



DEEP  
LEARNING  
INSTITUTE

# Image Classification with DIGITS

---

NVIDIA Deep Learning Institute



# 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

# Agenda

- Intro to Deep Learning
- Training vs. Programming
- Train our first neural network - Lab
- How networks “learn”
- Increasing performance - Lab
- Next Steps

**WHAT IS DEEP LEARNING?**

# ACCOMPLISHING COMPLEX GOALS

## ARTIFICIAL INTELLIGENCE

Early artificial intelligence stirs excitement.



## MACHINE LEARNING

Machine learning begins to flourish.



## DEEP LEARNING

Deep learning breakthroughs drive AI boom.



1950's

1960's

1970's

1980's

1990's

2000's

2010's

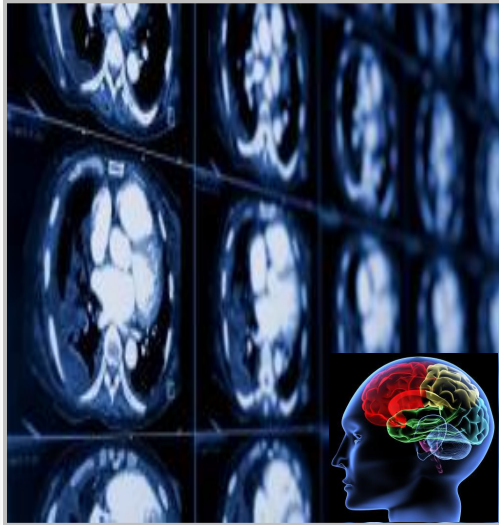
# SWEEPING ACROSS INDUSTRIES

## Internet Services



- Image/Video classification
- Speech recognition
- Natural language processing

## Medicine



- Cancer cell detection
- Diabetic grading
- Drug discovery

## Media & Entertainment



- Video captioning
- Content based search
- Real time translation

## Security & Defense



- Face recognition
- Video surveillance
- Cyber security

## Autonomous Machines

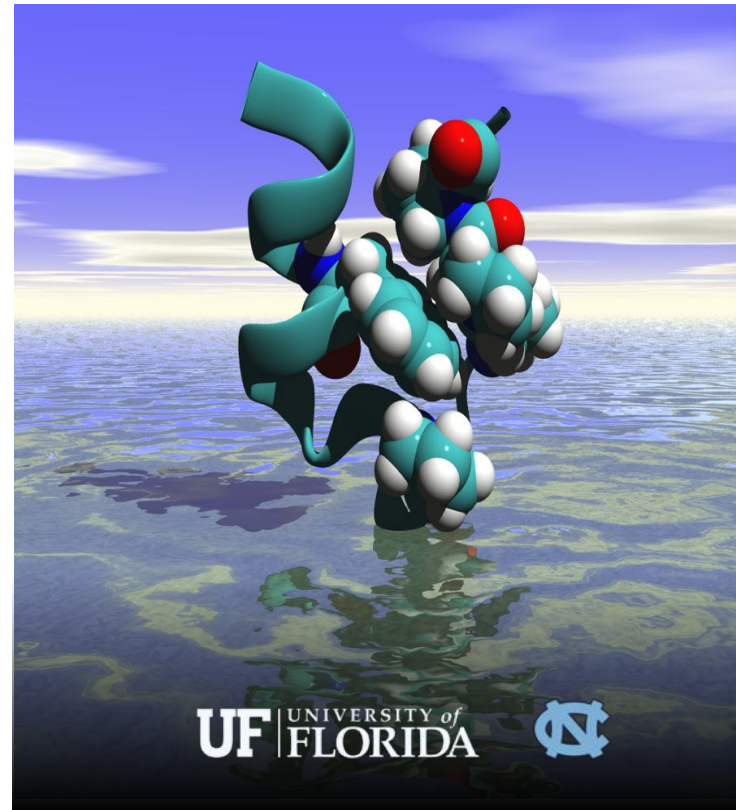


- Pedestrian detection
- Lane tracking
- Recognize traffic signs

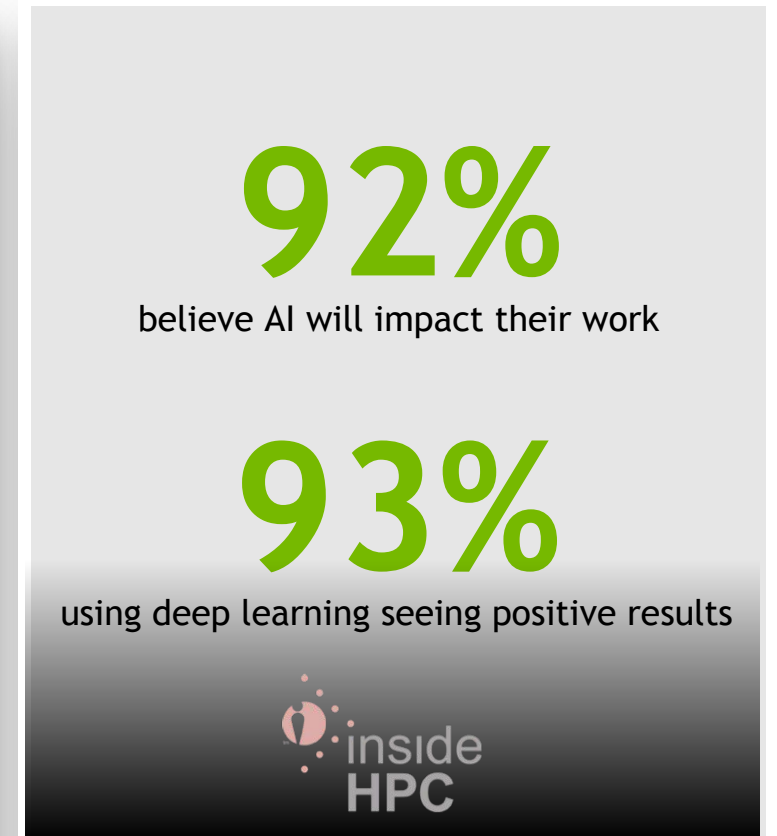
# TRANSFORMING RESEARCH



“Seeing” Gravity In Real Time



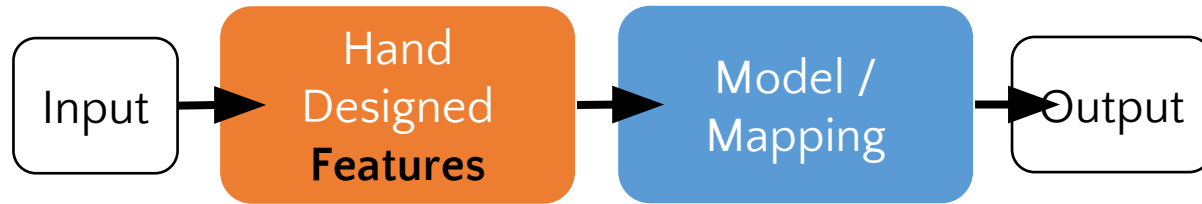
Accelerating Drug Discovery



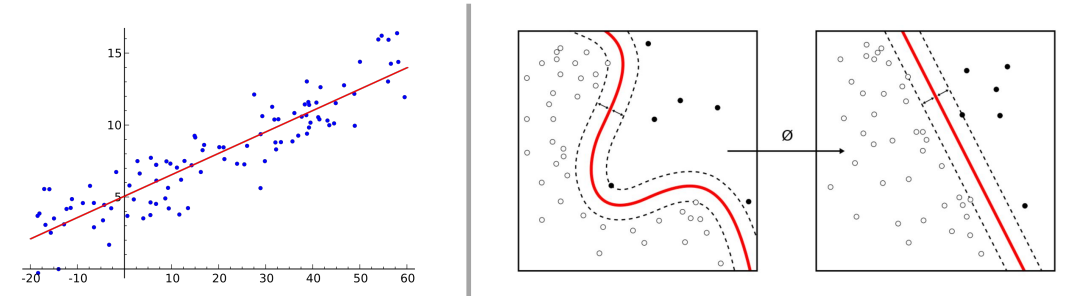
insideHPC.com Survey  
November 2016

# Difference in Workflow

Classic Machine Learning [ 1990 : now ]



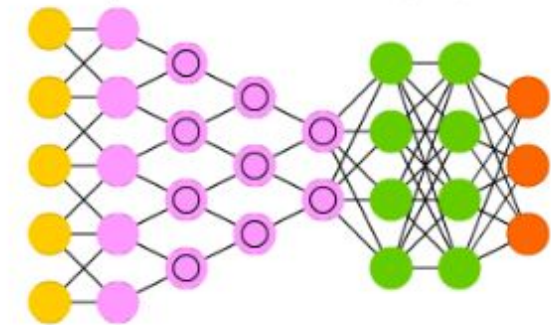
Examples [ Regression and SVMs ]



Deep/End-to-End Learning [ 2012 : now ]



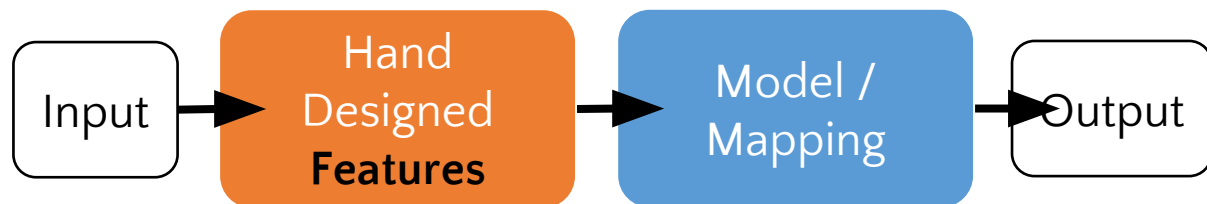
Example [ Conv Net ]



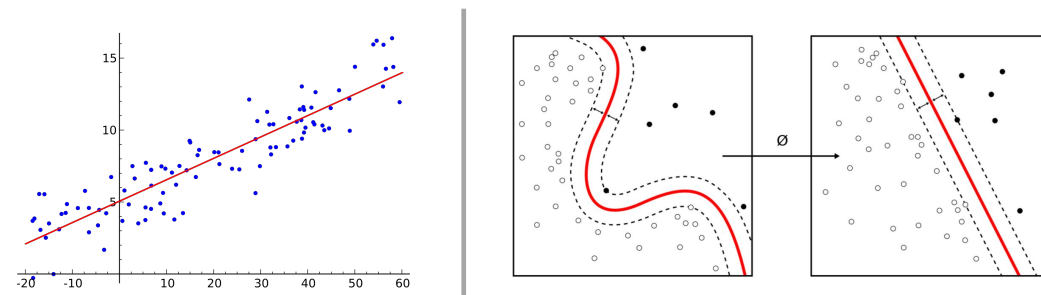


# Traditional Workflow

Classic Machine Learning [ 1990 : now ]



Examples [ Regression and SVMs ]



**Challenge in Slack channel: How would you describe this image to someone (or something) blind?**

Difficult: From it's raw pixels.

Medium: From geometric primitives (lines, curves, colors)

Easy: Using any words that you may know



# Deep Learning Workflow

Experience: Trust Neural Network to learn features and model by providing inputs and outputs.

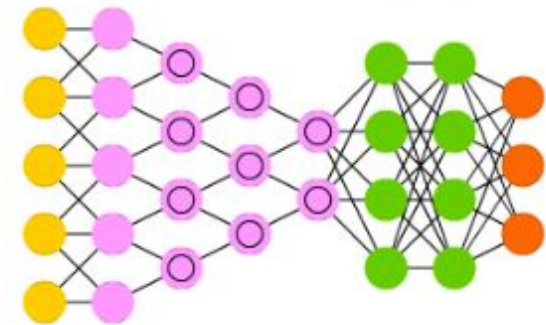
Key Skill: Experience (data) creation



Deep/End-to-End Learning [ 2012 : now ]



Example [ Conv Net ]



# INPUT TO OUTPUT

Louie or Not Louie?

1 = Louie

0 = Not Louie

.85 = 85% confident Louie



# INPUT TO OUTPUT

Louie or Not Louie?

1 = Louie

0 = Not Louie

.85 = 85% confident Louie



Yes, this beagle is Louie!

# INPUT TO OUTPUT

Louie or Not Louie?

1 = Louie

0 = Not Louie

.85 = 85% confident Louie



No, not Louie!

# INPUT TO OUTPUT

Louie or Not Louie?

1 = Louie

0 = Not Louie

.85 = 85% confident Louie



No, not Louie!

# INPUT TO OUTPUT

Louie or Not Louie?

1 = Louie

0 = Not Louie

.85 = 85% confident Louie



Yup, that's Louie!

# INPUT TO OUTPUT

Louie or Not Louie?

1 = Louie

0 = Not Louie

.85 = 85% confident Louie



Yea, that's Louie!



# INPUT TO OUTPUT

Louie or Not Louie?

1 = Louie

0 = Not Louie

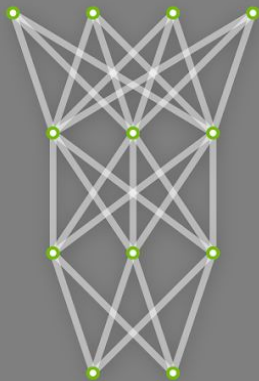
.85 = 85% confident Louie



Yes! Another epoch?

# DEEP LEARNING

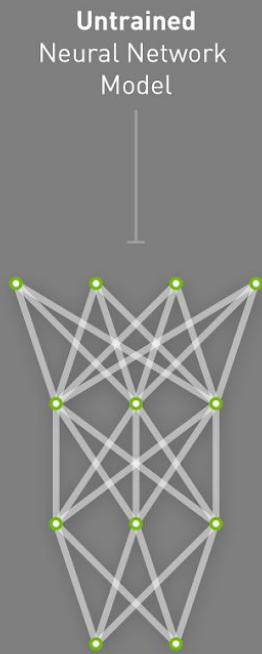
Untrained  
Neural Network  
Model



# DEEP LEARNING

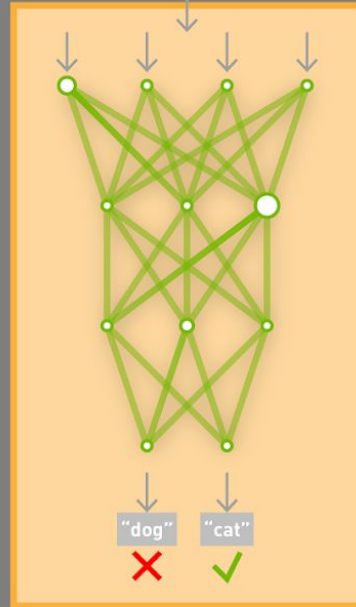
## TRAINING

Learning a new capability  
from existing data



Deep Learning  
Framework

TRAINING  
DATASET

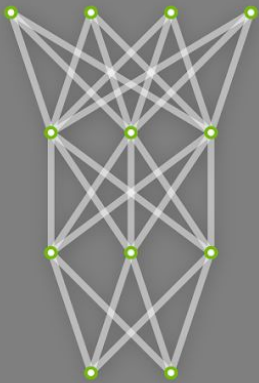


# DEEP LEARNING

## TRAINING

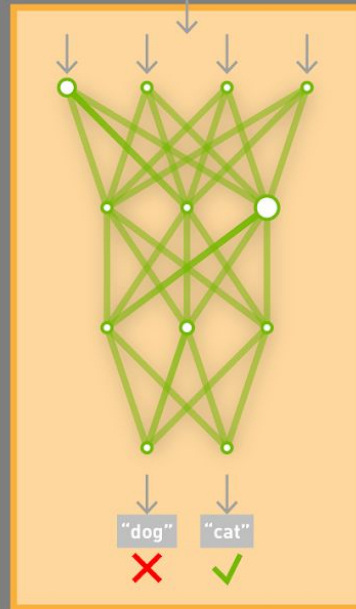
Learning a new capability  
from existing data

Untrained  
Neural Network  
Model

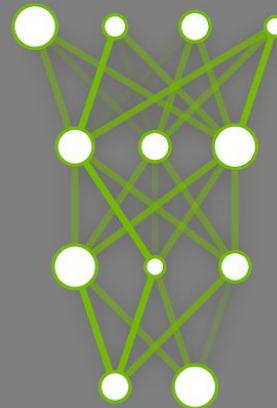


Deep Learning  
Framework

TRAINING  
DATASET



Trained Model  
New Capability

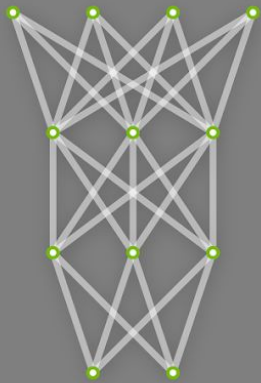


# DEEP LEARNING

## TRAINING

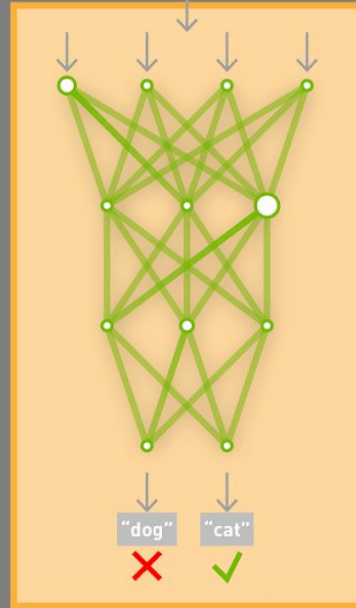
Learning a new capability  
from existing data

Untrained  
Neural Network  
Model

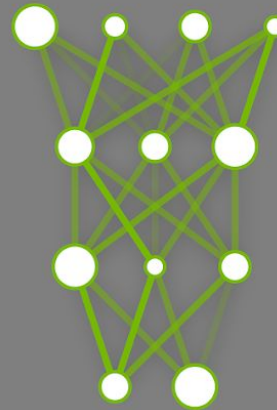


Deep Learning  
Framework

TRAINING  
DATASET



Trained Model  
New Capability



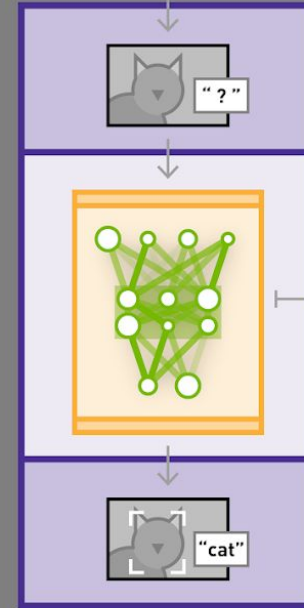
## INFERENCE

Applying this capability  
to new data

NEW  
DATA



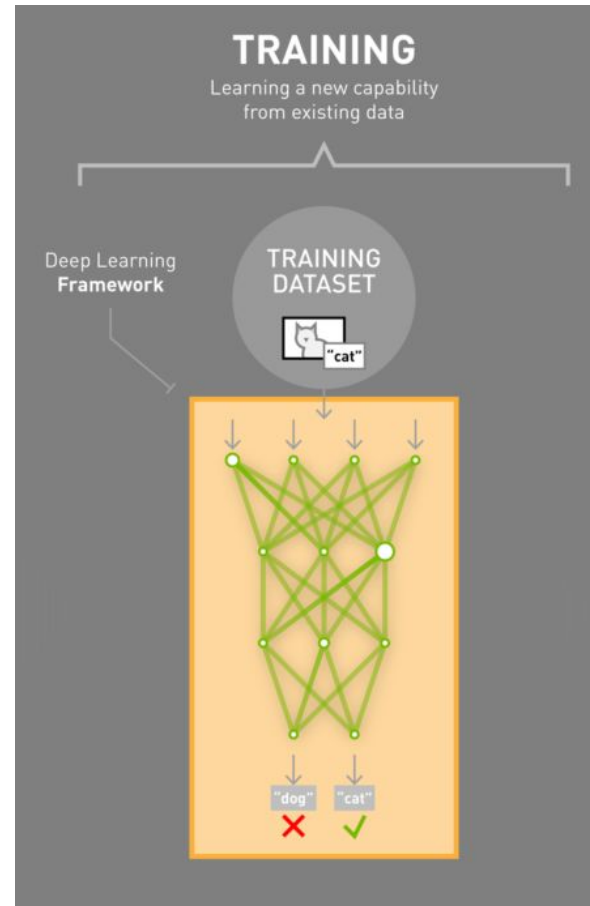
App or Service  
Featuring Capability



Trained Model  
Optimized for  
Performance

# Training a network with data

## Lab



# HANDWRITTEN DIGIT RECOGNITION

HELLO WORLD of machine learning



# WHAT THIS LAB IS

- An introduction to:
  - Deep Learning
  - Workflow of training a network
  - Understanding the results
- Hands-on exercises using DIGITS for computer vision and classification



# NVIDIA'S DIGITS

# NVIDIA DIGITS

## Interactive Deep Learning GPU Training System

### Process Data

The screenshot shows the 'aerial' dataset page in DIGITS. It includes a 'Job Information' section with details like 'Job Directory' and 'Image Type'. A 'Job Status Core' section shows a timeline of job events. Below, the 'Create DB (train)' section features a bar chart of 'Image Count' per category and an 'Image mean' preview.

### Configure DNN

The screenshot displays the 'New Image Classification Model' configuration page. It includes sections for 'Select Dataset', 'Data Transformations', 'Solver Options', and 'Custom Network'. A 'Pretrained model' section is also visible. At the bottom, there's a 'Use this many GPUs (next available)' section with a dropdown for GPU selection.

### Monitor Progress

The screenshot shows the 'ship\_type3' model monitoring page. It features a 'Job Status Running' section with a progress bar. Below, a 'Statistics' section displays a line graph of 'Loss (train)', 'Accuracy (val)', and 'Loss (all)' over epochs. A 'GPU Usage' section shows a bar chart of GPU utilization.

### Visualization

The screenshot displays the 'Visualization' page for the 'ship\_type3' model. It includes a 'Predictions' table with categories like 'military', 'cruise', 'boats', 'cargo', and 'sat'. Below, there are three rows of visualizations: 'Data' (mean and std deviation), 'conv1' (weights and activation), and 'norm1' (mean and std deviation). Each row includes a histogram and a corresponding image or heatmap.

# WHAT THIS LAB IS NOT

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

# ASSUMPTIONS

- No background in Deep Learning needed
- Understand how to:
  - Navigate a web browser
  - Download files
  - Locate files in file managers

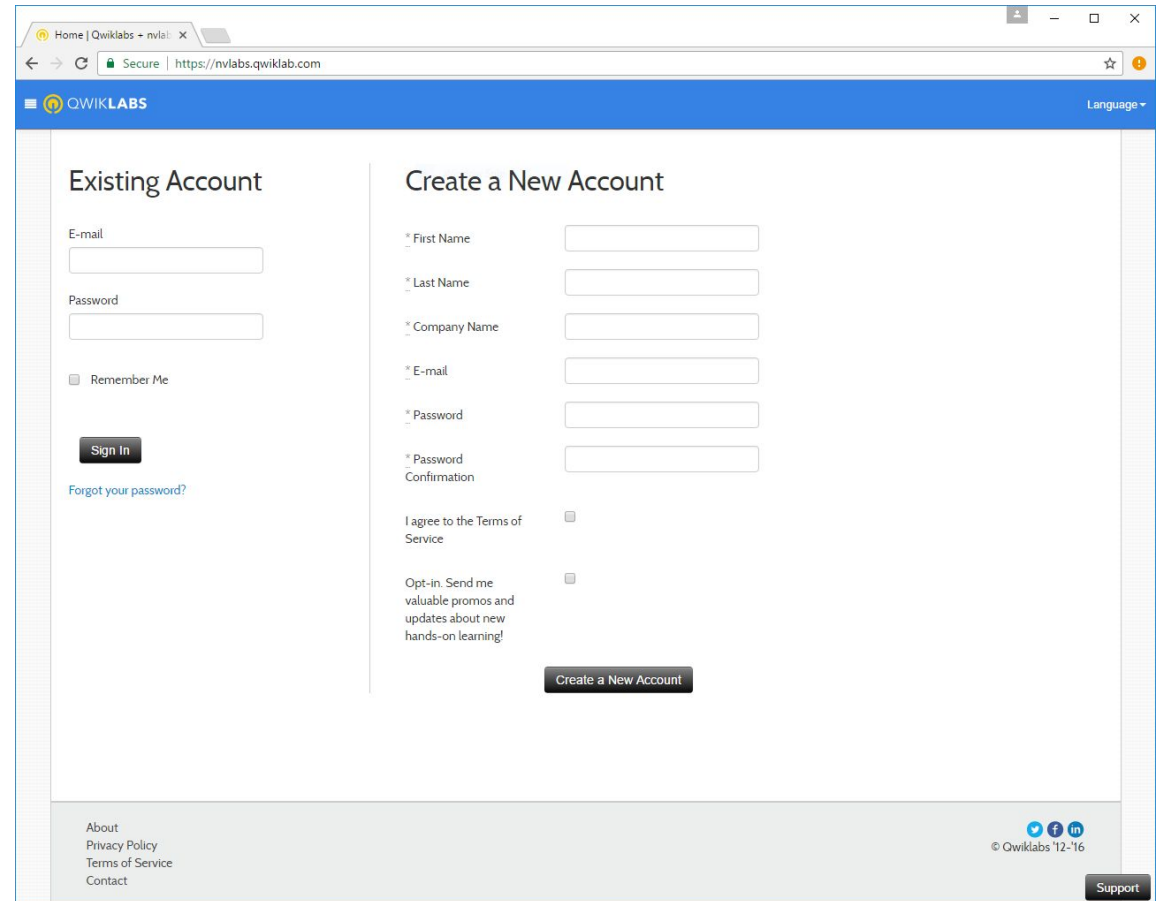
# LAB OVERVIEW

- Learn about the workflow of Deep Learning
  - Load data
  - Expose a network to data
  - Evaluate model results
  - Try different techniques to improve initial results

# LAUNCHING THE LAB

# NAVIGATING TO QWIKLABS

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



The screenshot shows a web browser window displaying the Qwiklabs login and account creation page. The browser's address bar shows the URL <https://nvlabs.qwiklab.com>. The page has a blue header with the Qwiklabs logo and a language dropdown menu. The main content area is divided into two columns: "Existing Account" and "Create a New Account".

**Existing Account:**

- E-mail:
- Password:
- Remember Me
- 
- [Forgot your password?](#)

**Create a New Account:**

- \* First Name:
- \* Last Name:
- \* Company Name:
- \* E-mail:
- \* Password:
- \* Password Confirmation:
- I agree to the Terms of Service:
- Opt-in. Send me valuable promos and updates about new hands-on learning!:
- 

At the bottom of the page, there is a footer with links for "About", "Privacy Policy", "Terms of Service", and "Contact". On the right side of the footer, there are social media icons for Twitter, Facebook, and LinkedIn, along with the text "© Qwiklabs 12-'16" and a "Support" button.

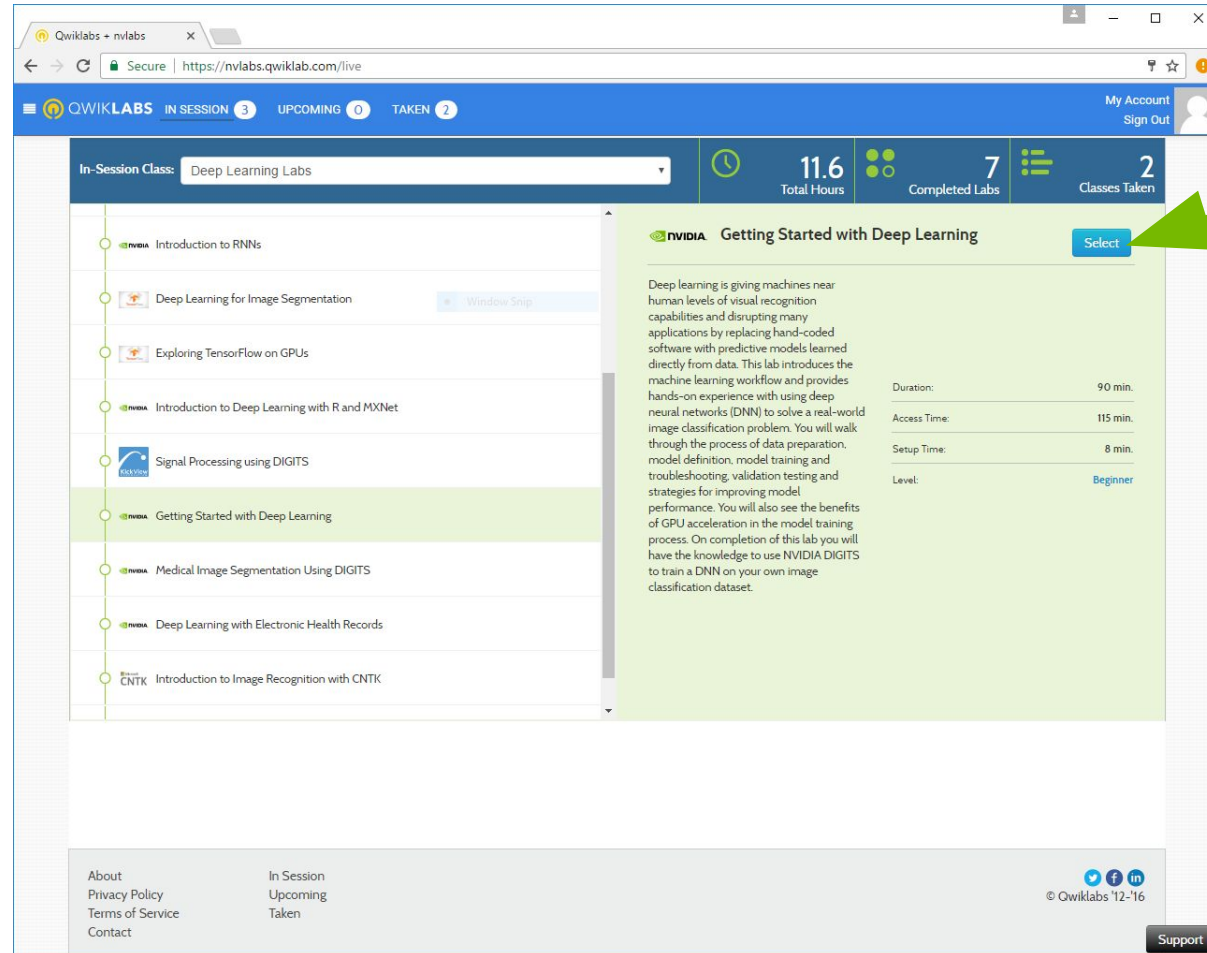
# ACCESSING LAB ENVIRONMENT

3. Select the event specific In-Session Class in the upper left
4. Click the “Image Classification with DIGITS” Class from the list

The screenshot shows the Qwiklabs interface. At the top, there is a navigation bar with 'QWIKLABS', 'IN SESSION 3', 'UPCOMING 0', and 'TAKEN 2'. On the right, there is a 'My Account Sign Out' link. Below the navigation bar, there is a header section with 'In-Session Class: Deep Learning Labs', '11.6 Total Hours', '7 Completed Labs', and '2 Classes Taken'. The main content area displays a list of lab options, including 'Introduction to RNNs', 'Deep Learning for Image Segmentation', 'Exploring TensorFlow on GPUs', 'Introduction to Deep Learning with R and MXNet', 'Signal Processing using DIGITS', 'Getting Started with Deep Learning', 'Medical Image Segmentation Using DIGITS', 'Deep Learning with Electronic Health Records', and 'Introduction to Image Recognition with CNTK'. A green arrow points to the 'In-Session Class' dropdown, and another green arrow points to the 'Getting Started with Deep Learning' lab. A detailed view of the 'Getting Started with Deep Learning' lab is shown on the right, including a description, duration (90 min), access time (115 min), setup time (8 min), and level (Beginner).



# LAUNCHING THE LAB ENVIRONMENT



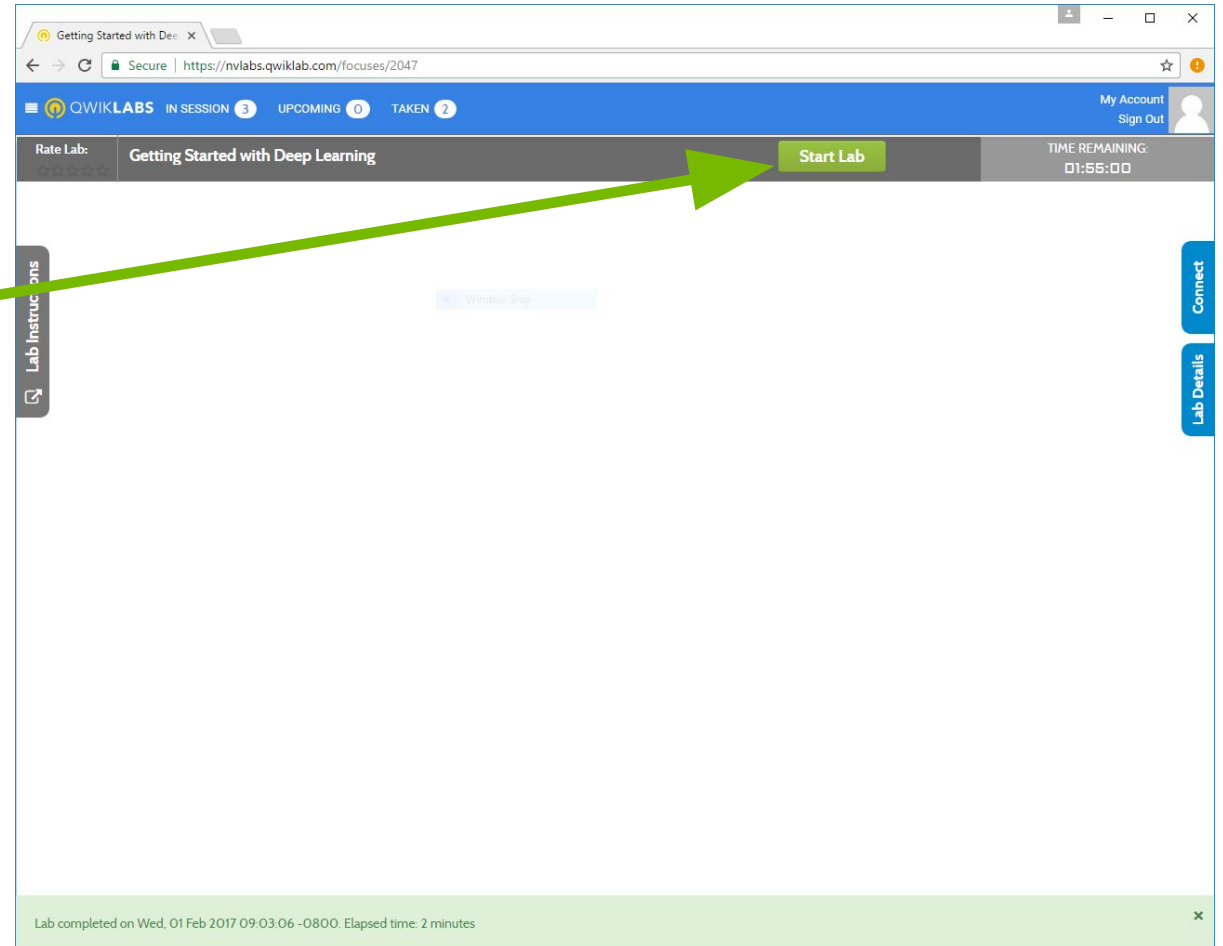
The screenshot shows the Qwiklabs web interface. At the top, there's a navigation bar with 'QWIKLABS' and status indicators for 'IN SESSION' (3), 'UPCOMING' (0), and 'TAKEN' (2). A user profile 'My Account Sign Out' is visible. Below the navigation bar, a dropdown menu shows 'In-Session Class: Deep Learning Labs'. A summary bar displays '11.6 Total Hours', '7 Completed Labs', and '2 Classes Taken'. A list of labs is shown on the left, with 'Getting Started with Deep Learning' highlighted in green. The main content area shows the details for this lab, including a description, duration (90 min), access time (115 min), setup time (8 min), and level (Beginner). A blue 'Select' button is visible next to the lab title, which is pointed to by a green arrow.

5. Click on the Select button to launch the lab environment

- After a short wait, lab Connection information will be shown
- Please ask Lab Assistants for help!

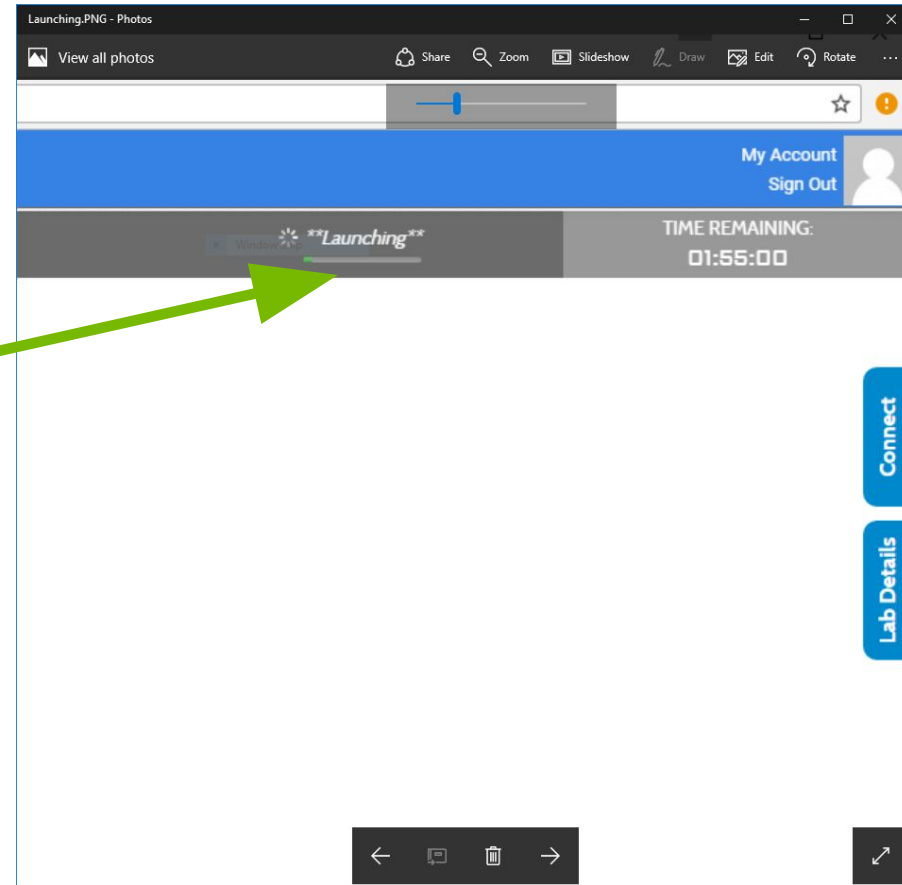
# LAUNCHING THE LAB ENVIRONMENT

6. Click on the Start Lab button



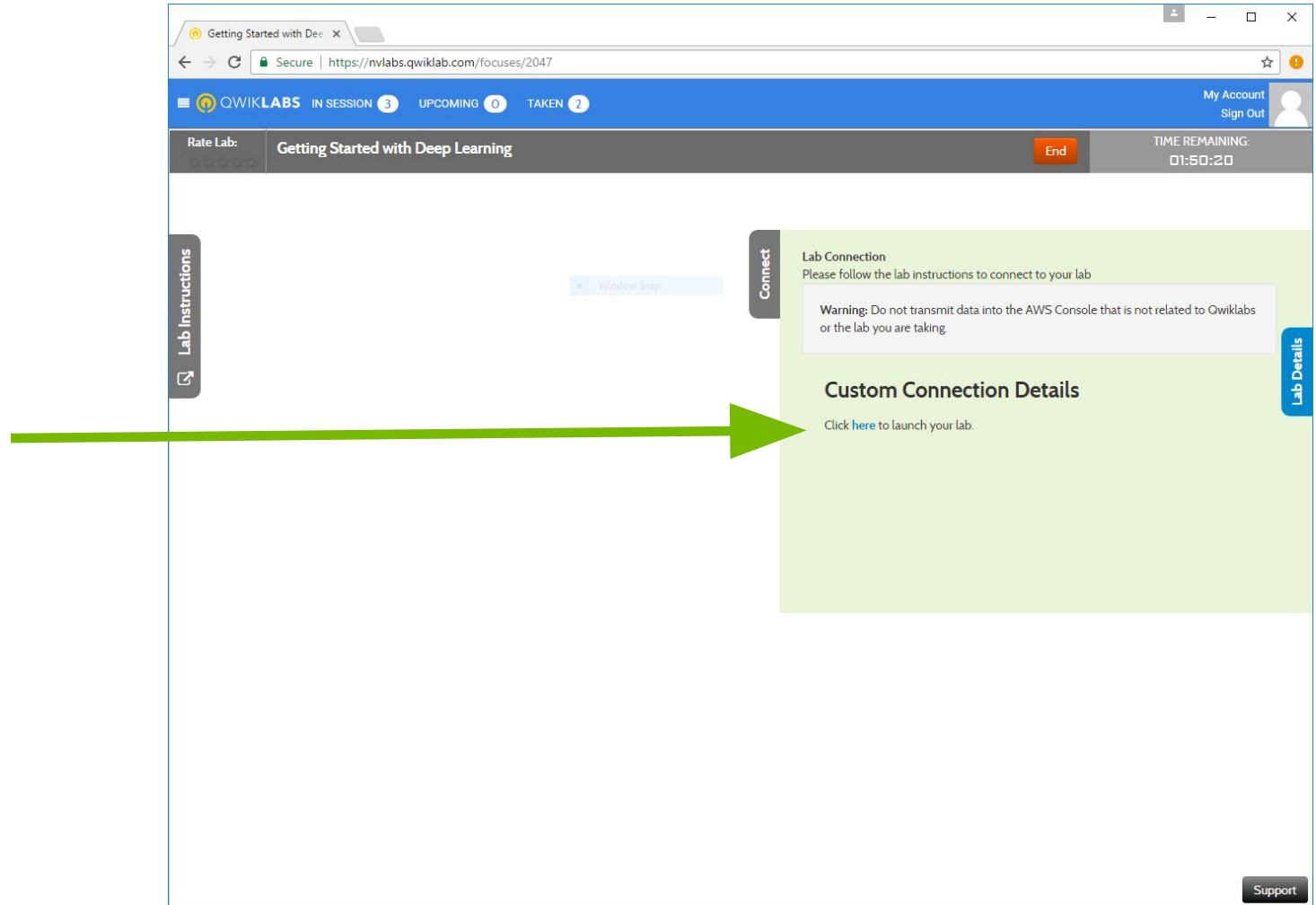
# LAUNCHING THE LAB ENVIRONMENT

You should see that the lab environment is “launching” towards the upper-right corner



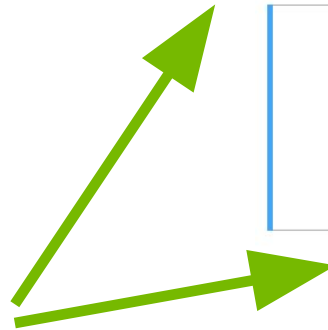
# CONNECTING TO THE LAB ENVIRONMENT

7. Click on “here” to access your lab environment / Jupyter notebook



# CONNECTING TO THE LAB ENVIRONMENT

You should see your  
“Image Classification  
with DIGITS” Jupyter  
notebook



The screenshot shows a JupyterLab interface. At the top, the title bar reads "jupyter Image Classification with DIGITS - Training a model" and "Last Checkpoint: Last Wednesday at 9:17 AM (unsaved changes)". Below the title bar is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", and "Help". A toolbar contains icons for file operations and a "Markdown" dropdown menu. The main content area displays the NVIDIA logo and the text "DEEP LEARNING INSTITUTE". Below this, the notebook title "Image Classification with DIGITS" is shown, followed by the subtitle "An introduction to Deep Learning". The notebook content includes an introductory paragraph, a list of topics to be examined, and a section titled "Training vs. programming".

**Image Classification with DIGITS**  
An introduction to Deep Learning

In this lab, you'll learn to **train** a **neural network** using clean **labeled data**. We'll introduce deep learning through the task of **supervised image classification**, where, given a large number of images and their labels, you'll build a tool that can *predict* labels of *new* images.

The intent is to build the skills to start experimenting with deep learning. You'll examine:

- What it means to *train* vs. to *program*
- The role of data in artificial intelligence
- How to load data for training a neural network
- The role of a *network* in deep learning
- How to train a model with data

At the end of this lab, you'll have a trained neural network that can successfully classify images to solve a classic deep learning challenge:

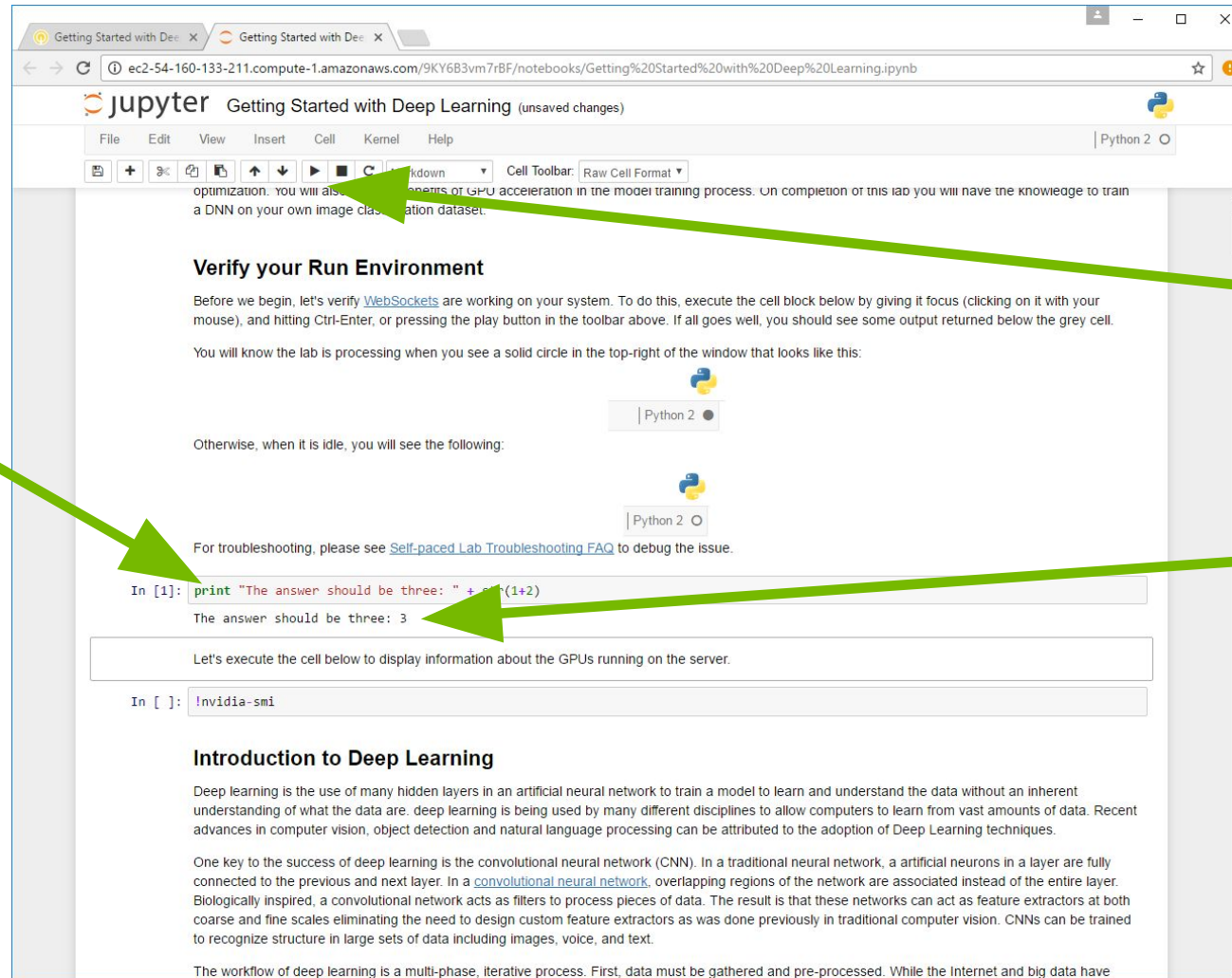
**How can we digitize handwriting?**

**Training vs. programming**

The fundamental difference between artificial intelligence (AI) and traditional programming is that AI *learns* while traditional algorithms are *programmed*. Let's examine the difference through an example:

Imagine you were asked to give a robot instructions to make a sandwich using traditional computer programming, instruction by instruction. How might you start?

# JUPYTER NOTEBOOK



1. Place your cursor in the code

2. Click the "run cell" button

3. Confirm you receive the same result

# STARTING DIGITS

Instruction in Jupyter notebook will link you to DIGITS

The screenshot shows a Jupyter notebook titled "Getting Started with Deep Learning (autosaved)". The browser address bar shows the URL: `ec2-54-160-133-211.compute-1.amazonaws.com/9KY6B3vm7rBF/notebooks/Getting%20Started%20with%20Deep%20Learning.ipynb`. The notebook interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with various icons. A dropdown menu is open, showing options: "Classification", "Object Detection", and "Other". Below the menu, there is a table with columns: "framework", "status", "elapsed", and "s".

To start DIGITS, [click here](#).

### Task - Create a Database

First, we want to create a database from the MNIST data. To create a database, select **Classification** from the **New Dataset** menu. At this point you may need to enter a username. If requested, just enter any name in lower-case.

In the **New Dataset** window, you want to set the following fields to the values specified:

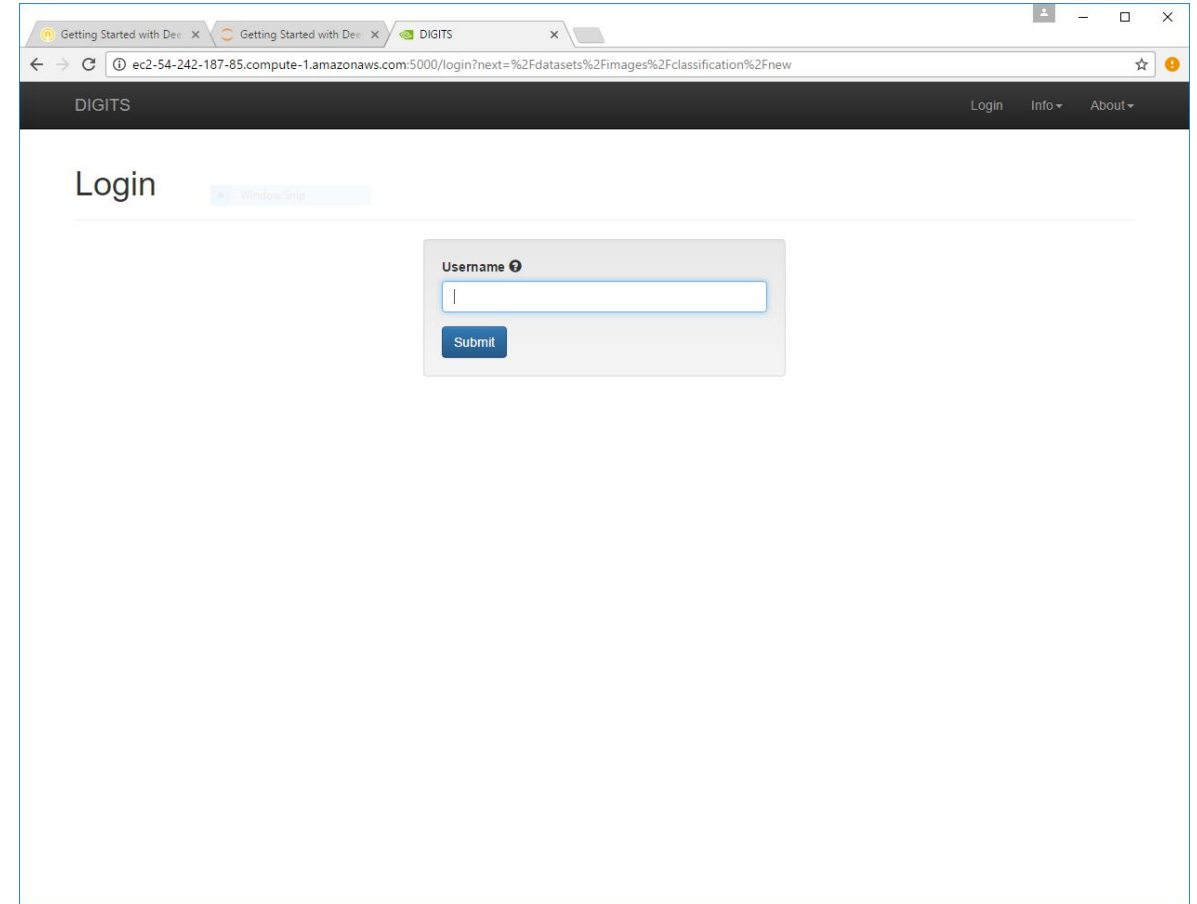
- Image Type : Grayscale
- Image Size : 28 x 28
- Training Images : /home/ubuntu/data/train\_small
- Select **Separate test images folder** checkbox
- Test Images : /home/ubuntu/data/test\_small
- Dataset Name : MNIST Small

Your screen should look like the image below.

DIGITS New Dataset mebersole (Logout) Info About

# ACCESSING DIGITS

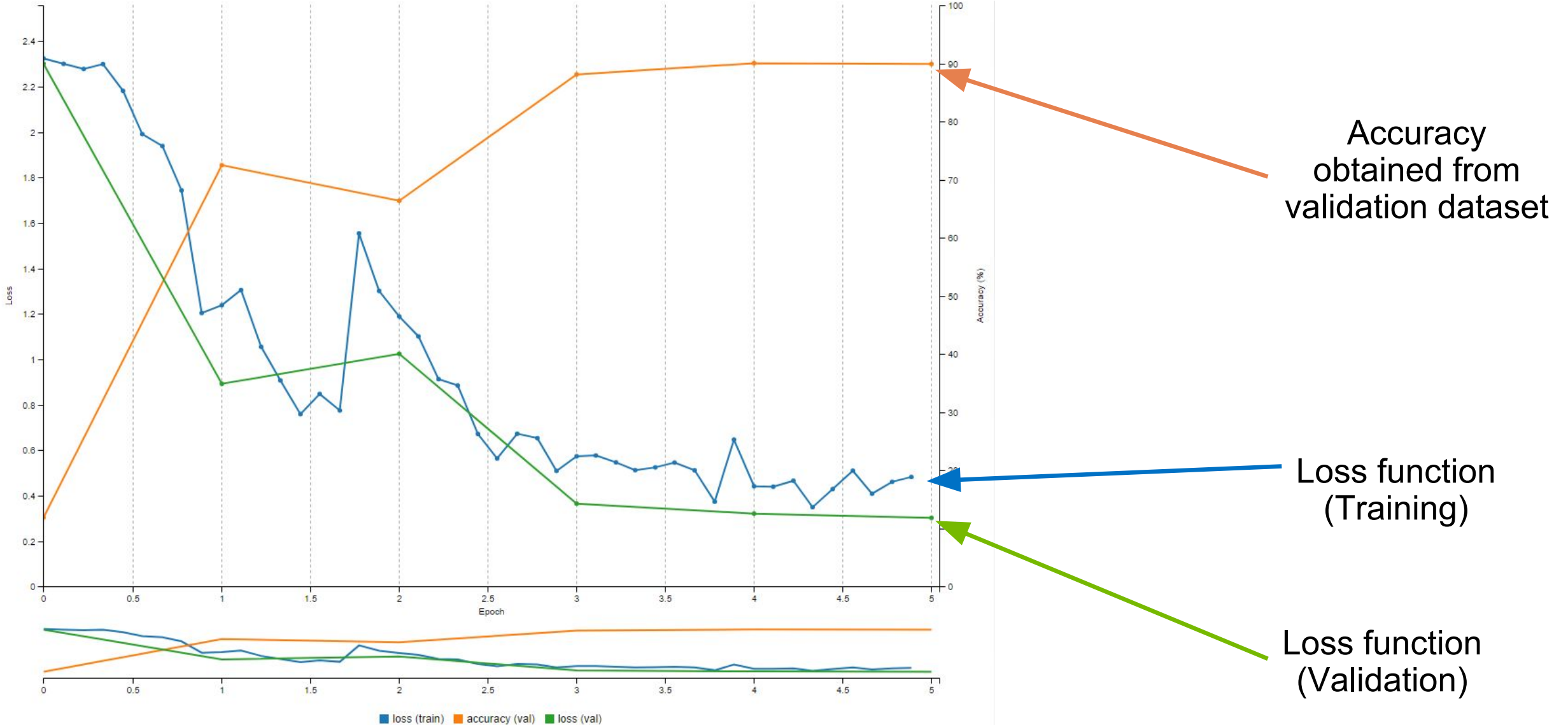
- Will be prompted to enter a username to access DIGITS
  - Can enter any username
  - Use lower case letters





# Evaluating Performance

# EVALUATE THE MODEL

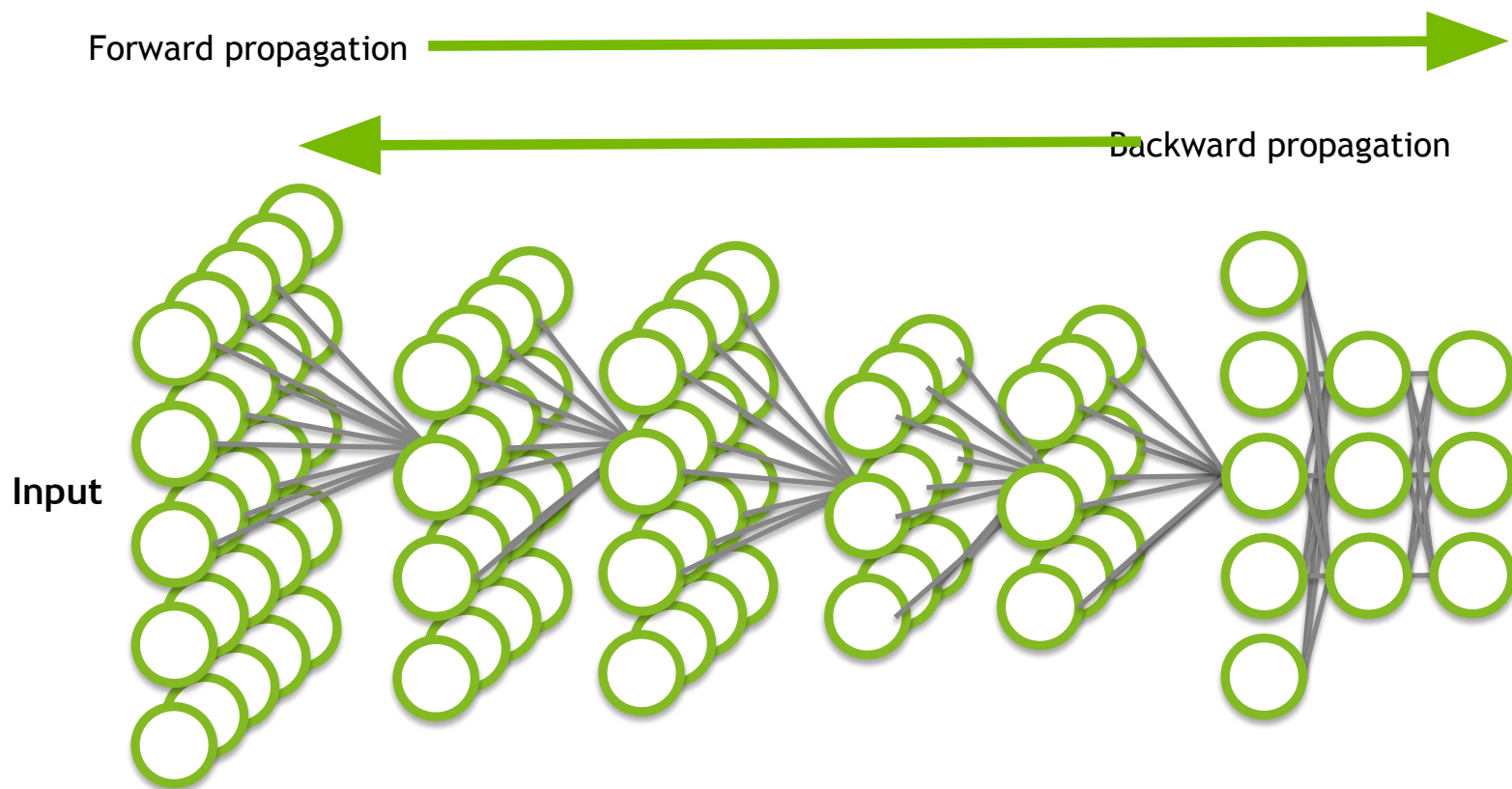


Accuracy  
obtained from  
validation dataset

Loss function  
(Training)

Loss function  
(Validation)

# DEEP LEARNING APPROACH - TRAINING



## Process

- Forward propagation yields an inferred label for each training image
- Loss function used to calculate difference between known label and predicted label for each image
- Weights are adjusted during backward propagation
- Repeat the process

# Next Challenges

Ideas?



- Increase accuracy and confidence with similar data







- Generalize performance to more diverse data

# Lab Review

# More data








Full dataset ( 10 epochs )

- 99% of accuracy achieved
- No improvements in recognizing real-world images

	Defaults	Training+Data
	1 : 99.90 %	0 : 93.11 %
	2 : 69.03 %	2 : 87.23 %
	8 : 71.37 %	8 : 71.60 %
	8 : 85.07 %	8 : 79.72 %
	0 : 99.00 %	0 : 95.82 %
	8 : 99.69 %	8 : 100.0 %
	8 : 54.75 %	2 : 70.57 %

# DATA AUGMENTATION

## Adding inverted images ( 10 epochs )

	SMALL DATASET	FULL DATASET	+INVERTED
	1 : 99.90 %	0 : 93.11 %	1 : 90.84 %
	2 : 69.03 %	2 : 87.23 %	2 : 89.44 %
	8 : 71.37 %	8 : 71.60 %	3 : 100.0 %
	8 : 85.07 %	8 : 79.72 %	4 : 100.0 %
	0 : 99.00 %	0 : 95.82 %	7 : 82.84 %
	8 : 99.69 %	8 : 100.0 %	8 : 100.0 %
	8 : 54.75 %	2 : 70.57 %	2 : 96.27 %

# DATA AUGMENTATION

## Adding Inverted Images

















DIGITS Image Classification Dataset smorino (Logout) Info

### Exploring MNIST invert (train\_db) images

Show all images or filter by class: 0 1 2 3 4 5 6 7 8 9

Items per page: 10 - 25 - 50 - 100

« 0 1 2 3 4 5 ... 3600 »








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

```
keras.preprocessing.image.ImageDataGenerator(featurewise_center=False,
samplewise_center=False,
featurewise_std_normalization=False,
samplewise_std_normalization=False,
zca_whitening=False,
zca_epsilon=1e-6,
rotation_range=0.,
width_shift_range=0.,
height_shift_range=0.,
shear_range=0.,
zoom_range=0.,
channel_shift_range=0.,
fill_mode='nearest',
cval=0.,
horizontal_flip=False,
vertical_flip=False,
rescale=None,
preprocessing_function=None,
data_format=K.image_data_format())
```



# MODIFIED NETWORK

Adding filters and ReLU layer ( 10 epochs )

	SMALL DATASET	FULL DATASET	+INVERTED	ADDING LAYER
	1 : 99.90 %	0 : 93.11 %	1 : 90.84 %	1 : 59.18 %
	2 : 69.03 %	2 : 87.23 %	2 : 89.44 %	2 : 93.39 %
	8 : 71.37 %	8 : 71.60 %	3 : 100.0 %	3 : 100.0 %
	8 : 85.07 %	8 : 79.72 %	4 : 100.0 %	4 : 100.0 %
	0 : 99.00 %	0 : 95.82 %	7 : 82.84 %	2 : 62.52 %
	8 : 99.69 %	8 : 100.0 %	8 : 100.0 %	8 : 100.0 %
	8 : 54.75 %	2 : 70.57 %	2 : 96.27 %	8 : 70.83 %

# MODIFY THE NETWORK

Necessary for less “solved” challenges.

```
layer {
  name: "pool1"
  type: "Pooling"
  ...
}

layer {
  name: "reluP1"
  type: "ReLU"
  bottom: "pool1"
  top: "pool1"
}

layer {
  name: "reluP1"
```

```
layer {
  name: "conv1"
  type: "Convolution"
  ...
  convolution_param {
    num_output: 75
  }
  ...
}

layer {
  name: "conv2"
  type: "Convolution"
  ...
  convolution_param {
    num_output: 100
  }
  ...
}
```

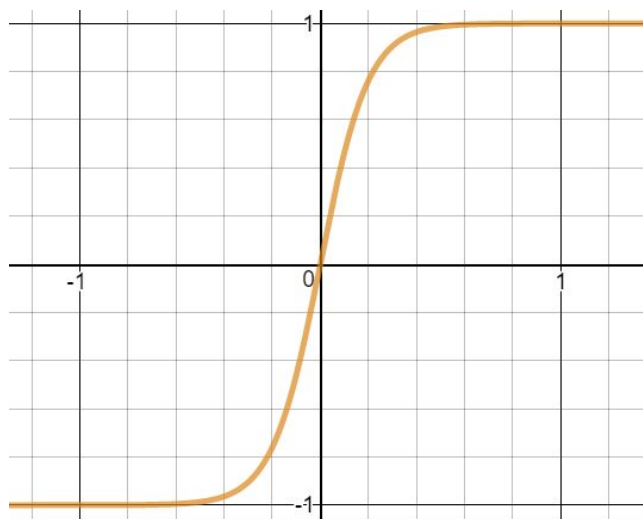
# Next Steps

- Experiment with Image Classification
  - Different datasets
  - Increase performance
- Learn to train existing networks with data for other challenges
- Learn about network construction
- Learn about how to create an image classifier with other frameworks
  - Caffe/Keras
  - Tensorflow
  - Etc.

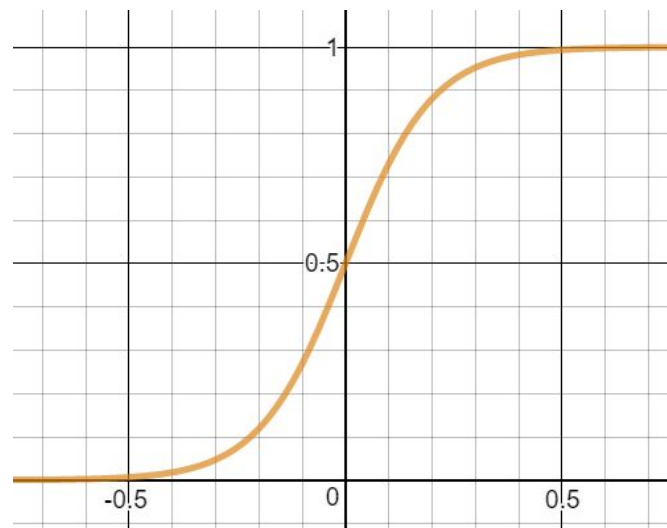
# Appendix



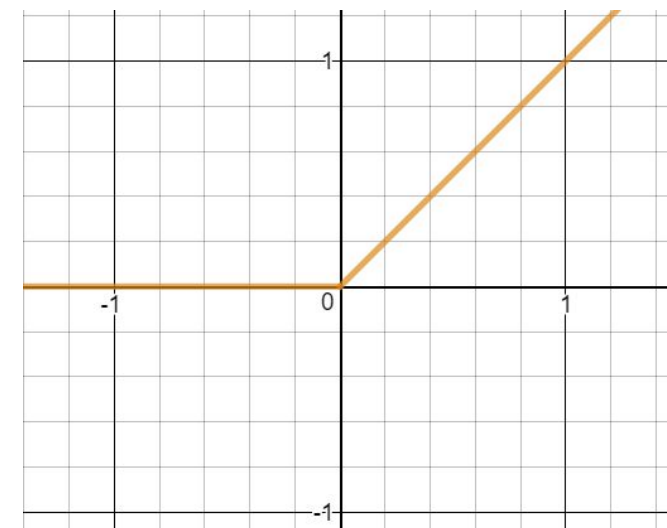
# Activation functions



tanh



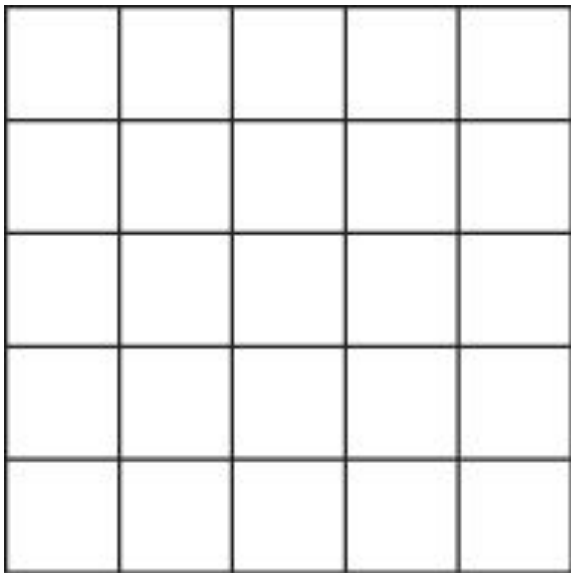
Sigmoid



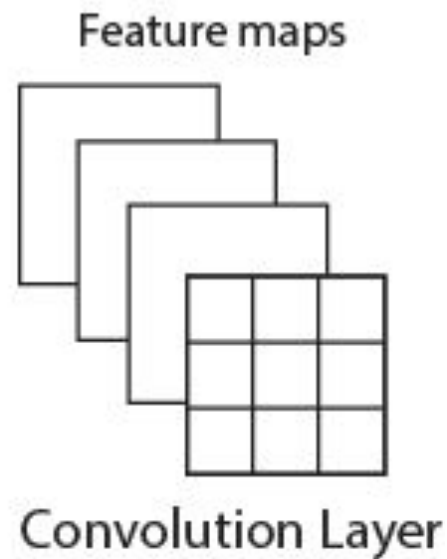
ReLU

# CNN - Example

Each pixel is a neuron

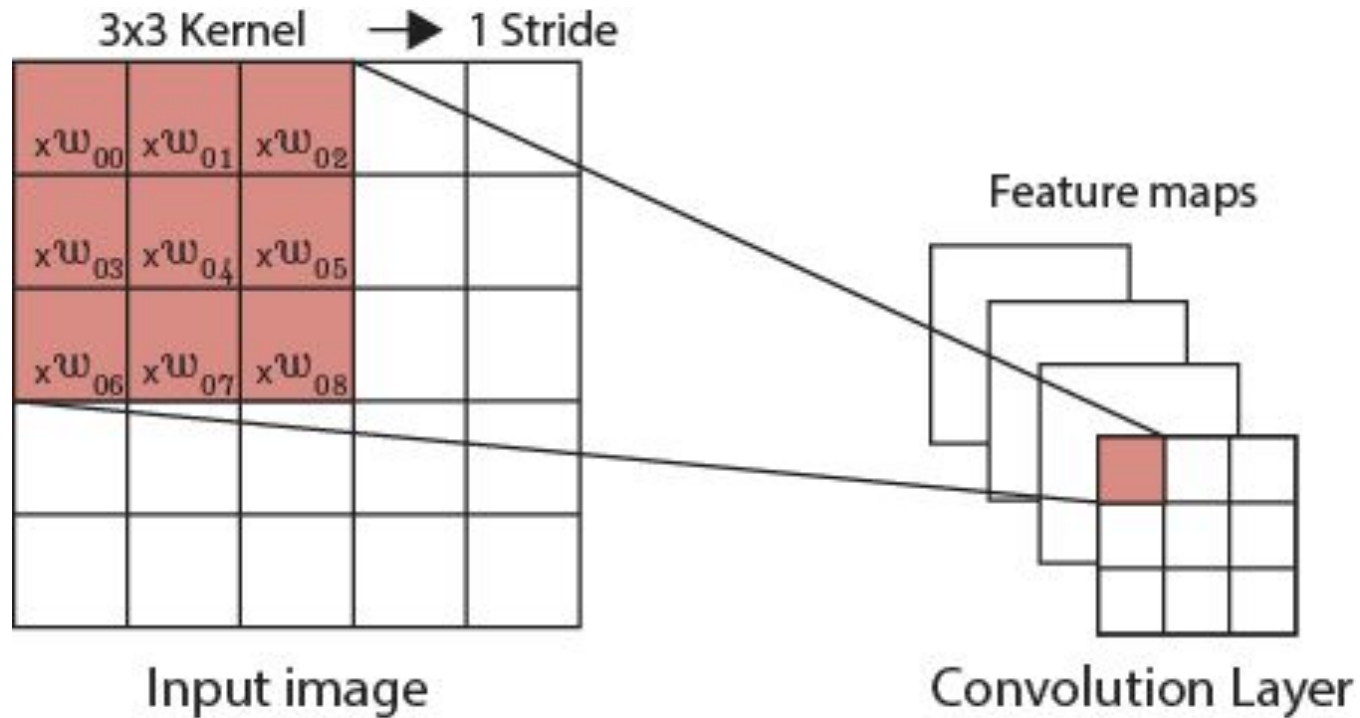


Input image



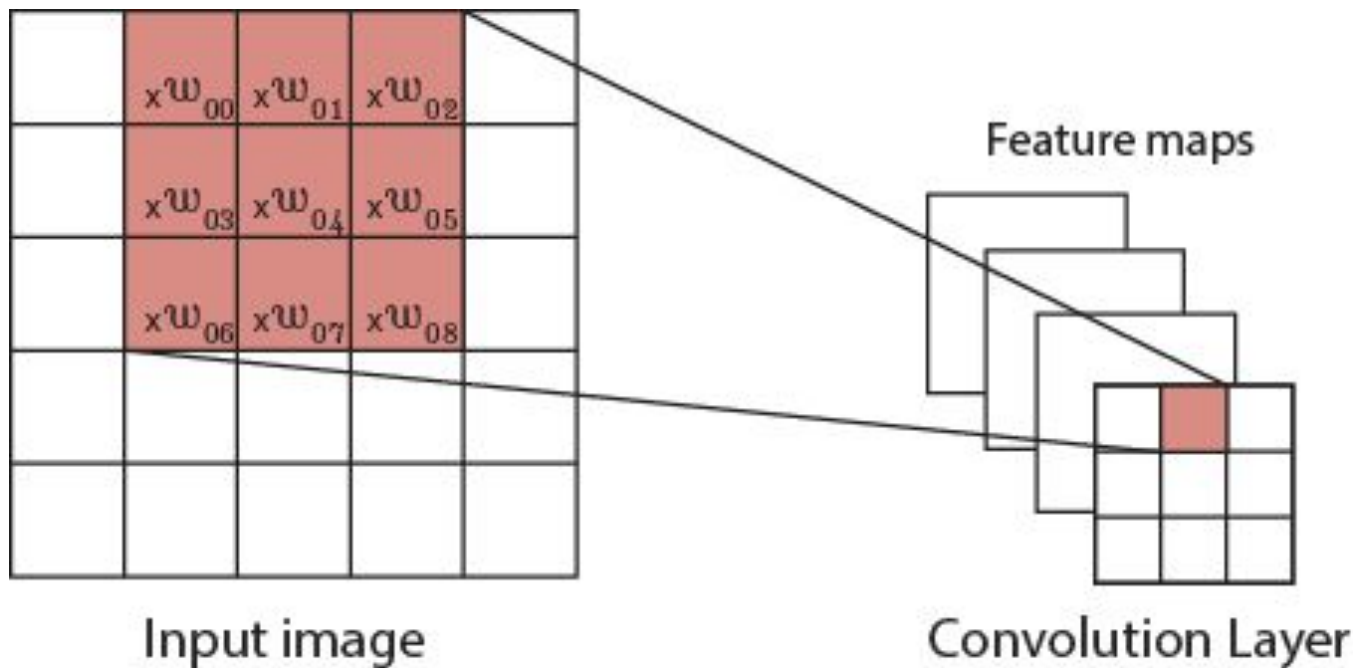
# CNN - Example - 1st Feature Map

3x3 Kernel, 1 Stride, weights constant per kernel



# CNN - Example - 1st Feature Map

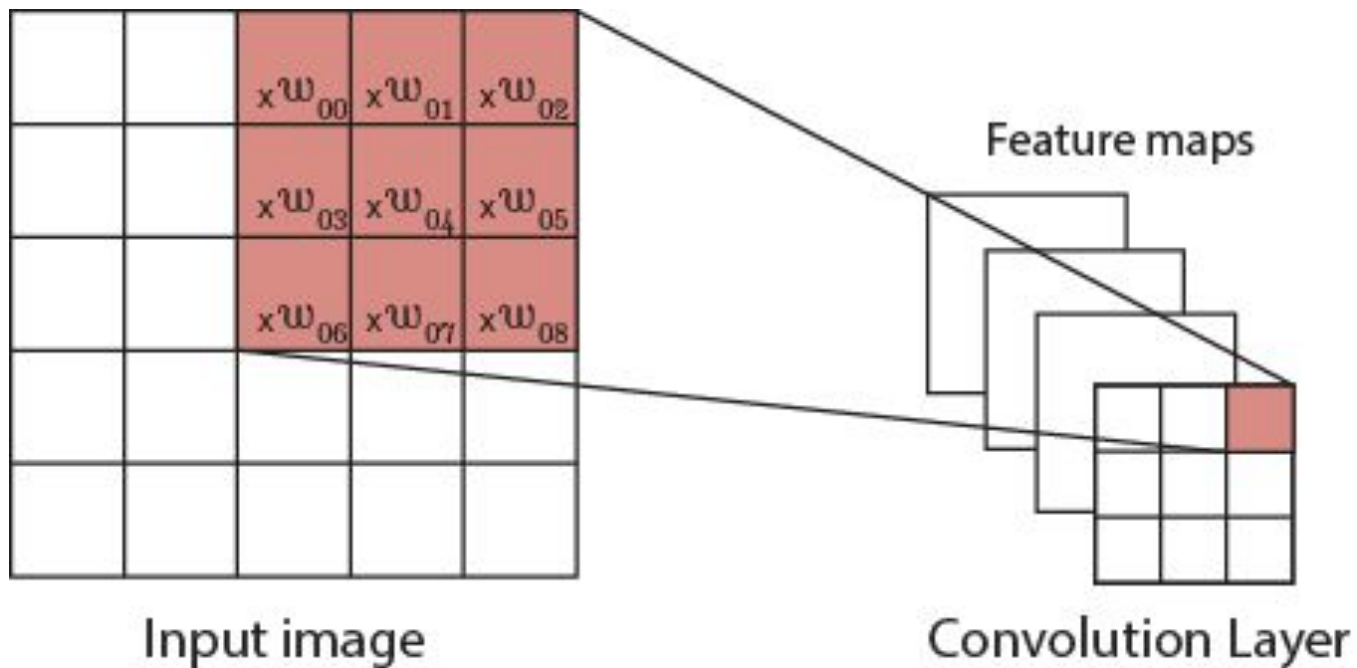
3x3 Kernel, 1 Stride, weights constant per kernel





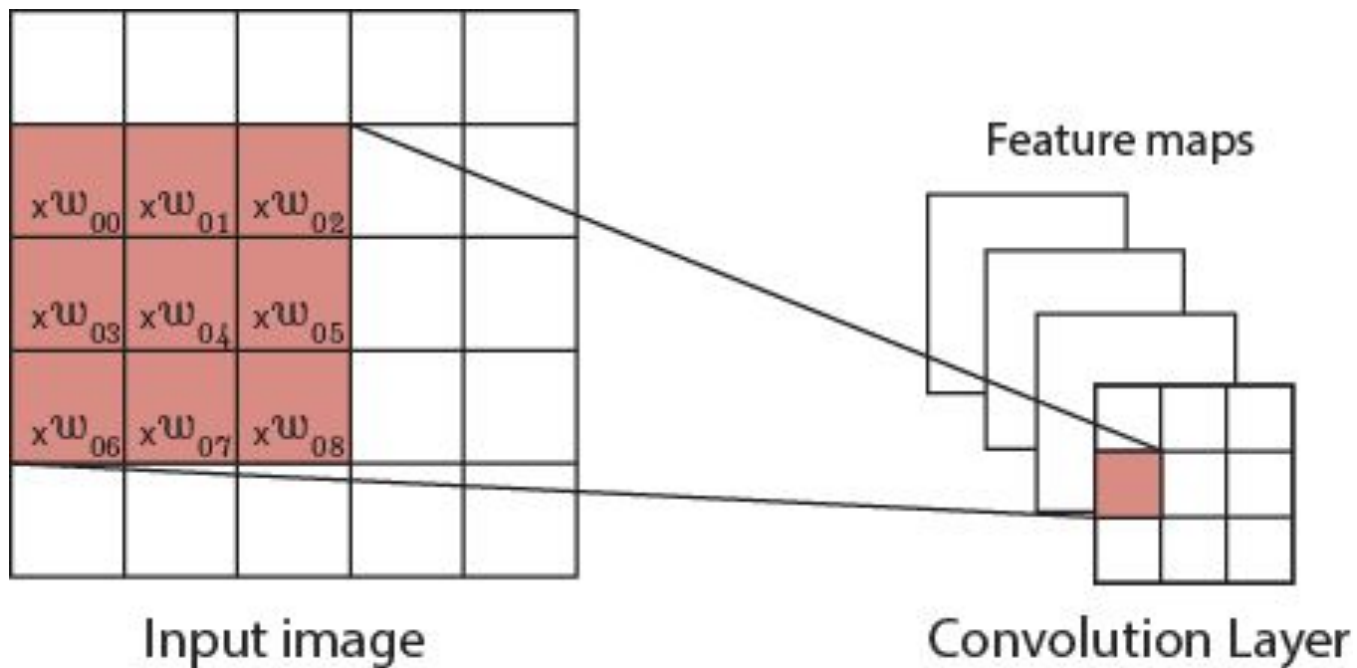
# CNN - Example - 1st Feature Map

3x3 Kernel, 1 Stride, weights constant per kernel



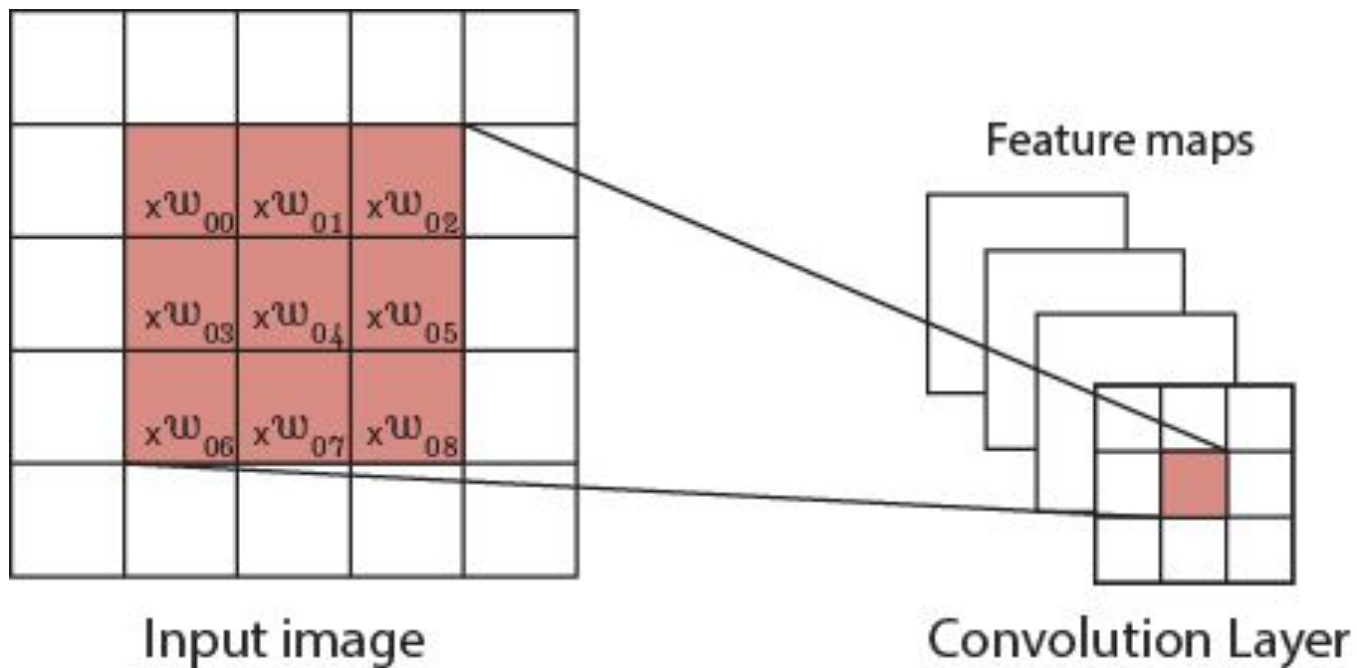
# CNN - Example - 1st Feature Map

3x3 Kernel, 1 Stride, weights constant per kernel



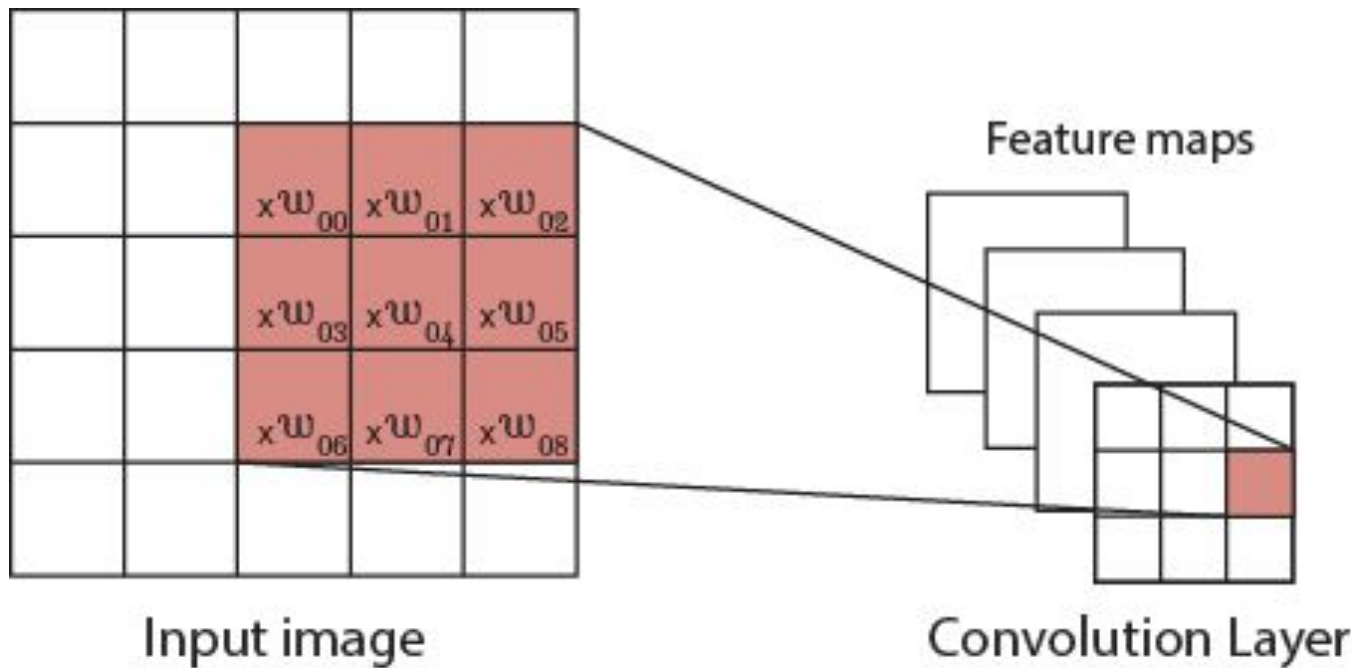
# CNN - Example - 1st Feature Map

3x3 Kernel, 1 Stride, weights constant per kernel

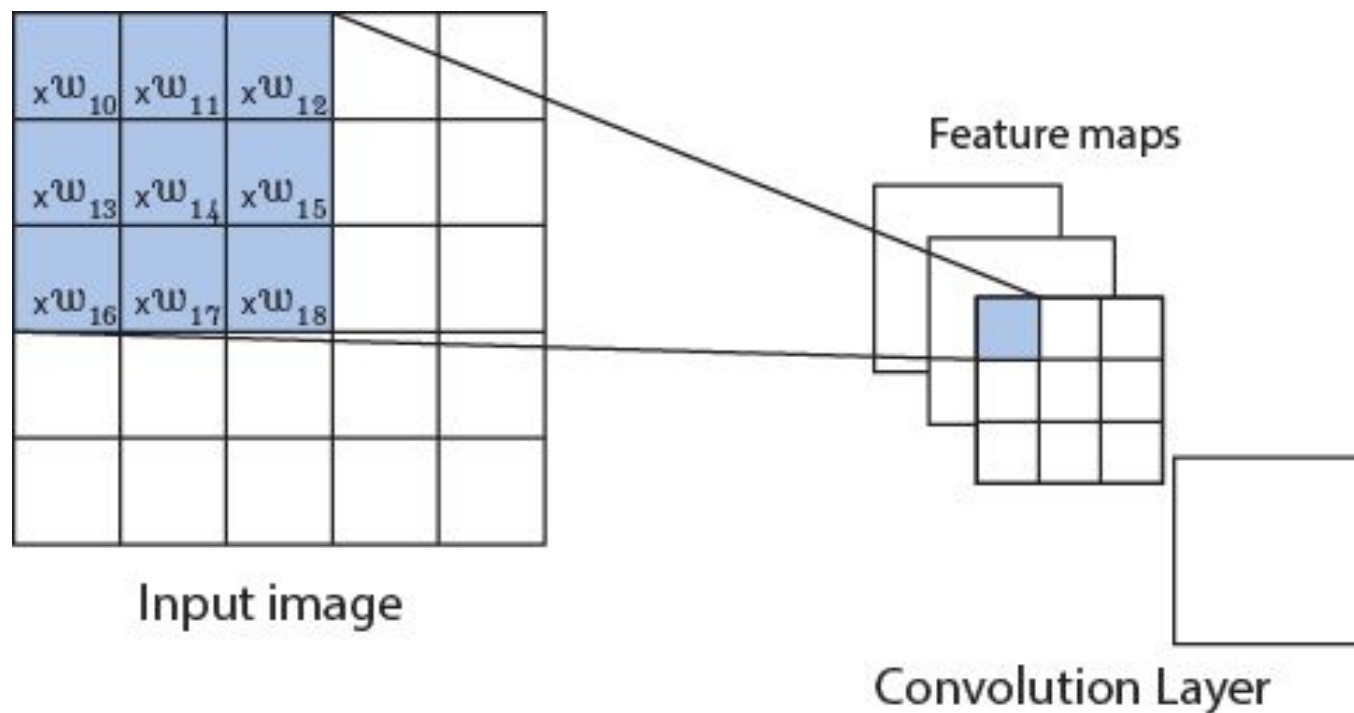


# CNN - Example - 1st Feature Map

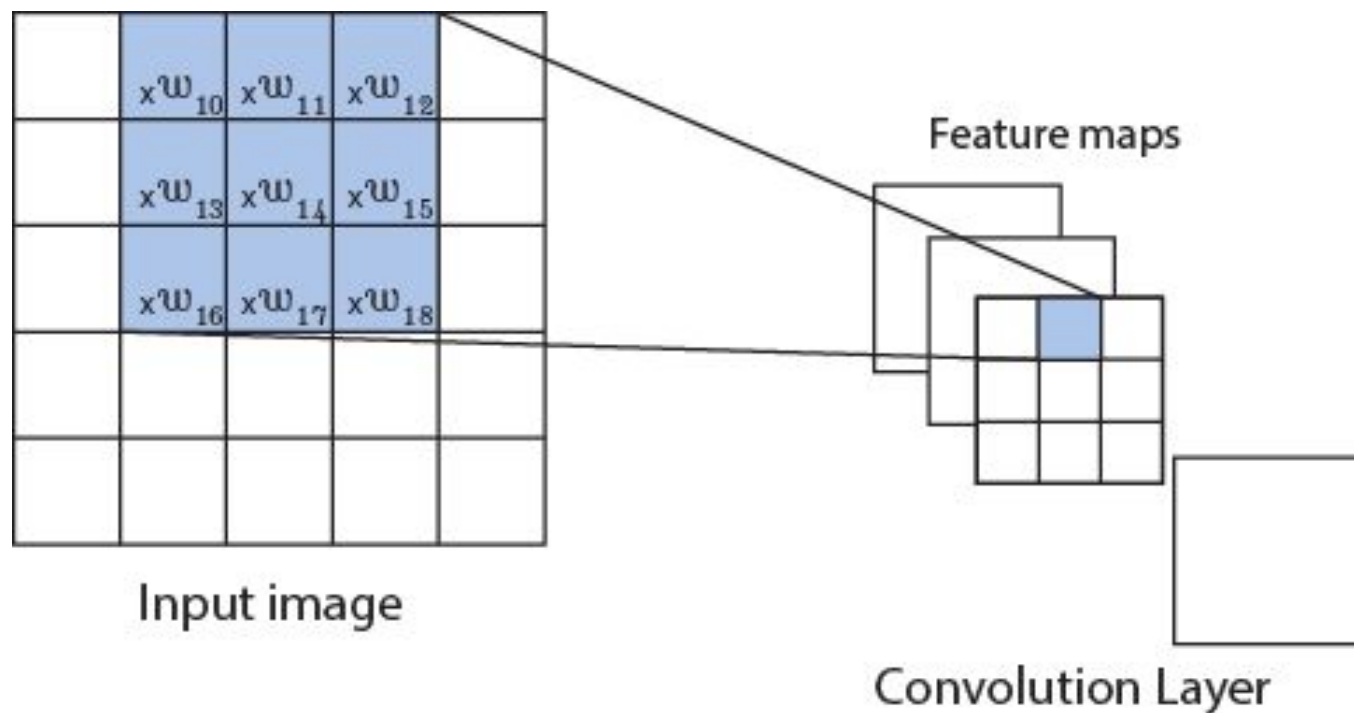
3x3 Kernel, 1 Stride, weights constant per kernel



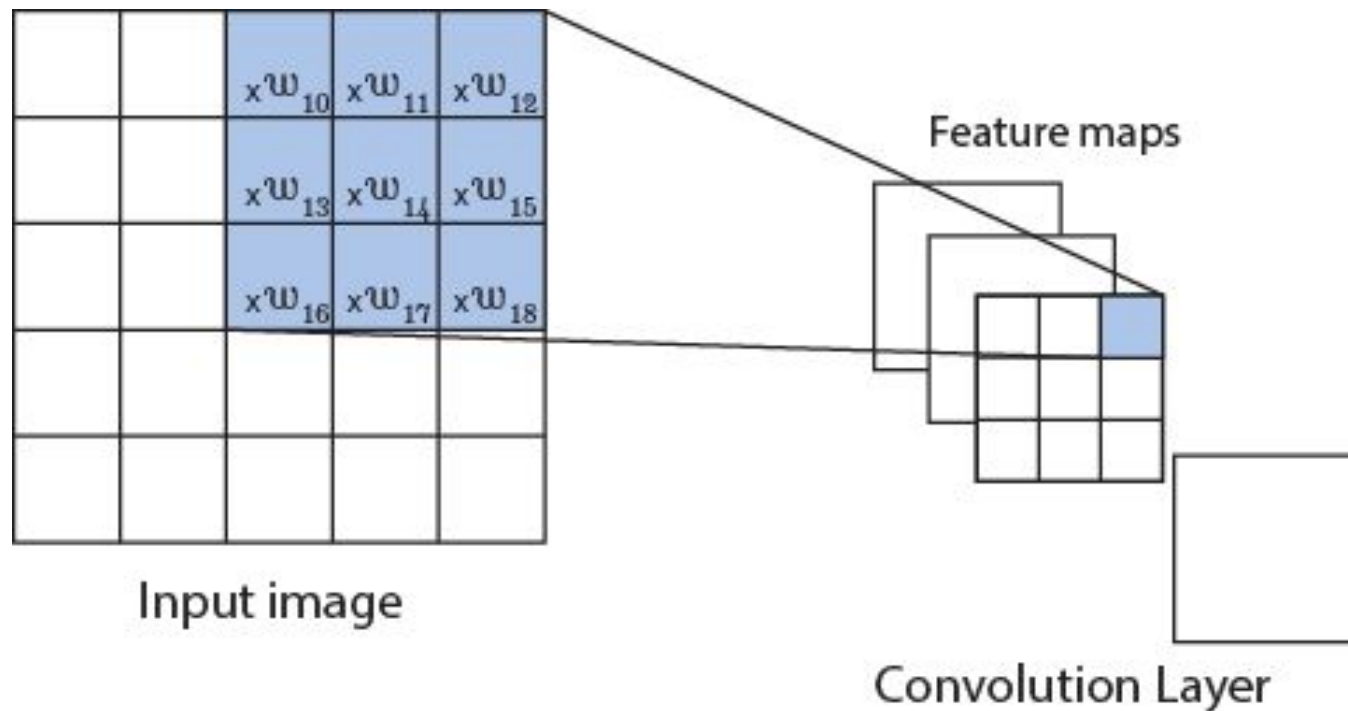
# CNN - Example - 2nd Feature Map



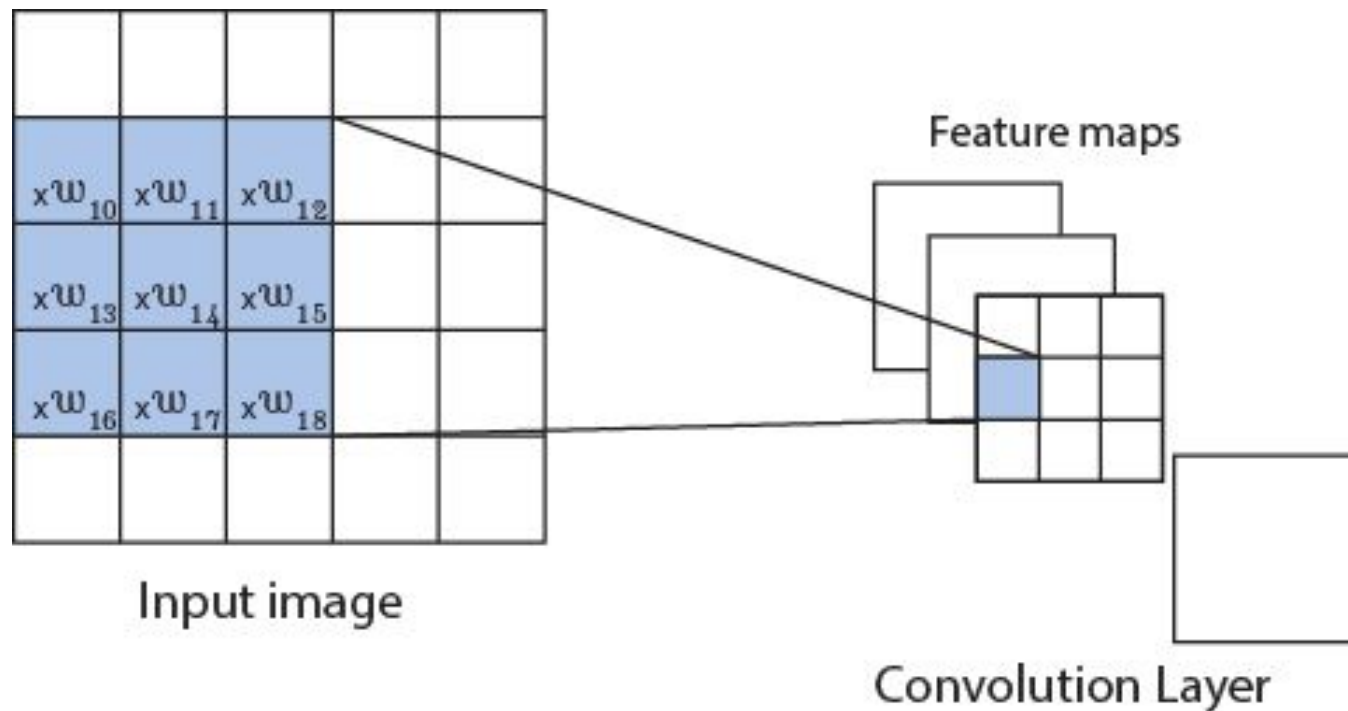
# CNN - Example - 2nd Feature Map



# CNN - Example - 2nd Feature Map

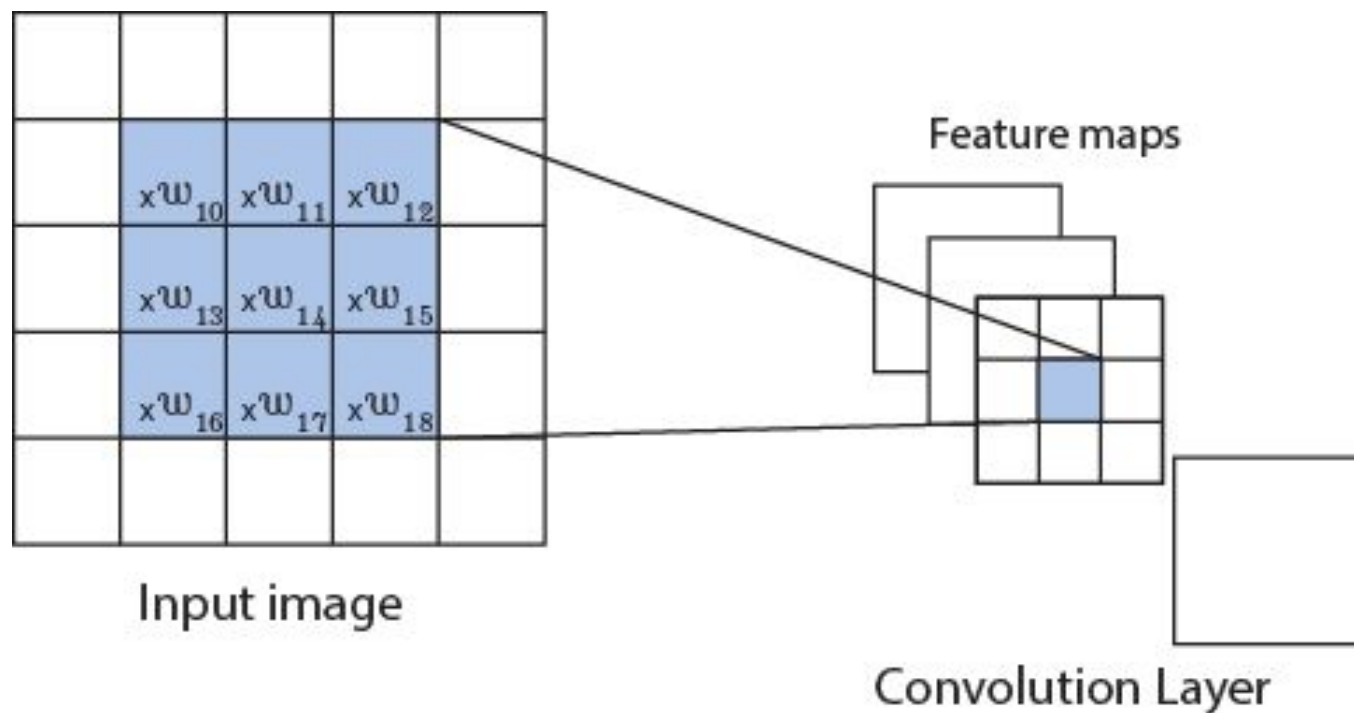


# CNN - Example - 2nd Feature Map

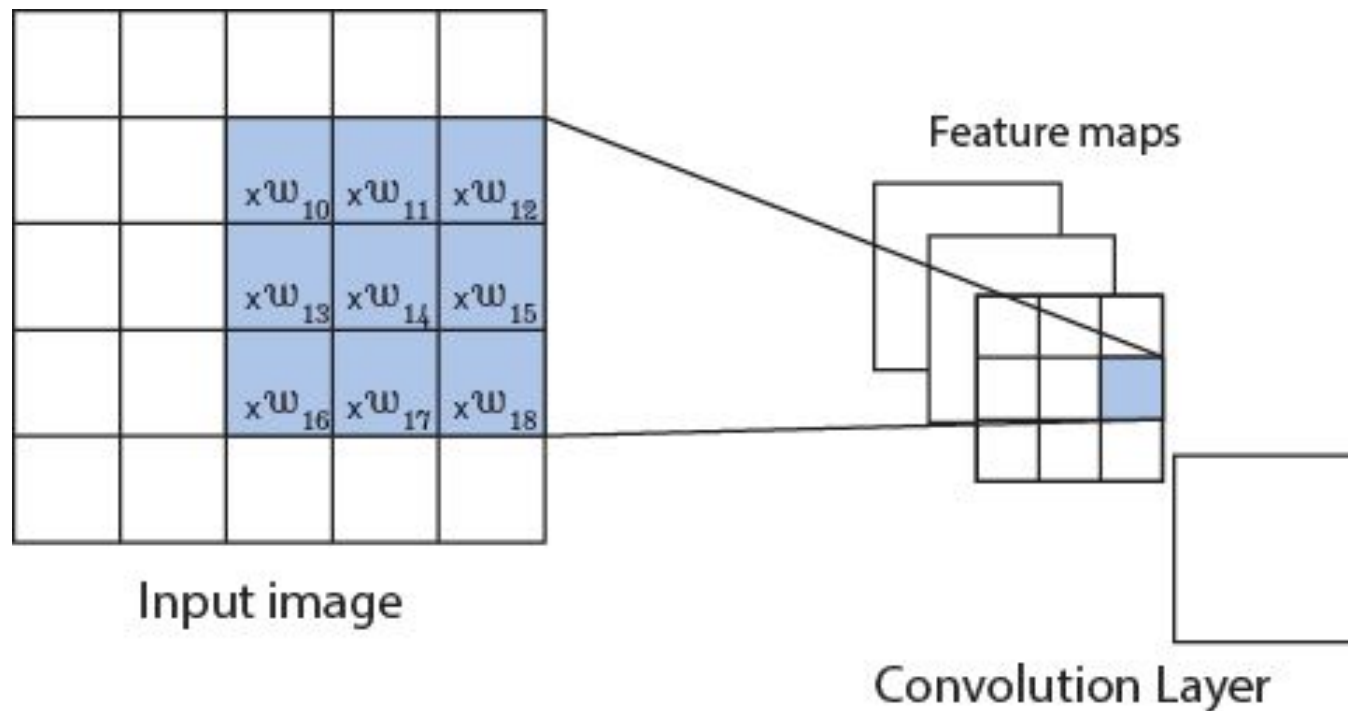




# CNN - Example - 2nd Feature Map



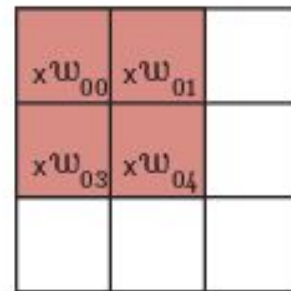
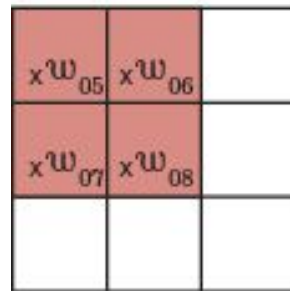
# CNN - Example - 2nd Feature Map



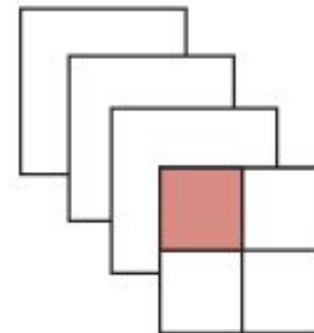
# CNN - Example - Consecutive Convolutions

- Each filter in above layer performs convolution on all filters in previous layer, same for colour channels.

2x2 Kernel → 1 Stride



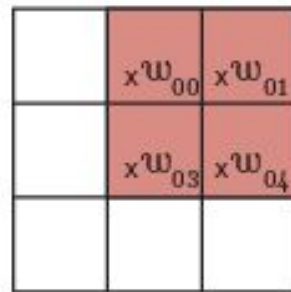
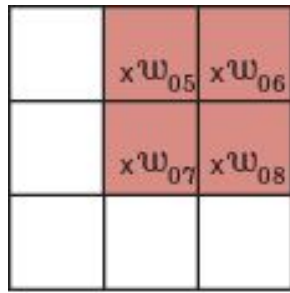
Convolution with  
2 feature maps



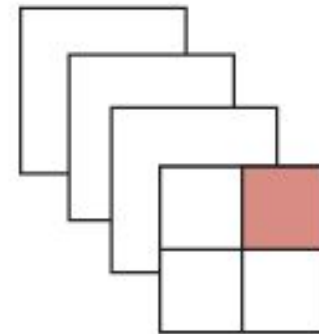
Convolution Layer

# CNN - Example - Consecutive Convolutions

- Each filter in above layer performs convolution on all filters in previous layer, same for colour channels.



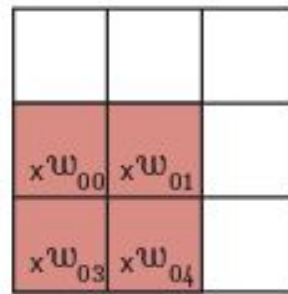
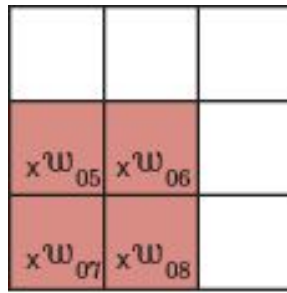
Convolution with  
2 feature maps



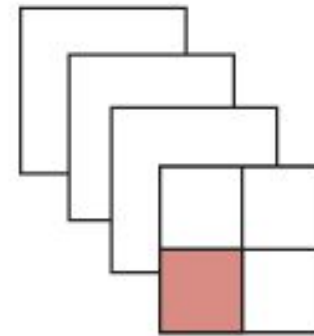
Convolution Layer

# CNN - Example - Consecutive Convolutions

- Each filter in above layer performs convolution on all filters in previous layer, same for colour channels.



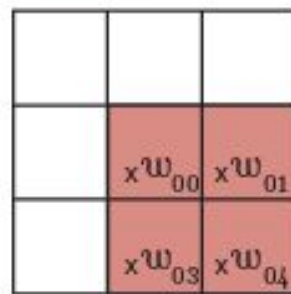
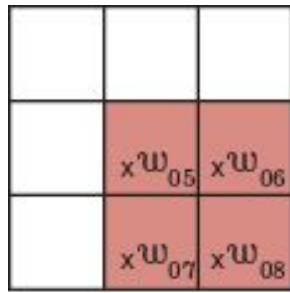
Convolution with  
2 feature maps



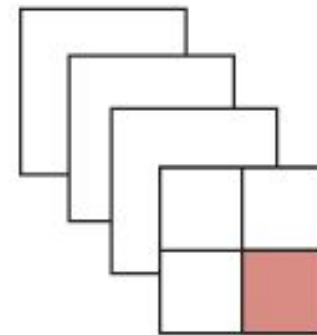
Convolution Layer

# CNN - Example - Consecutive Convolutions

- Each filter in above layer performs convolution on all filters in previous layer, same for colour channels.



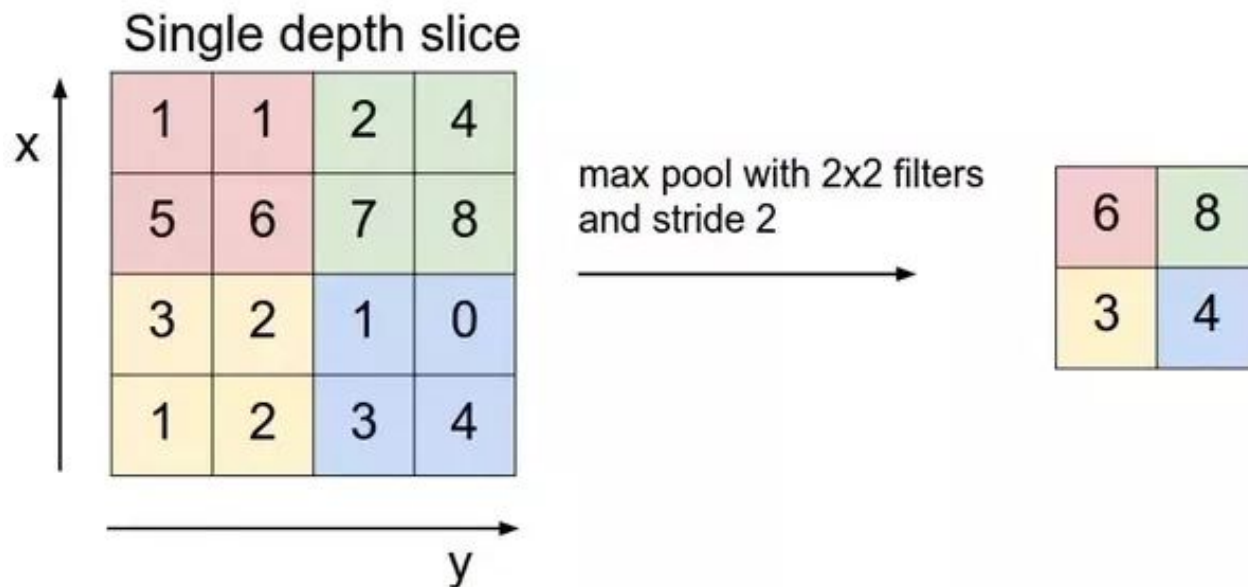
Convolution with  
2 feature maps



Convolution Layer

# Pooling

- Pooling performs subsampling and reduces network size
- Example of MAX pooling (selecting the maximum value)



[<http://cs231n.github.io/>]



Instructor:



DEEP  
LEARNING  
INSTITUTE

[www.nvidia.com/dli](http://www.nvidia.com/dli)