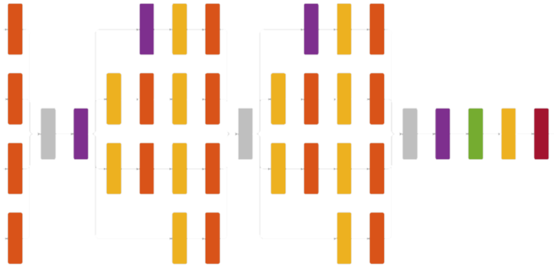
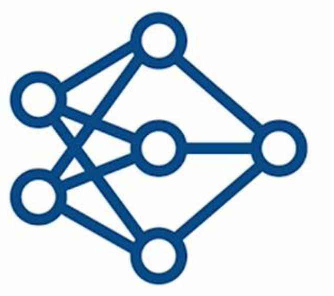
Deep Learning with MATLAB

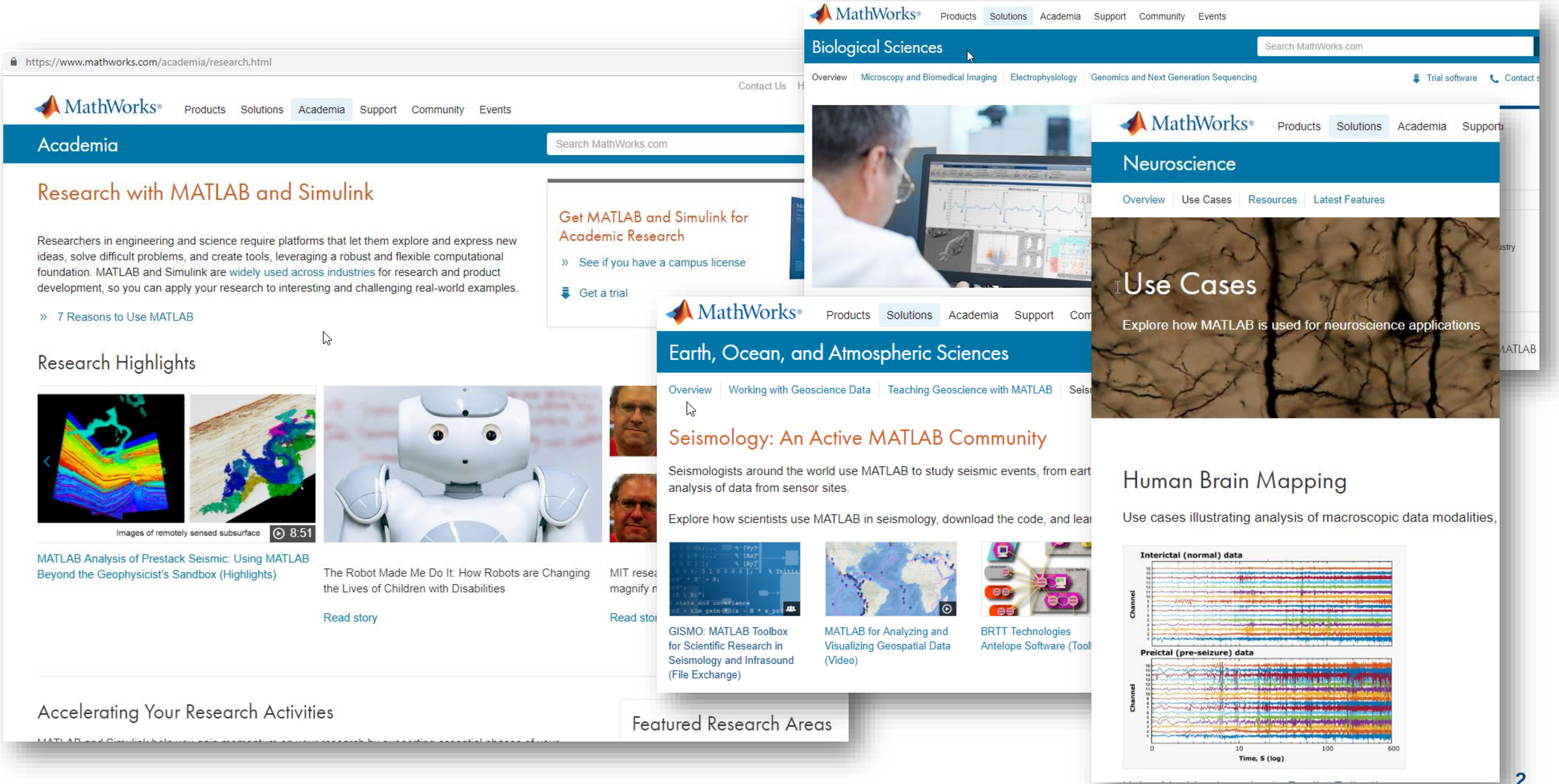
Cagliari (Italy), 19 Sep. 2019



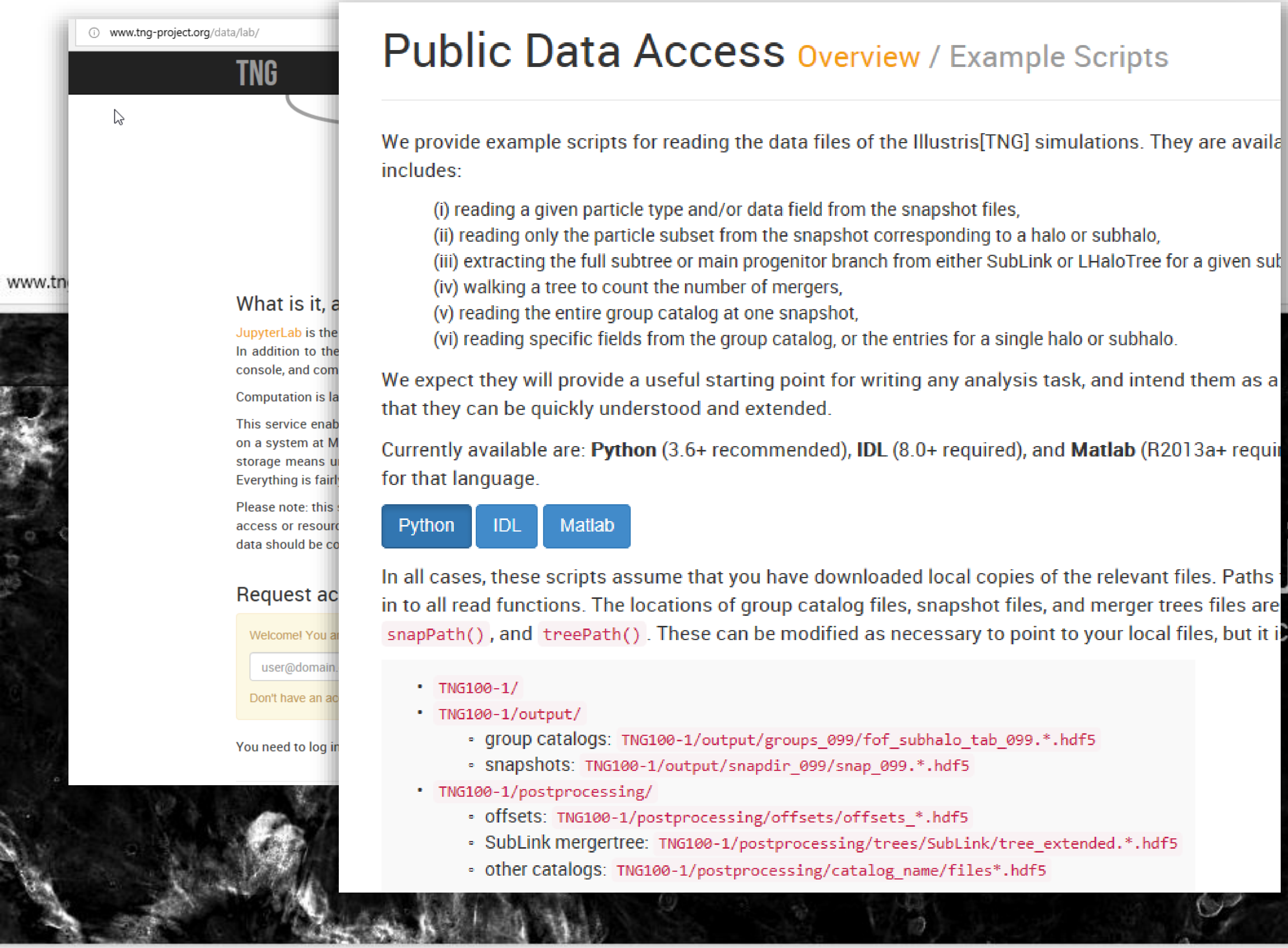


Giuseppe Ridinò
Senior Application Engineer
MathWorks Italy

MATLAB & Simulink in Research



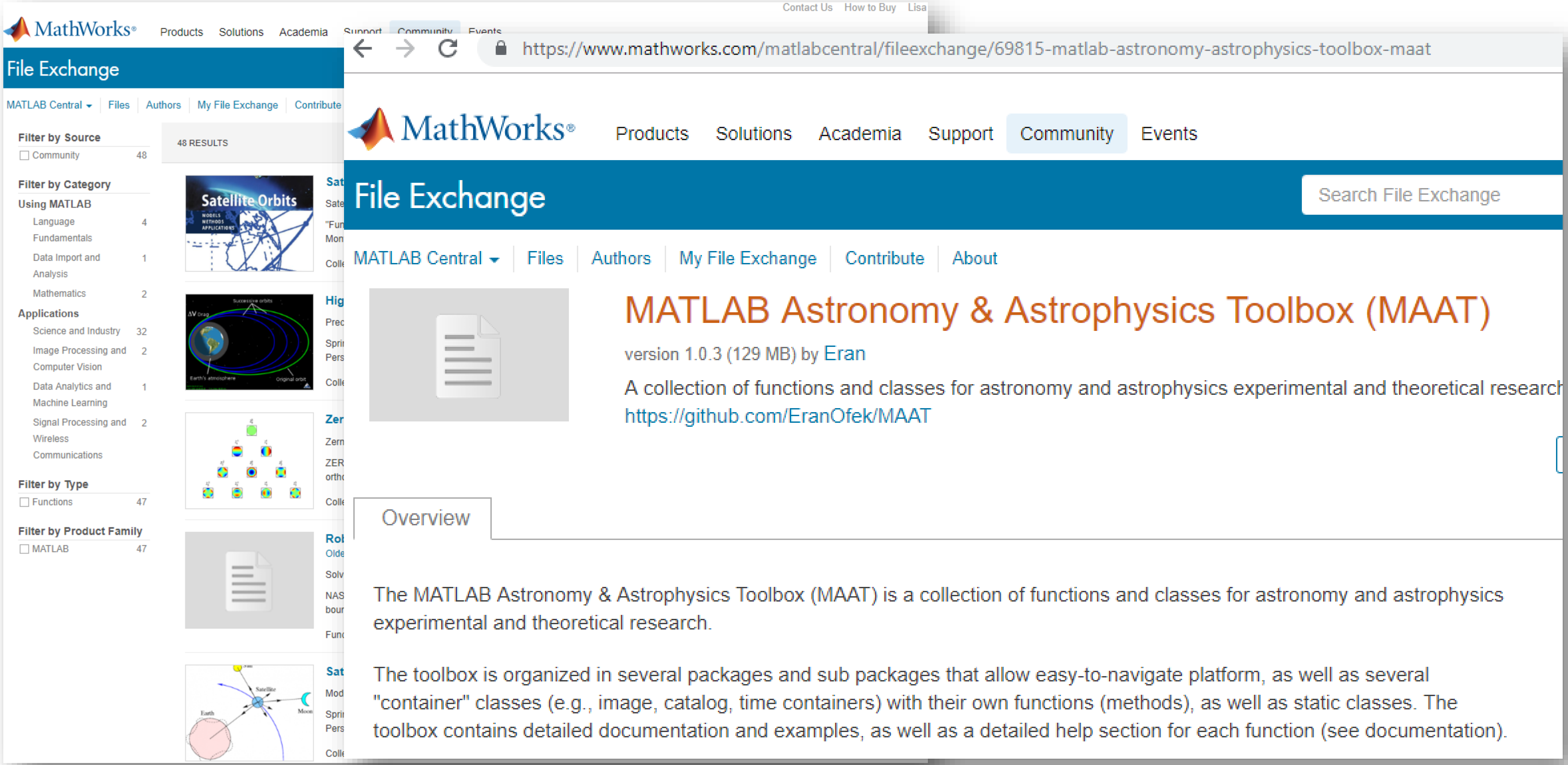
TNG Data Project and Jupyterlab Interface



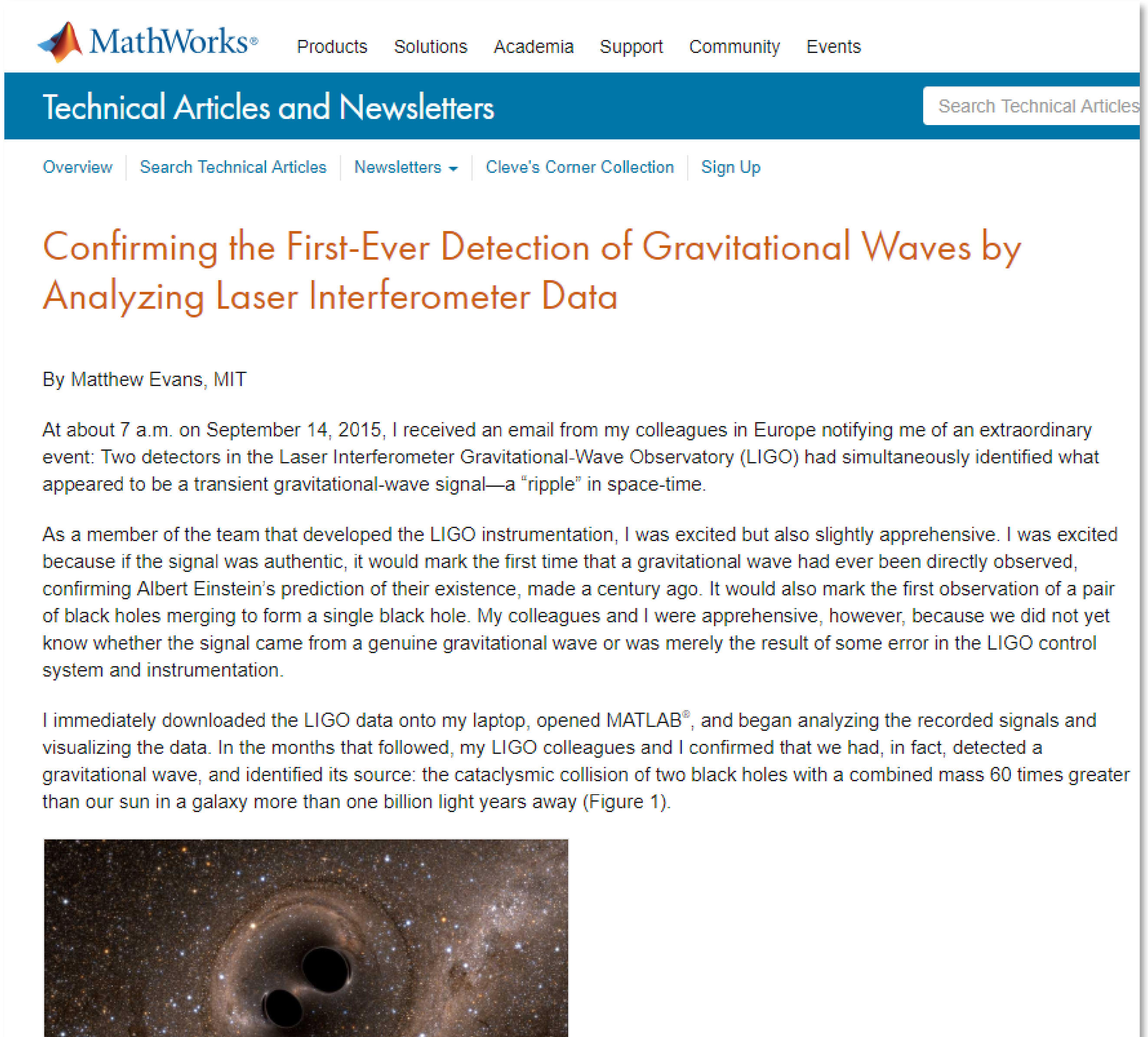
Project: http://www.tng-project.org/
Jupyterlab: tng-project.org/data/lab/
Scripts: http://www.tng-project.org/data/docs/scripts/

MATLAB Community Tools for Astronomy

https://www.mathworks.com/matlabcentral/fileexchange?sort=downloads_desc&q=astronomy



LIGO: Spot 1st-Ever Gravitational Waves with MATLAB



MathWorks® Products Solutions Academia Support Community Events

Technical Articles and Newsletters

Search Technical Articles

Overview | Search Technical Articles | Newsletters | Cleve's Corner Collection | Sign Up

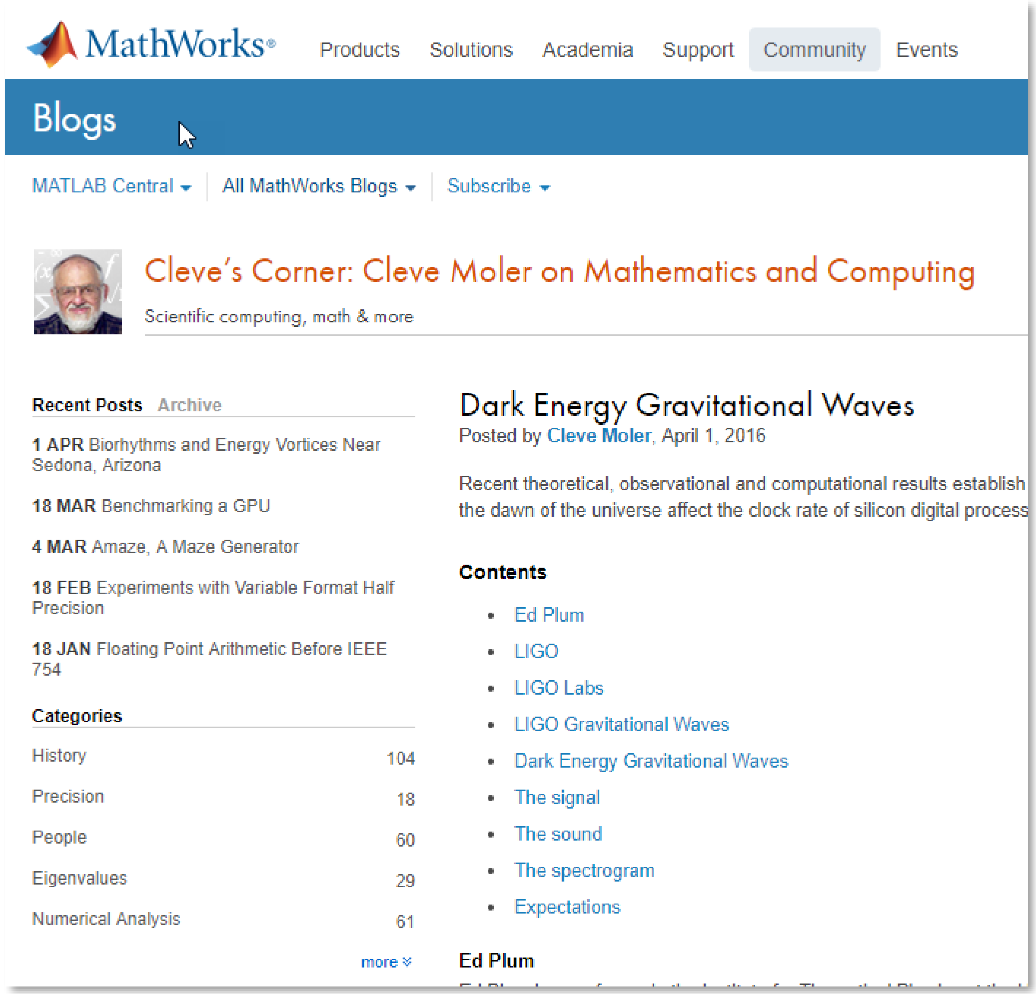
Confirming the First-Ever Detection of Gravitational Waves by Analyzing Laser Interferometer Data

By Matthew Evans, MIT

At about 7 a.m. on September 14, 2015, I received an email from my colleagues in Europe notifying me of an extraordinary event: Two detectors in the Laser Interferometer Gravitational-Wave Observatory (LIGO) had simultaneously identified what appeared to be a transient gravitational-wave signal—a "ripple" in space-time.

As a member of the team that developed the LIGO instrumentation, I was excited but also slightly apprehensive. I was excited because if the signal was authentic, it would mark the first time that a gravitational wave had ever been directly observed, confirming Albert Einstein's prediction of their existence, made a century ago. It would also mark the first observation of a pair of black holes merging to form a single black hole. My colleagues and I were apprehensive, however, because we did not yet know whether the signal came from a genuine gravitational wave or was merely the result of some error in the LIGO control system and instrumentation.

I immediately downloaded the LIGO data onto my laptop, opened MATLAB®, and began analyzing the recorded signals and visualizing the data. In the months that followed, my LIGO colleagues and I confirmed that we had, in fact, detected a gravitational wave, and identified its source: the cataclysmic collision of two black holes with a combined mass 60 times greater than our sun in a galaxy more than one billion light years away (Figure 1).



MathWorks® Products Solutions Academia Support Community Events

Blogs

MATLAB Central | All MathWorks Blogs | Subscribe

Cleve's Corner: Cleve Moler on Mathematics and Computing

Scientific computing, math & more

Recent Posts Archive

- **1 APR** Biorhythms and Energy Vortices Near Sedona, Arizona
- **18 MAR** Benchmarking a GPU
- **4 MAR** Amaze, A Maze Generator
- **18 FEB** Experiments with Variable Format Half Precision
- **18 JAN** Floating Point Arithmetic Before IEEE 754

Categories

History	104
Precision	18
People	60
Eigenvalues	29
Numerical Analysis	61

more

Dark Energy Gravitational Waves

Posted by Cleve Moler, April 1, 2016

Recent theoretical, observational and computational results establish the dawn of the universe affect the clock rate of silicon digital process

Contents

Ed Plum

[Article](#)
[Blog post](#)

NASA Build Kepler Pipeline tools with MATLAB

download: github.com/nasa/kepler-pipeline

GitHub, Inc. [US] | https://github.com/nasa/kepler-pipeline

Kepler Science Data Processing Pipeline

The Kepler telescope launched into orbit in March 2009, initiating NASA's first mission to discover Earth-size planets orbiting Sun-like stars. Kepler simultaneously collected data for ~160,000 target stars over its four-year mission, identifying over 4700 planet candidates, 2300 confirmed or validated planets, and 2100 eclipsing binaries. While Kepler was designed to discover exoplanets, the long term, ultra-high photometric precision measurements it achieved also make it a premier observational facility for stellar astrophysics, especially in the field of asteroseismology, and for variable stars, such as RR Lyrae stars. The Kepler Science Operations Center (SOC) was developed at NASA Ames Research Center to process the data acquired by Kepler starting with pixel-level calibrations all the way to identifying transiting planet signatures and subjecting them to a suite of diagnostic tests to establish or break confidence in their planetary nature. Detecting small, rocky planets transiting Sun-like stars presents a variety of daunting challenges, including achieving an unprecedented photometric precision of ~20 ppm on 6.5-hour timescales, supporting the science operations, management, and repeated reprocessing of the accumulating data stream.

The scientific objective of the Kepler Mission is to explore the structure and diversity of planetary systems. This is achieved by surveying a large sample of stars to:

- Determine the abundance of terrestrial and larger planets in or near the habitable zone of a wide variety of stars;
- Determine the distribution of sizes and shapes of the orbits of these planets;
- Estimate how many planets are in multiple-star systems;
- Determine the variety of orbit sizes and planet reflectivities, radii, masses and densities of short-period giant planets;
- Identify additional members of each discovered planetary system using other techniques; and
- Determine the properties of those stars that harbor planetary systems.

Agenda

Deep Learning Overview

Deep Learning in 6 Lines of MATLAB Code

Deep Learning with…

Images	Signals

Labeling Ground Truth Data

Working with Other Frameworks

Performance, Deployment, and Next Steps

Artificial Intelligence

- Development of computer systems to perform tasks that normally require human intelligence

Machine Learning

Decision Trees

Deep Learning

Logistic Regression

K-means

SVM

Nearest Neighbor

Reinforcement Learning

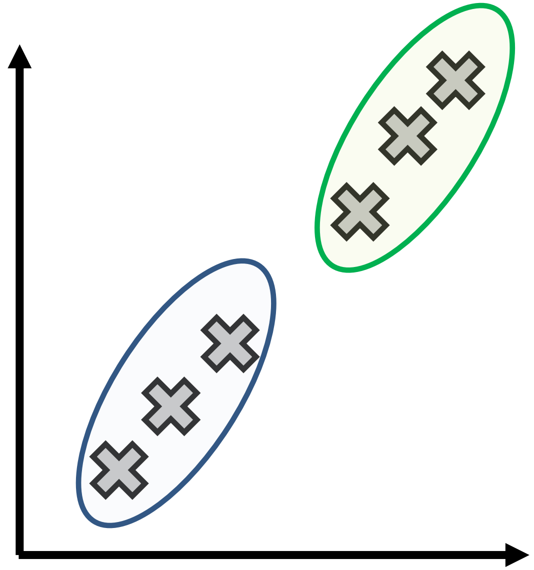
Gaussian Mixture

Machine Learning and Deep Learning

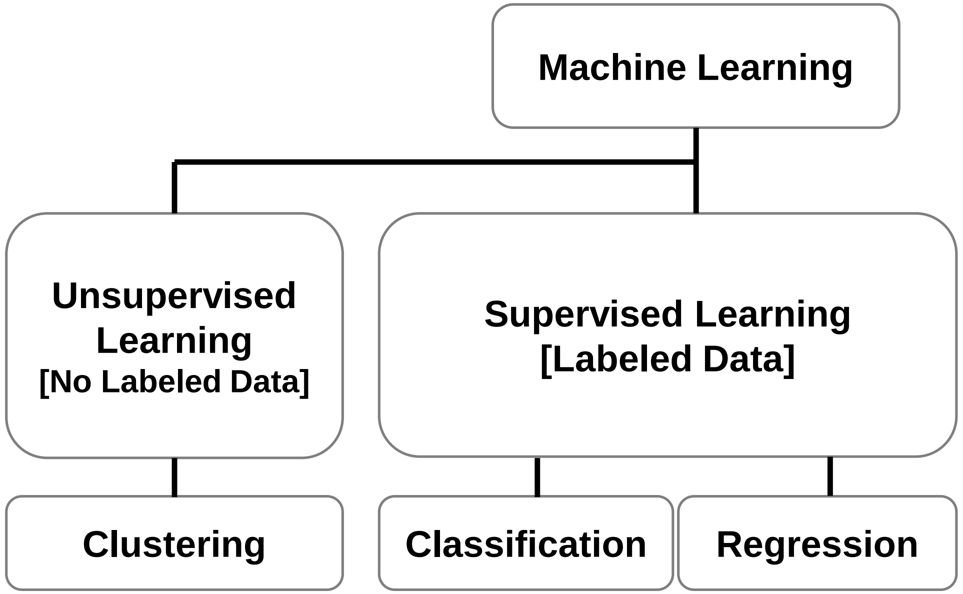
Machine Learning

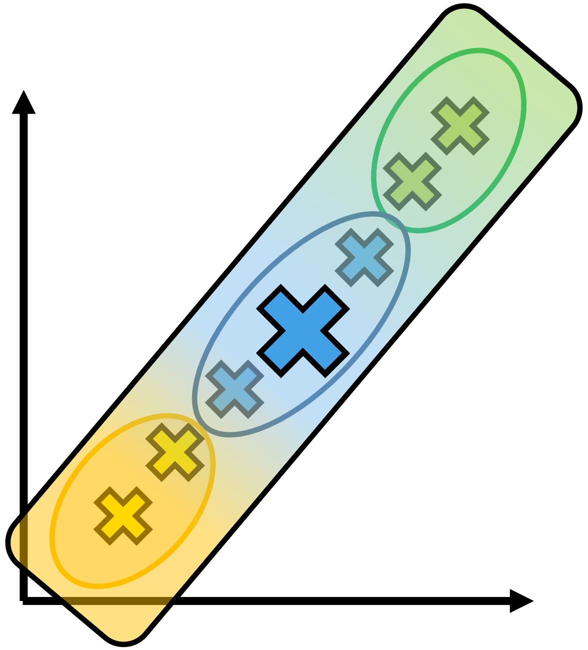
Unsupervised Learning
[No Labeled Data]

Clustering

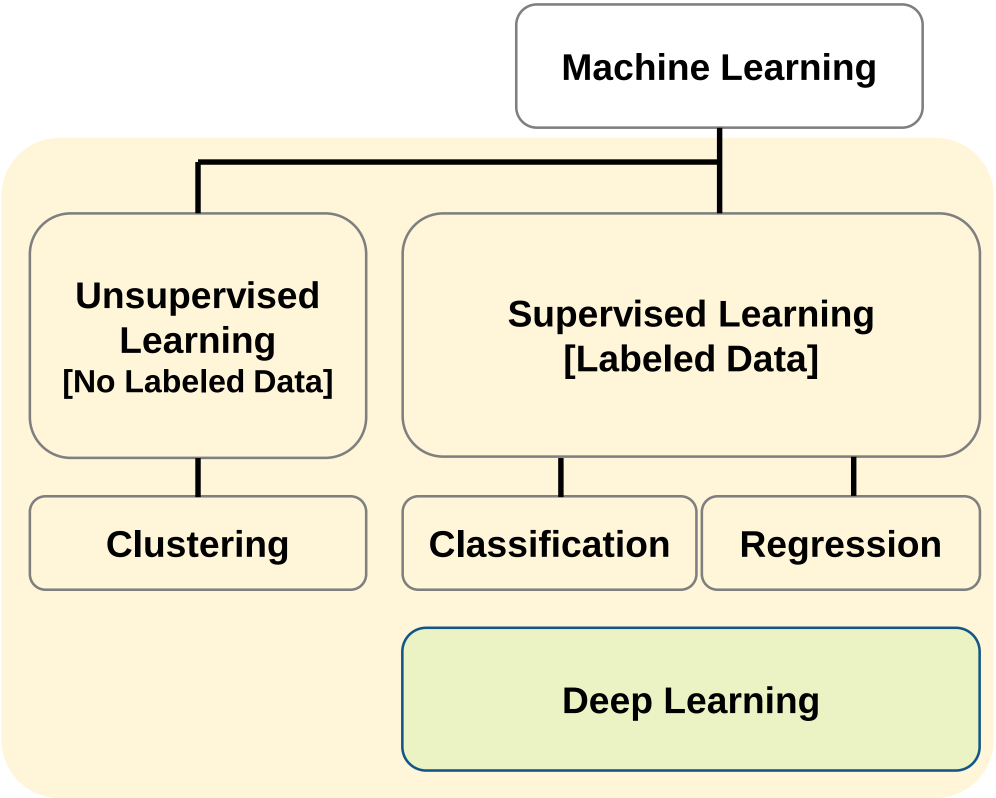


Machine Learning and Deep Learning





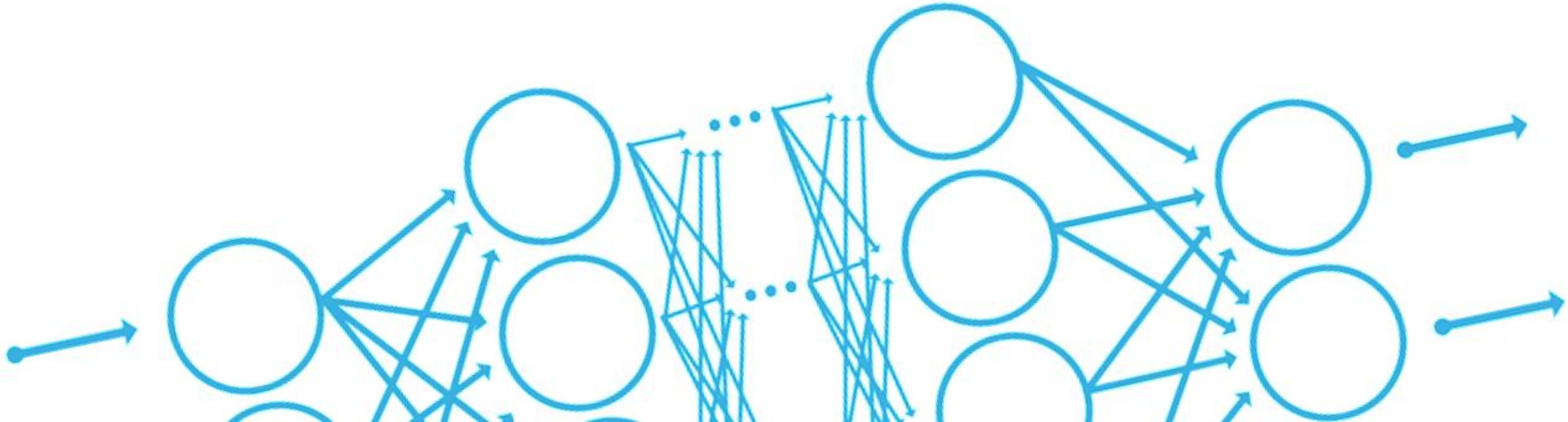
Machine Learning and Deep Learning



Supervised learning typically involves feature extraction

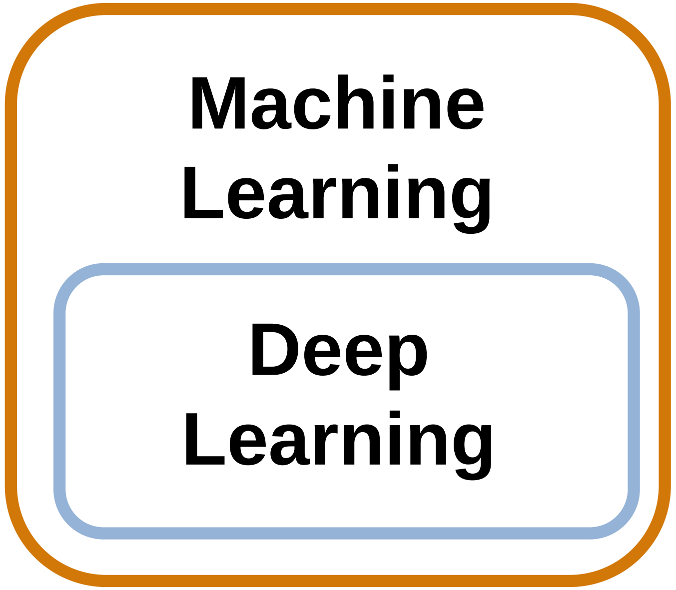
Deep learning typically does not involve feature extraction

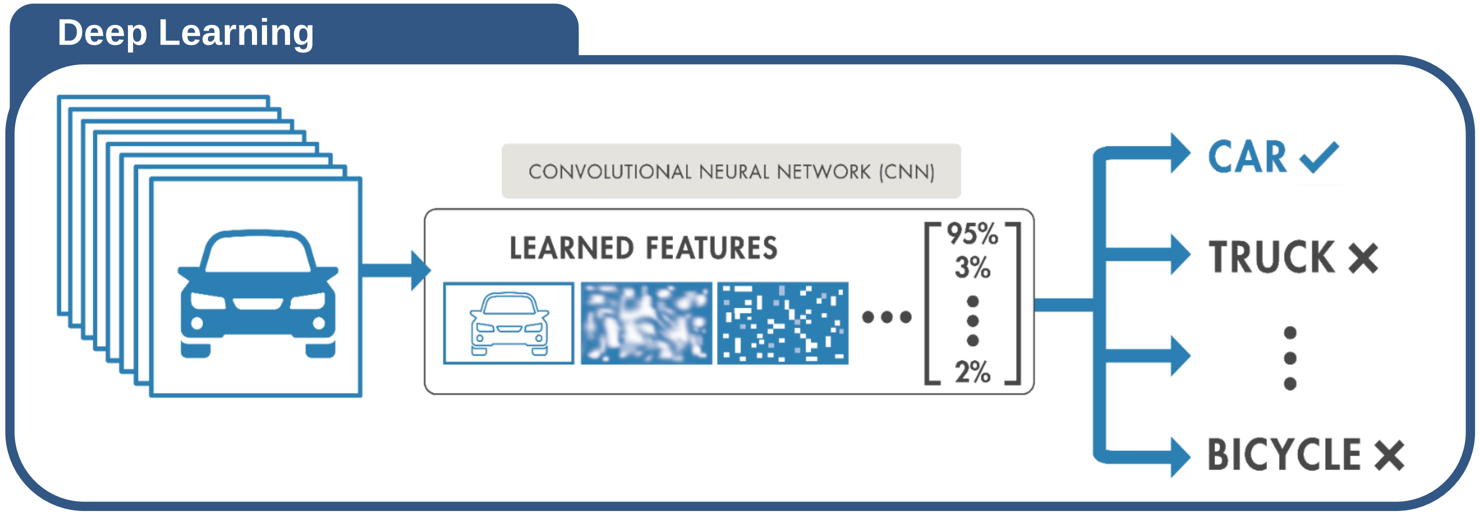
What is Deep Learning?



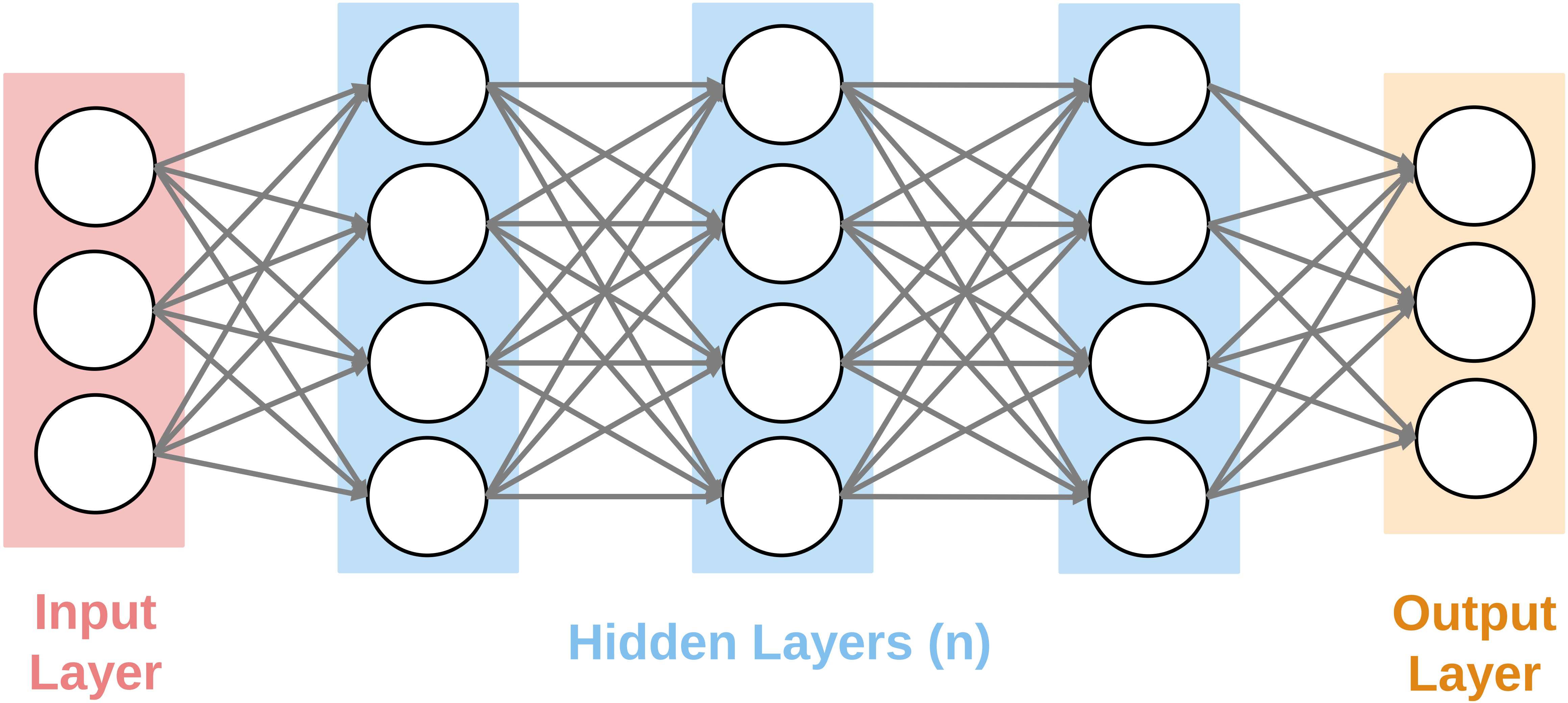
Deep Learning

- Subset of machine learning with automatic feature extraction
  – Learns features and tasks directly from data
  – More Data = better model



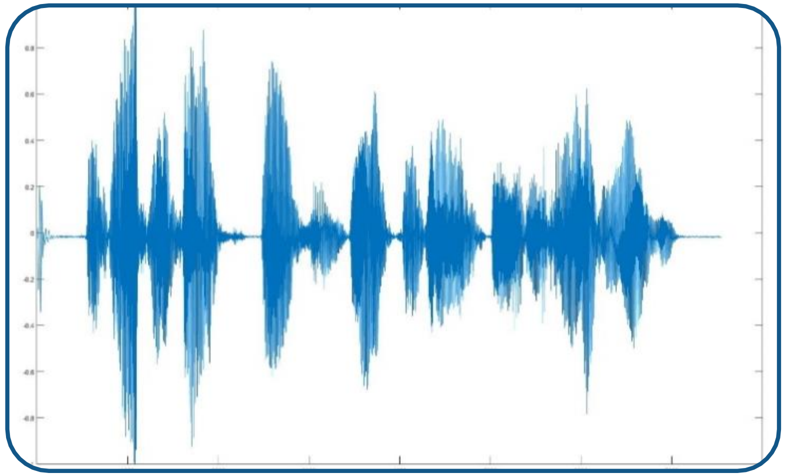


Deep Learning uses a neural network architecture



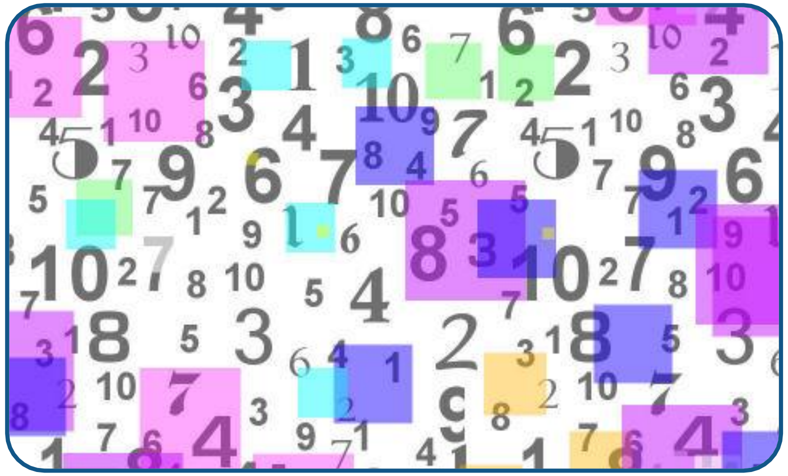
Deep Learning datatypes

Image



Signal

Numeric



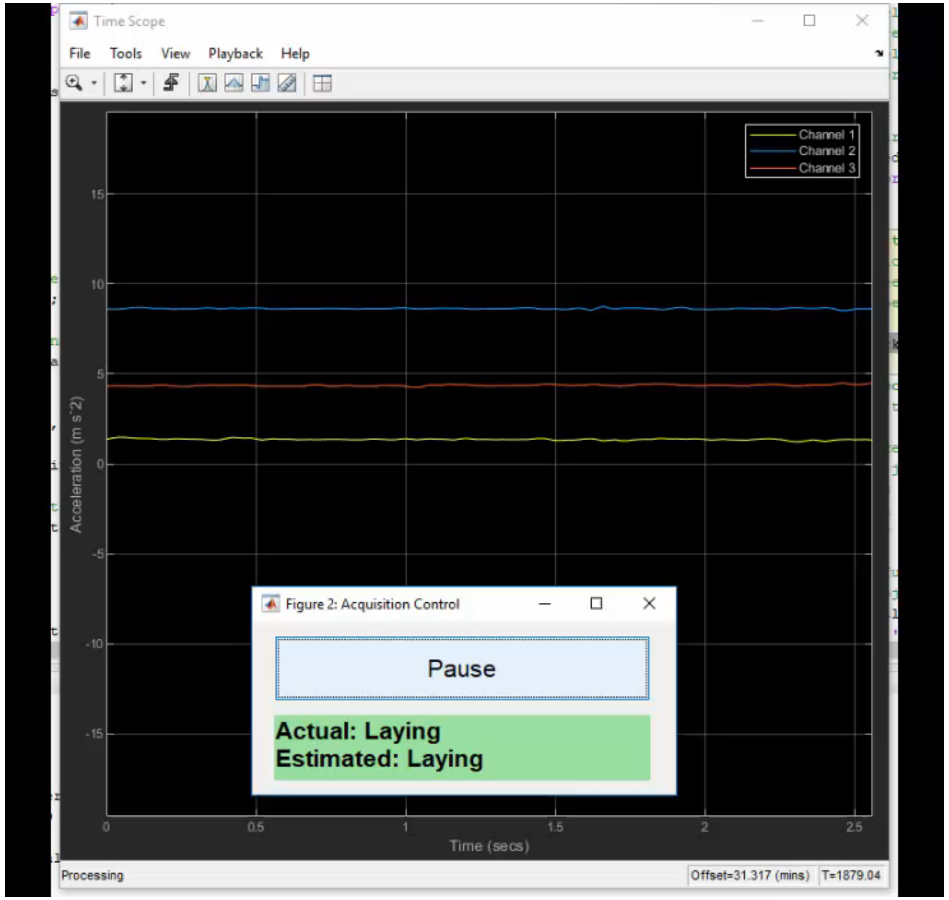
Text

Example 1: Detection and localization using deep learning

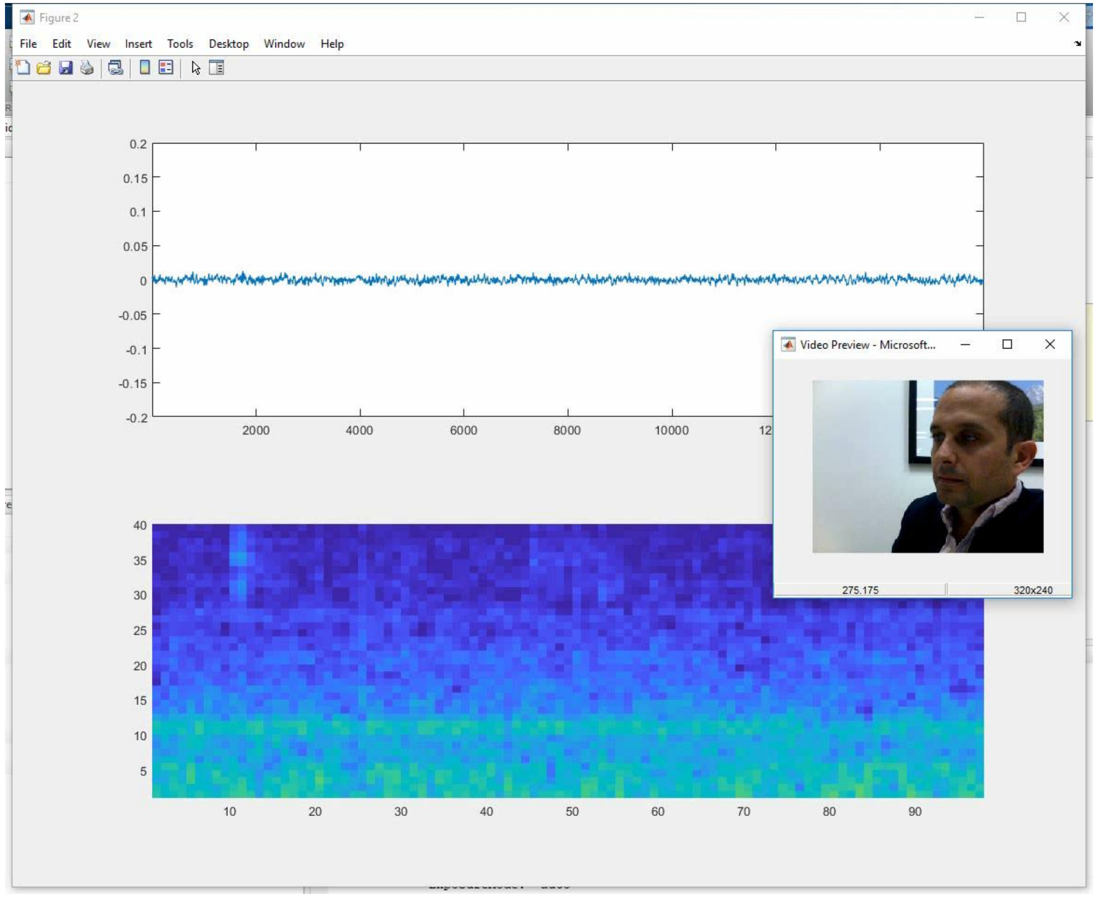
YOLO v2 (You Only Look Once)

Semantic Segmentation using SegNet

Example 2: Analyzing signal data using deep learning



Signal Classification using LSTMs

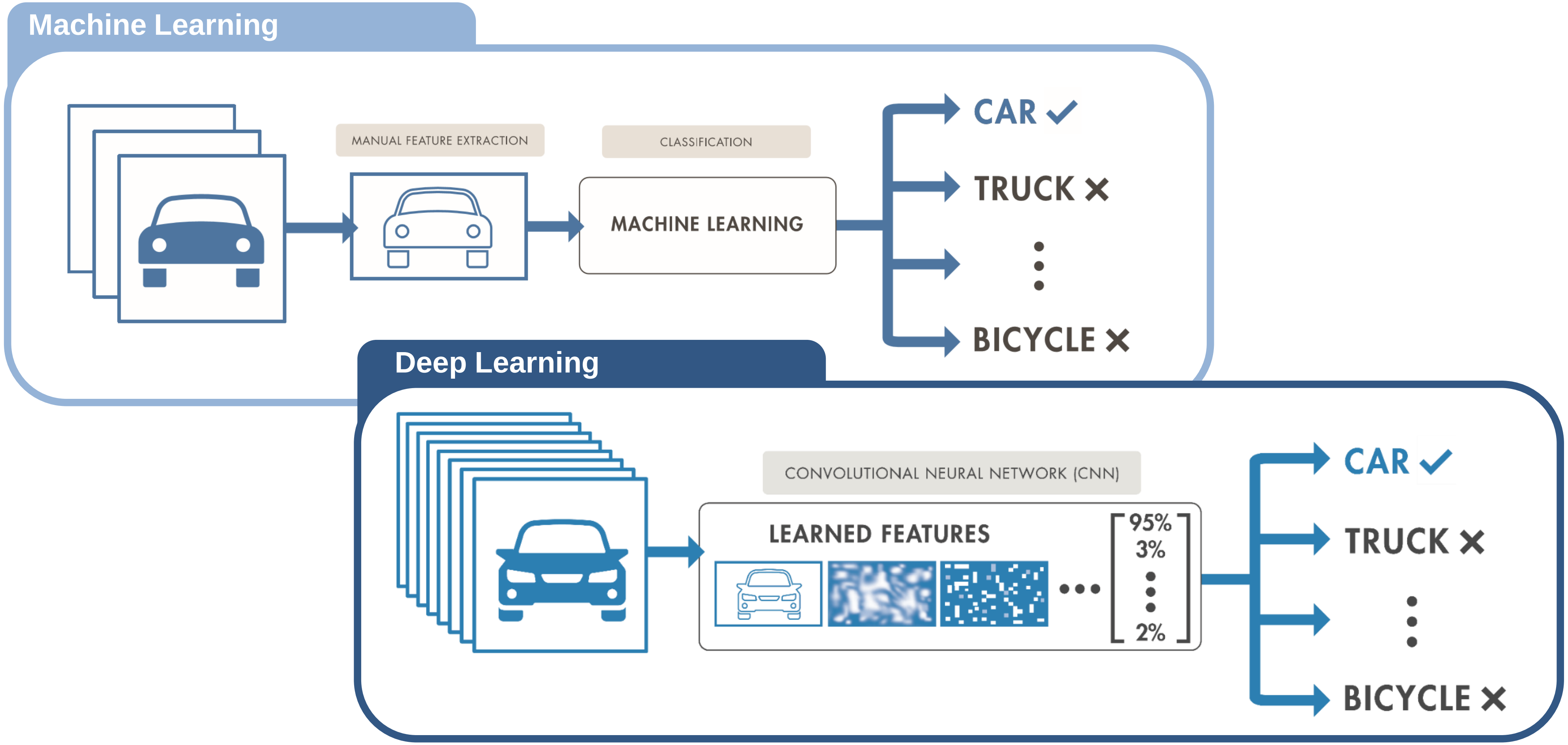


Speech Recognition using CNNs

Machine Learning vs Deep Learning

Deep learning performs **end-to-end learning** by learning **features, representations and tasks** directly from **images, text and sound**

Deep learning algorithms also **scale with data** – traditional machine learning **saturates**



Deep Learning workflow

PREPARE DATA

- Data access and preprocessing
- Ground truth labeling
- Simulation-based data generation

TRAIN MODEL

- Model design, Hyperparameter tuning
- Model exchange across frameworks
- Hardware-accelerated training

DEPLOY SYSTEM

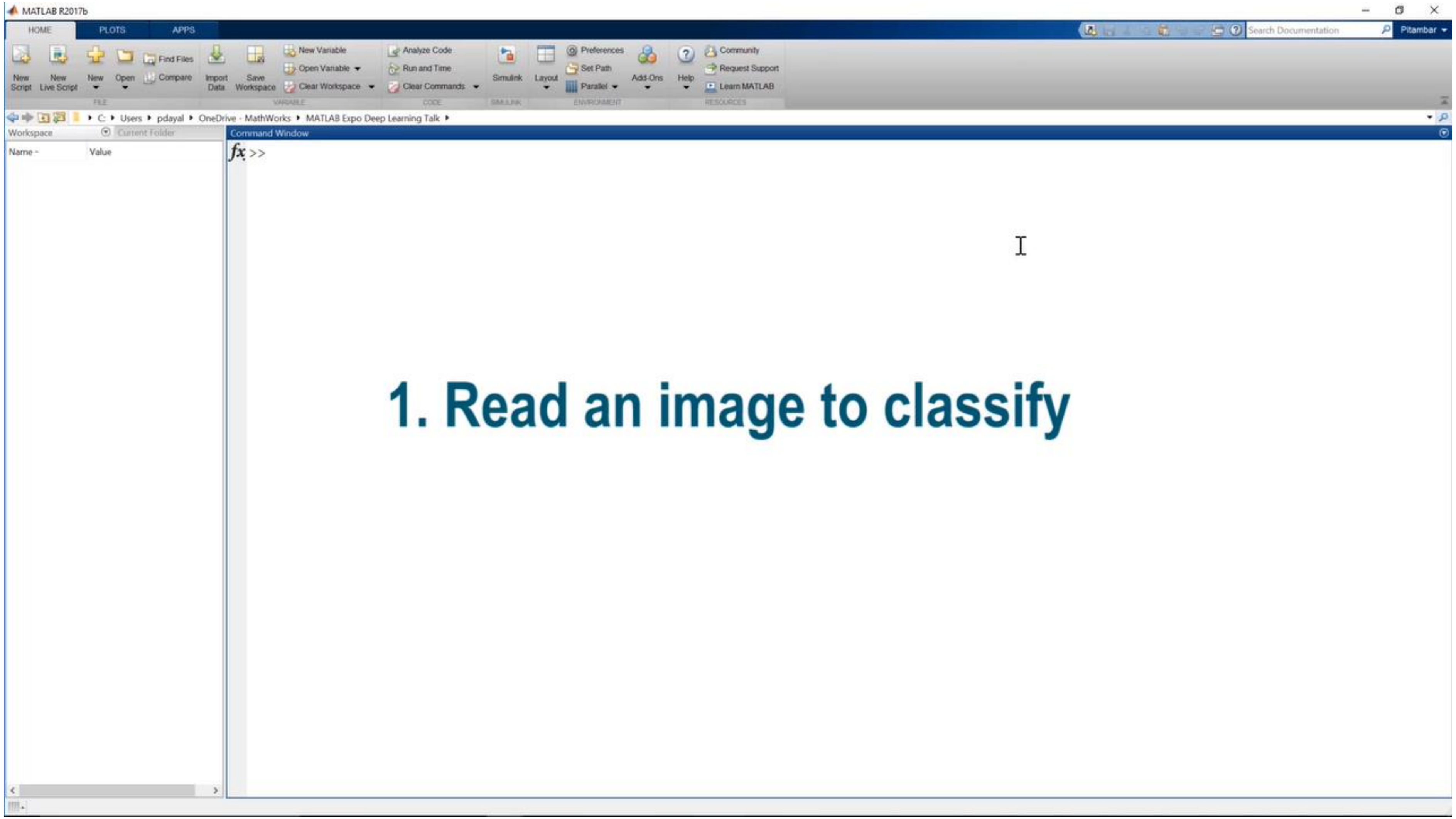
- Multiplatform code generation (CPU, GPU)
- Embedded deployment
- Enterprise Deployment

Iteration and Refinement

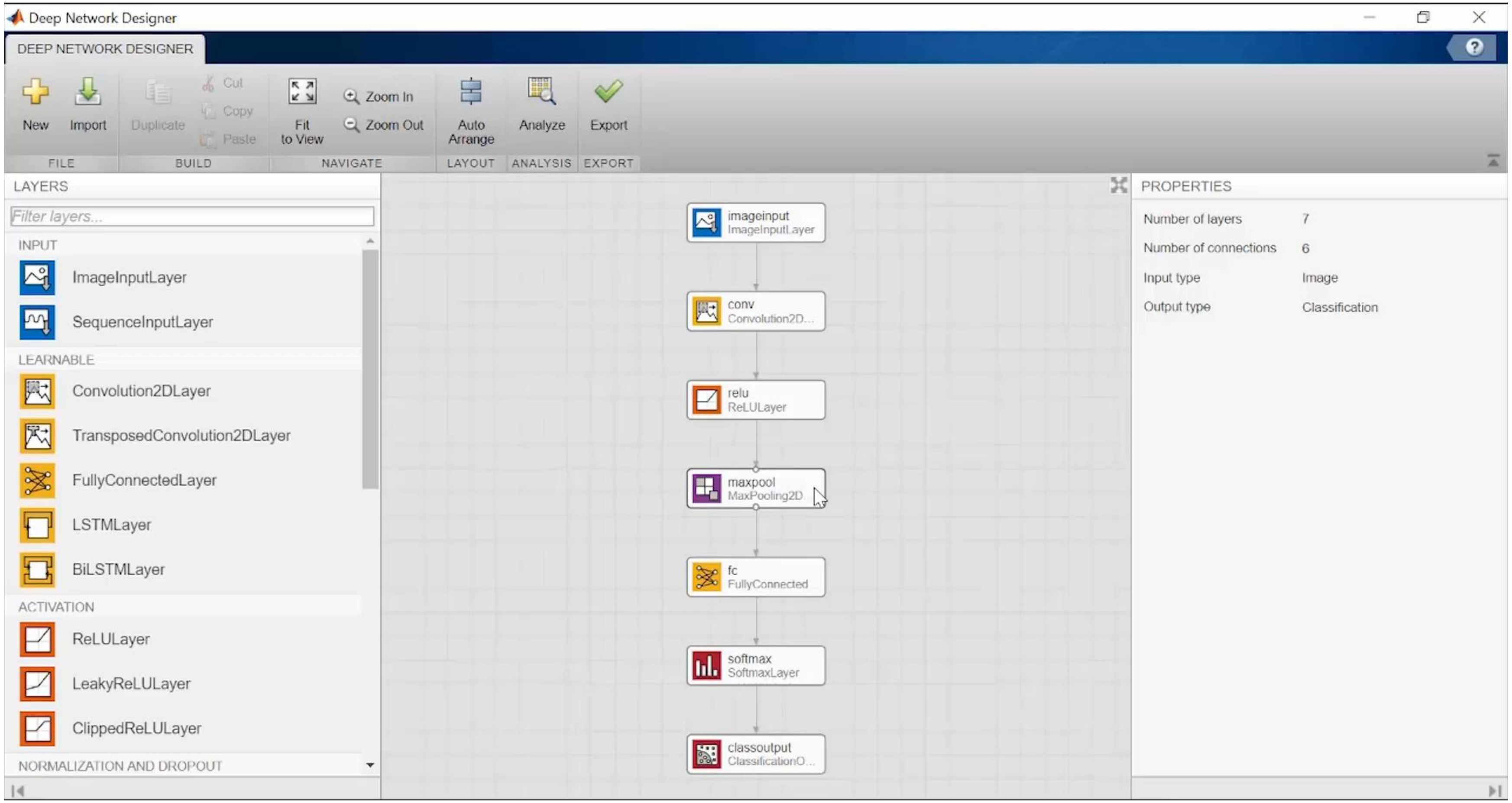
Agenda

Deep Learning Overview	
Deep Learning in 6 Lines of MATLAB Code	
Deep Learning with…	
Images	Signals
Labeling Ground Truth Data	
Working with Other Frameworks	
Performance, Deployment, and Next Steps	

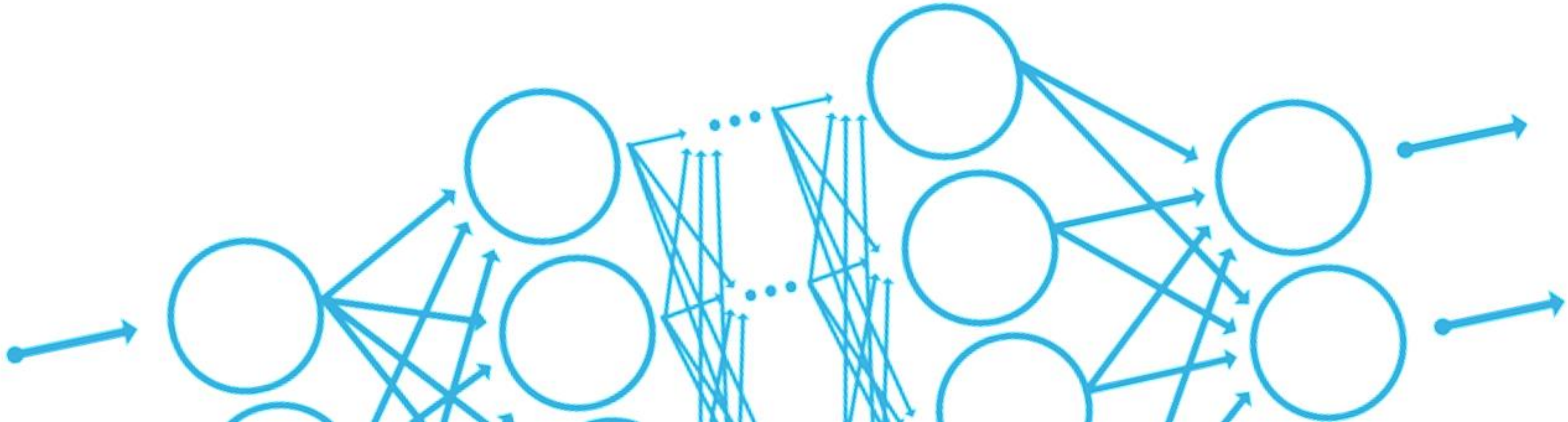
Deep Learning in 6 lines of MATLAB code



Using Apps for designing Deep Learning networks



Different Neural Networks for Different Data



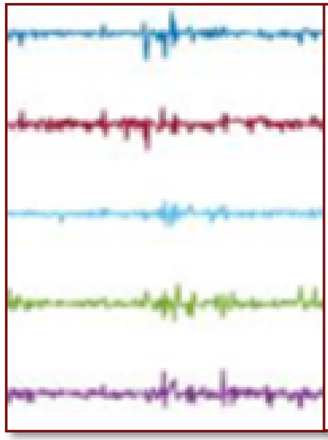
Types of Data

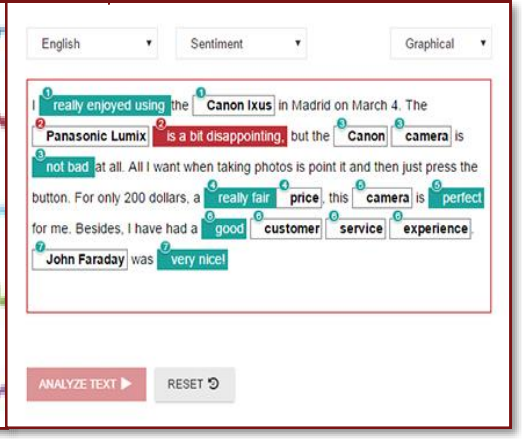
Numeric Data

ID	WC_TA	RE_TA	EBIT_TA	MVE_BVTD	S_TA	Industry	Rating
62394	0.013	0.104	0.036	0.447	0.142	3	BB
48608	0.232	0.335	0.062	1.969	0.281	8	A
42444	0.311	0.367	0.074	1.935	0.366	1	A
48631	0.194	0.263	0.062	1.017	0.228	4	BBB
43768	0.121	0.413	0.057	3.647	0.466	12	AAA
39255	-0.117	-0.799	0.01	0.179	0.082	4	CCC
62236	0.087	0.158	0.049	0.816	0.324	2	BBB
39354	0.005	0.181	0.034	2.597	0.388	7	AA
40326	0.47	0.752	0.07	11.596	1.12	8	AAA
51681	0.11	0.337	0.045	3.835	0.812	4	AAA

Machine Learning or LSTM

Time Series/ Text Data



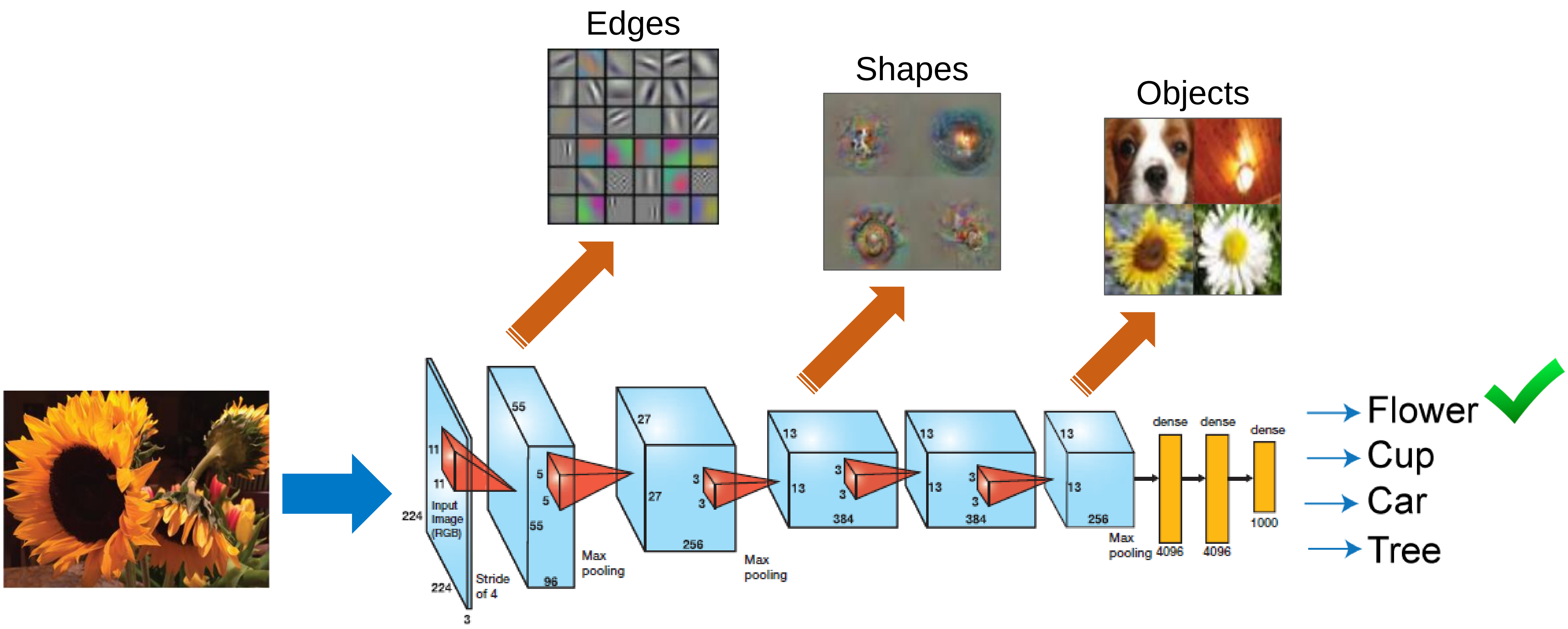


LSTM or CNN

Image Data

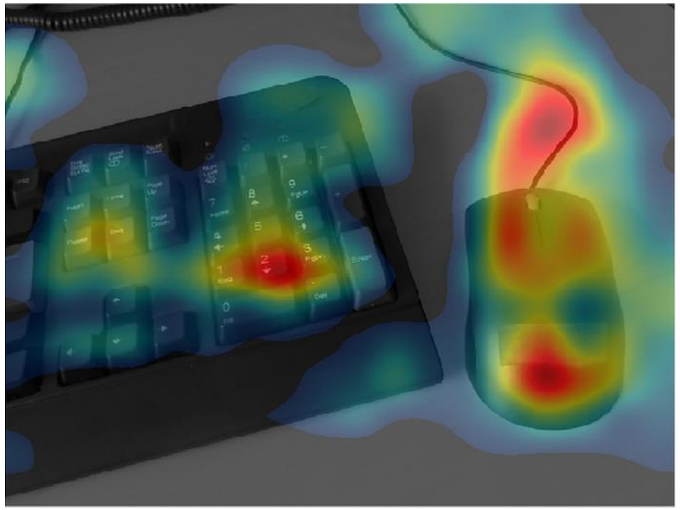
CNN

Convolutional Neural Networks (CNN)

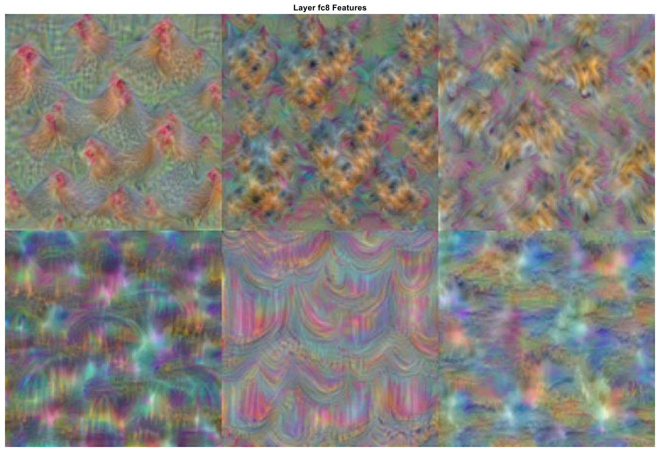


CNN Visualization methods

Layer Activations



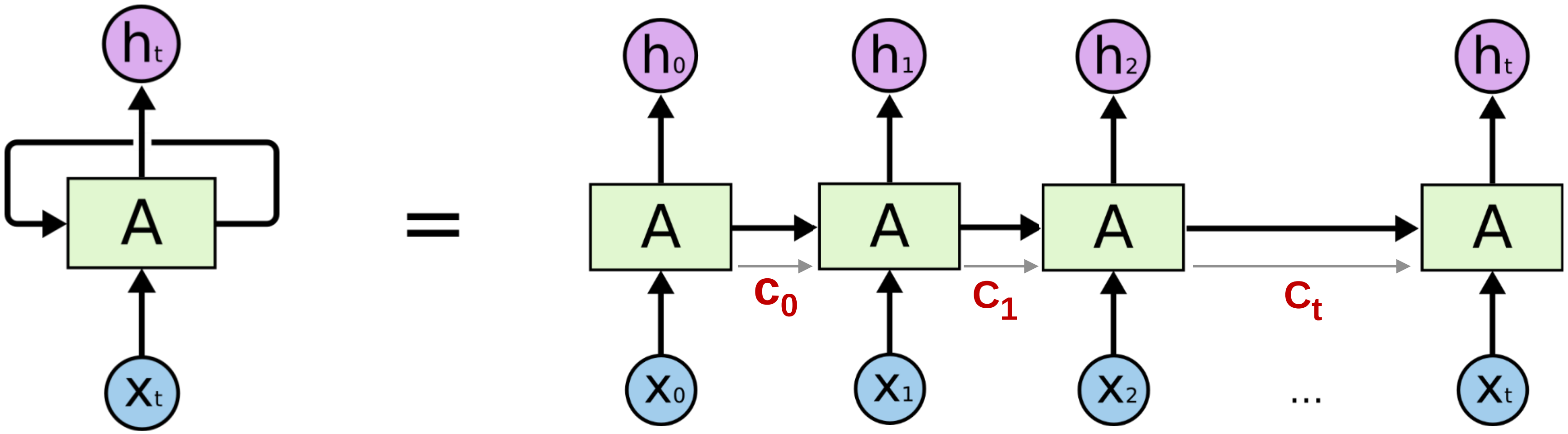
Class Activations



DeepDream Images

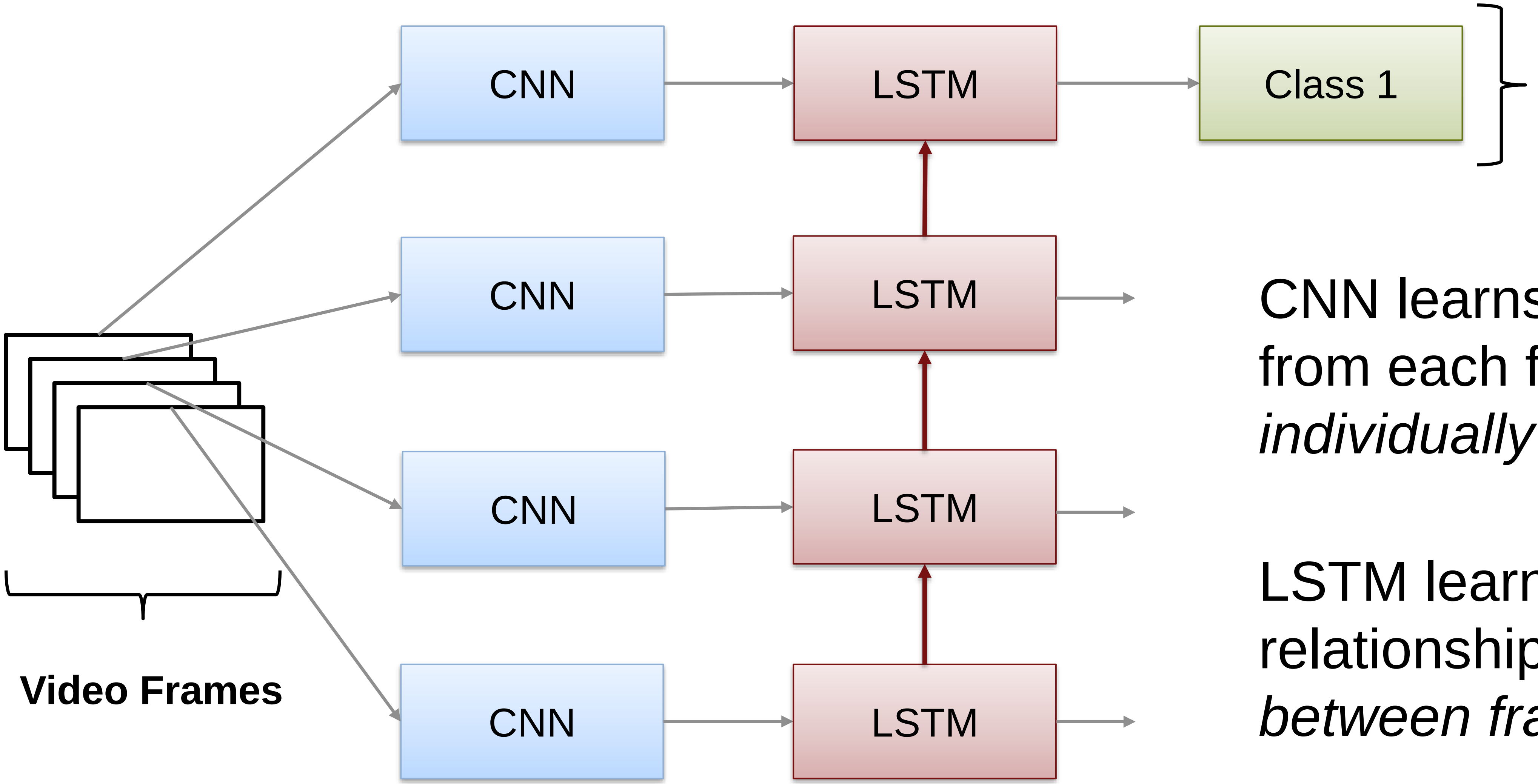
Another network for signals - LSTM

- LSTM = Long Short Term Memory (Networks)
  - Signal, text, time-series data
  - Use previous data to predict new information
- I live in France. I speak ___________.



Combining convolution and recurrent layers

Learn spatial and temporal features simultaneously



Response
class label

CNN learns features from each frame *individually*

LSTM learns relationships *between frames*

Agenda

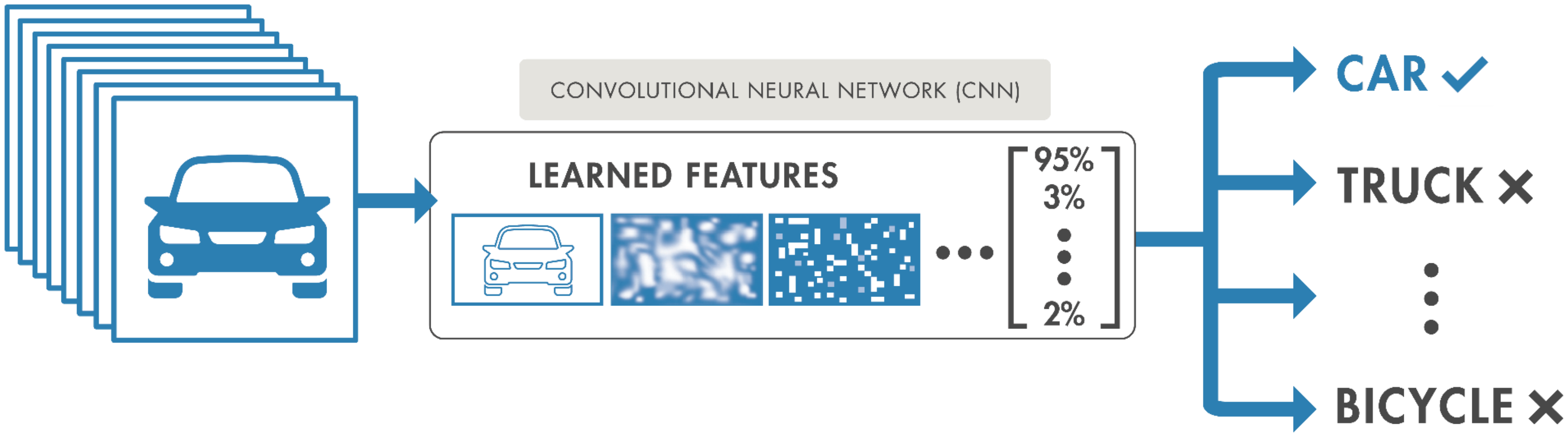
Deep Learning Overview	
Deep Learning in 6 Lines of MATLAB Code	
Deep Learning with…	
Images	Signals
Labeling Ground Truth Data	
Working with Other Frameworks	
Performance, Deployment, and Next Steps	

Agenda

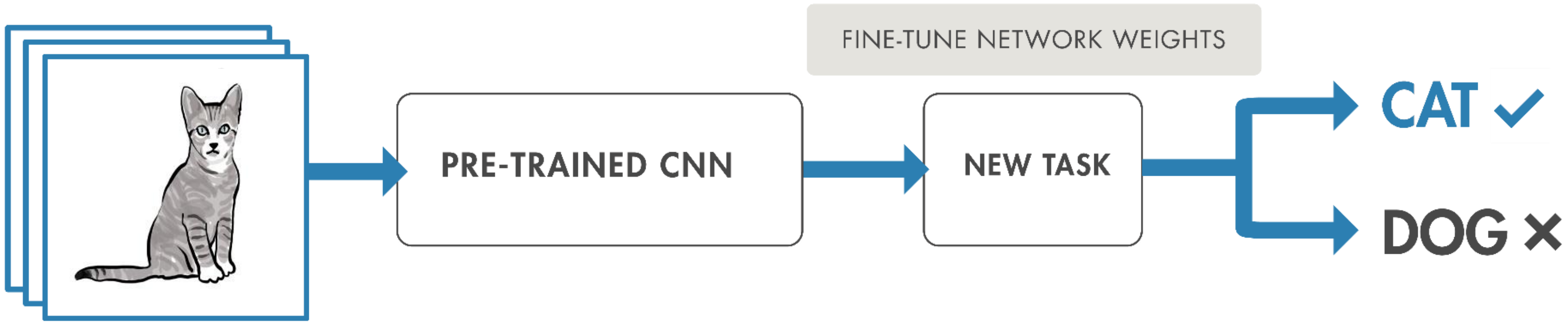
Deep Learning Overview	
Deep Learning in 6 Lines of MATLAB Code	
Deep Learning with…	
Images	Signals
Labeling Ground Truth Data	
Working with Other Frameworks	
Performance, Deployment, and Next Steps	

Two approaches for Deep Learning

1. Train a Deep Neural Network from Scratch



2. Fine-tune a pre-trained model (transfer learning)

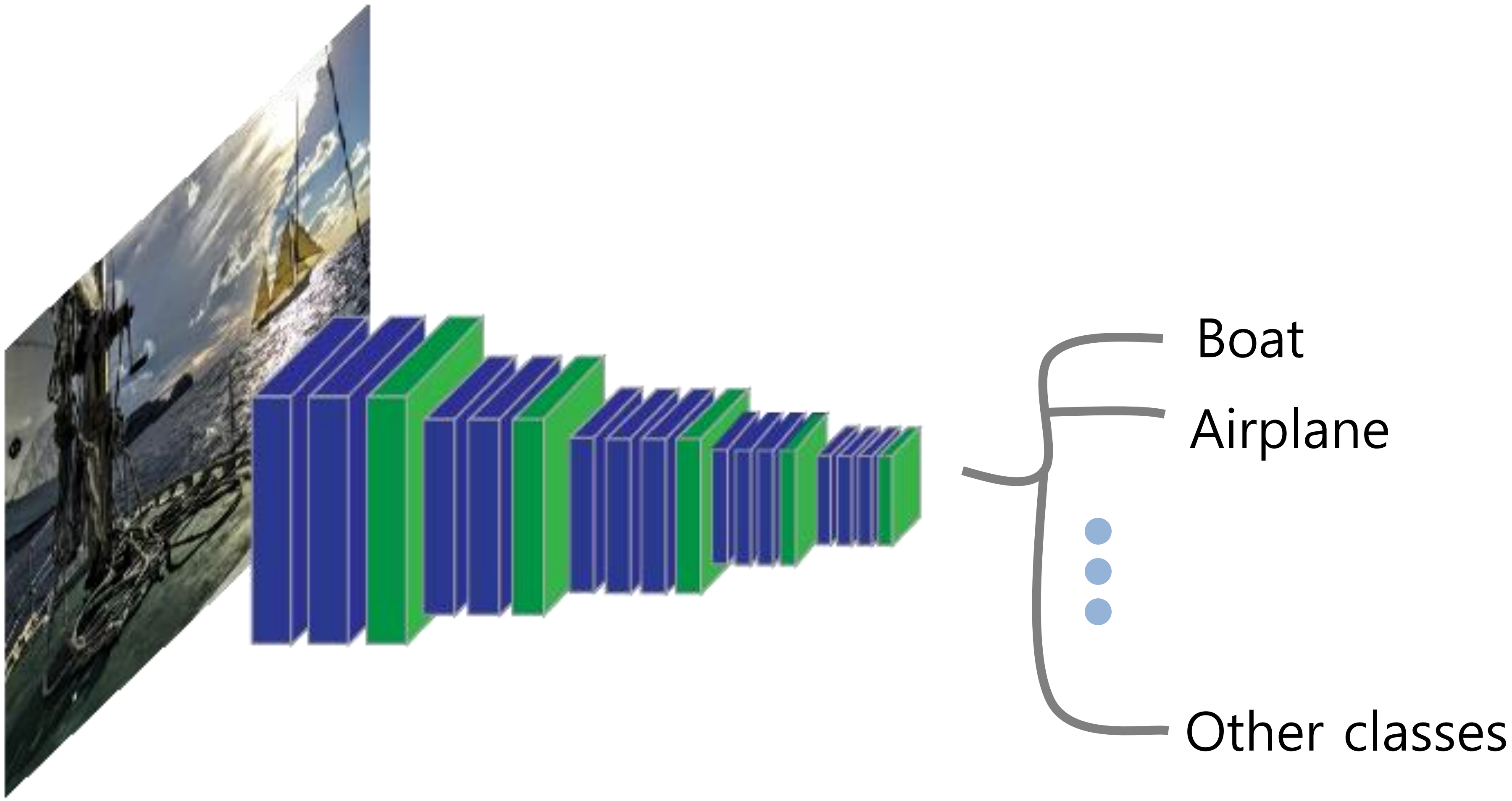


Semantic Segmentation

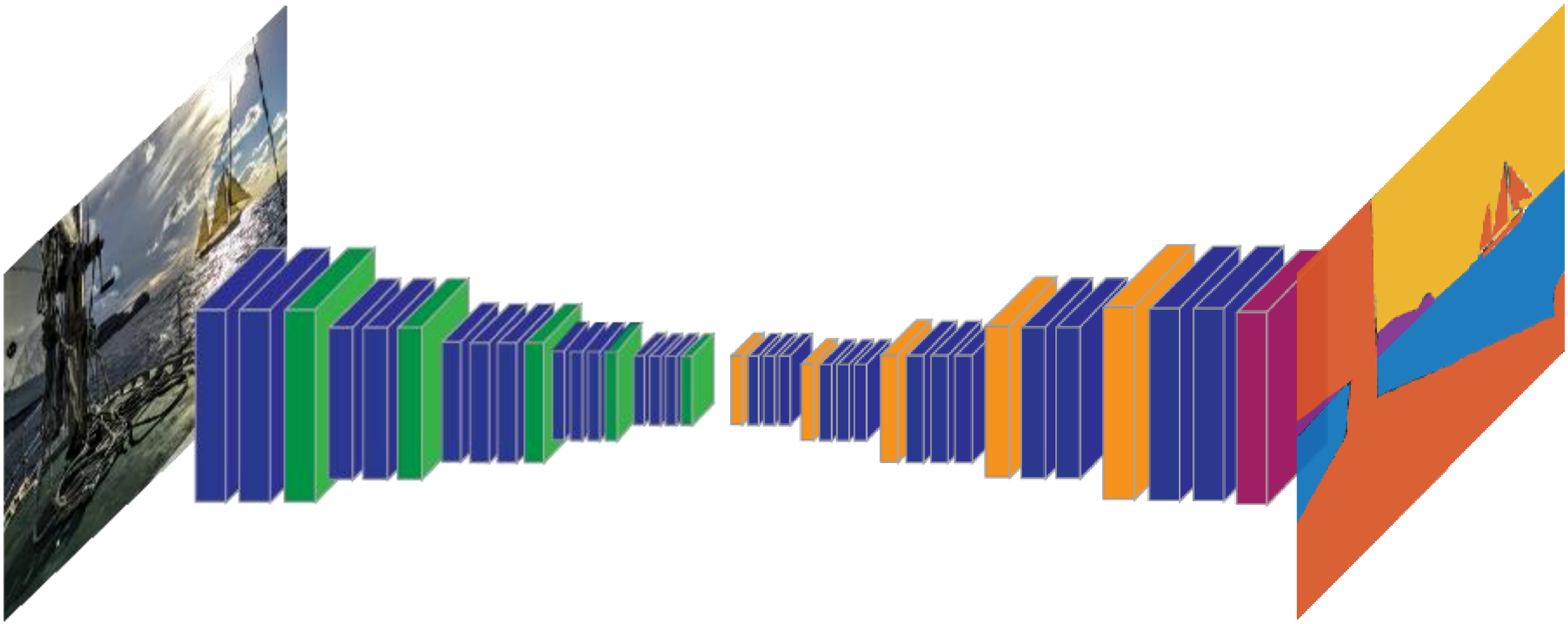
CamVid Dataset

1. Segmentation and Recognition Using Structure from Motion Point Clouds, ECCV 2008
2. Semantic Object Classes in Video: A High-Definition Ground Truth Database ,Pattern Recognition Letters

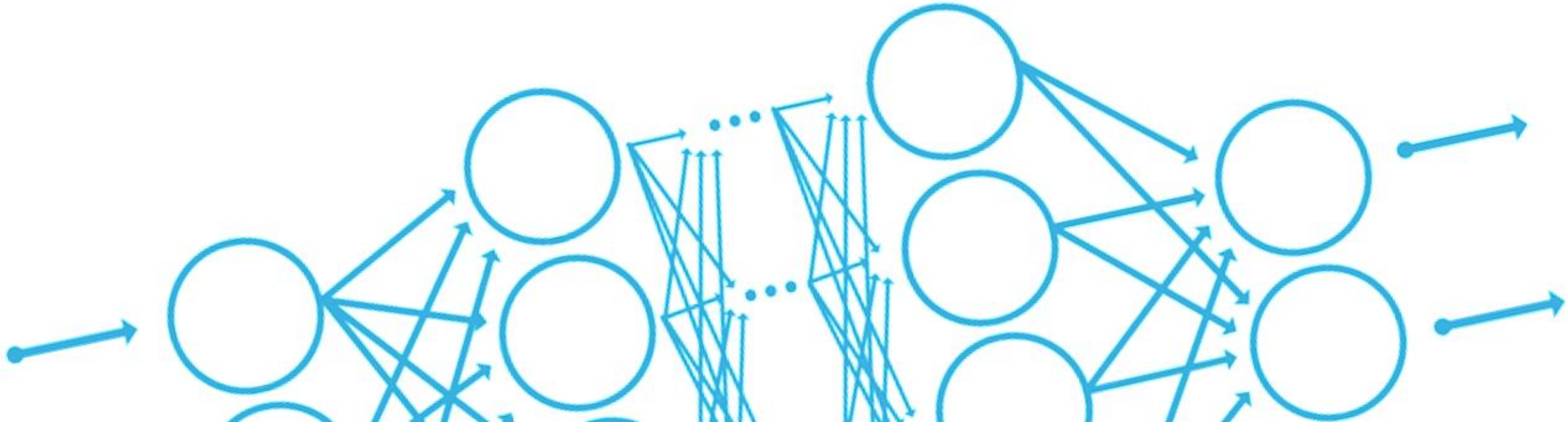
Semantic Segmentation Network



Semantic Segmentation Network



Example: Semantic Segmentation for Free Road Detection



1. Use CamVid dataset

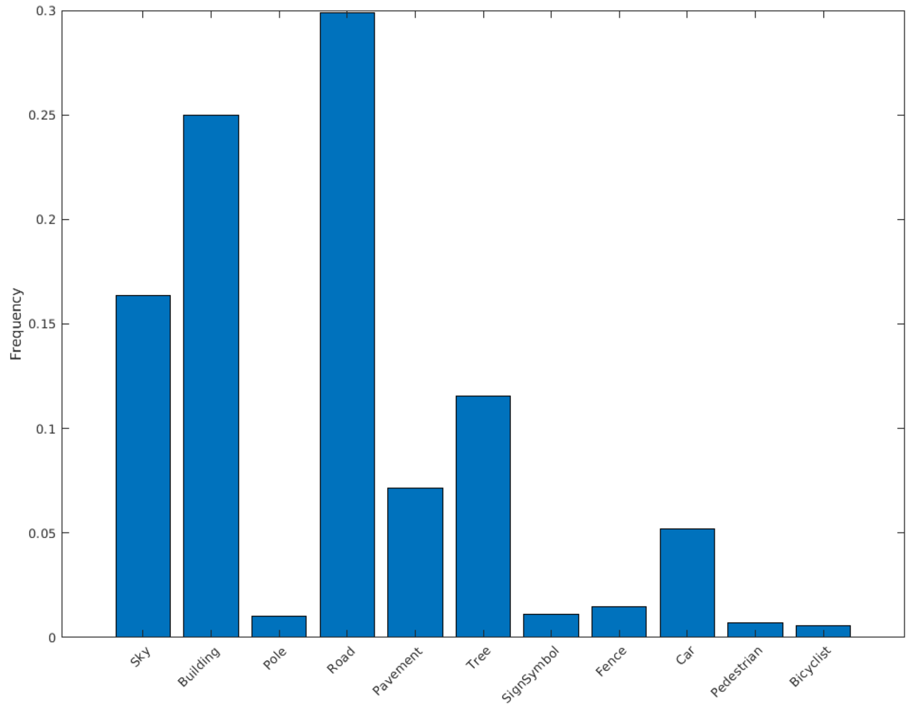
- Download the dataset (images and labels)
- Resize images to fit the input sizes of our neural network

2. Balance classes using class weighting

Ideally, all classes would have an equal number of observations.

However, the classes in CamVid are imbalanced.

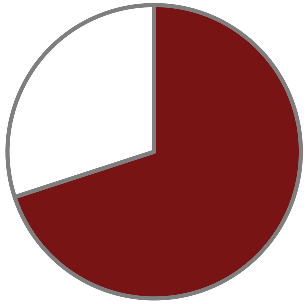
We use the pixel label counts and calculate the median frequency class weights.



3. Split dataset

Training

60%



- Trains the model
- Computer "learns" from this data

Validation

20%

- Checks accuracy of model during training

Test

20%

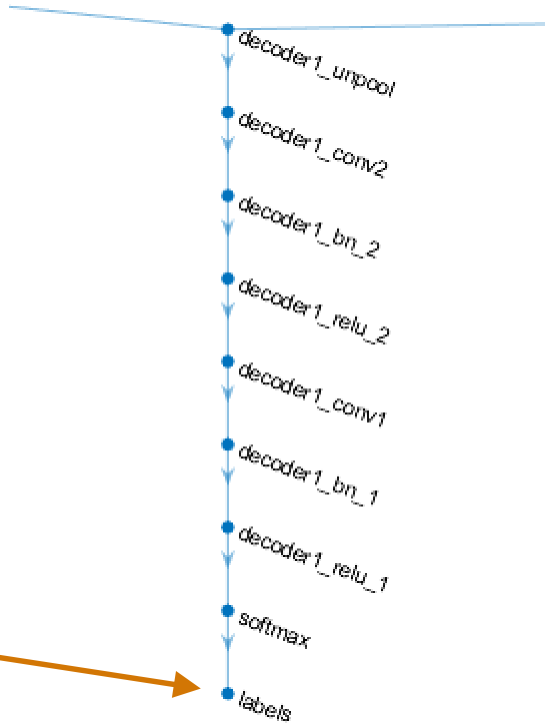
- Tests model accuracy
- Not used until validation accuracy is good

4. Define network

```
91x1 Layer array with layers:

     1   'inputImage'         Image Input                  360x480x3 images with 'zerocenter' normalization
     2   'conv1_1'            Convolution                  64 3x3x3 convolutions with stride [1  1] and padding [1  1  1  1]
     3   'bn_conv1_1'         Batch Normalization          Batch normalization
     4   'relu1_1'            ReLU                         ReLU
     5   'conv1_2'            Convolution                  64 3x3x64 convolutions with stride [1  1] and padding [1  1  1  1]
     6   'bn_conv1_2'         Batch Normalization          Batch normalization
     7   'relu1_2'            ReLU                         ReLU
     8   'pool1'              Max Pooling                  2x2 max pooling with stride [2  2] and padding [0  0  0  0]
     9   'conv2_1'            Convolution                  128 3x3x64 convolutions with stride [1  1] and padding [1  1  1  1]
    10   'bn_conv2_1'         Batch Normalization          Batch normalization
    11   'relu2_1'            ReLU                         ReLU
...
```

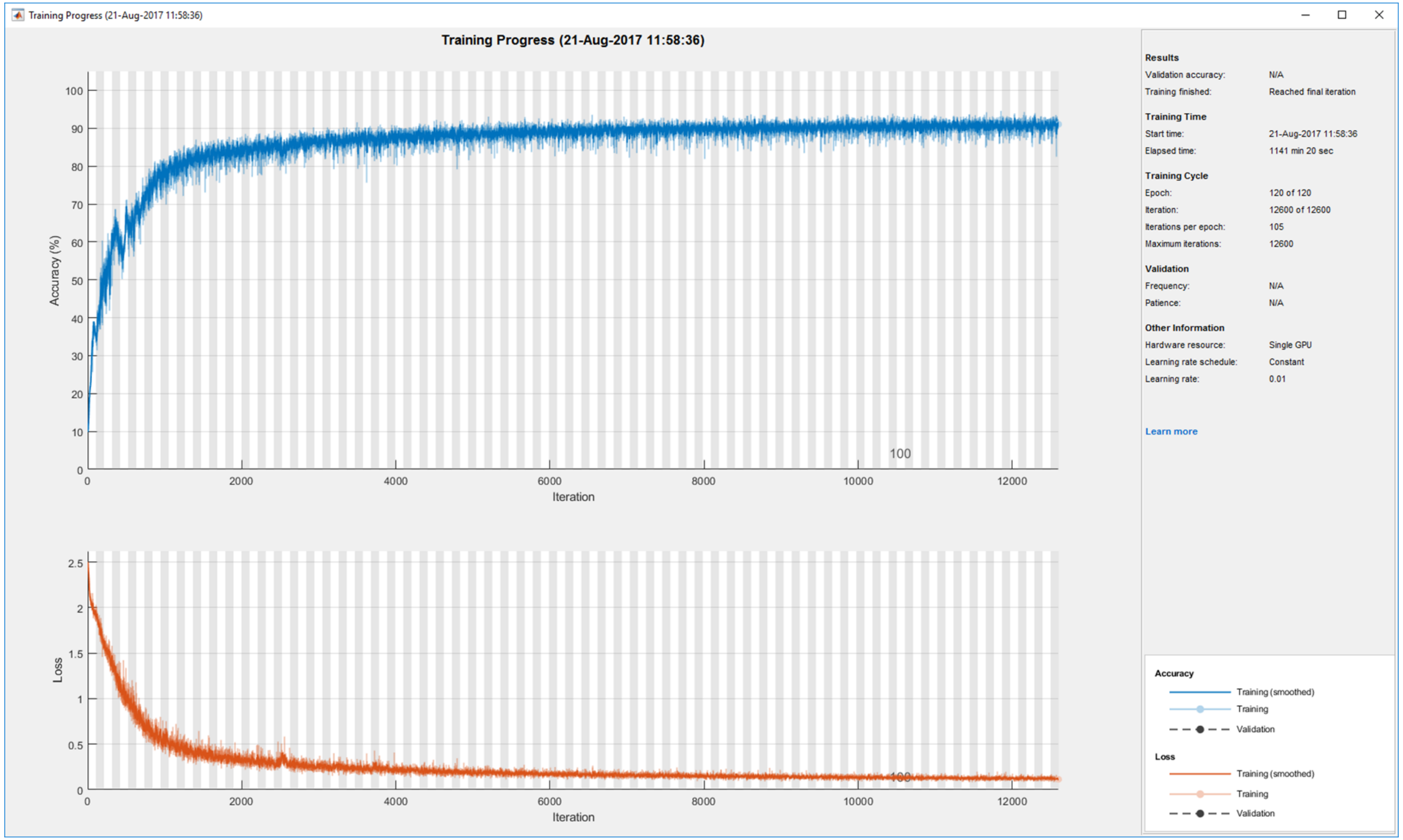
- Use segnetLayers to create a SegNet network initialized from a pre-trained VGG-16.

- Another option: Deeplab v3+ network with weights initialized from a pre-trained Resnet-18.



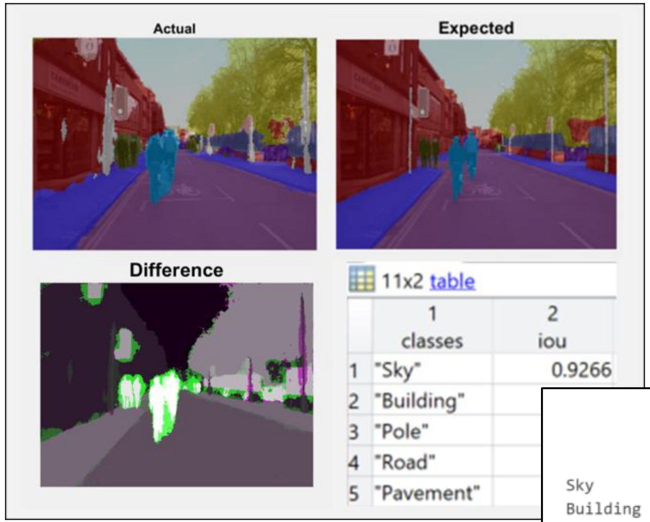
Last 9 layers: the last one provides also the **Class Weighting**

5. Train network

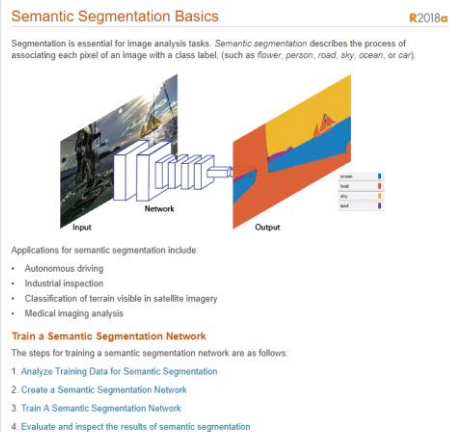


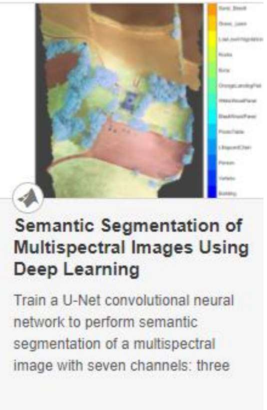
Useful tools for Semantic Segmentation

- **Automatically create network structures**
  - Using **`segnetLayers`** and **`fcnLayers`**
- **Handle pixel labels**
  - Using the **`pixelLabelImageDatastore`** and **`pixelLabelDatastore`**
- **Evaluate network performance**
  - Using `evaluateSemanticSegmentation`
- **Examples and tutorials to learn concepts**



	Accuracy	IoU	MeanBFScore
Sky	0.93544	0.89279	0.88239
Building	0.79978	0.75543	0.59861
Pole	0.73166	0.18361	0.51426
Road	0.93644	0.90663	0.7086
Pavement	0.90624	0.72932	0.70585
Tree	0.86587	0.73694	0.67097
SignSymbol	0.76118	0.35339	0.44175
Fence	0.83258	0.49648	0.50265
Car	0.90961	0.75263	0.64837
Pedestrian	0.83751	0.35409	0.46796
Bicyclist	0.84156	0.5472	0.46933



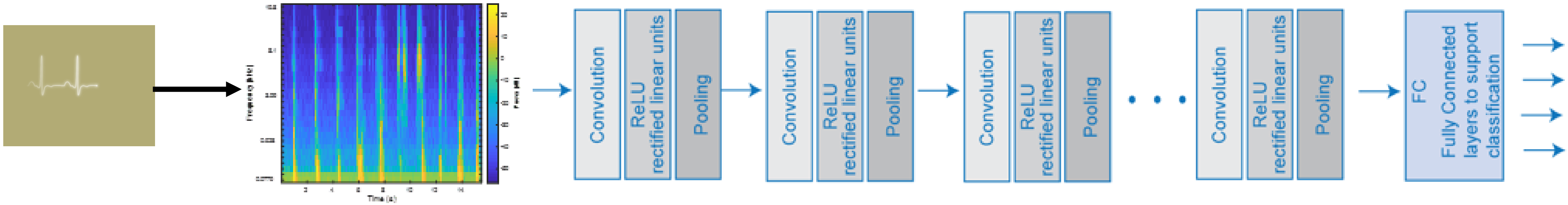


Agenda

Deep Learning Overview
Deep Learning in 6 Lines of MATLAB Code
Deep Learning with…
Images / Signals
Labeling Ground Truth Data
Working with Other Frameworks
Performance, Deployment, and Next Steps

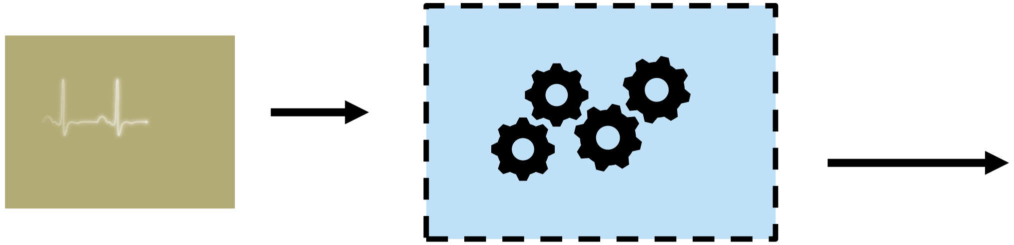
Common network architectures - signal processing

Convolutional Neural Networks (**CNN**)

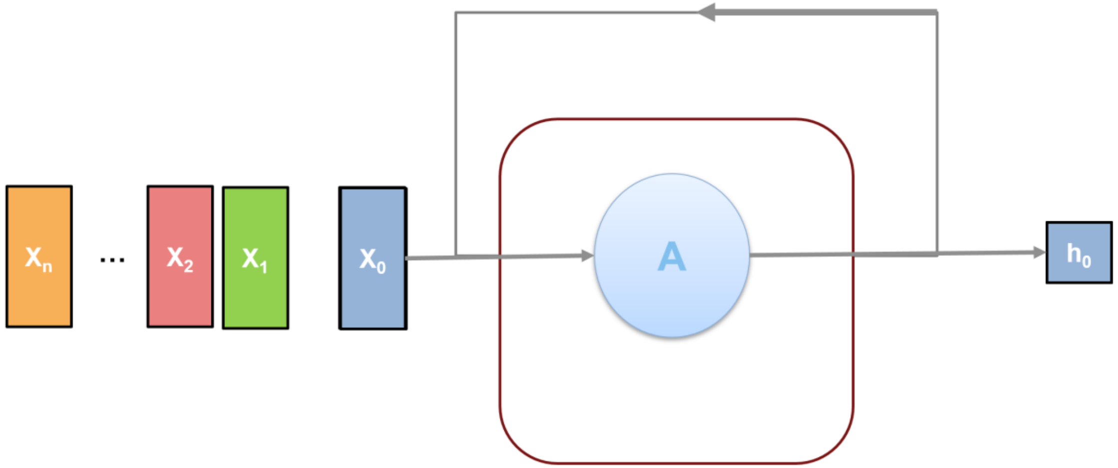


Time-Frequency Transformation

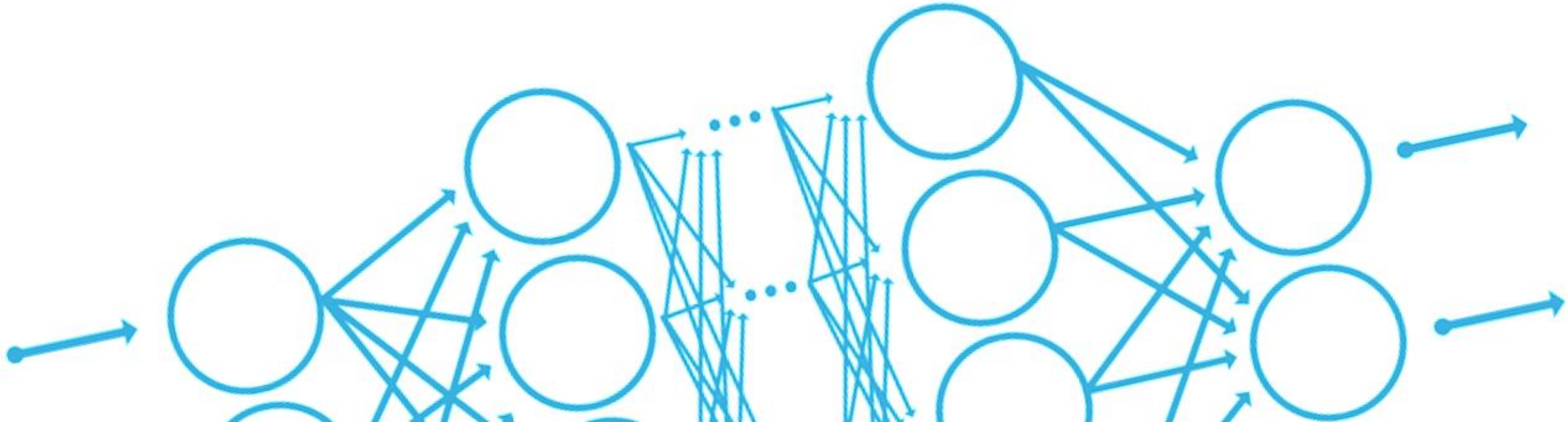
Long Short Term Memory (**LSTM**) Networks



Feature Engineering



Example: Speech Recognition



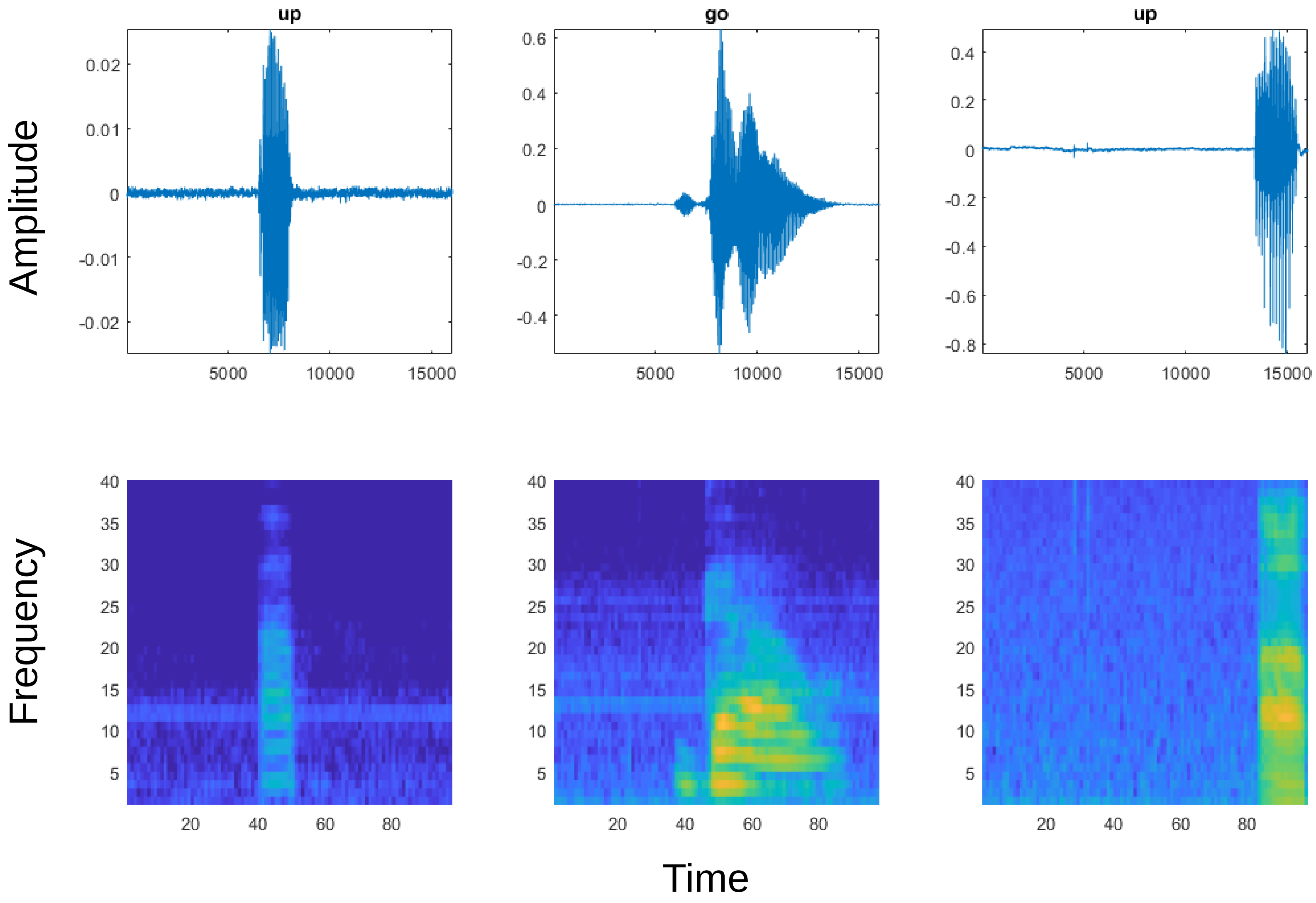
1. Create Datastore

- Datastore creates reference for data
- Do not have to load in all objects into memory

Name	Date modified
_background_noise_	2/12/2018 9:32 AM
Data	2/12/2018 9:39 AM
go	2/12/2018 9:34 AM
left	2/12/2018 9:35 AM
no	2/12/2018 9:36 AM
off	2/12/2018 9:37 AM
on	2/12/2018 9:38 AM
right	2/12/2018 9:31 AM
up	2/12/2018 9:31 AM
yes	2/12/2018 9:32 AM

```
datafolder = fullfile(tempdir,'speech_commands_v0.01');

addpath(fullfile(matlabroot,'toolbox','audio','audiodemos'))
ads = audioexample.Datastore(datafolder, ...
    'IncludeSubfolders',true, ...
    'FileExtensions','.wav', ...
    'LabelSource','foldernames', ...
    'ReadMethod','File')
```

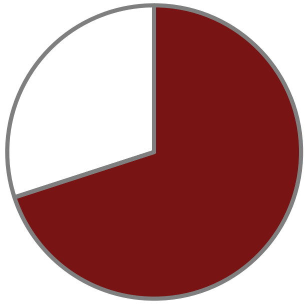
2. Compute speech spectrograms



3. Split datastores

Training

70%



- Trains the model
- Computer “learns” from this data

Validation

15%

- Checks accuracy of model during training

Test

15%

- Tests model accuracy
- Not used until validation accuracy is good

4. Define architecture and parameters

```
layers = [
    imageInputLayer(imageSize)

    convolution2dLayer(3,16,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,32,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2,'Padding',[0,1])

    dropoutLayer(dropoutProb)
    convolution2dLayer(3,64,'Padding','same')
    batchNormalizationLayer
    reluLayer

    dropoutLayer(dropoutProb)
```

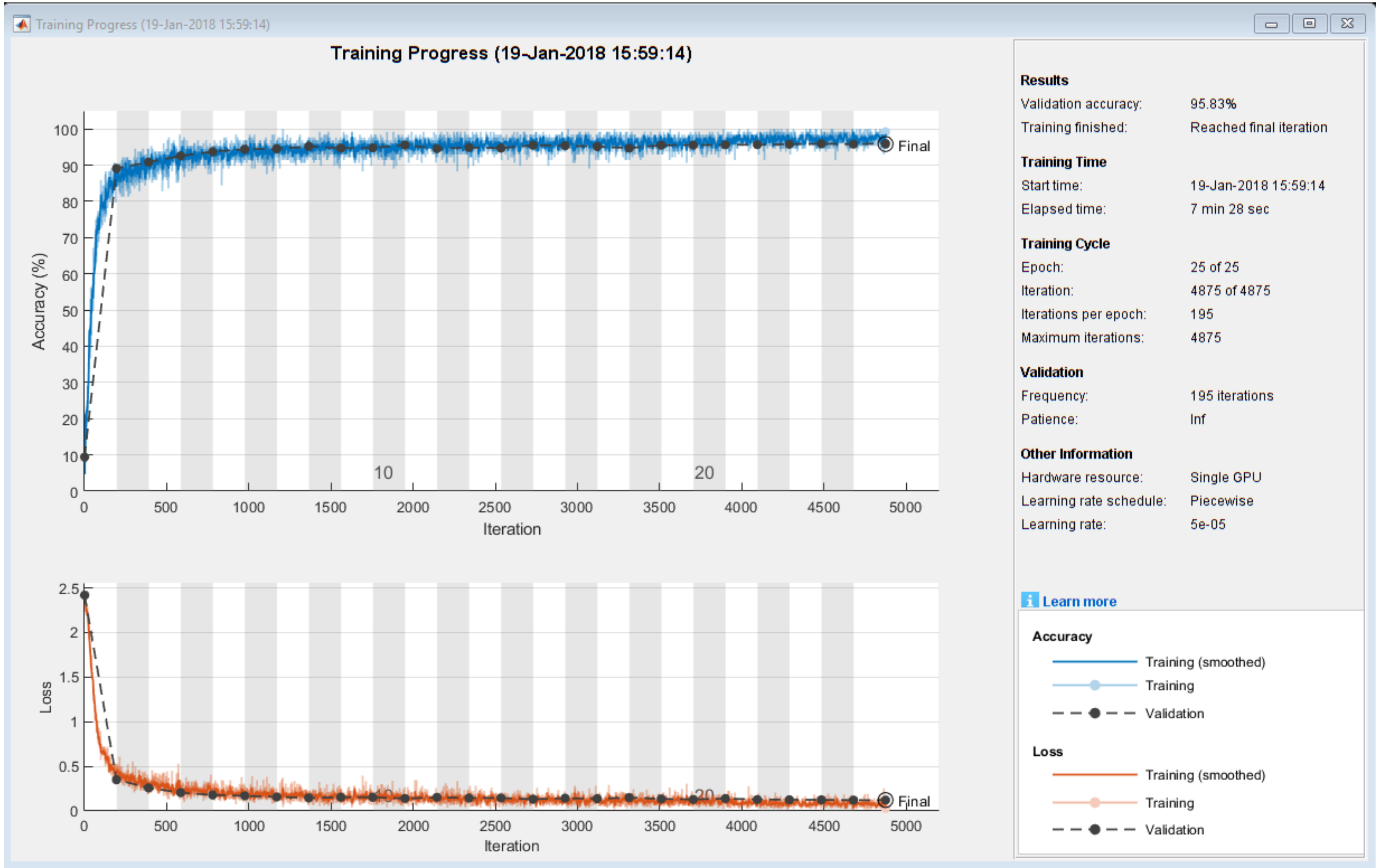
```
    convolution2dLayer(3,64,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2,'Padding',[0,1])

    dropoutLayer(dropoutProb)
    convolution2dLayer(3,64,'Padding','same')
    batchNormalizationLayer
    reluLayer

    dropoutLayer(dropoutProb)
    convolution2dLayer(3,64,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer([1 13])

    fullyConnectedLayer(numClasses)
    softmaxLayer
    weightedCrossEntropyLayer(classNames,classWeights)];
```

Neural Network Architecture

```
miniBatchSize = 128;
validationFrequency = floor(numel(YTrain)/miniBatchSize);
options = trainingOptions('adam', ...
    'InitialLearnRate',5e-4, ...
    'MaxEpochs',25, ...
    'MiniBatchSize',miniBatchSize, ...
    'Shuffle','every-epoch', ...
    'Plots','training-progress', ...
    'Verbose',false, ...
    'ValidationData',{XValidation,YValidation}, ...
    'ValidationFrequency',validationFrequency, ...
    'ValidationPatience',Inf, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropFactor',0.1, ...
    'LearnRateDropPeriod',20);
```

Model Parameters

5. Train network



Training is an iterative process

```
miniBatchSize = 128;
validationFrequency = floor(numel(YTrain)/miniBatchSize);
options = trainingOptions('adam', ...
    'InitialLearnRate',5e-4, ...
    'MaxEpochs',25, ...
    'MiniBatchSize',miniBatchSize, ...
    'Shuffle','every-epoch', ...
    'Plots','training-progress', ...
    'Verbose',false, ...
    'ValidationData',{XValidation,YValidation}, ...
    'ValidationFrequency',validationFrequency, ...
    'ValidationPatience',Inf, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropFactor',0.1, ...
    'LearnRateDropPeriod',20);
```

Parameters adjusted according to performance

Agenda

Deep Learning Overview	
Deep Learning in 6 Lines of MATLAB Code	
Deep Learning with…	
Images	Signals
Labeling Ground Truth Data	
Working with Other Frameworks	
Performance, Deployment, and Next Steps	

Labeling for deep learning is repetitive, tedious, and time-consuming…

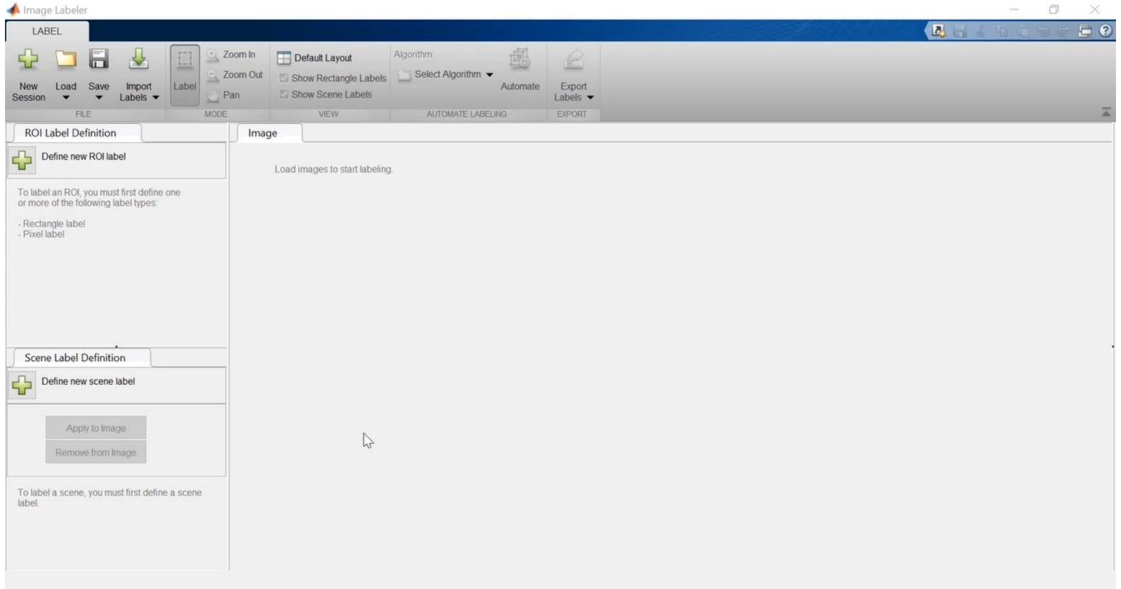
but necessary

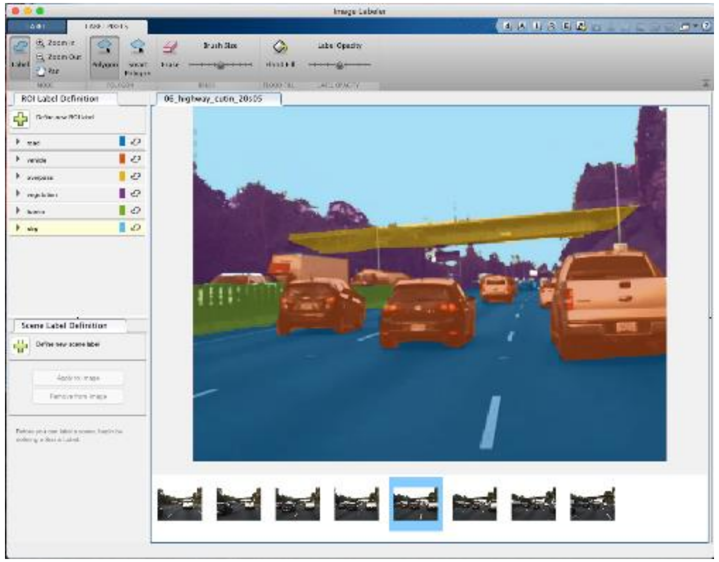
Ground truth labeling for images and video

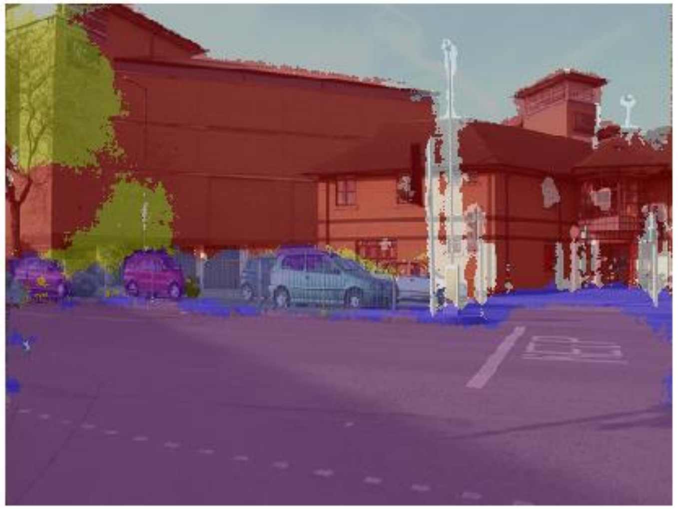
“How do I *label* my data?”

App for Ground Truth Labeling

Label regions and pixels for semantic segmentation





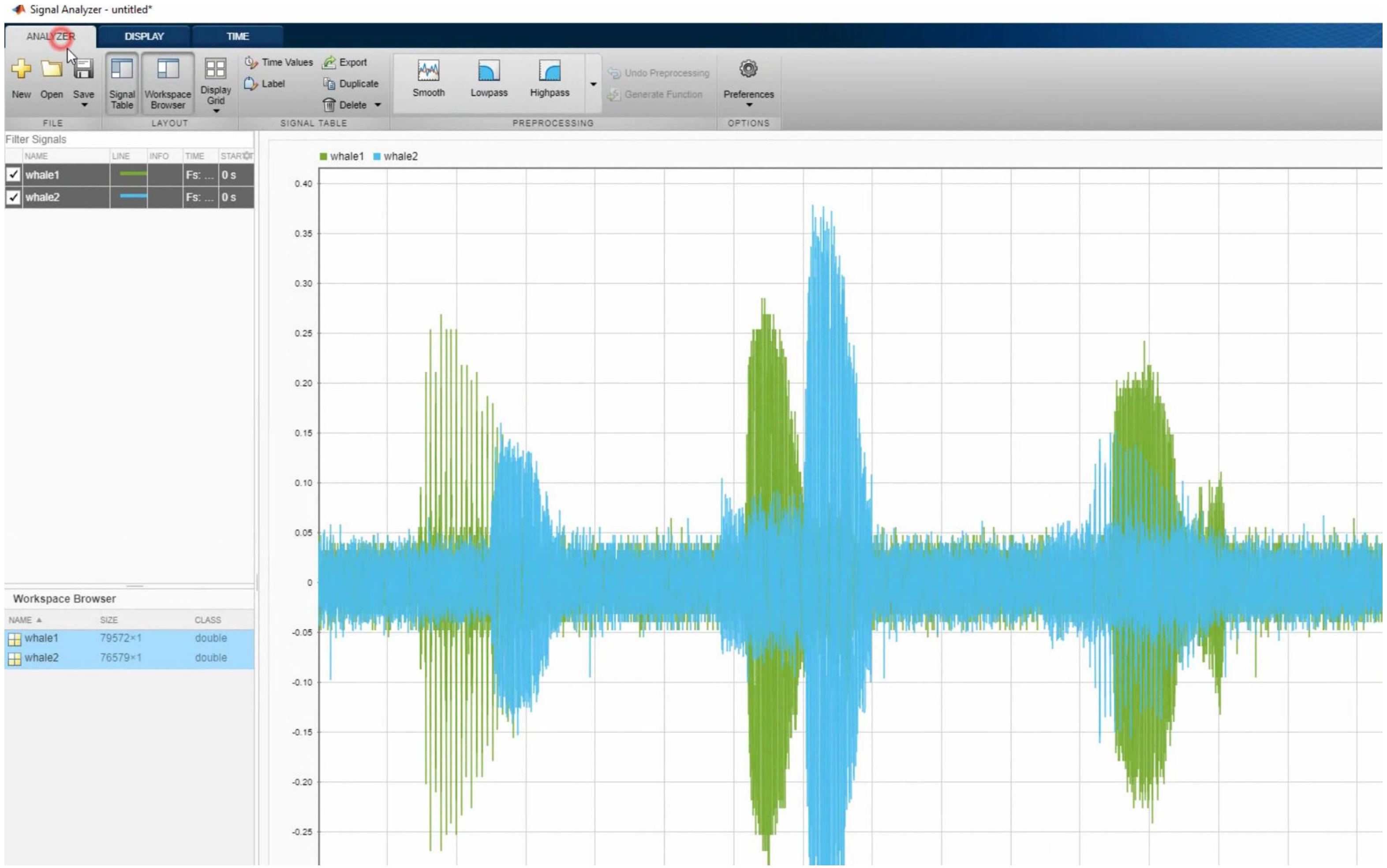


Ground truth labeling for signals and audio

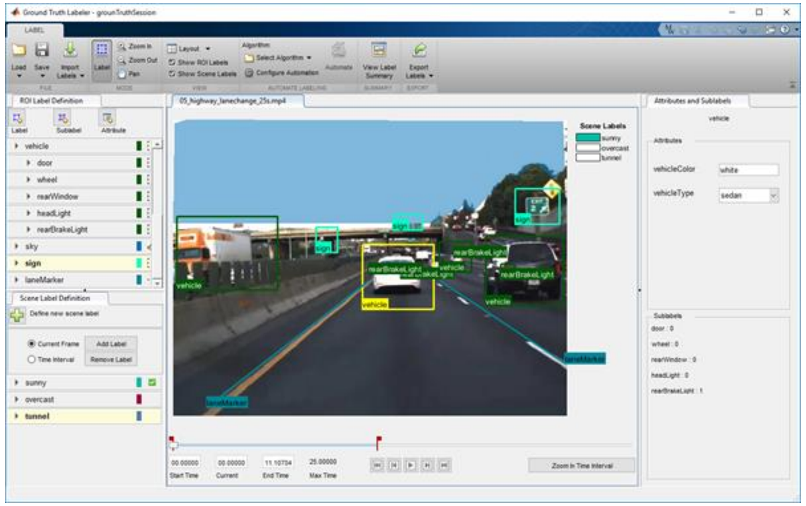
"How do I *label* my data?"

App for Signal Analysis and Labeling

Label signal waveforms, attributes, and samples for machine learning



MATLAB Apps help you speed up labeling work



For labeling **images** and **videos**:

1. **Image Labeler** (Computer Vision Toolbox)
   - https://www.mathworks.com/help/vision/ref/imagelabeler-app.html
2. **Video Labeler** (Computer Vision Toolbox)
   - https://www.mathworks.com/help/vision/ref/videolabeler-app.html
3. **Ground Truth Labeler** (Automated Driving Toolbox)
   - https://www.mathworks.com/help/driving/ref/groundtruthlabeler-app.html

How to choose the correct app for your project?

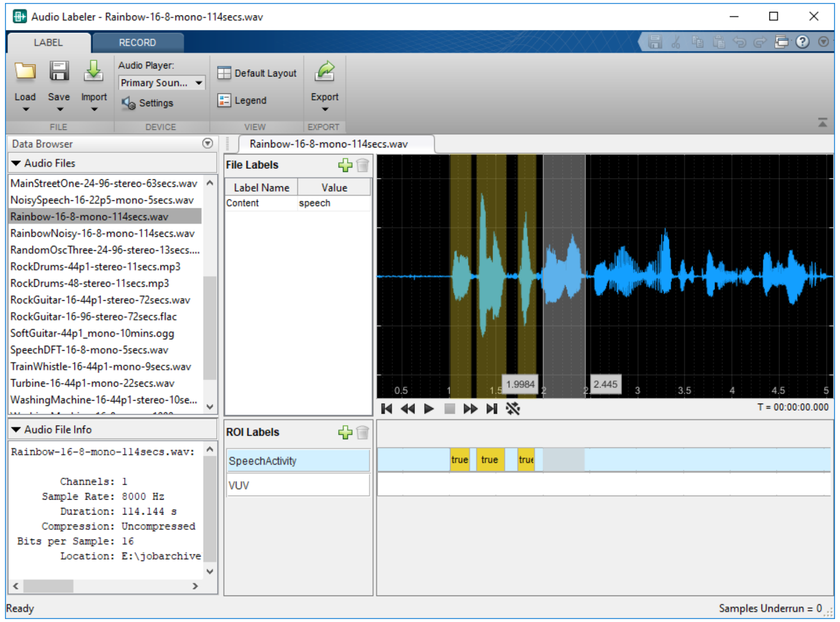
– https://www.mathworks.com/help/vision/ug/choose-a-labeling-app.html

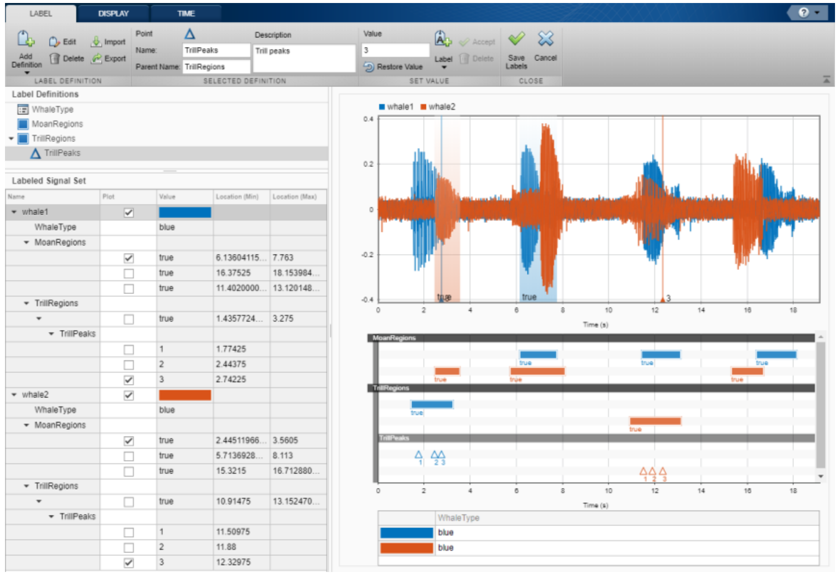
MATLAB Apps help you speed up labeling work

For labeling **signals** and **audio**:

1. **Audio Labeler**
   (Audio Toolbox)
   https://www.mathworks.com/help/audio/ug/audio-labeler-walkthrough.html

2. **Signal Labeler**
   (Signal Processing Toolbox)
   https://www.mathworks.com/help/signal/ref/signallabeler.html





Agenda

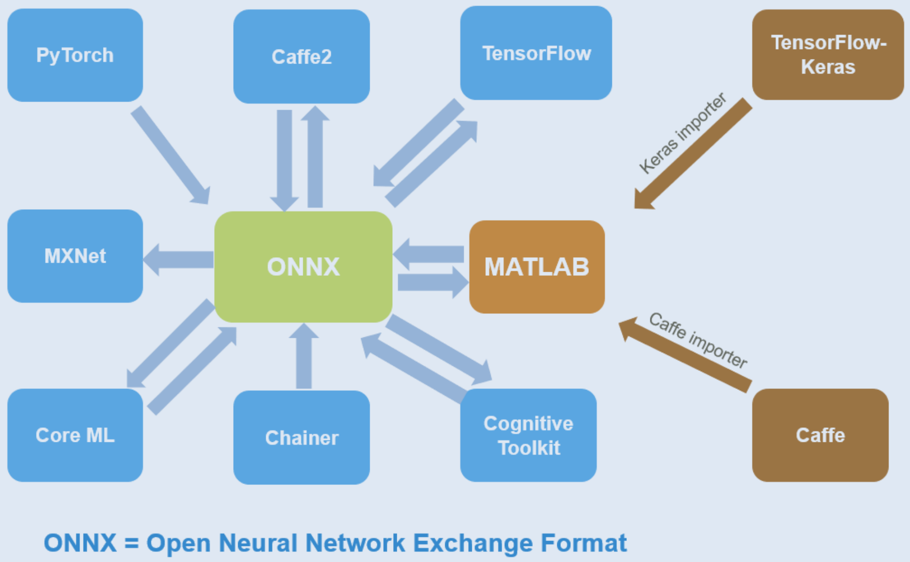
Deep Learning Overview
Deep Learning in 6 Lines of MATLAB Code

Deep Learning with…

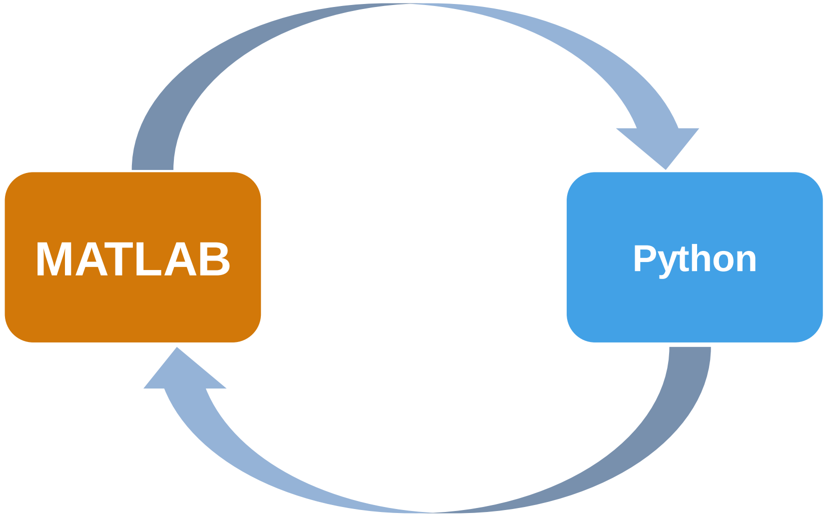
Images	Signals

Labeling Ground Truth Data
Working with Other Frameworks
Performance, Deployment, and Next Steps

Two ways to work with TensorFlow and PyTorch



Model Exchange

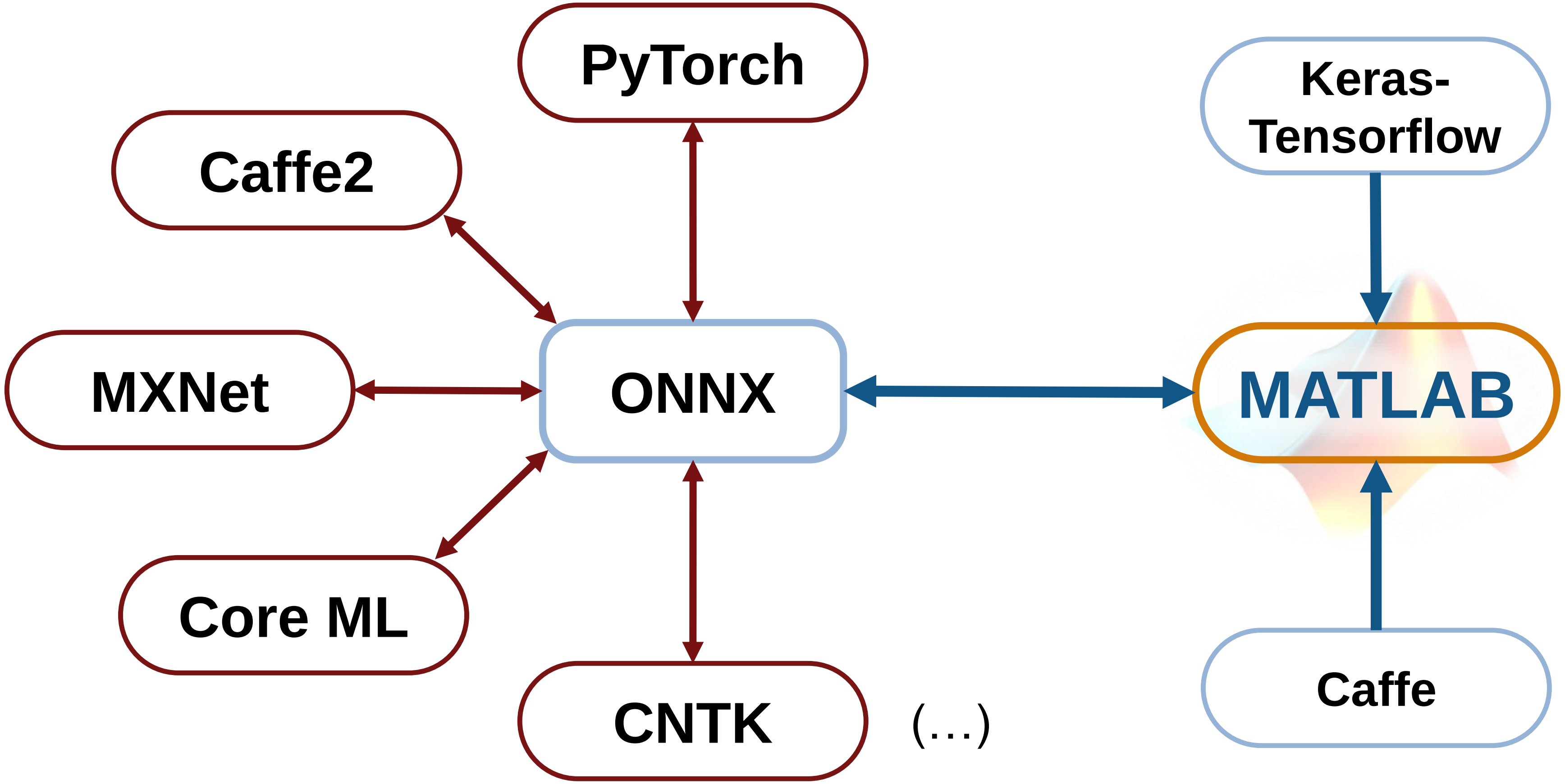


Co-execution (Python and C++)

Three use cases for model exchange

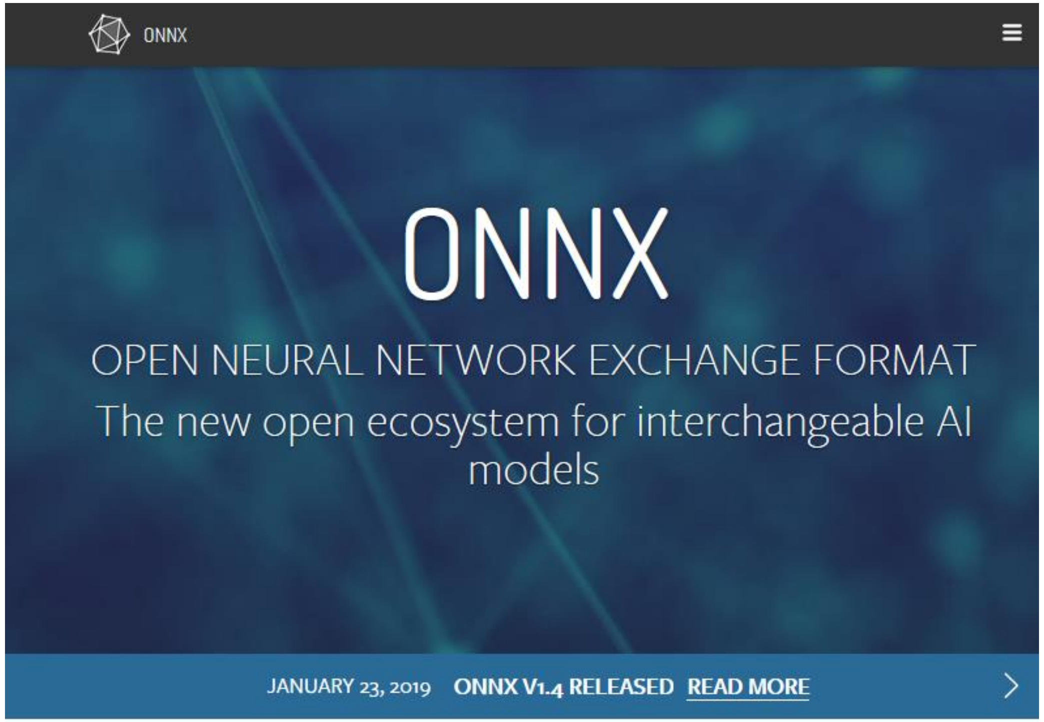
1. Enable teams to use different frameworks and tools
2. Access to code generation and other downstream tools in MATLAB
3. MATLAB users gain access to models in research/DL ecosystem

Model exchange with MATLAB



Open Neural Network Exchange

ONNX – Industry Standard for Model Exchange



aws Facebook Open Source Microsoft

What is ONNX?

ONNX is a open format to represent deep learning models. With ONNX, AI developers can more easily move models between state-of-the-art tools and choose the combination that is best for them. ONNX is developed and supported by a community of partners.

Frameworks

Caffe2 Chainer Cognitive Toolkit mxnet
PyTorch PaddlePaddle MATLAB sas

Converters

K dmlc XGBoost LibSVM

Runtimes

NVIDIA Qualcomm BITMAIN
Tencent vespa Windows

Source: https://onnx.ai/

Transfer Learning with Pretrained Models

- Inception-v3
- ResNet-101
- VGG-16
- Inception-ResNet-v2
- ResNet-18
- GoogLeNet
- DenseNet-201
- VGG-19
- SqueezeNet
- AlexNet
- ResNet-50

Import & Export Models Between Frameworks

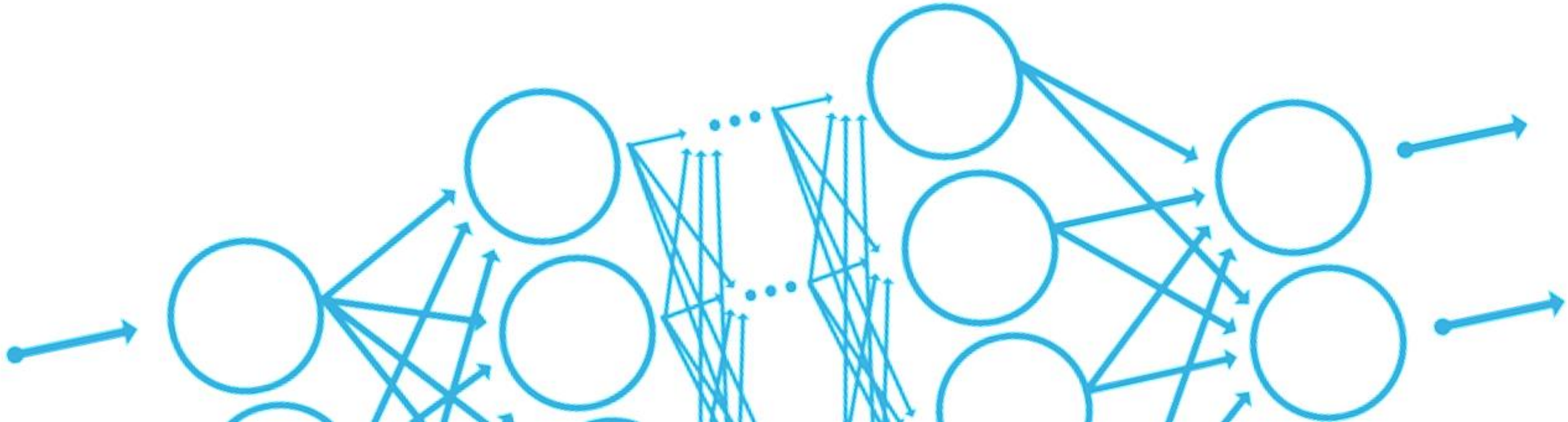
- Keras-Tensorflow Importer
- Caffe Model Importer
- ONNX Model Converter

More comprehensive list here: https://www.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html

Agenda

Deep Learning Overview
Deep Learning in 6 Lines of MATLAB Code
Deep Learning with…
Images / Signals
Labeling Ground Truth Data
Working with Other Frameworks
Performance, Deployment, and Next Steps

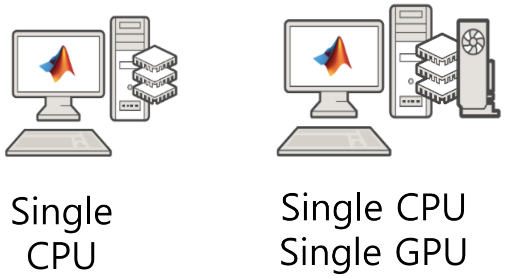
Training Performance and Scalability



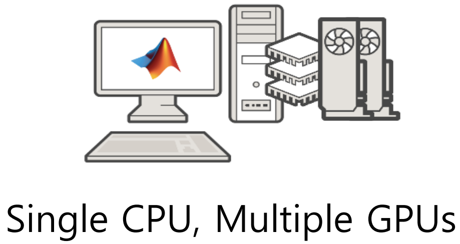
Deep Learning on CPU, GPU, Multi-GPU and Clusters

How to target?

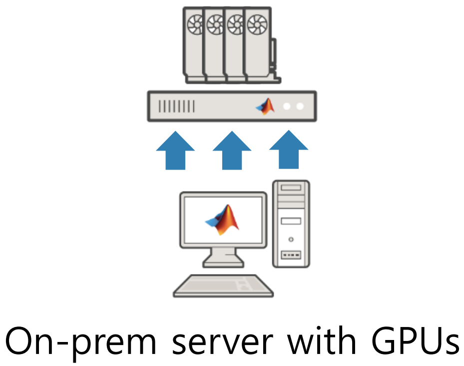
```
opts = trainingOptions('sgdm', ...
      'MaxEpochs', 100, ...
      'MiniBatchSize', 250, ...
      'InitialLearnRate', 0.00005, ...
'ExecutionEnvironment', 'auto' );
```

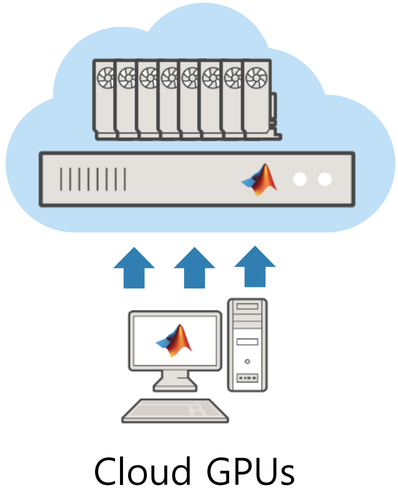



```
opts = trainingOptions('sgdm', ...
      'MaxEpochs', 100, ...
      'MiniBatchSize', 250, ...
      'InitialLearnRate', 0.00005, ...
'ExecutionEnvironment', 'multi-gpu' );
```

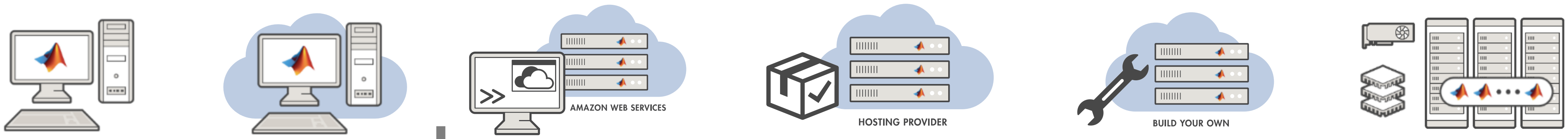



```
opts = trainingOptions('sgdm', ...
      'MaxEpochs', 100, ...
      'MiniBatchSize', 250, ...
      'InitialLearnRate', 0.00005, ...
'ExecutionEnvironment', 'parallel' );
```





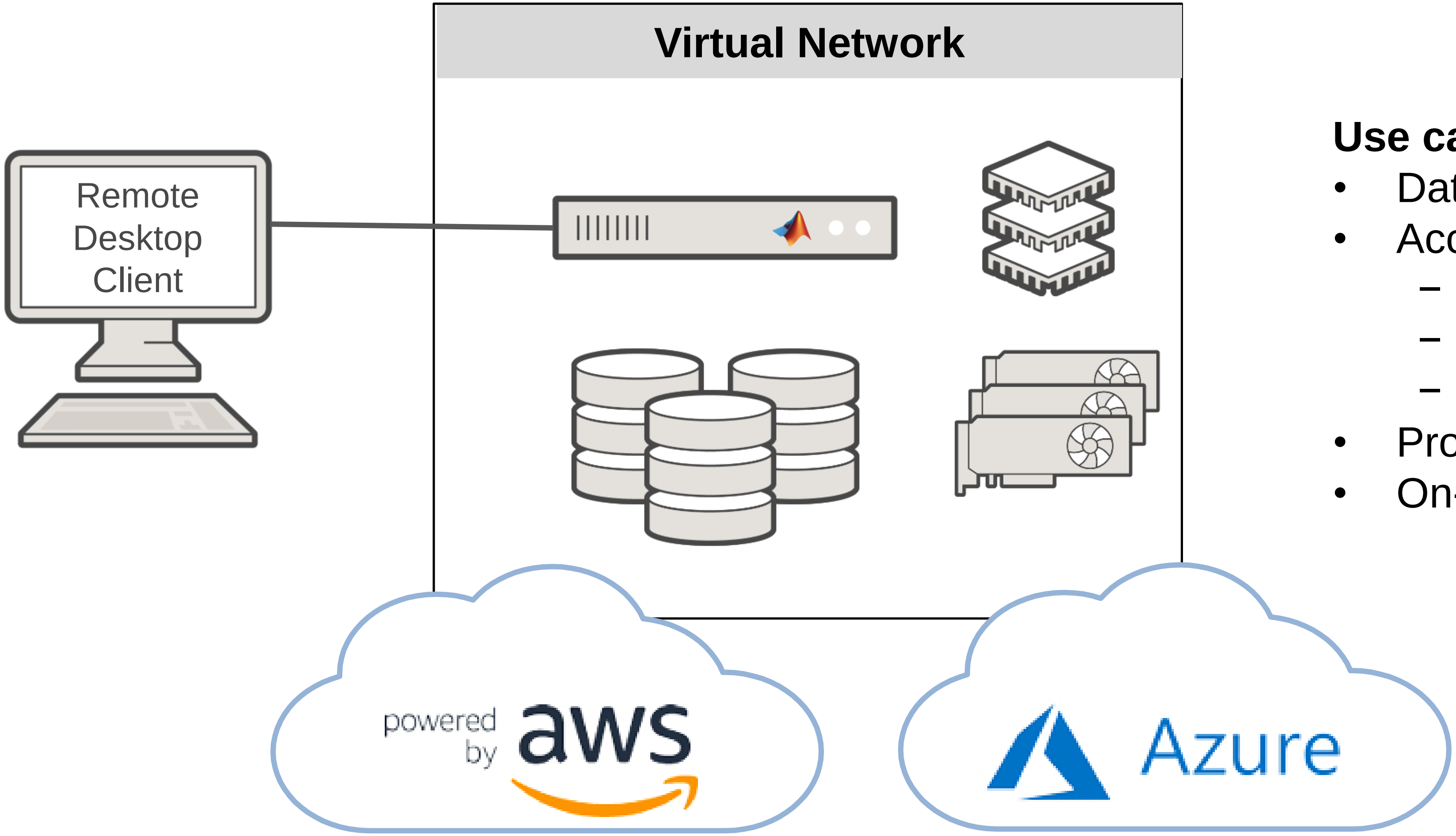
MATLAB parallel capabilities for cloud are evolving



Option	**Parallel Computing Toolbox**		**MATLAB Parallel Server (formally called MDCS)**			
	Desktop	*Custom Cloud* ***Reference Architecture***	*User-managed Cloud* ***Cloud Center***	*Pre-configured Cloud* ***Hosting Providers***	*Custom Cloud* ***Reference Architecture***	*On Premise*
Description	Explicit desktop scaling	Explicit desktop scaling	Preconfigured clusters in Amazon Web Services	Cloud solutions from MathWorks partners	Custom infrastructure for AWS, Azure, and others	Scale to clusters in your organization
When to use	Prototyping, small scaling	Data in cloud, access high-end machines, especially with GPUs	User-centric workflows, small studies	Managed environments with operating expense rather than capital expense	Internally-managed environments with operating expense rather than capital expense	Traditional scaling on assets you manage at your facilities
Maximum workers	Physical cores in machine	Physical cores in machine	1024 per cluster	Defined by Hosting Provider	No limit	Physical cores in cluster
License Options	• Network • Online	• Online • Network (requires support)	• Online only	• Network • Online	• Online • Network (requires support)	• Network • Online

http://www.mathworks.com/programs/mdcs-cloud.html

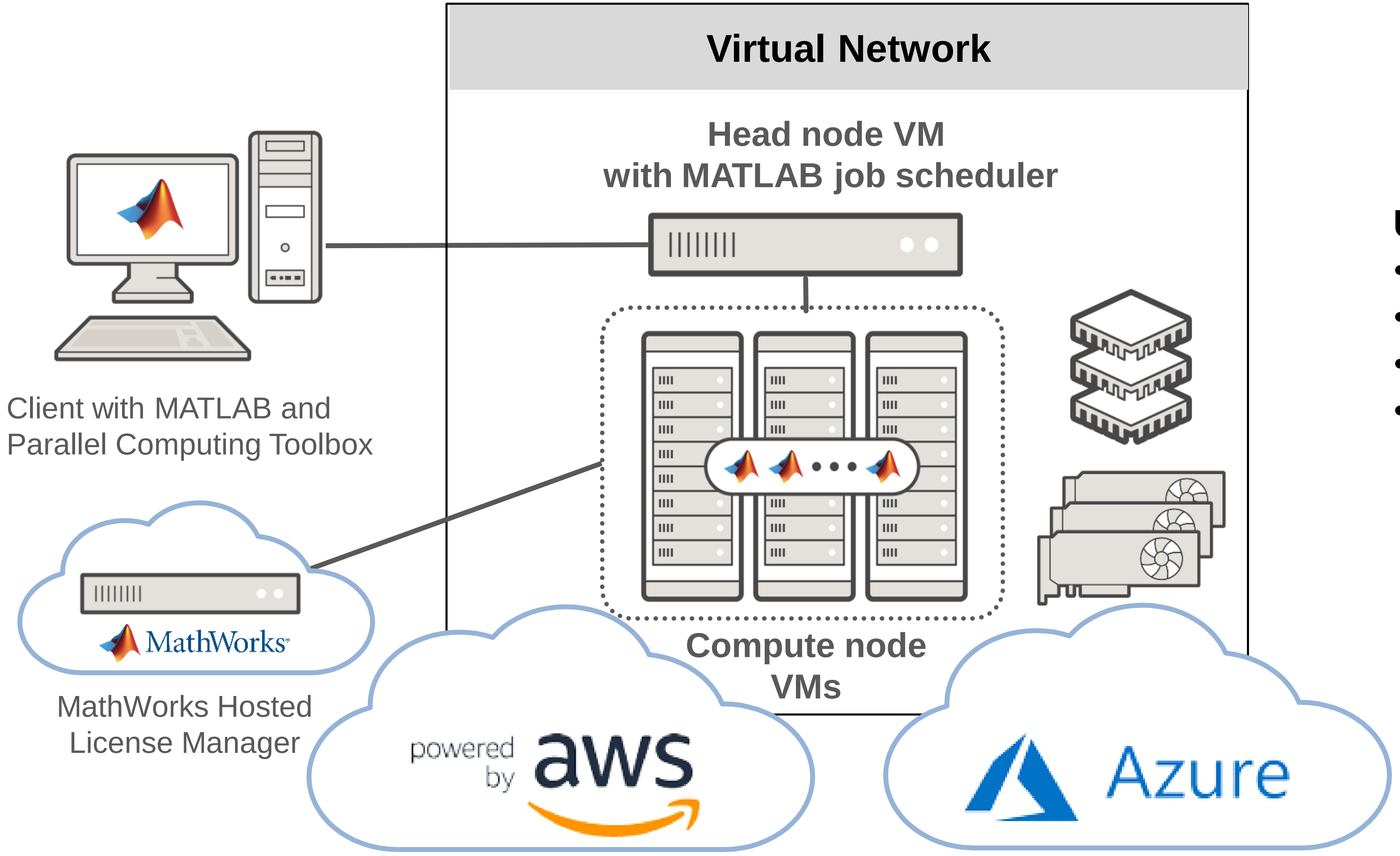
Cloud Reference Architecture: MATLAB & Simulink



Use cases:

- Data analytics on cloud-stored data
- Access to high-end hardware:
  - multi-core
  - GPUs
  - FPGAs
- Prototyping parallel algorithms
- On-server pre-production testing

Reference Architecture: MATLAB Distributed Computing Server



Use cases:
- Parameter sweeps
- Monte Carlo runs
- Optimization
- Distributed array calculations

NVIDIA NGC & DGX Supports MATLAB for Deep Learning

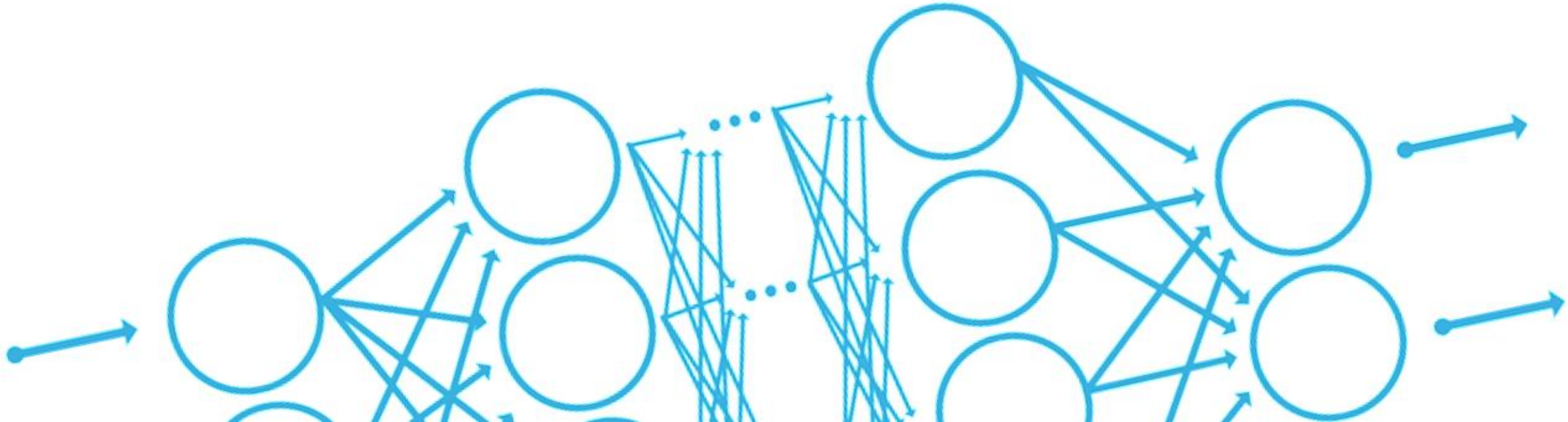
- GPU-accelerated MATLAB Docker container for deep learning
  - Leverage multiple GPUs on NVIDIA DGX Systems and in the Cloud
    - Cloud providers include: AWS, Azure, Google, Oracle, and Alibaba

- NVIDIA DGX System / Station
  - Interconnects 4/8/16 Volta GPUs in one box

- Containers available for R2018a through R2019a
  - New Docker container with every major release (a/b)

- Download MATLAB container from NGC Registry
  - https://ngc.nvidia.com/registry/partners-matlab

Inference Performance and Deployment



Deployment process

PREPARE DATA

- Data access and preprocessing
- Ground truth labeling
- Simulation-based data generation

TRAIN MODEL

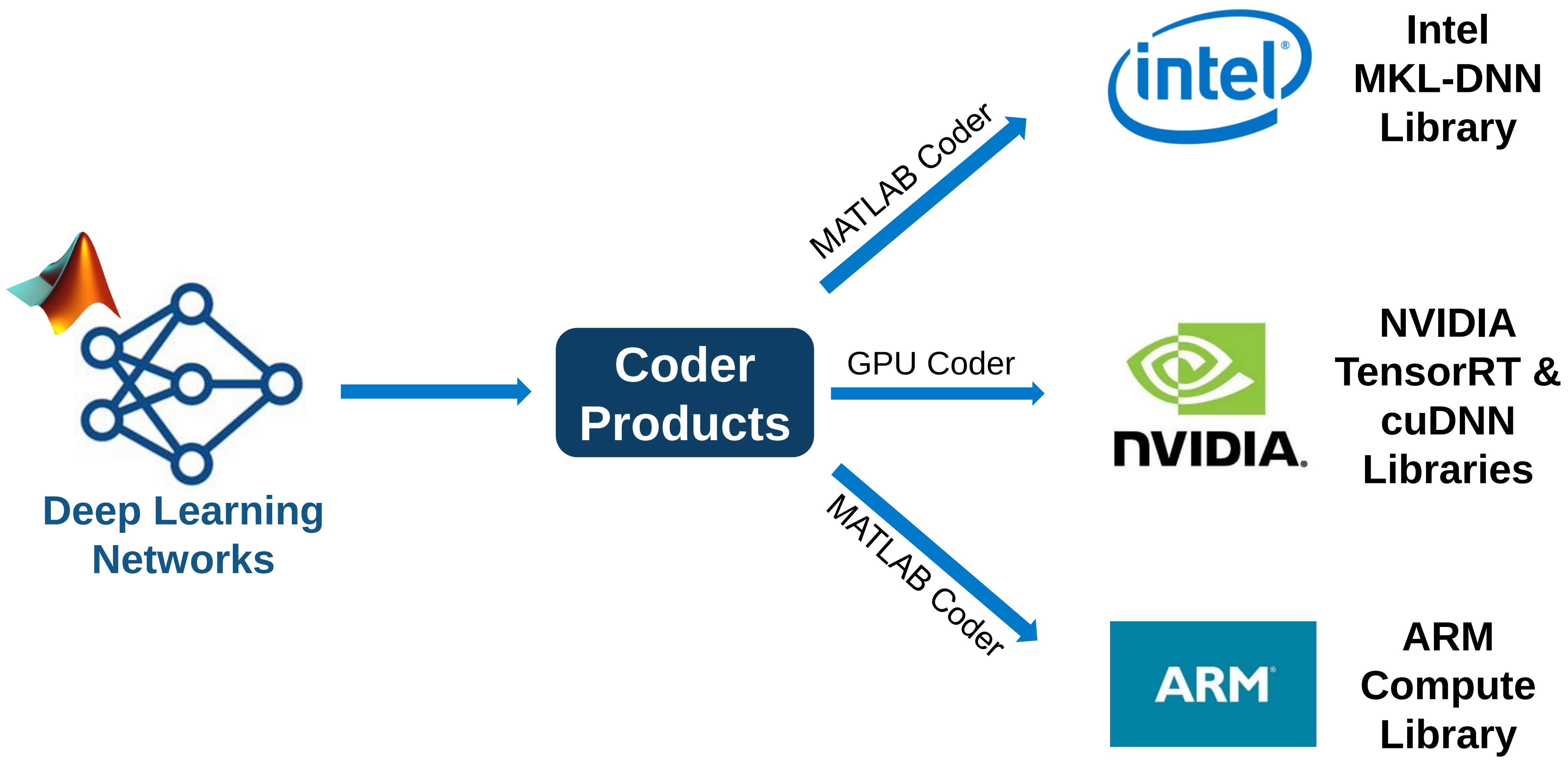
- Model design, Hyperparameter tuning
- Model exchange across frameworks
- Hardware-accelerated training

DEPLOY SYSTEM

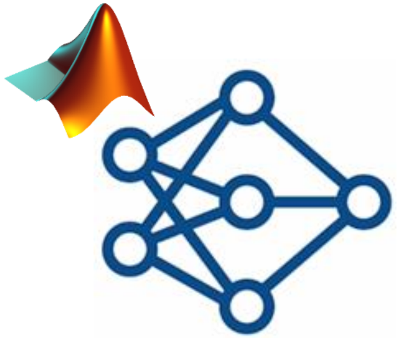
- Multiplatform code generation (CPU, GPU)
- Edge deployment
- Enterprise Deployment

Iteration and Refinement

Deploying Deep Learning models for inference



Deploying to HW targets



Deep Learning Networks

Coder Products

MATLAB Coder

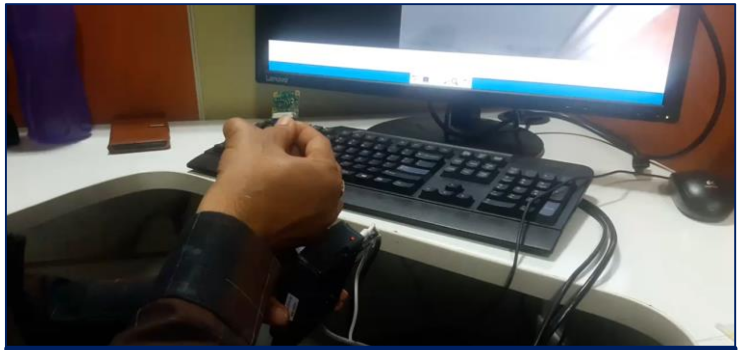
GPU Coder

MATLAB Coder

Desktop CPU

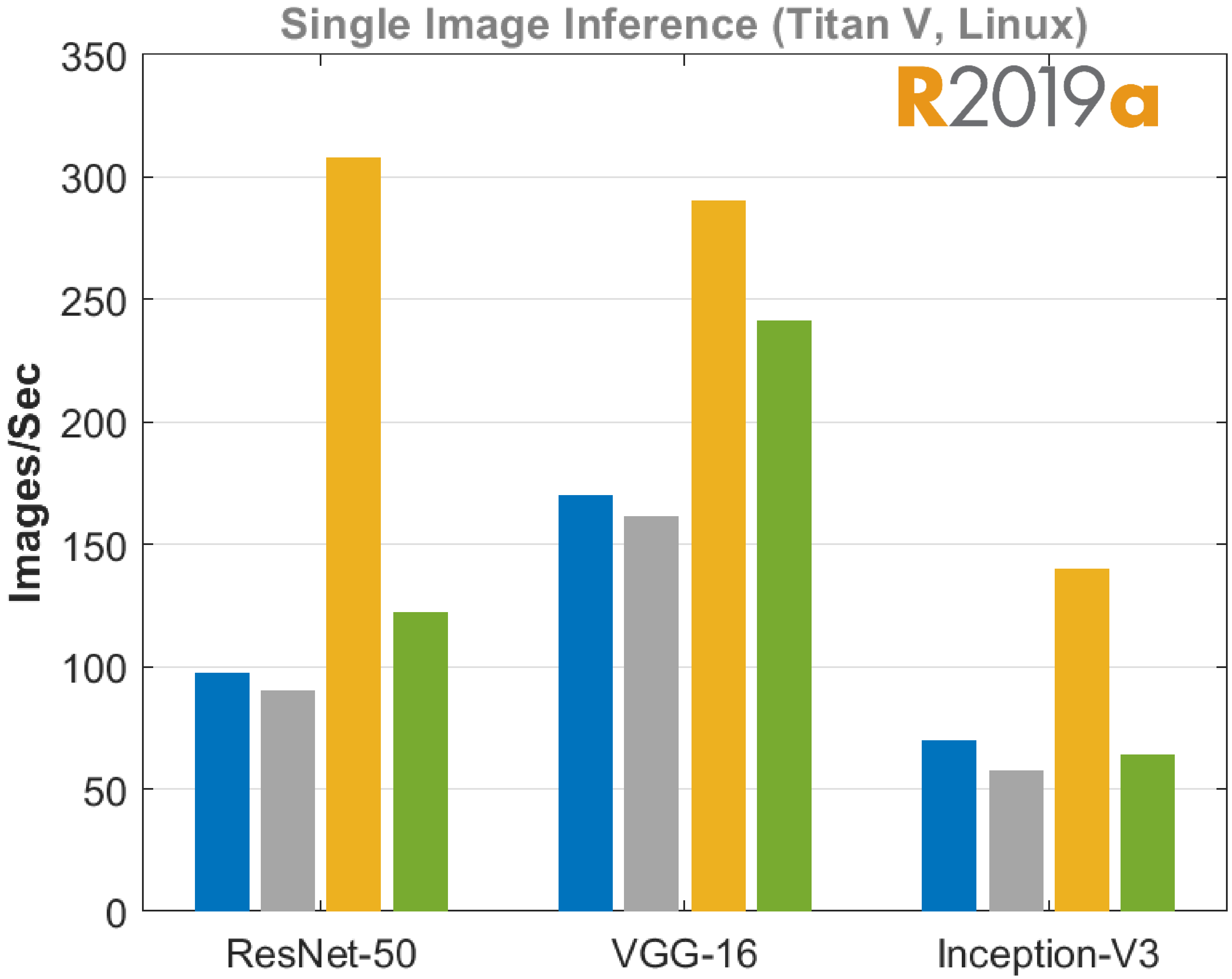
NVIDIA

NVIDIA TensorRT & cuDNN Libraries



Raspberry Pi board

With GPU Coder, MATLAB is fast



GPU Coder is faster than TensorFlow, MXNet and PyTorch

- TensorFlow
- MXNet
- GPU Coder
- PyTorch

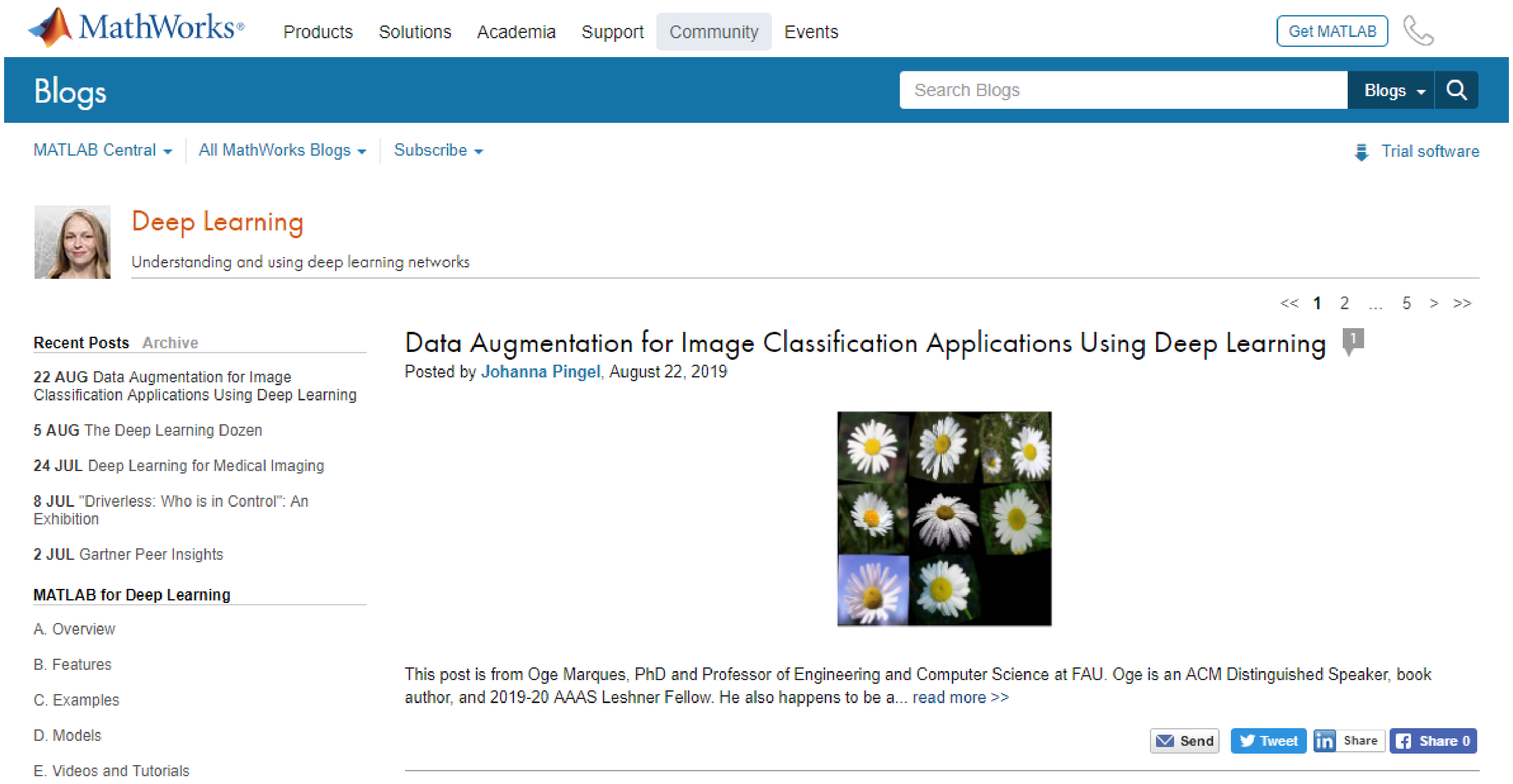
Intel® Xeon® CPU 3.6 GHz - NVIDIA libraries: CUDA10 - cuDNN 7 - Frameworks: TensorFlow 1.13.0, MXNet 1.4.0 PyTorch 1.0.0

Semantic Segmentation speedup

Running in MATLAB

Generated Code from GPU Coder

Blog: Deep Learning



http://blogs.mathworks.com/deep-learning/

Introduction to Deep Learning

MathWorks® Products Solutions Academia Support Community Events

Video and Webinar Series

Search Videos | Videos

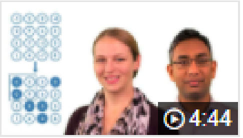
Videos Home | Search

Contact sales | Trial software

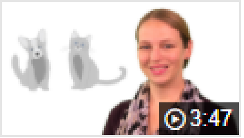
Introduction to Deep Learning

Watch this series of MATLAB® Tech Talks to explore key deep learning concepts. Learn to identify when to use deep learning, discover what approaches are suitable for your application, and explore some of the challenges you might encounter.

What Is Deep Learning?
Explore deep learning fundamentals in this MATLAB Tech Talk. You'll learn why deep learning has become so popular, and you'll walk through 3 concepts: what deep learning is, how it is used in the real world, and how you can get started.



What Are Convolutional Neural Networks?
Explore the basics of convolutional neural networks (also called CNNs or ConvNets) in this MATLAB Tech Talk. You'll learn 3 concepts: local receptive fields, shared weights & biases, and activation & pooling. You'll also learn 3 ways to train CNNs.



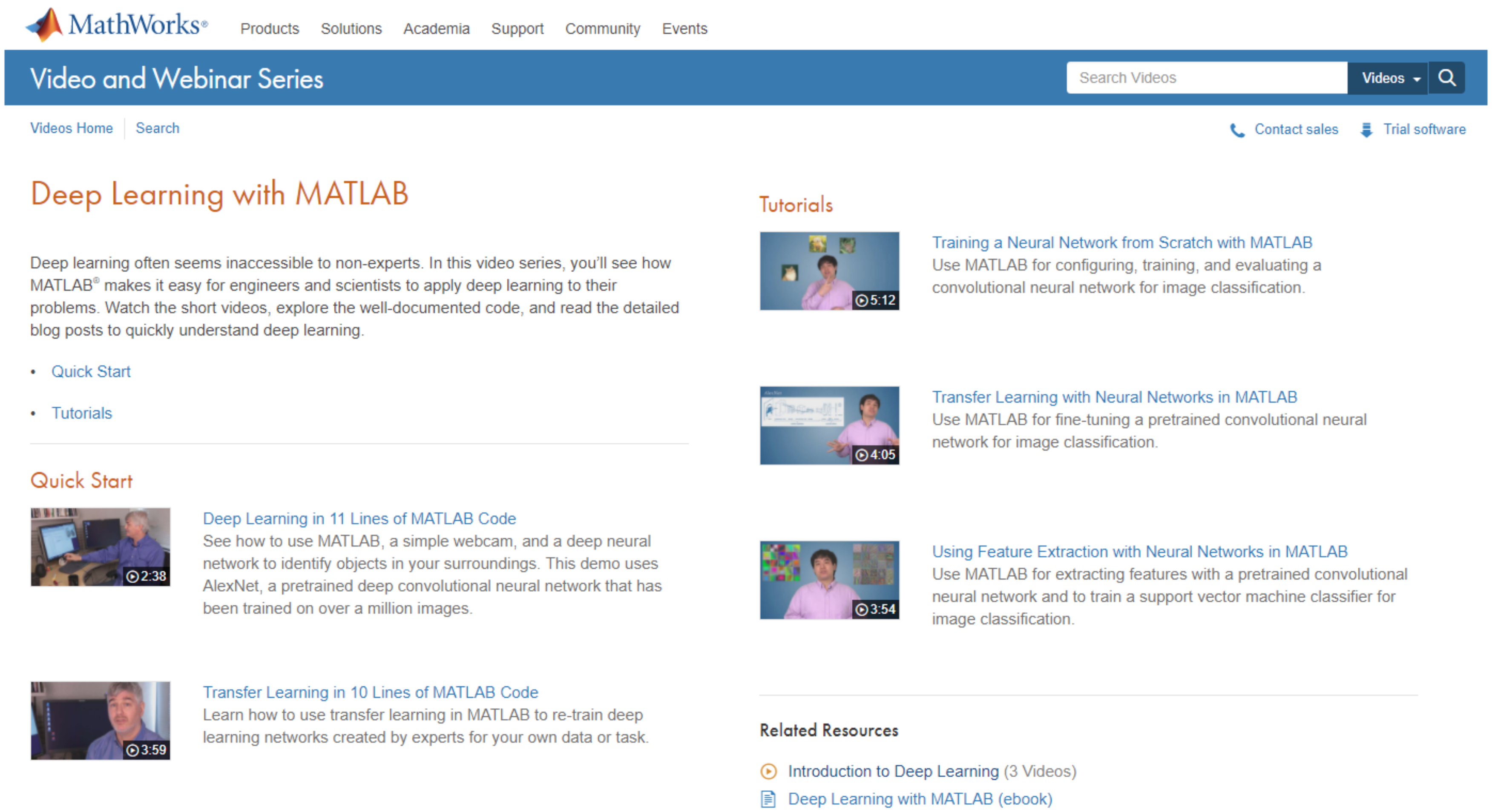
Machine Learning vs. Deep Learning
Learn about the differences between deep learning and machine learning in this MATLAB Tech Talk. Walk through several examples, and learn about how decide which method to use.

Related Resources

- Deep Learning with MATLAB (5 Videos)
- Deep Learning with MATLAB (ebook)

http://www.mathworks.com/videos/series/introduction-to-deep-learning.html

Deep Learning with MATLAB



MathWorks® Products Solutions Academia Support Community Events

Video and Webinar Series

Search Videos | Videos

Videos Home | Search

Contact sales | Trial software

Deep Learning with MATLAB

Deep learning often seems inaccessible to non-experts. In this video series, you'll see how MATLAB® makes it easy for engineers and scientists to apply deep learning to their problems. Watch the short videos, explore the well-documented code, and read the detailed blog posts to quickly understand deep learning.

- Quick Start
- Tutorials

Quick Start

2:38 **Deep Learning in 11 Lines of MATLAB Code**
See how to use MATLAB, a simple webcam, and a deep neural network to identify objects in your surroundings. This demo uses AlexNet, a pretrained deep convolutional neural network that has been trained on over a million images.

3:59 **Transfer Learning in 10 Lines of MATLAB Code**
Learn how to use transfer learning in MATLAB to re-train deep learning networks created by experts for your own data or task.

Tutorials

5:12 **Training a Neural Network from Scratch with MATLAB**
Use MATLAB for configuring, training, and evaluating a convolutional neural network for image classification.

4:05 **Transfer Learning with Neural Networks in MATLAB**
Use MATLAB for fine-tuning a pretrained convolutional neural network for image classification.

3:54 **Using Feature Extraction with Neural Networks in MATLAB**
Use MATLAB for extracting features with a pretrained convolutional neural network and to train a support vector machine classifier for image classification.

Related Resources

- Introduction to Deep Learning (3 Videos)
- Deep Learning with MATLAB (ebook)

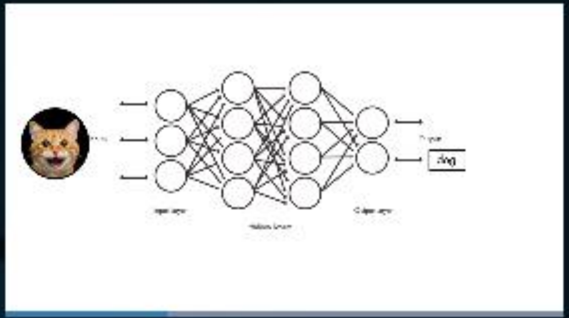
http://www.mathworks.com/videos/series/deep-learning-with-MATLAB.html

Deep Learning Onramp

This free, two-hour deep learning tutorial provides an interactive introduction to practical deep learning methods. You will learn to use deep learning techniques in MATLAB® for image recognition.

Prerequisites: MATLAB Onramp or basic knowledge of MATLAB

Launch the course



Access to MATLAB through your web browser

Engaging video tutorials

Hands-on exercises with automated assessments and feedback

Lessons available in English and Japanese

http://www.mathworks.com/learn/tutorials/deep-learning-onramp.html

Thank you!