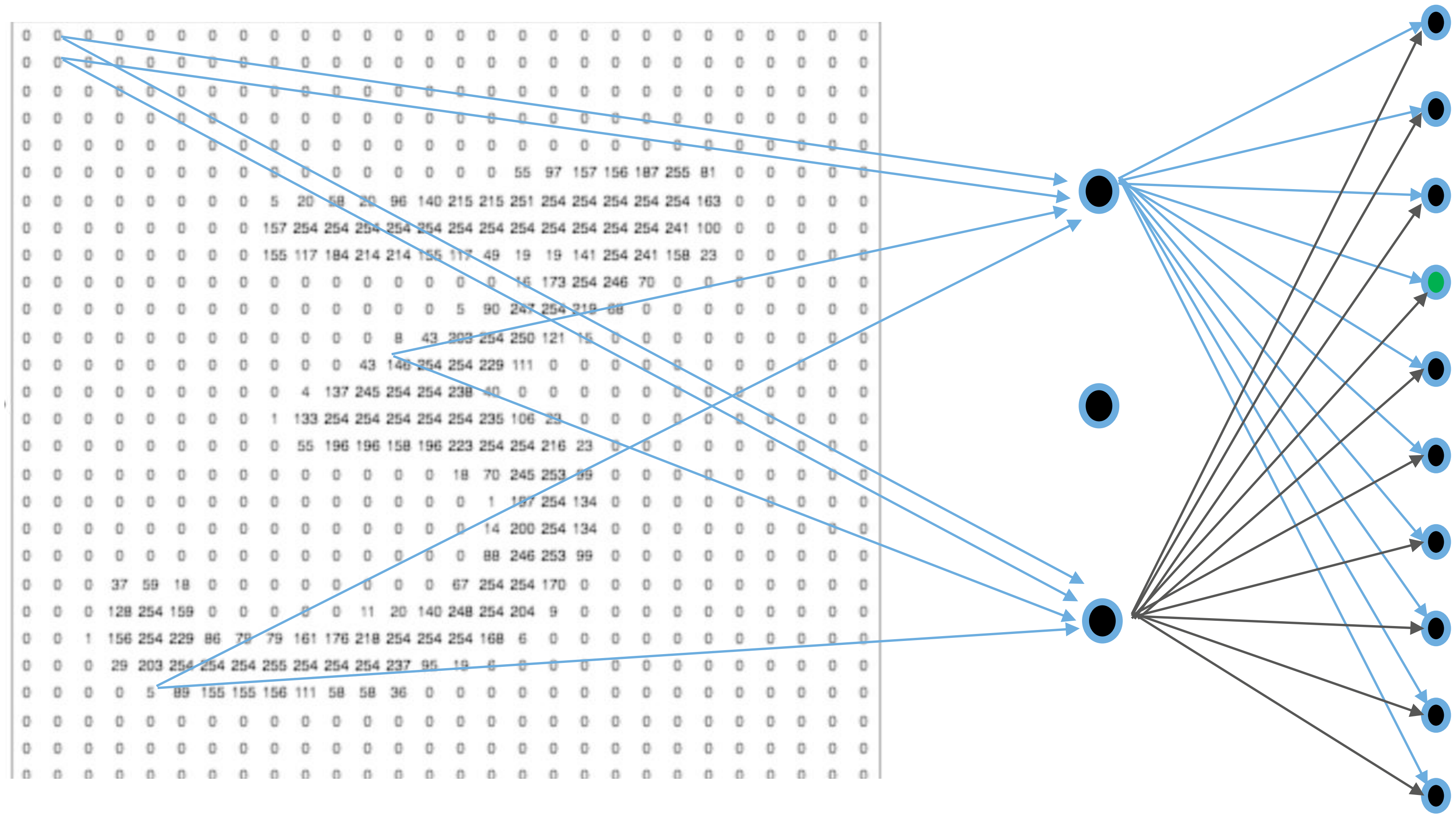
Neural Networks for Image Classification

Fully Connected Neural Network



Neural Networks for Image Classification

Fully Connected Neural Network



It is a three. The idea in training, modify the weights from previous layer to this one, so this output neuron provides 1 given that input and the rest of output neurons provides 0 given that input

Neural Networks for Image Classification

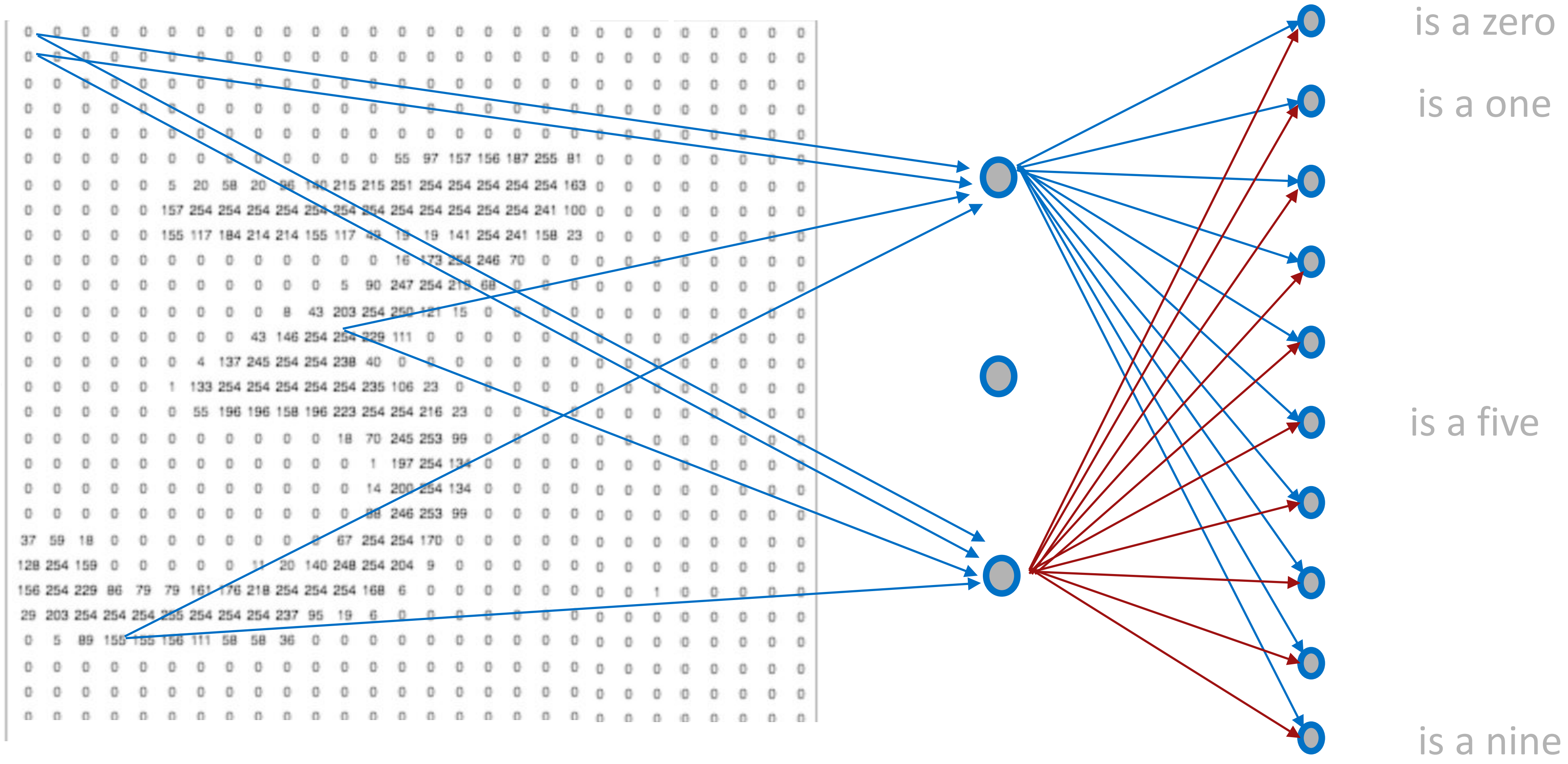
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	55	97	157	156	187	255	81	0	0	0	0	0
0	0	0	0	0	0	0	0	5	20	58	20	96	140	215	215	251	254	254	254	254	254	163	0	0	0	0	0
0	0	0	0	0	0	0	0	157	254	254	254	254	254	254	254	254	254	254	254	254	241	100	0	0	0	0	0
0	0	0	0	0	0	0	0	155	117	184	214	214	155	117	49	19	19	141	254	241	158	23	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	173	254	246	70	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	90	247	254	219	68	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	8	43	203	254	250	121	15	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	43	146	254	254	229	111	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	4	137	245	254	254	238	40	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	133	254	254	254	254	254	235	106	23	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	55	196	196	158	196	223	254	254	216	23	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	70	245	253	99	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	197	254	134	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	200	254	134	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	88	246	253	99	0	0	0	0	0	0	0	0	0
0	0	0	37	59	18	0	0	0	0	0	0	0	0	67	254	254	170	0	0	0	0	0	0	0	0	0	0
0	0	0	128	254	159	0	0	0	0	0	11	20	140	248	254	204	9	0	0	0	0	0	0	0	0	0	0
0	0	1	156	254	229	86	79	79	161	176	218	254	254	254	168	6	0	0	0	0	0	0	0	0	0	0	0
0	0	0	29	203	254	254	254	255	254	254	254	237	95	19	6	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	5	89	155	155	156	111	58	58	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

shift to the left

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	55	97	157	156	187	255	81	0	0	0	0	0	0	0	0
0	0	0	0	0	5	20	58	20	96	140	215	215	251	254	254	254	254	254	163	0	0	0	0	0	0	0	0
0	0	0	0	0	157	254	254	254	254	254	254	254	254	254	254	254	254	241	100	0	0	0	0	0	0	0	0
0	0	0	0	0	155	117	184	214	214	155	117	49	19	19	141	254	241	158	23	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	16	173	254	246	70	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	5	90	247	254	219	68	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	8	43	203	254	250	121	15	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	43	146	254	254	229	111	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	4	137	245	254	254	238	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	133	254	254	254	254	254	235	106	23	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	55	196	196	158	196	223	254	254	216	23	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	18	70	245	253	99	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	197	254	134	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	14	200	254	134	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	88	246	253	99	0	0	0	0	0	0	0	0	0	0	0	0
37	59	18	0	0	0	0	0	0	0	0	67	254	254	170	0	0	0	0	0	0	0	0	0	0	0	0	0
128	254	159	0	0	0	0	0	11	20	140	248	254	204	9	0	0	0	0	0	0	0	0	0	0	0	0	0
156	254	229	86	79	79	161	176	218	254	254	254	168	6	0	0	0	0	0	0	0	0	1	0	0	0	0	0
29	203	254	254	254	255	254	254	254	237	95	19	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	5	89	155	155	156	111	58	58	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

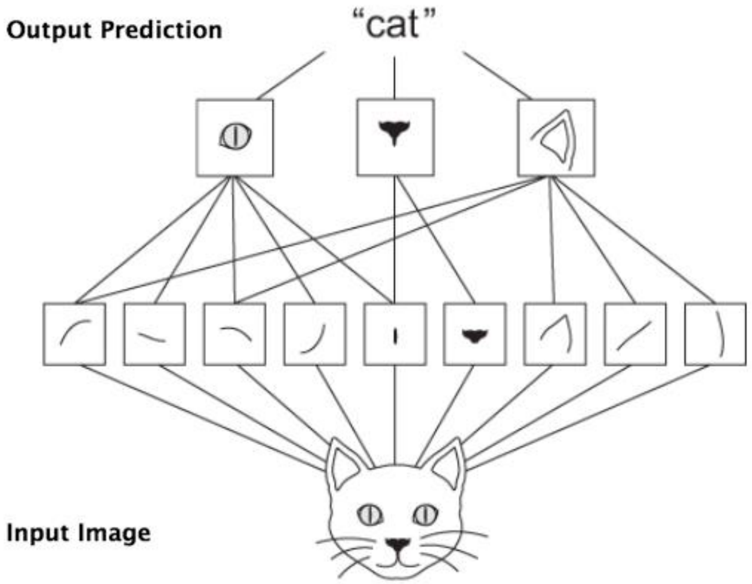
Neural Networks for Image Classification

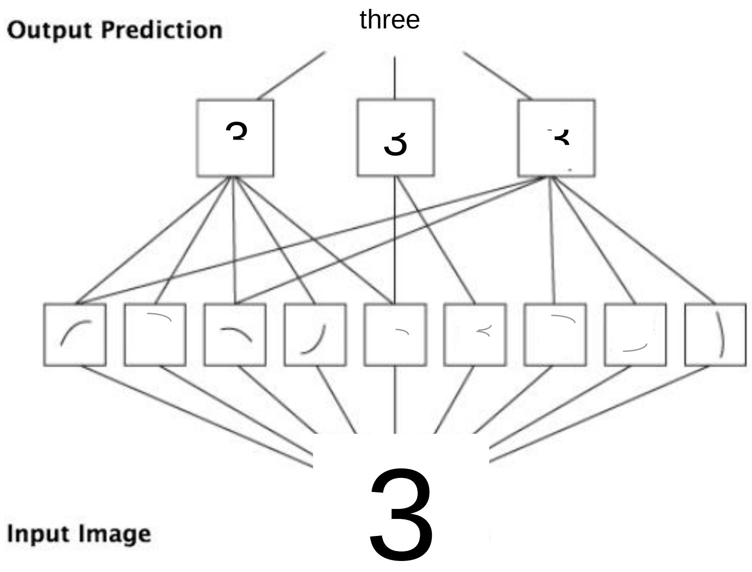
Fully Connected Neural Network



Neural Networks for Image Classification

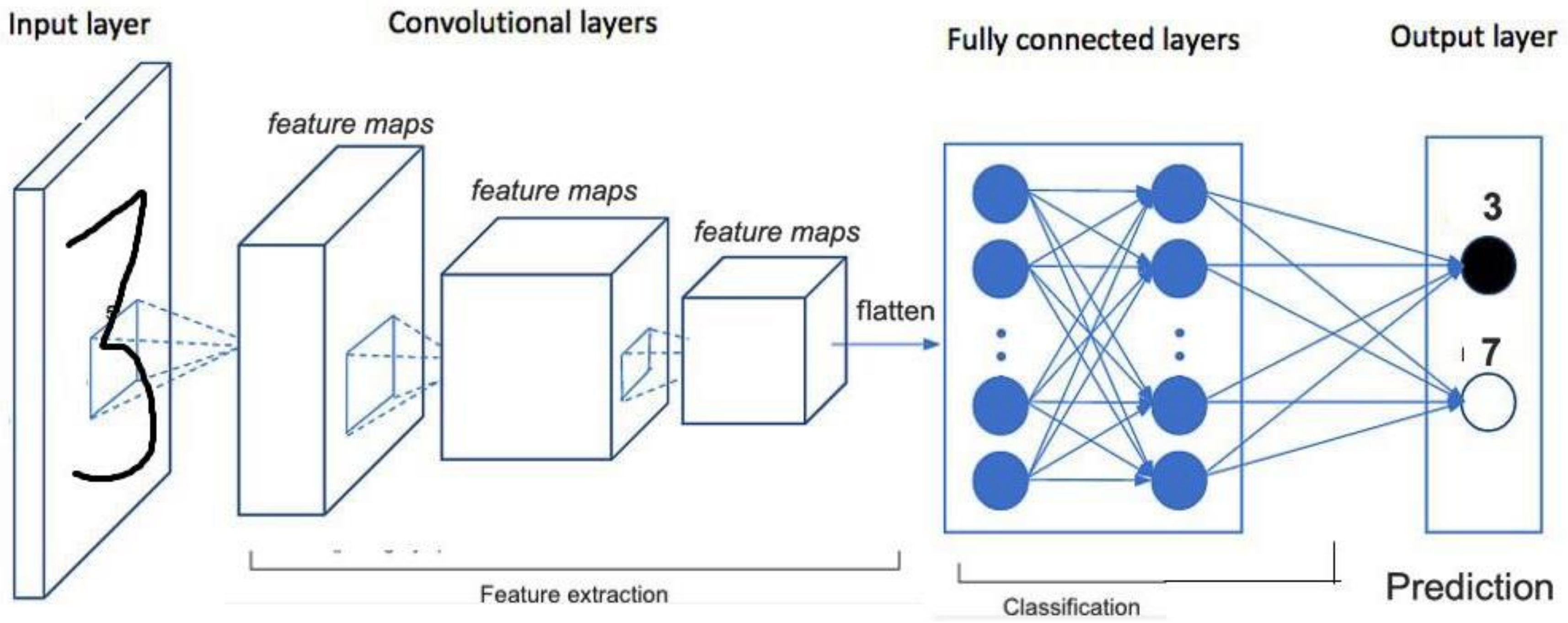
Convolutional Neural Networks





Neural Networks for Image Classification

Convolutional Neural Networks



TENSORFLOW

WHAT IS 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

RUNNING TENSORFLOW

- Construct a graph—this happens before any real computation happens
  - Specify your neural network as a graph
  - `Variables`--characteristics of the graph that can change over time
    - i.e., learned weights
  - Operations—computations that combine the variables and the data
    - e.g., convolution, activation, matrix multiply, etc.
- Launch a `session`
  - This is TF verbiage for executing a graph
  - Taking data and running it through a previously-created graph

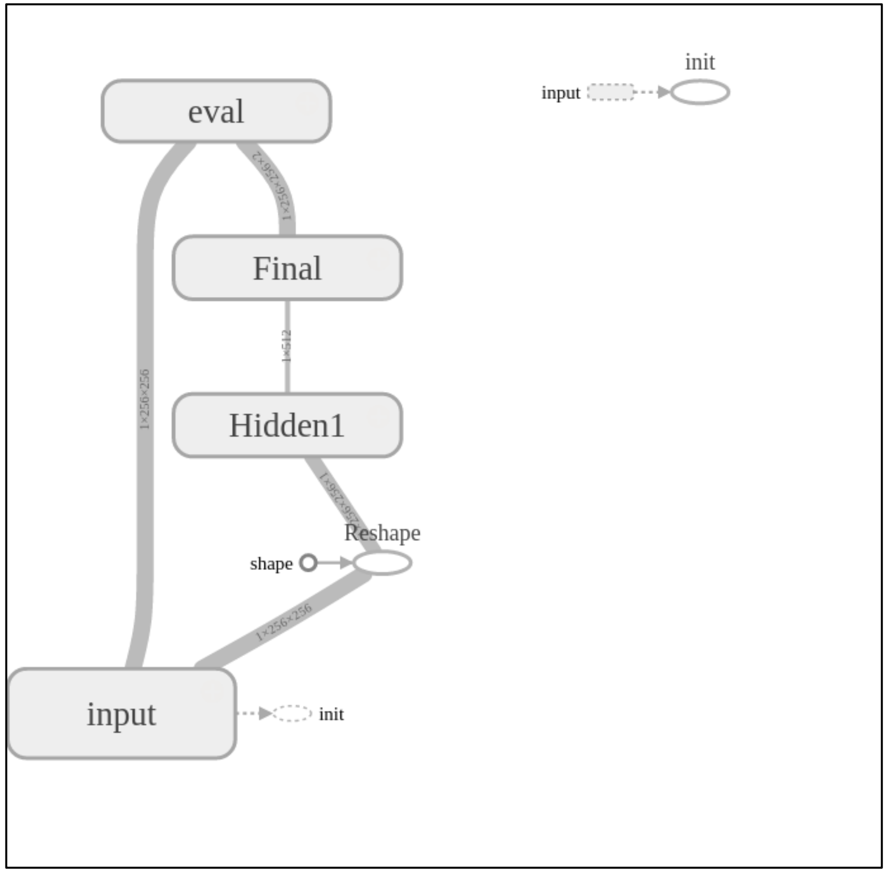
SAMPLE WORKFLOW

- Prepare input data
  - Can use numpy arrays but for very large datasets TFRecords are recommended
- Build the computation graph
  - Create inference, loss, training nodes
- Train the model
  - Inject input data into graph in a TF `session` and loop over your input data.
  - Specify things like batch size, number of epochs, learning rate, etc.
- Evaluate the model
  - Run inference on graph and then evaluate accuracy based on suitable metric

TENSORBOARD

- TF tool to visualize training progress
  - Plots of loss, learning rate, accuracy
  - Visualize computation graph
- Will use TensorBoard during this lab
  - Extremely useful to aggregate training and evaluation statistics for clear analysis of the model behavior

TENSORBOARD GRAPH EXAMPLE



- Evaluation graph for NN with 1 hidden layer

- Each box clicks to expand
  - Shows you the operations and the variables in each user-defined node

INFERENCE GRAPH EXAMPLE

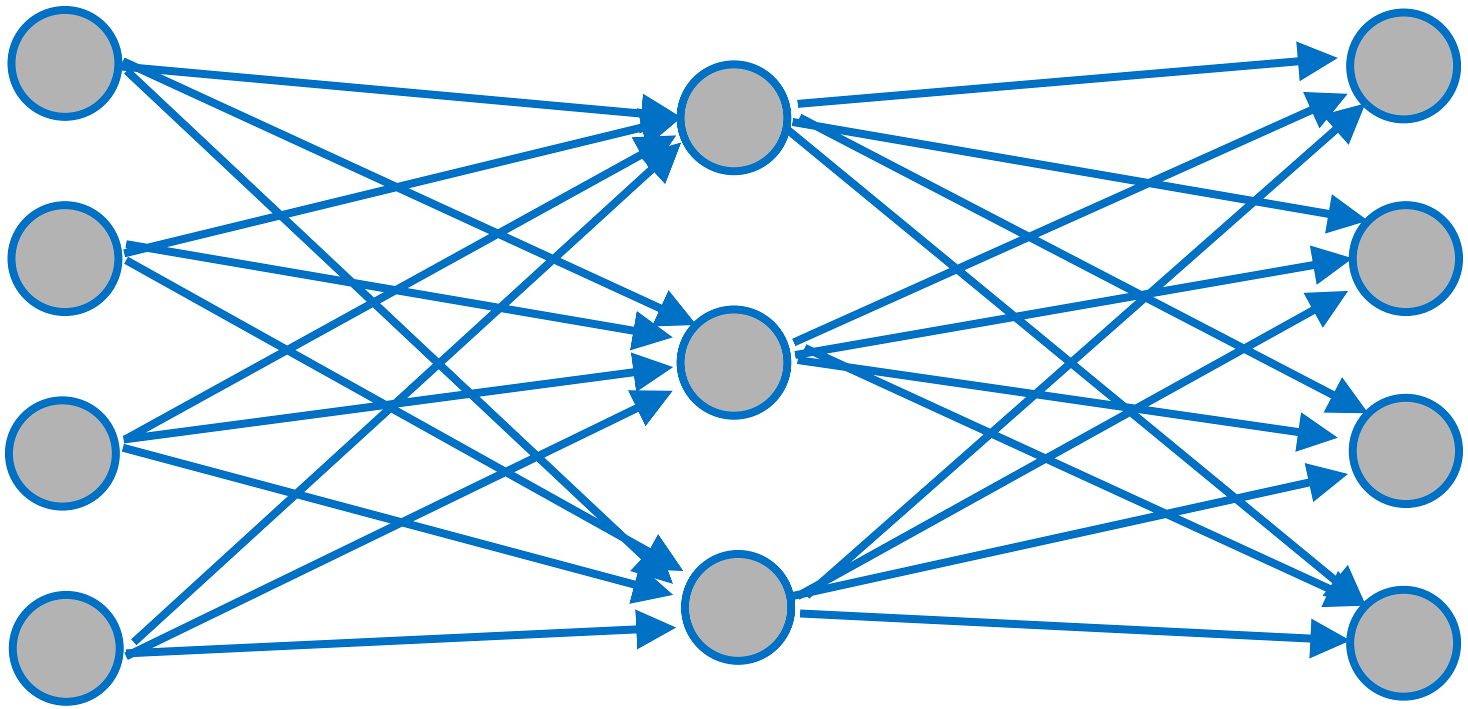
```
with tf.name_scope('Hidden1'):
    W_fc = tf.Variable(tf.truncated_normal( [256*256, 512],
                 stddev=0.1, dtype=tf.float32), name='W_fc')
    flatten1_op = tf.reshape( images_re, [-1, 256*256])
    h_fc1 = tf.matmul( flatten1_op, W_fc )

with tf.name_scope('Final'):
    W_fc2 = tf.Variable(tf.truncated_normal( [512, 256*256*2],
                stddev=0.1, dtype=tf.float32), name='W_fc2' )
    h_fc2 = tf.matmul( h_fc1, W_fc2 )
    h_fc2_re = tf.reshape( h_fc2, [-1, 256, 256, 2] )

return h_fc2_re
```

TASK 1 - NEURAL NETWORK

One hidden layer only



Input layer
65536 (256 x 256) pixels

hidden layer
512

output layer
256 x 256 x 2

Back to Today's Lab 1

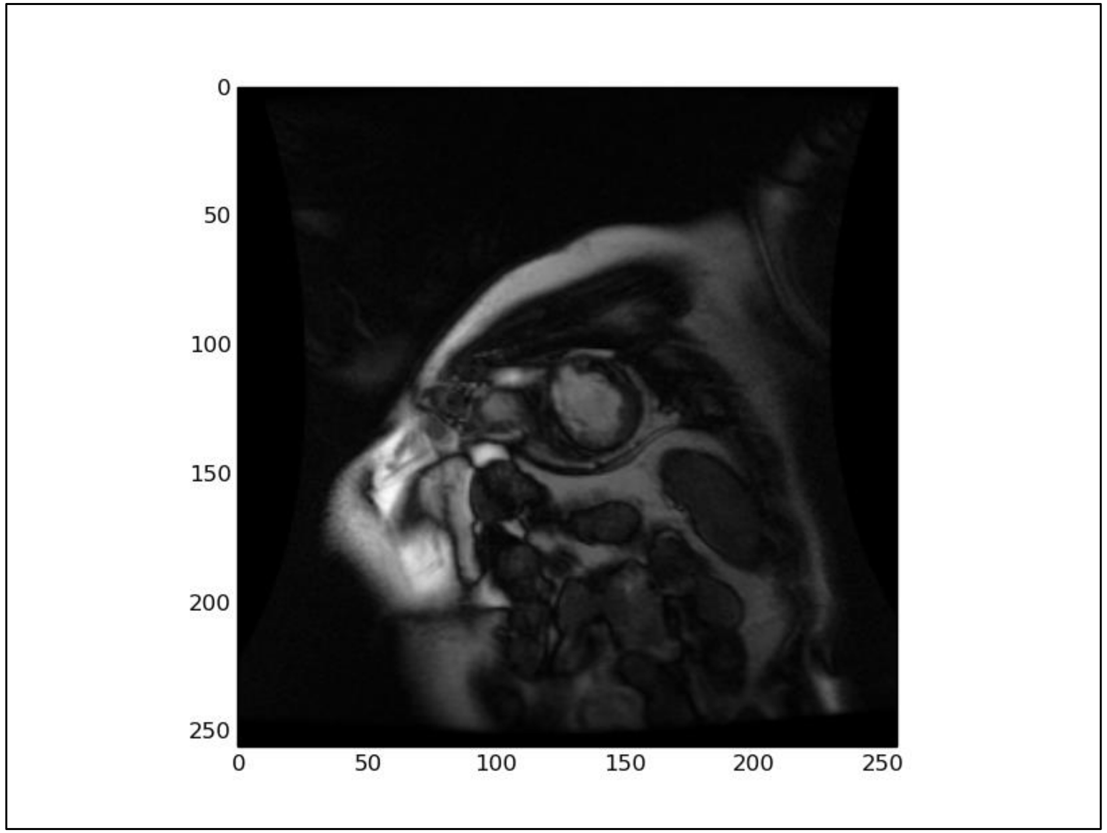
IMAGE SEGMENTATION

- “Segmentation” sometimes used to describe similar but slightly different tasks
- In this lab, semantic segmentation will be performed
  - i.e., in an image, each pixel will be placed into one of multiple classes
- In a sense it’s a classification problem where each pixel has a class, vs image recognition where each image (collection of pixels) has a class
- Specifically we’ll be looking at medical imaging data and attempting to determine where the left ventricle (LV) is
  - i.e., for each pixel is it part of LV or not?

DATASET

- Cardiac MRI short-axis (SAX) scans
  - Sunnybrook cardiac images from earlier competition
    http://smial.sri.utoronto.ca/LV_Challenge/Data.html
  - "Sunnybrook Cardiac MR Database" is made available under the CC0 1.0 Universal license described above, and with more detail here: http://creativecommons.org/publicdomain/zero/1.0/
  - Attribution:
    - Radau P, Lu Y, Connelly K, Paul G, Dick AJ, Wright GA. "Evaluation Framework for Algorithms Segmenting Short Axis Cardiac MRI." The MIDAS Journal -Cardiac MR Left Ventricle Segmentation Challenge, http://hdl.handle.net/10380/3070

IMAGE EXAMPLE



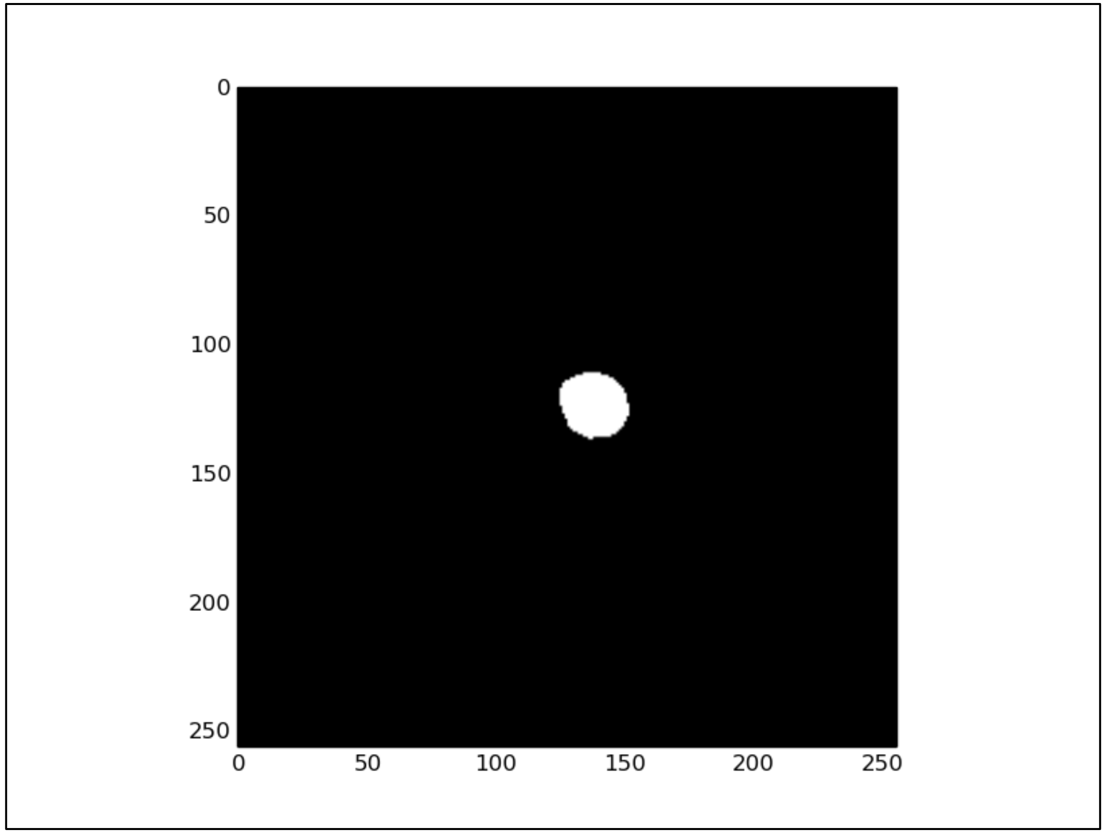
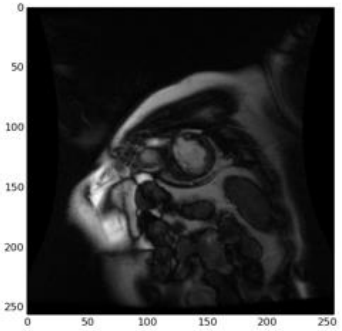
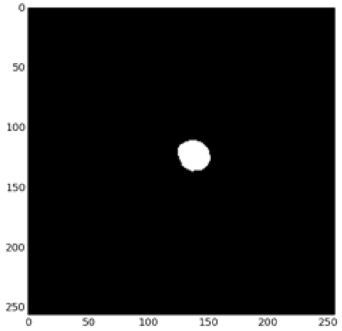
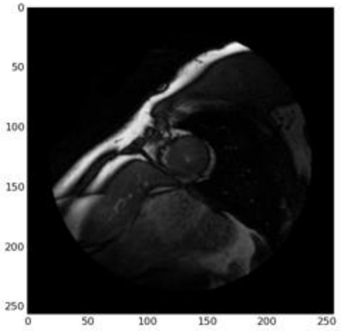


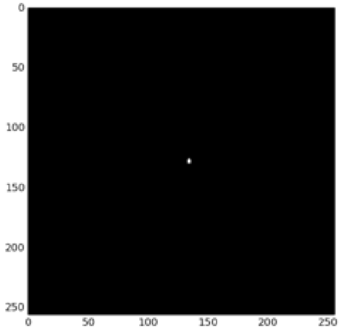
IMAGE EXAMPLES

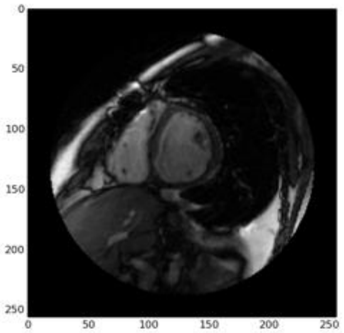
Complete images and expertly labeled contours of LV

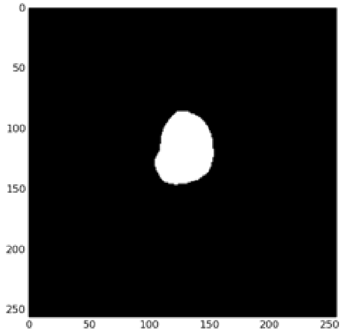


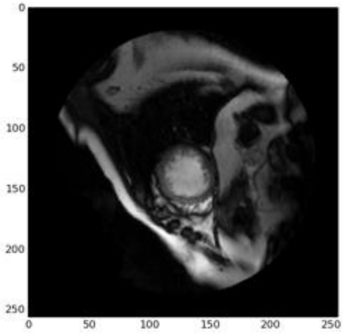


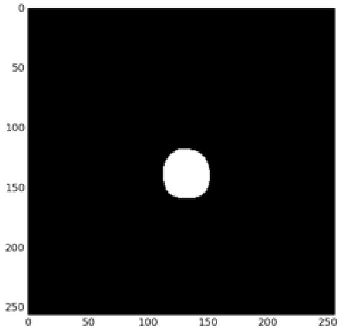












DATA DETAILS

- Original images are 256 x 256 grayscale DICOM format
- Output is a tensor of size 256 x 256 x 2
  - Each pixel belongs to one of two classes
- Training set consist of 234 images
- Validation set consist of 26 images

BACKGROUND DATA SETUP

- Lots of guidance and code for how to setup/extract data taken from here:
  - https://www.kaggle.com/c/second-annual-data-science-bowl/details/deep-learning-tutorial
- Images and contours have been extracted from the raw data and packaged up for ingest into TensorFlow
  - Data extraction code is included but won't be demo'd.
- TensorFlow data records provided but raw data is NOT provided for this lab
  - If interested you can download yourself

TASK 1

`Ensure things are working properly!`

- Train and test a fully-connected neural network with one hidden layer
  - Visual representation of this network appears on next slide
- For the loss computation we'll use TF built-in **`sparse_softmax_cross_entropy_with_logits`**
  - Computes softmax of the inference output then cross entropy against the correct labels

TASK 1 - TRAINING OUTPUT

```
!python exercises/simple/runTraining.py --data_dir /data
```

Output:

```
OUTPUT: Step 0: loss = 2.621 (0.169 sec)

OUTPUT: Step 100: loss = 4.958 (0.047 sec)

OUTPUT: Step 200: loss = 4.234 (0.047 sec)

OUTPUT: Done training for 1 epochs, 231 steps.
```

Lots of messages printed to the screen - look for “OUTPUT”

TASK 1 - EVALUATION

```
!python exercises/simple/runEval.py --data_dir /data
```

Output:

```
OUTPUT: 2016-08-23 15:37:26.752794: accuracy = 0.504

OUTPUT: 26 images evaluated from file
/tmp/sunny_data/val_images.tfrecords
```

- Output shows the accuracy of the predictions and which data was utilized
  - 1.0 means the NN classified all the data the same as the label, ie 100% correct

TASK 2 - ADDITIONAL LAYERS

- Convolution layers
  - Previous example focused on each input pixel
  - What if features encompass multiple input pixels
  - Can use convolutions to capture larger receptive fields
- Pooling layers
  - Essentially a down-sampling method retaining information while eliminating some computational complexity

TASK 2 - FULLY CONVOLUTIONAL NETWORK (FCN)

- Image classification layers—Convolutions, pooling, activations, fully connected
  - Output is an N-dimensional vector where N == Number_of_classes
- Can we leverage this network to do segmentation? YES!
- Reconsider the problem as pixel classification
  - i.e., each pixel has a class
- Reuse most of the image classification network
- Replace fully connected layer(s) with deconvolution (transpose convolution)
  - Output is a 256 x 256 x N tensor where N == Number_of_classes
  - In this lab N == 2

TASK 2 - ADDITIONAL LAYER

- Deconvolution (transpose convolution) layer
  - Up-sampling method to bring a smaller image data set back up to it's original size for final pixel classification
- Long et al (CVPR2015) has nice paper re: FCN for segmentation
  - Created FCNs from AlexNet and other canonical networks
- Zeiler et al (CVPR2010) describes deconvolution
- Network we will use is very similar to Vu Tran's kaggle example here: https://www.kaggle.com/c/second-annual-data-science-bowl/details/deep-learning-tutorial

TASK 2

exercises/tf/segmentation/cnn/neuralnetwork.py

- Finish the CNN, replace "FIXME"
  - vi / vim the file and type /FIXME to identify where to make changes
  - You need to figure out the dimensions
- Convolution1, 5x5 kernel, stride 2; Maxpooling1, 2x2 window, stride 2
- Convolution2, 5x5 kernel, stride 2; Maxpooling2, 2x2 window, stride 2
- Convolution3, 3x3 kernel, stride 1; Convolution4, 3x3 kernel, stride 1
- Score_classes, 1x1 kernel, stride 1; Upscore (DeConv), 31x31 kernel, stride 16
- Optional / Time Permitting: Experiment with num_epochs

TASK 2 - EVALUATION RESULTS

- 1 epoch of training

```
OUTPUT: 2016-08-26 20:44:55.012370: precision = 0.571
```

- 30 epochs of training

```
OUTPUT: 2016-08-26 20:48:16.593103: precision = 0.985
```

- 98.5% accurate!
    - Very good accuracy, are we done?

TASK 2 - ACCURACY

- How are we determining accuracy
  - We are comparing the pixel value in the label with the value computed by the CNN
    - So 98.5% of the time we are predicting the pixel correctly
- However, the size of the contour is relatively small compared to the entire image Class imbalance problem
- If we simply output the notLV class for every pixel we'd have over 95% accuracy
  - Clearly this isn't what we want

TASK 3 - DICE METRIC

- Metric to compare the similarity of two samples:

$$\frac{2A_{nl}}{A_n + A_l}$$

  - Where:
    - A_n is the area of the contour predicted by the network
    - A_l is the area of the contour from the label
    - A_{nl} is the intersection of the two
      - The area of the contour that is predicted correctly by the network
      - 1.0 means perfect score.

- More accurately compute how well we're predicting the contour against the label

- We can just count pixels to give us the respective areas

TASK 3 - TRAINING PARAMETERS

Important to search the space of parameters

- learning_rate: initial learning rate
- decay_rate: the rate that the initial learning rate decays
    - e.g., 1.0 is no decay, 0.5 means cut the decay rate in half each number of (decay) steps
- decay_steps: number of steps to execute before changing learning rate
- num_epochs: number of times to cycle through the input data
- batch_size: keep at 1 for now
- Experiment with **learning_rate, decay_rate, decay_steps, num_epoch**
- Record the parameters that give you the best Dice score

TASK 3 - EVALUTION RESULTS

- Recall result from prior example:
  - 1 epoch: `precision = 0.501`
  - 30 epochs: `precision = 0.985`
- Now with Dice metric (recall 1.0 is perfect accuracy)
  - 1 epoch: Dice metric = 0.033
  - 30 epochs: Dice metric = 0.579
- Not as good as we originally thought

TASK 3 - RESULT

One possible result

--learning_rate=0.03

--decay_rate=0.75

--num_epochs=100

--decay_steps=10000

`OUTPUT: 2016-08-26 21:22:15.590642: Dice metric = 0.861`

Accuracy now looking much better!

LAB REVIEW

LAB SUMMARY

- Intro to image segmentation
    - Classifying pixels vs images
- Converted image recognition network into FCN for segmentation.
- Used TensorFlow as framework to explore various optimizations to FCN
- Explored new accuracy metric (Dice metric) to better capture true accuracy

WHAT ELSE?

- Run training longer
  - For demo purposes we ran really short training runs
  - Need more epochs
- More training data
  - We only had 236 images in our training set
  - Gather more data
  - Augment images that we have with rotations, inversions, etc.
    - TF has functions to flip/rotate/transpose automatically
- Larger more complicated networks

DEEP
LEARNING
INSTITUTE

WORD GENERATION WITH TENSORFLOW

PD Dr. Juan J. Durillo

Certified Instructor, NVIDIA Deep Learning Institute
NVIDIA Corporation

TOPICS

- Overview
- Recurrent Neural Networks
- One-Hot Encoding
- Lab
  - Discussion / Overview
  - Launching the Lab Environment
  - Lab Review

NON-IMAGE DATA

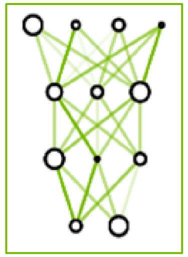
- Convert to images
  - Sound waves
  - Stock prices
- New workflows
  - Different input and output types
  - Handle new components like time
  - Still learned input->output mappings

NON-IMAGE DATA

IMAGES – INPUT AND OUTPUT

Classifier data flow

100	37	59	87	55	29	13	44
62	79	54	62	23	93	93	26
50	57	93	17	67	53	60	75
3	54	70	37	17	20	69	7
86	42	2	55	90	45	74	77
59	39	100	52	10	8	20	37
61	2	62	92	83	18	12	82
11	7	87	20	5	13	4	34



Deep Neural Network

0.04	0	0.02	0.01	0.92	0.01
Kites	Harrier	Vulture	Hawk	Eagle	Buzzards

Eagle

One-Hot: Turning words into Numbers

- Numerical vector representation for each word
- Dictionary of N words
- Each word is a vector with N-1 zeros and one 1, at the position of the word in the dictionary
- A document can be represented as a sequence of these one-hot vectors
- One interesting property of this representation is that no information gets lost

ONE-HOT ENCODING

```
small_dict=['EOS','a','my','sleeps','on','dog','cat','the','bed','floor'] #'EOS' means end of sentence.
```

```
import numpy as np #numpy is "numerical python" and is used in deep learning mostly for its n-dimensional array
X=np.array([[2,6,3,4,2,8,0],[1,5,3,4,7,9,0]],dtype=np.int32)
print([small_dict[ind] for ind in X[1,:]]) #Feel free to change 1 to 0 to see the other sentence.
```

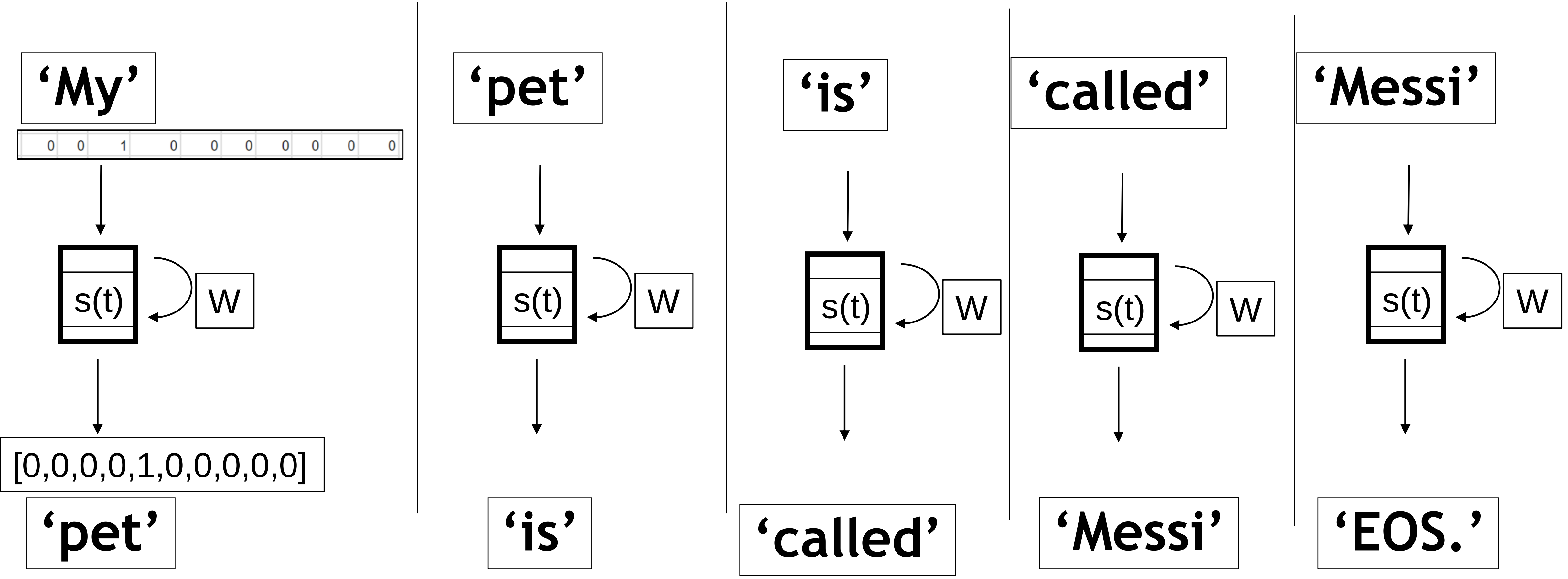
```
['a', 'dog', 'sleeps', 'on', 'the', 'floor', 'EOS']
```

```
one-hot encoded inputs
[[[ 0.  0.  1.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  1.  0.  0.  0.]
  [ 0.  0.  0.  1.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  1.  0.  0.  0.  0.  0.]
  [ 0.  0.  1.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.  1.  0.]
  [ 1.  0.  0.  0.  0.  0.  0.  0.  0.  0.]]

 [[ 0.  1.  0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  1.  0.  0.  0.  0.]
  [ 0.  0.  0.  1.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  1.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  1.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  1.]
  [ 1.  0.  0.  0.  0.  0.  0.  0.  0.  0.]]]
shape of the input
(2, 7, 10)
```

RECURRENT NEURAL NETWORKS

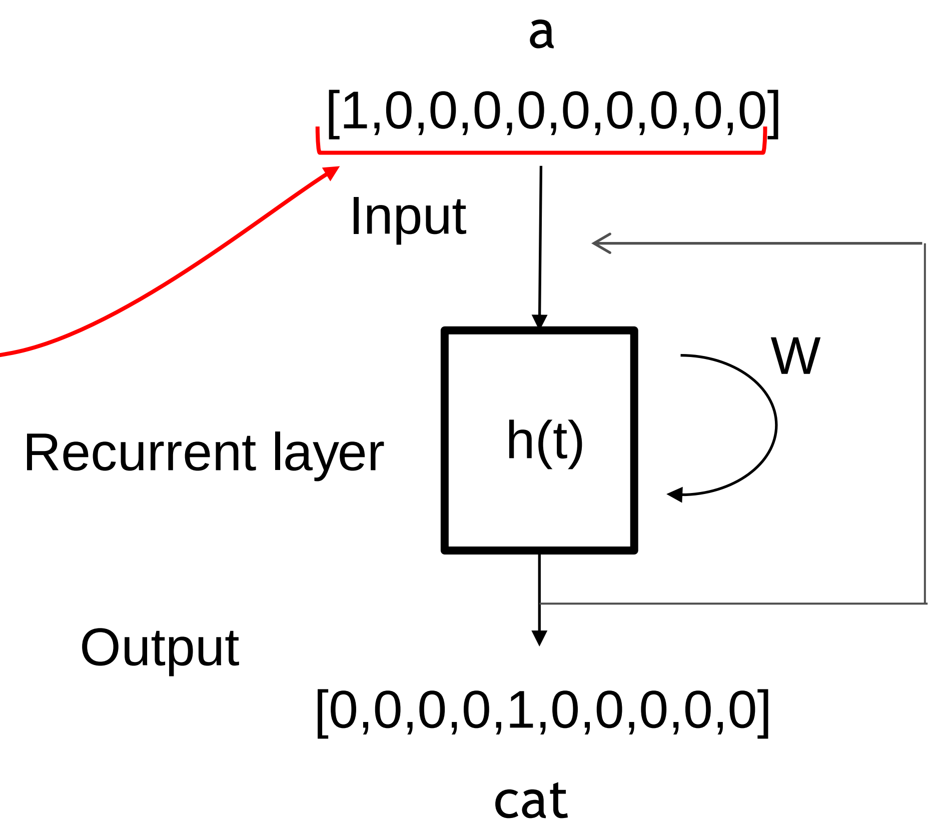
Generating Language



RECURRENT NETWORK EXAMPLE

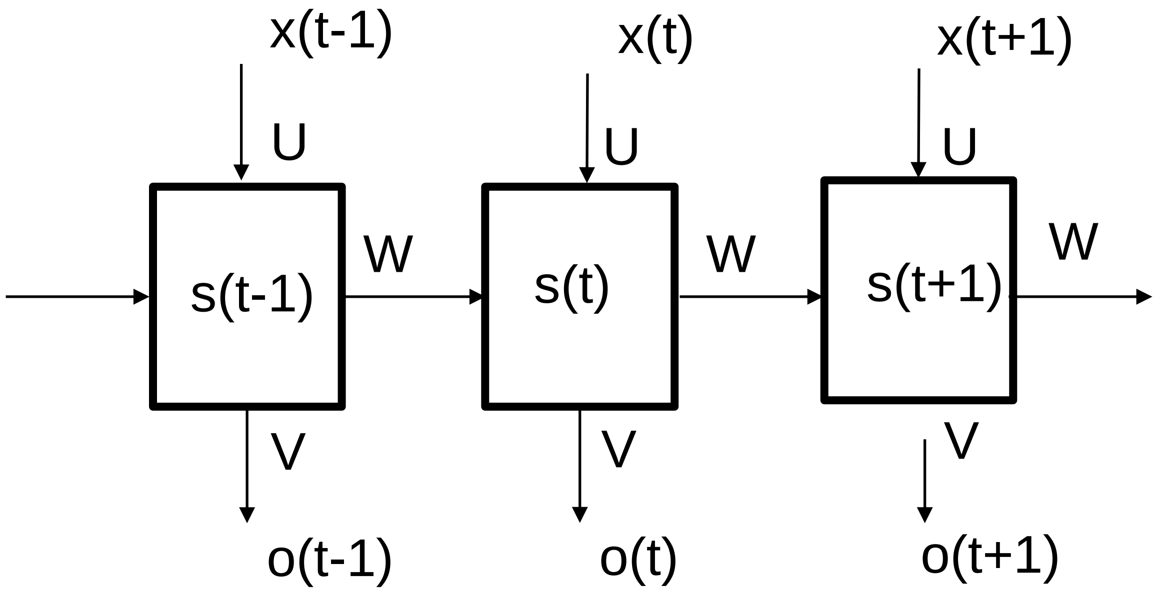
Word prediction example

a	1	0	0	0	0	0	0	0	0	0
cat	0	0	0	0	1	0	0	0	0	0
is	0	0	0	1	0	0	0	0	0	0
on	0	0	1	0	0	0	0	0	0	0
the	0	1	0	0	0	0	0	0	0	0
grass	0	0	0	0	0	0	0	0	1	0



RECURRENT NETWORK EXAMPLE

a	the	on	is	cat	park	play	swin g	grass	sitting
0	1	2	3	4	5	6	7	8	9



Unrolled Recurrent Layer

[0 , 4 , 3 , 2 , 1 , 8]

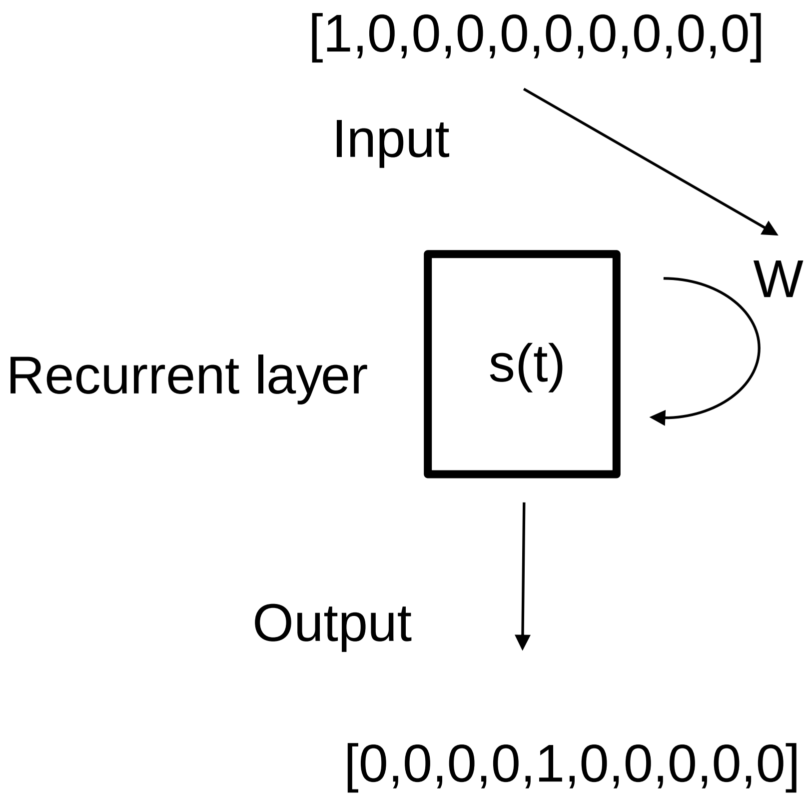
A cat is on the grass.

RNNs learn by reducing the error between their predicted next word and the actual next word in a corpus. RNNs are structured to "remember" the words that led to their prediction.

TIME SERIES INFORMATION

Recurrent neural networks are a popular approach

Demonstrated effectiveness with sentence and code creation as well as language translation.



LAB TASK 1

- Task 1:
  - How does an RNN learn?
  - Why use a deeper network?
  - What does dropout do?

LAB TASK 1

- Task 1:
  - How does an RNN learn?
  - Why use a deeper network?
  - What does dropout do?

LAB TASK 2

- What could we do to improve performance?

- How many steps are you using?

- How many layers do you have?

PART 2 RECURRENT NEURAL NETWORK

- What could we do to improve performance?
  - Answer: Increase the number of hidden units, change dropout, change learning rate and add a learning policy
- How many steps are you using?
  - Answer: 20
- How many layers do you have?
  - Answer: 2

DEEP
LEARNING
INSTITUTE

IMAGE CAPTIONING

PD Dr. Juan J. Durillo

Senior Deep Learning Certified Instructor, NVIDIA Deep Learning Institute
NVIDIA Corporation

TOPICS

- Lab Structure
- Image Captioning
- Video Captioning

LAB STRUCTURE

JUPYTER NOTEBOOKS

- Landing notebook contain links to:
  - Image Captioning notebook
  - Video Captioning notebook
  - Reference notebook - from Lab 2

TRAINING DATA / NETWORK

IMAGE CAPTIONING

- Microsoft Common Object in Common (MSCOCO)
  - Images
  - Five captions for each image
- VGG 16 network
  - Visual Geometry Group

THE PROCESS – IMAGE CAPTIONING

1. Import libraries
2. Evaluate data / Pixel to Content
   a. Feature vector – FC7
3. Align captions with images
   a. Will work with a subset of the data
4. Predict next word
   a. Similar to Lab 2
   b. Parse, tokenize, etc.

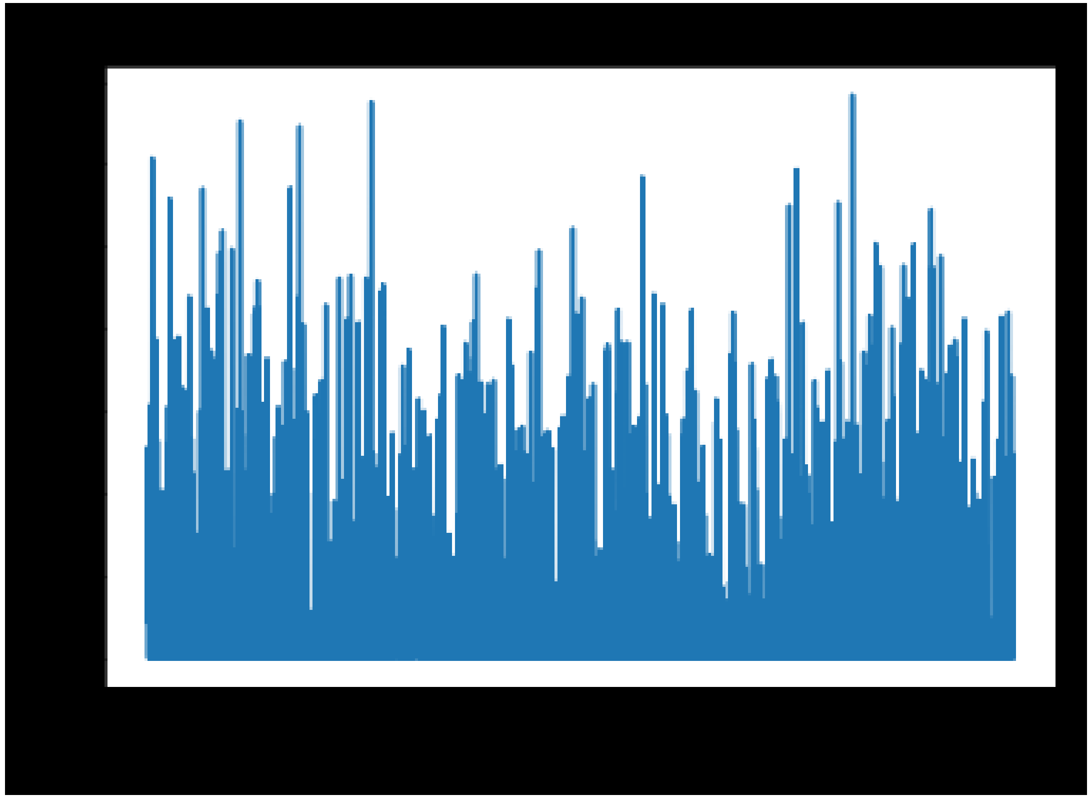


IMAGE CAPTIONING

image	0	0	0	0	0	0	0	0	0	0
a	1	0	0	0	0	0	0	0	0	0
dog	0	0	0	0	1	0	0	0	0	0
with	0	0	0	1	0	0	0	0	0	0
cake	0	0	1	0	0	0	0	0	0	0

a

[0.2,0.001,5e-2,….,0.2,0.9,0,0,0,0,0,0,0,0,0,0]

Input

Recurrent layer

h(t)

W

Output

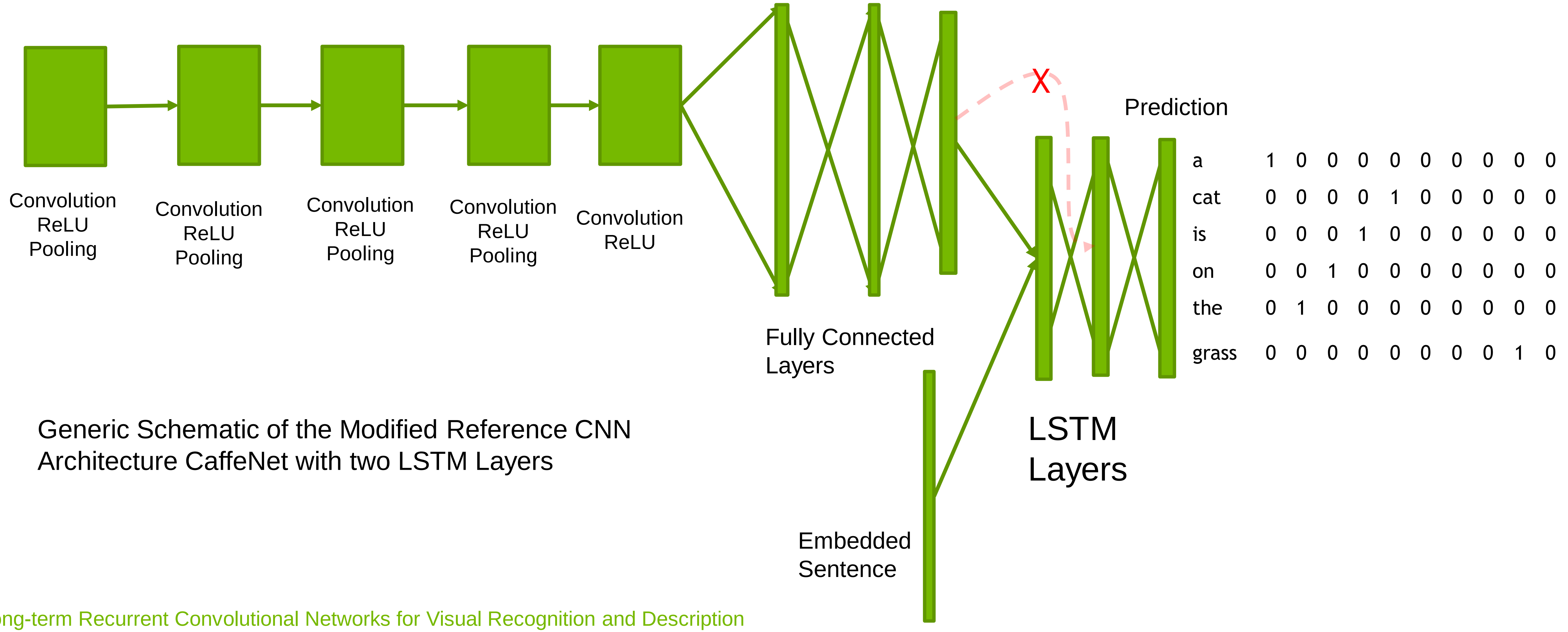
[1,0,0,0,0,0,0,0,0,0]

a

THE PROCESS – IMAGE CAPTIONING

5. Architect the network (RNN)
6. Train / build model
7. Evaluate a training image & captions
8. Generate a caption for a validation image
9. RUN LAST CODE BLOCK TO FREE GPU MEMORY

LAB 3 - IMAGE CAPTIONING



Generic Schematic of the Modified Reference CNN Architecture CaffeNet with two LSTM Layers

Long-term Recurrent Convolutional Networks for Visual Recognition and Description
Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, Trevor Darrell

EXAMPLE CAPTION RESULTS

CaffeNet	A white bird standing on top of a sandy beach.
VGG	A small bird standing on the ground.

CaffeNet	A white cat sitting on a chair.
VGG	A white and white cat laying on a white chair.

CaffeNet	A white horse standing in a lush field of grass.
VGG	A white horse standing in a field next to a fence.

CaffeNet	A bunch of bananas that are on a table.
VGG	A close up of a bunch of white flowers.

*Results shown here were generated using work from this paper.
Long-term Recurrent Convolutional Networks for Visual Recognition and Description
Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, Trevor Darrell

CONCLUSION

- Image and video captioning based on two papers
  - Translating Videos to Natural Language Using Deep Recurrent Neural Networks
  - Long-term Recurrent Convolutional Networks for Visual Recognition and Description
- Multiple approaches for image and video captioning – only one was used here

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

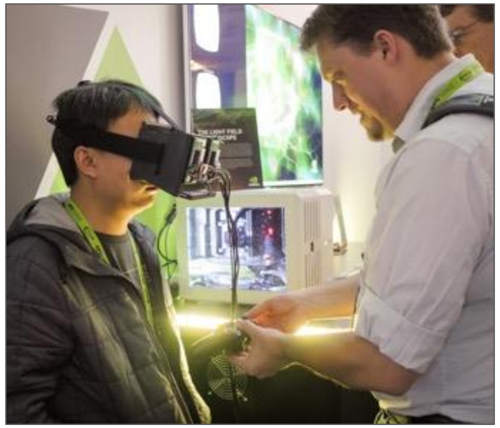
Visit www.nvidia.com/dli for workshops in your area.

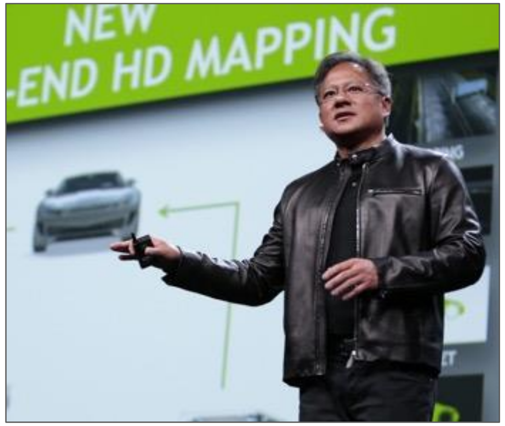
JOIN DEVELOPER PROGRAM

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