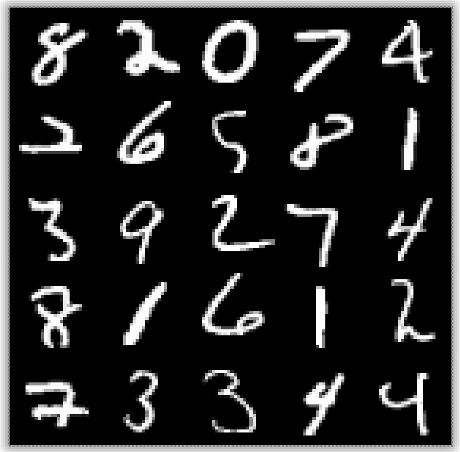
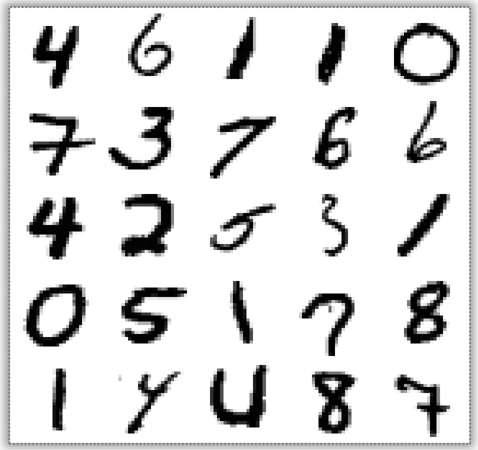
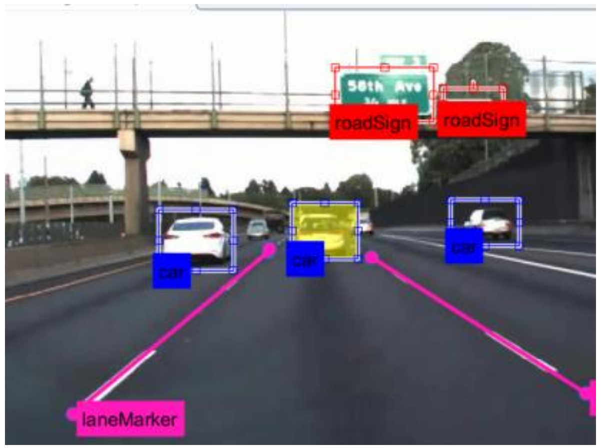
MATLAB EXPO 2018

Demystifying Deep Learning

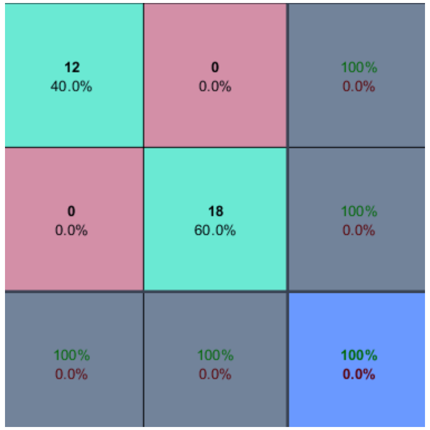
Dr. Amod Anandkumar
Senior Team Lead – Signal Processing & Communications







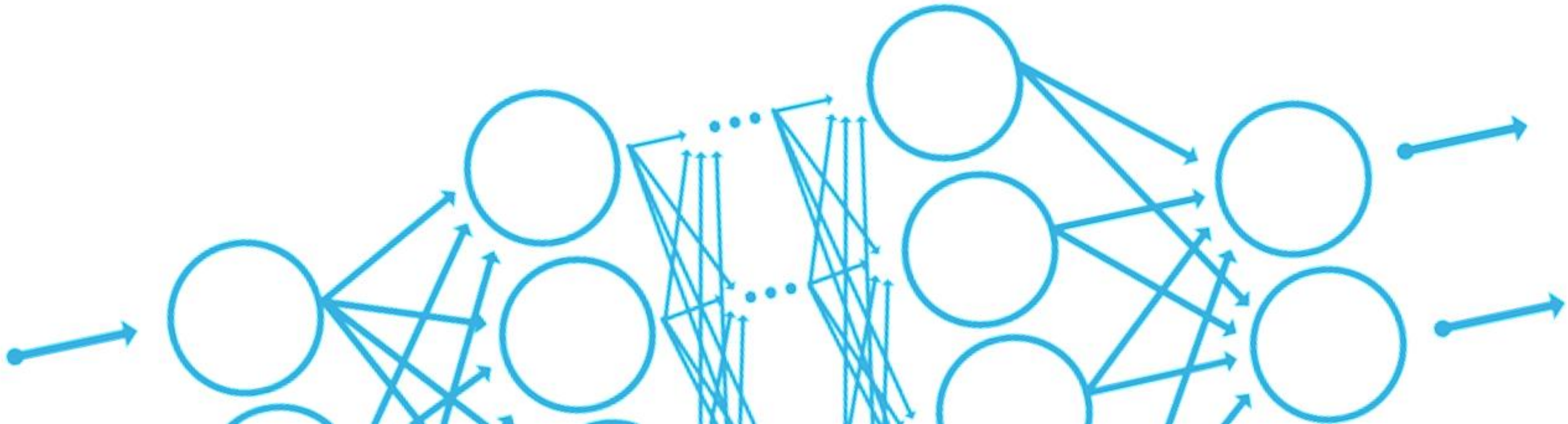
What is Deep Learning?



Deep learning is a type of machine learning in which a model learns to perform classification tasks directly from images, text, or sound.

Deep learning is usually implemented using a **neural network.**

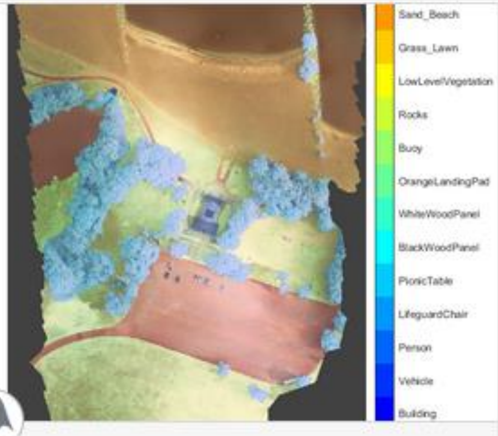
The term “deep” refers to the **number of layers** in the network—the more layers, the deeper the network.



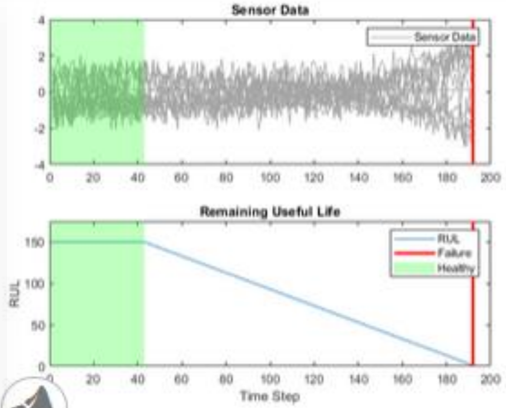
Deep Learning is **Versatile**

[MATLAB Examples Available Here]

Object Detection Using Faster R-CNN Deep Learning



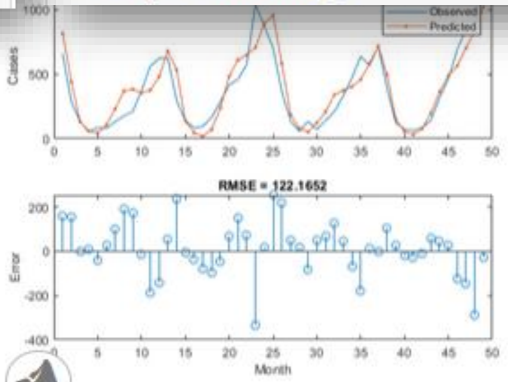
Semantic Segmentation of Multispectral Images Using Deep Learning



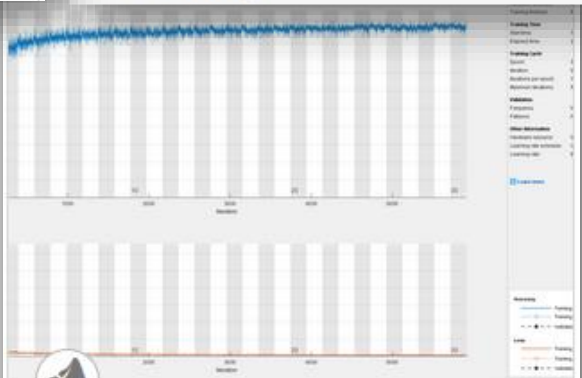
Sequence-to-Sequence Regression Using Deep Learning

Single Image Super-Resolution Using Deep Learning

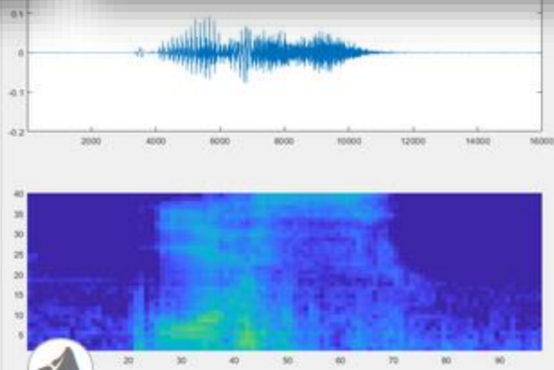
Classify Image Using GoogLeNet



Time Series Forecasting Using Deep Learning



Classify Text Data Using Deep Learning



Deep Learning Speech Recognition

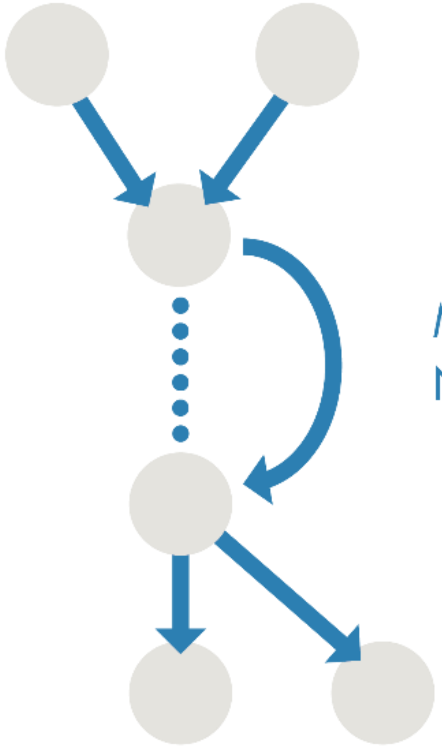
Many Network Architectures for Deep Learning

Series Network

Single-in
single-out

AlexNet
YOLO

Directed Acyclic Graph Network



Multi-in, multi-out
No feedback loops

ResNet
R-CNN

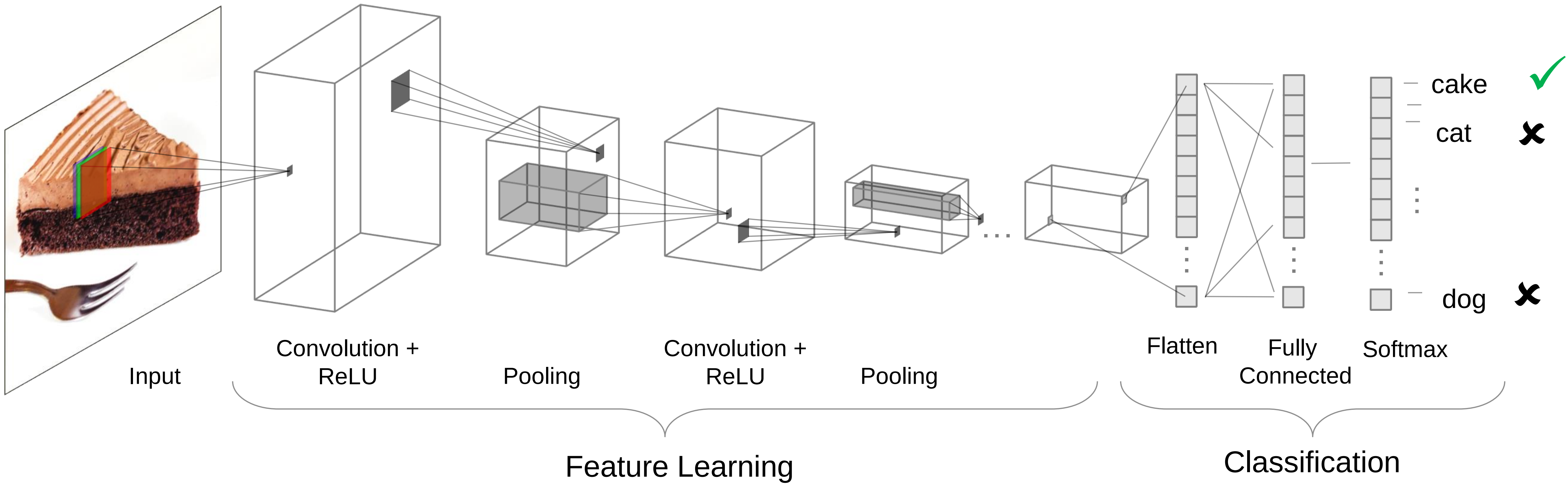
Recurrent Network

Memory

LSTM

and more
(GAN, DQN,…)

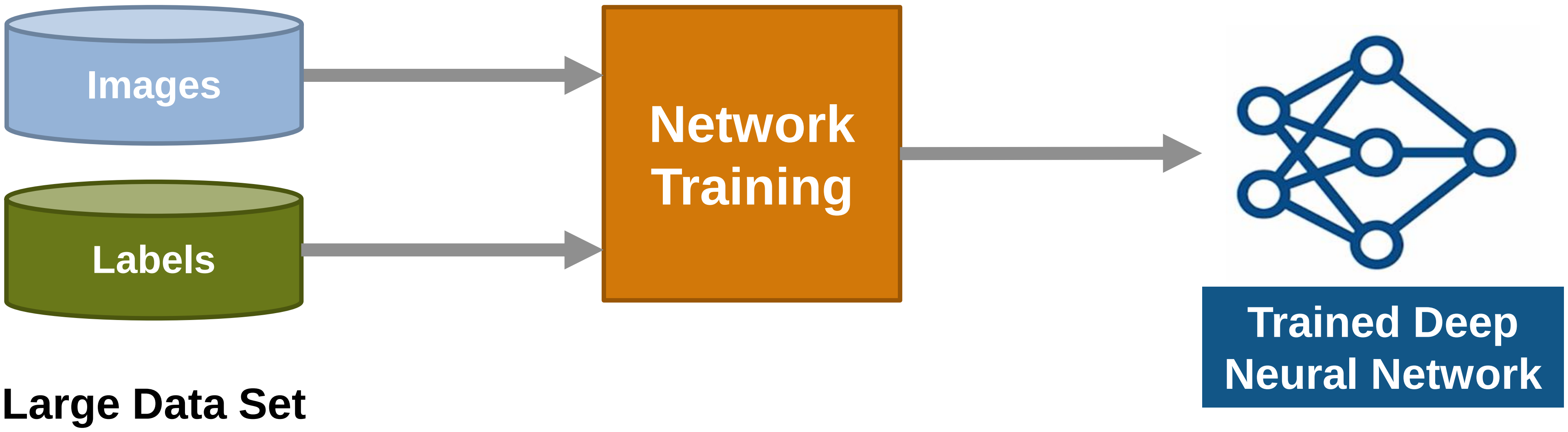
Convolutional Neural Networks



Deep Learning Inference in 4 Lines of Code

```
>> net = alexnet;
>> I = imread('peacock.jpg')
>> I1 = imresize(I,[227 227]);
>> classify(net,I1)
ans =
   categorical
    peacock
```

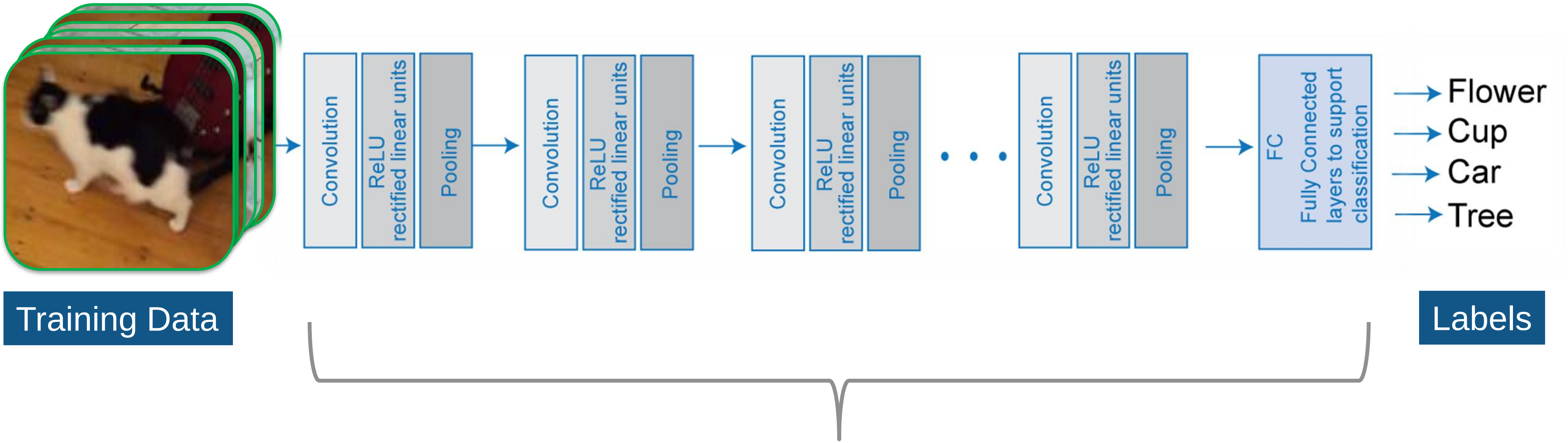
What is Training?



During training, neural network architectures learn features directly from the data without the need for manual feature extraction

What Happens During Training?

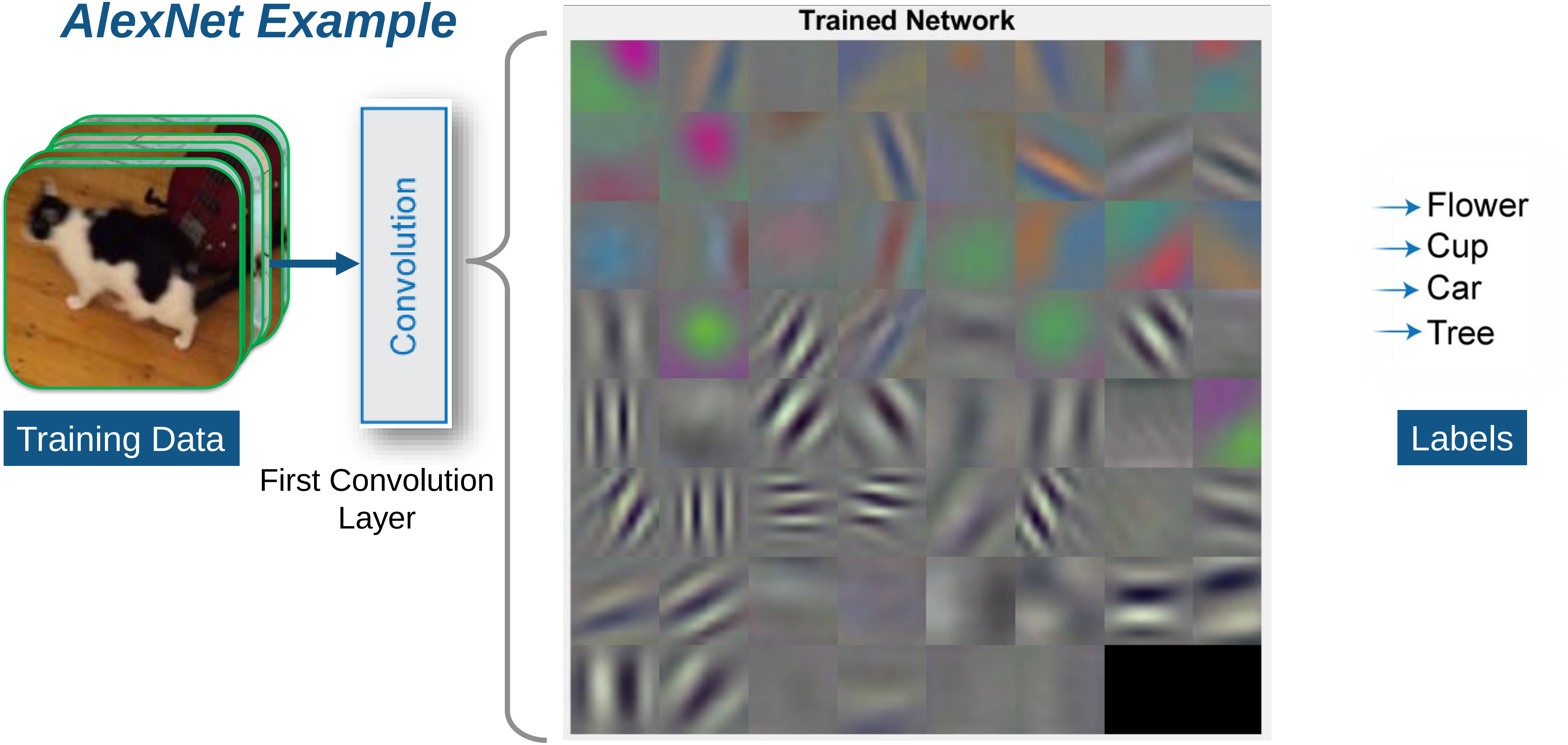
AlexNet Example



Layer weights are learned during training

Visualize Network Weights During Training

AlexNet Example



Visualization Technique – Deep Dream

```
deepDreamImage(...
      net, 'fc5', channel,
     'NumIterations', 50, ...
     'PyramidLevels', 4,...
     'PyramidScale', 1.25);
```

Synthesizes images that strongly activate a channel in a particular layer

Example Available Here

Visualize Features Learned During Training

AlexNet Example

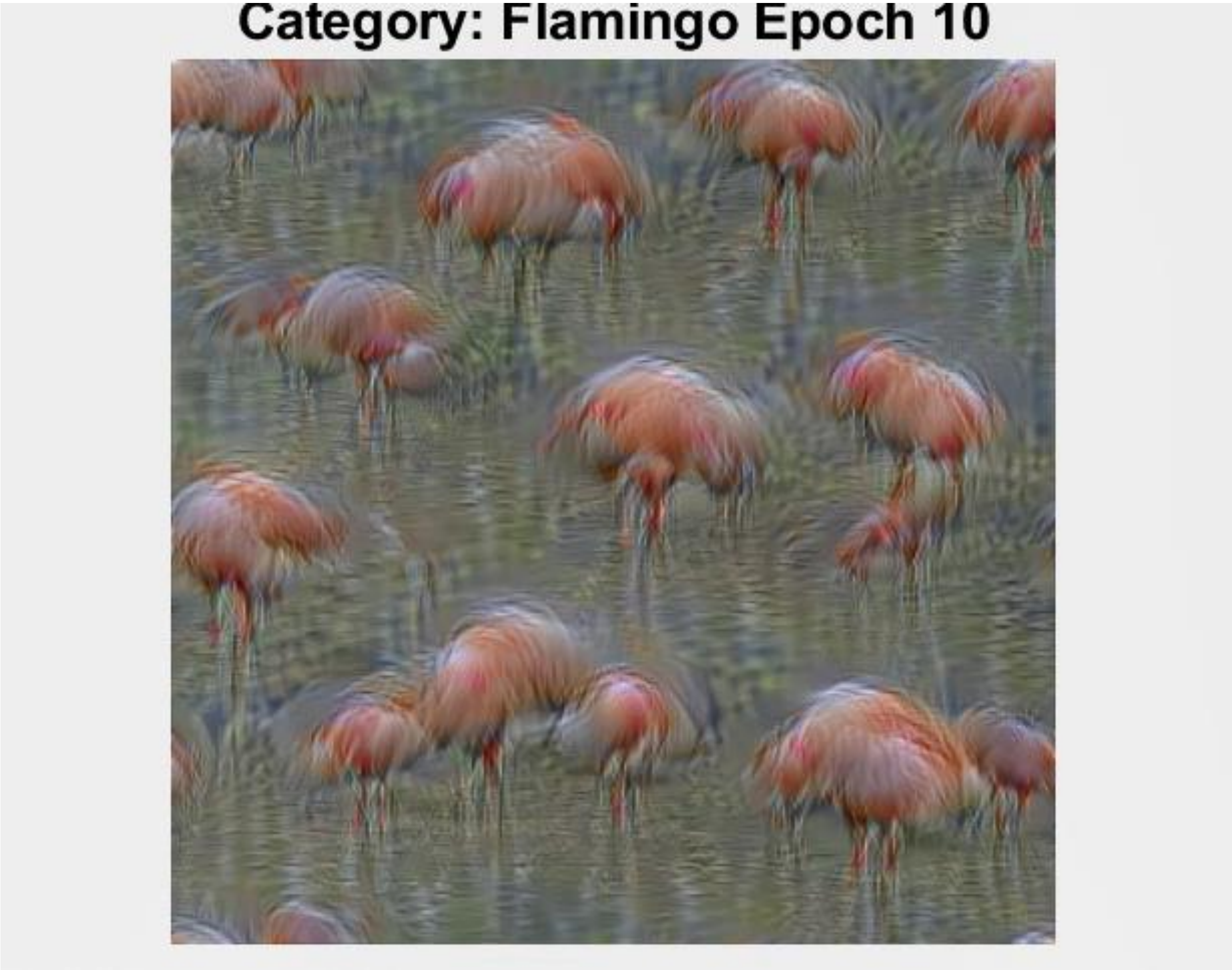
Sample Training Data

Features Learned by Network

Visualize Features Learned During Training

AlexNet Example

Sample Training Data



Features Learned by Network

Deep Learning Challenges

Data

- Handling large amounts of data
- Labeling thousands of images & videos

Training and Testing Deep Neural Networks

- Accessing reference models from research
- Optimizing hyperparameters
- Training takes hours-days

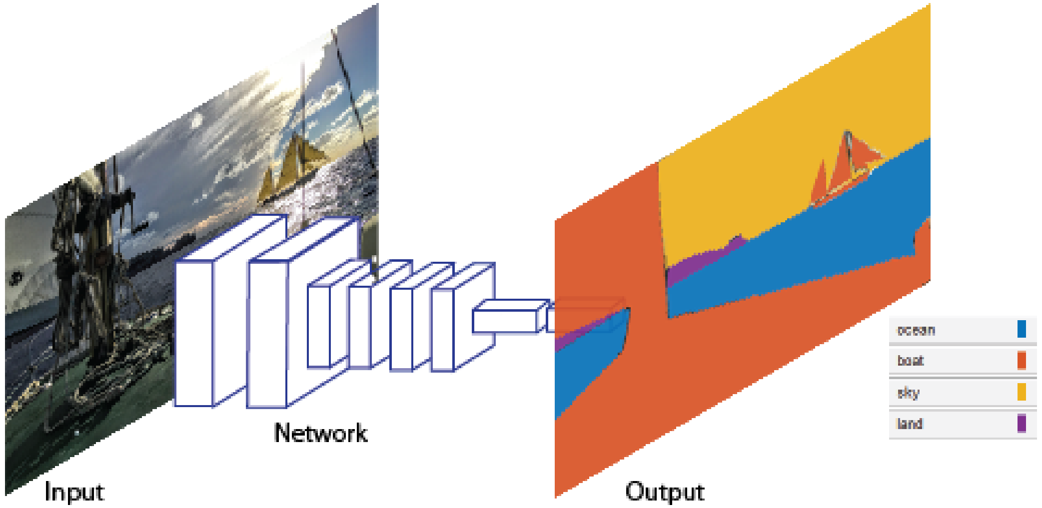
Rapid and Optimized Deployment

- Desktop, web, cloud, and embedded hardware

Not a deep learning expert

Example – Semantic Segmentation

Available Here

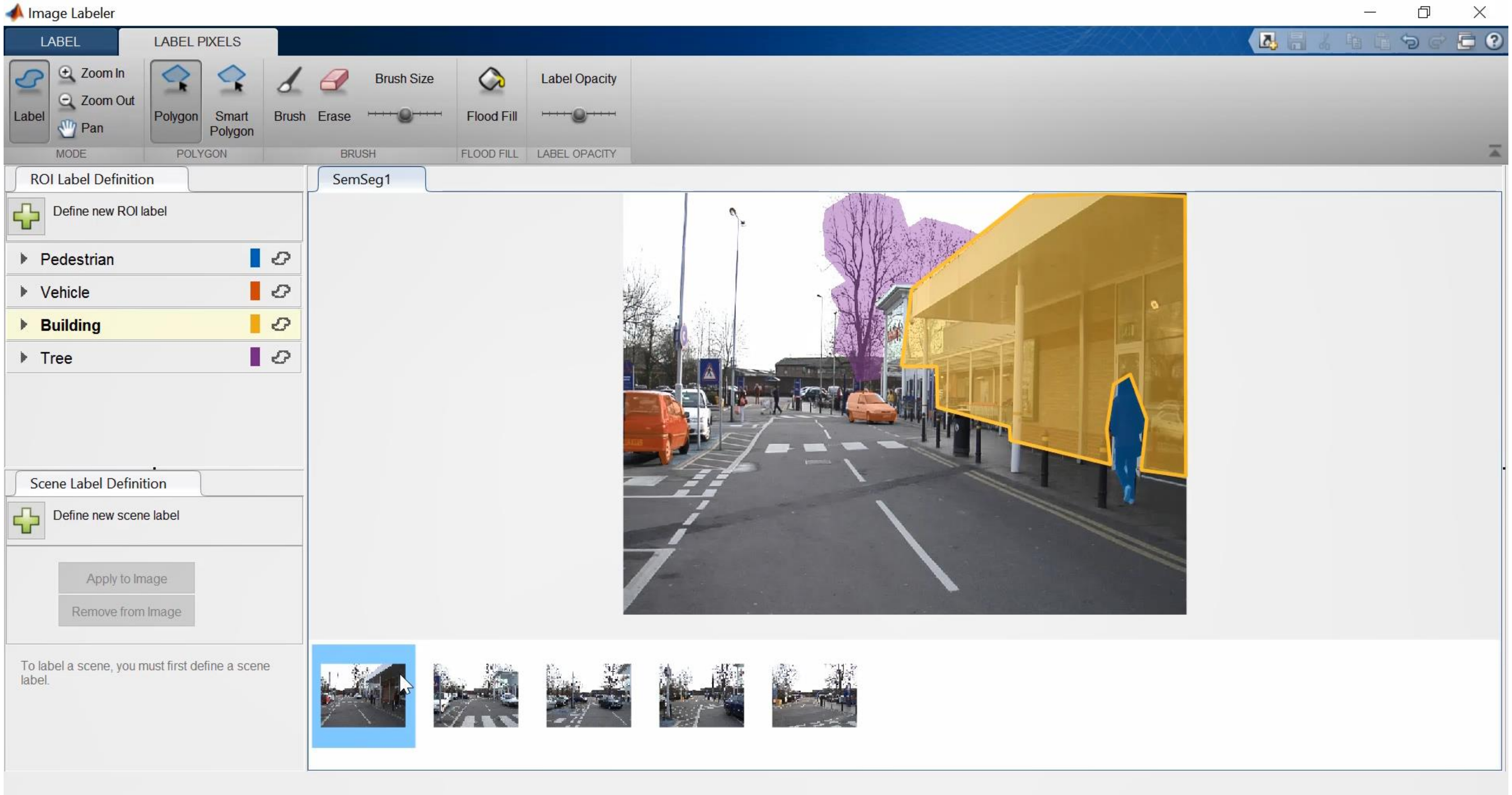


- Classify pixels into 11 classes
  - Sky, Building, Pole, Road, Pavement, Tree, SignSymbol, Fence, Car, Pedestrian, Bicyclist

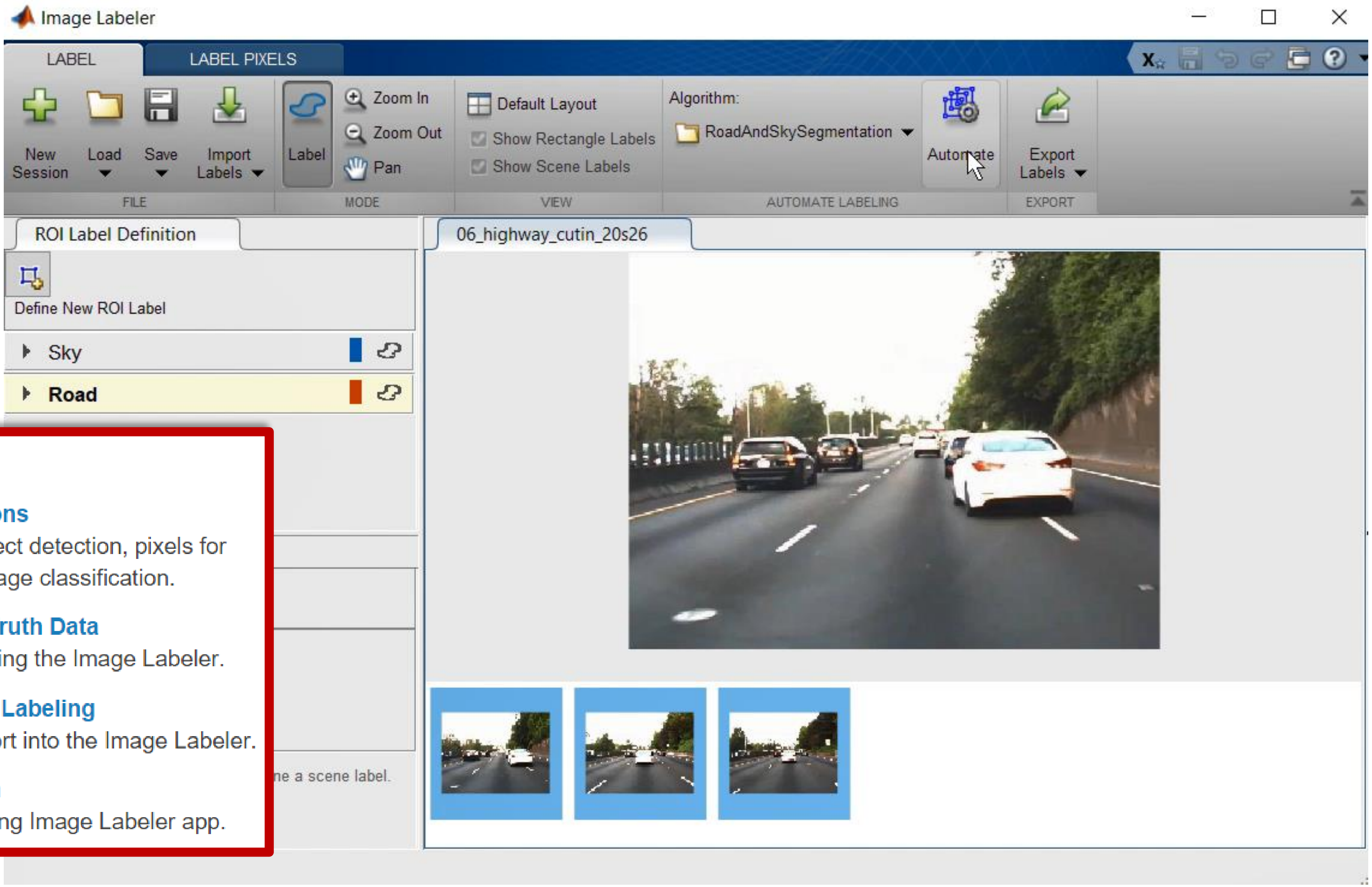
- CamVid dataset

Brostow, Gabriel J., Julien Fauqueur, and Roberto Cipolla. "Semantic object classes in video: A high-definition ground truth database." Pattern Recognition Letters Vol 30, Issue 2, 2009, pp 88-97.

Label Images Using Image Labeler App



Accelerate Labeling With Automation Algorithms



Learn More

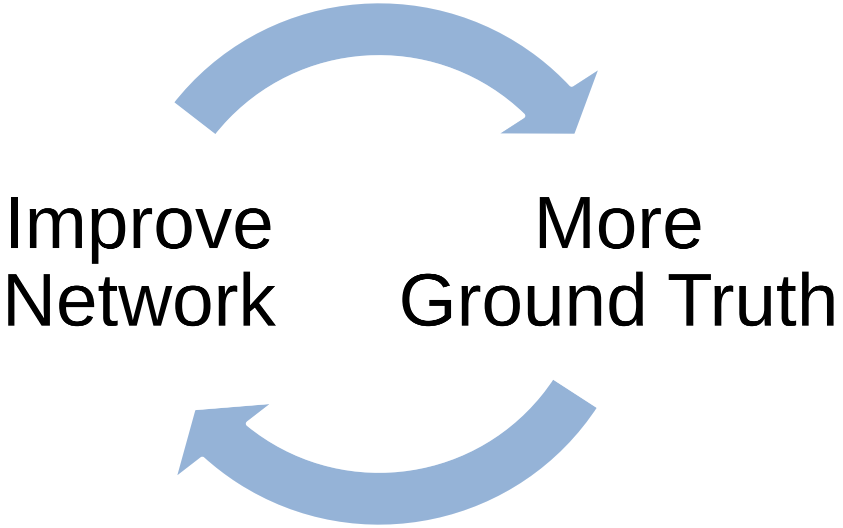
Define Ground Truth for Image Collections
Interactively label rectangular ROIs for object detection, pixels for semantic segmentation, and scenes for image classification.

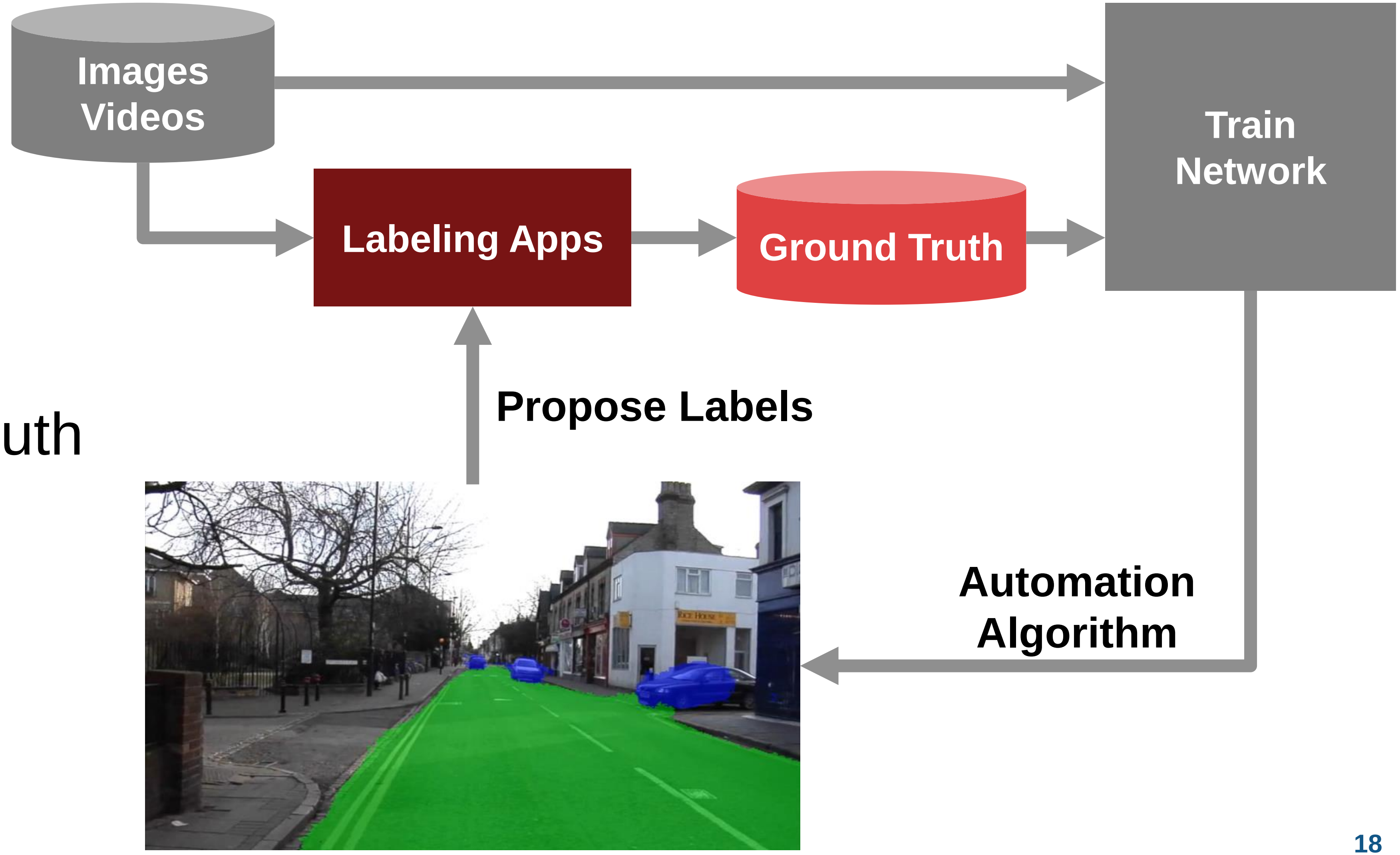
Train an Object Detector from Ground Truth Data
Create training data for object detection using the Image Labeler.

Create Automation Algorithm for Image Labeling
Create a custom tracking algorithm to import into the Image Labeler.

Label Pixels for Semantic Segmentation
Label pixels for semantic segmentation using Image Labeler app.

Perform Bootstrapping to Label Large Datasets





Example – Semantic Segmentation

[Available Here](#)

Examples

Search Help

Semantic Segmentation Using Deep Learning

This example shows how to train a semantic segmentation network using deep learning.

A semantic segmentation network classifies every pixel in an image, resulting in an image that is segmented by class. Applications for semantic segmentation include road segmentation for autonomous driving and cancer cell segmentation for medical diagnosis. To learn more, see Semantic Segmentation Basics.

To illustrate the training procedure, this example trains SegNet [1], one type of convolutional neural network (CNN) designed for semantic image segmentation. Other types networks for semantic segmentation include fully convolutional networks (FCN) and U-Net. The training procedure shown here can be applied to those networks too.

This example uses the CamVid dataset [2] from the University of Cambridge for training. This dataset is a collection of images containing street-level views obtained while driving. The dataset provides pixel-level labels for 32 semantic classes including car, pedestrian, and road.

Learn more about

This example also uses:
Neural Network Toolbox
vgg16

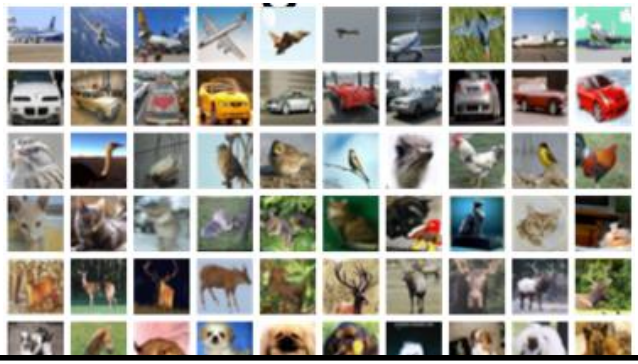
Open Script

Setup

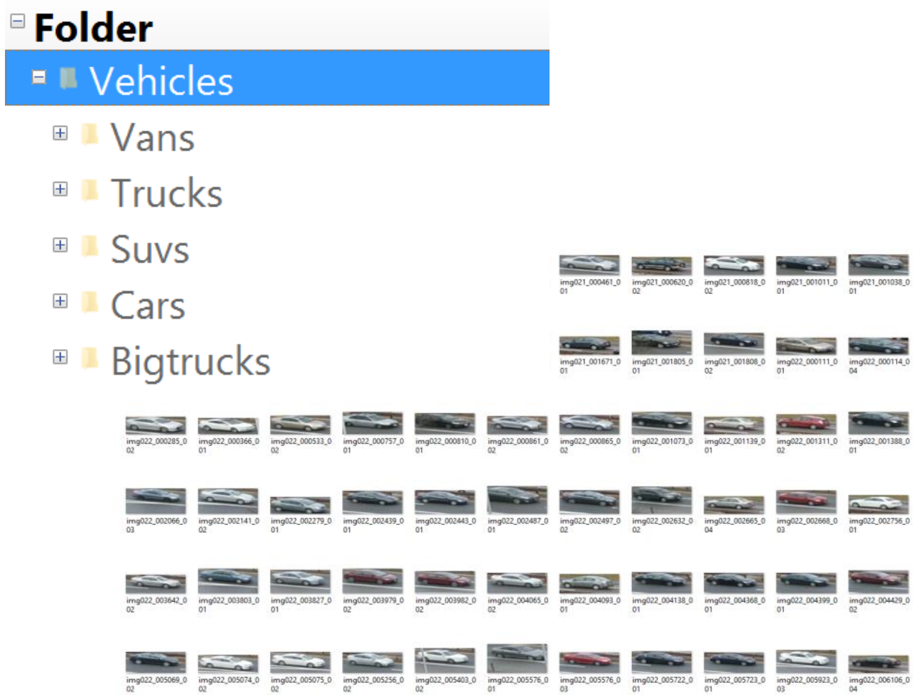
This example creates the SegNet network with weights initialized from the VGG-16 network. To get VGG-16, install Neural Network Toolbox™ Model for VGG-16 Network. After installation is complete, run the following code to verify that the installation is correct.

```
vgg16();
```

Access Large Sets of Images



Handle Large Sets of Images



Organize Images in Folders
(~ 10,000 images , 5 folders)

```
imageData =
imageDataStore('vehicles')
```

Easily manage large sets of images
- Single line of code to access images
- Operates on disk, database, big-data file system

Handle Big Image Collections without Big Changes

Images in local directory

```
fileLoc = 'FoodImages';
ds = imageDatastore(fileLoc,'IncludeSubfolders',true,...
    'LabelSource','foldernames')
```

```
ds =

  ImageDatastore with p

      Files: {
             ' ...\Wor
             ' ...\Wor
             ' ...\Wor
              ... and
             }
     Labels: [chocolat
   ReadSize: 1
    ReadFcn: @readData
```

Images on HDFS

```
fileLoc = 'hdfs://hadoop01glnxa64:54310/datasets/FoodImages';
ds = imageDatastore(fileLoc,'IncludeSubfolders',true,...
    'LabelSource','foldernames')
```

```
ds =

  ImageDatastore with properties:

      Files: {
             'hdfs://hadoop01glnxa64:54310/datasets/FoodImages/apple_pie/1005649.jpg';
             'hdfs://hadoop01glnxa64:54310/datasets/FoodImages/apple_pie/1011328.jpg';
             'hdfs://hadoop01glnxa64:54310/datasets/FoodImages/apple_pie/101251.jpg'
              ... and 100997 more
             }
     Labels: [apple_pie; apple_pie; apple_pie ... and 100997 more categorical]
   ReadSize: 1
    ReadFcn: @readDatastoreImage
```

Import Pre-Trained Models and Network Architectures

Pretrained Models

- `alexnet`
- `vgg16`
- `vgg19`
- `googlenet`
- `inceptionv3`
- `resnet50`
- `resnet101`
- `inceptionresnetv2`
- `squeezenet`

Import Models from Frameworks

- Caffe Model Importer (including Caffe Model Zoo)
  - `importCaffeLayers`
  - `importCaffeNetwork`
- TensorFlow-Keras Model Importer
  - `importKerasLayers`
  - `importKerasNetwork`

Caffe MODELS

KERAS IMPORTER

Importer for TensorFlow-Keras Models

Download from within MATLAB

Add-Ons

Example – Semantic Segmentation

Available Here

Examples

Search Help

Semantic Segmentation Using Deep Learning

This example shows how to train a semantic segmentation network using deep learning.

A semantic segmentation network classifies every pixel in an image, resulting in an image that is segmented by class. Applications for semantic segmentation include road segmentation for autonomous driving and cancer cell segmentation for medical diagnosis. To learn more, see Semantic Segmentation Basics.

To illustrate the training procedure, this example trains SegNet [1], one type of convolutional neural network (CNN) designed for semantic image segmentation. Other types networks for semantic segmentation include fully convolutional networks (FCN) and U-Net. The training procedure shown here can be applied to those networks too.

This example also uses:
Neural Network Toolbox
vgg16

Open Script

This example uses the CamVid dataset [2] from the University of Cambridge for training. This dataset is a collection of images containing street-level views obtained while driving. The dataset provides pixel-level labels for 32 semantic classes including car, pedestrian, and road.

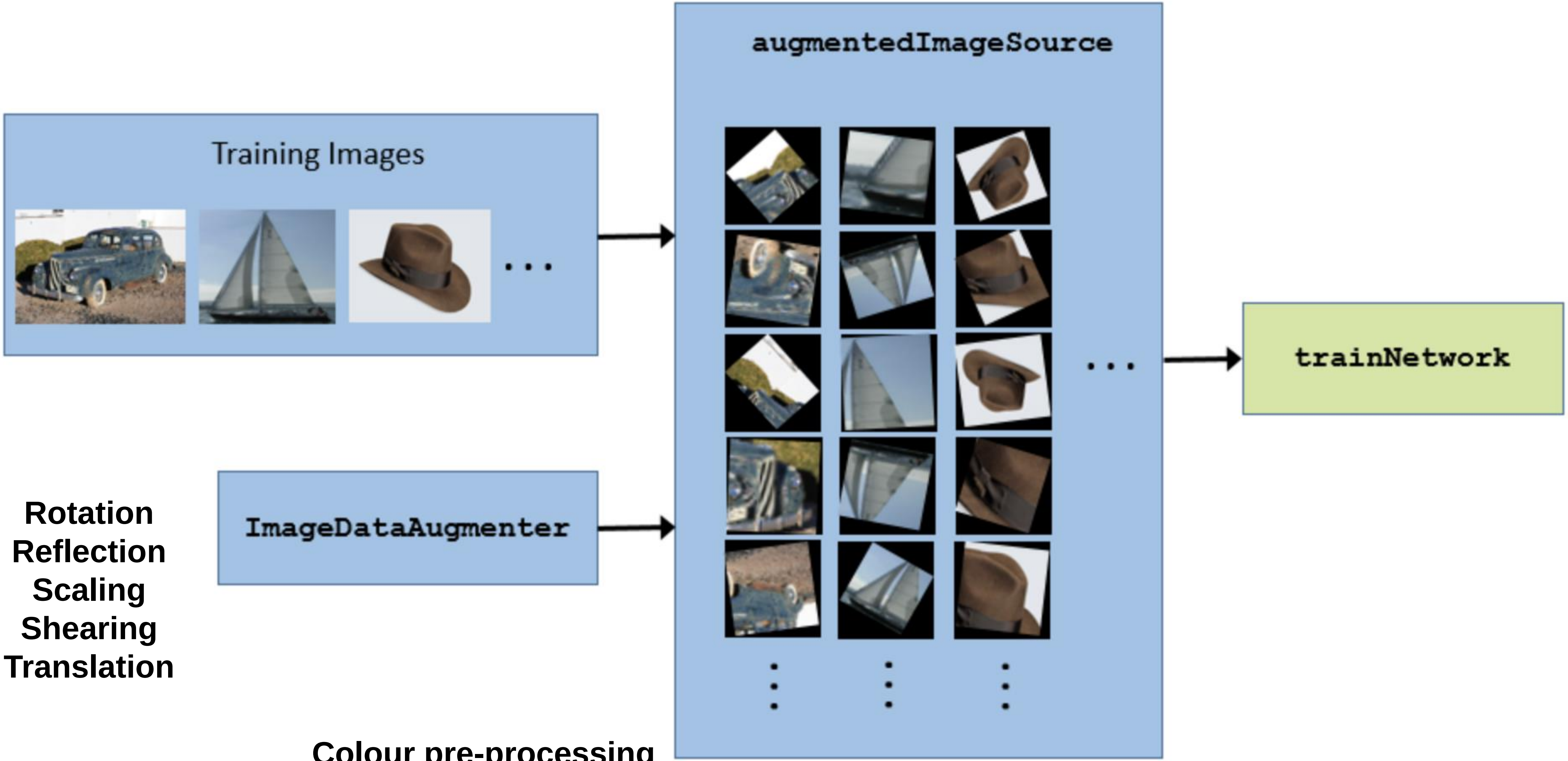
Learn more about

Setup

This example creates the SegNet network with weights initialized from the VGG-16 network. To get VGG-16, install Neural Network Toolbox™ Model for VGG-16 Network. After installation is complete, run the following code to verify that the installation is correct.

```
vgg16();
```

Augment Training Images



Rotation
Reflection
Scaling
Shearing
Translation

Colour pre-processing
Resize / Random crop / Centre crop

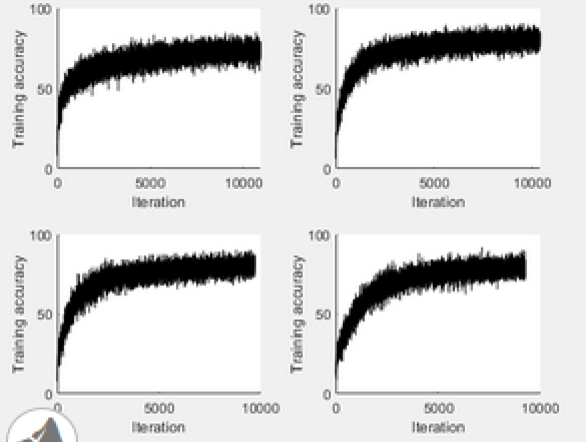
Tune Hyperparameters to Improve Training

Many hyperparameters

- depth, layers, solver options, learning rates, regularization, …

Techniques

- Parameter sweep
- Bayesian optimization



Use parfeval to Train Multiple Deep Learning Networks

Use parfeval for a parameter sweep on the depth of the network architecture. Deep Learning training often takes hours or days, and

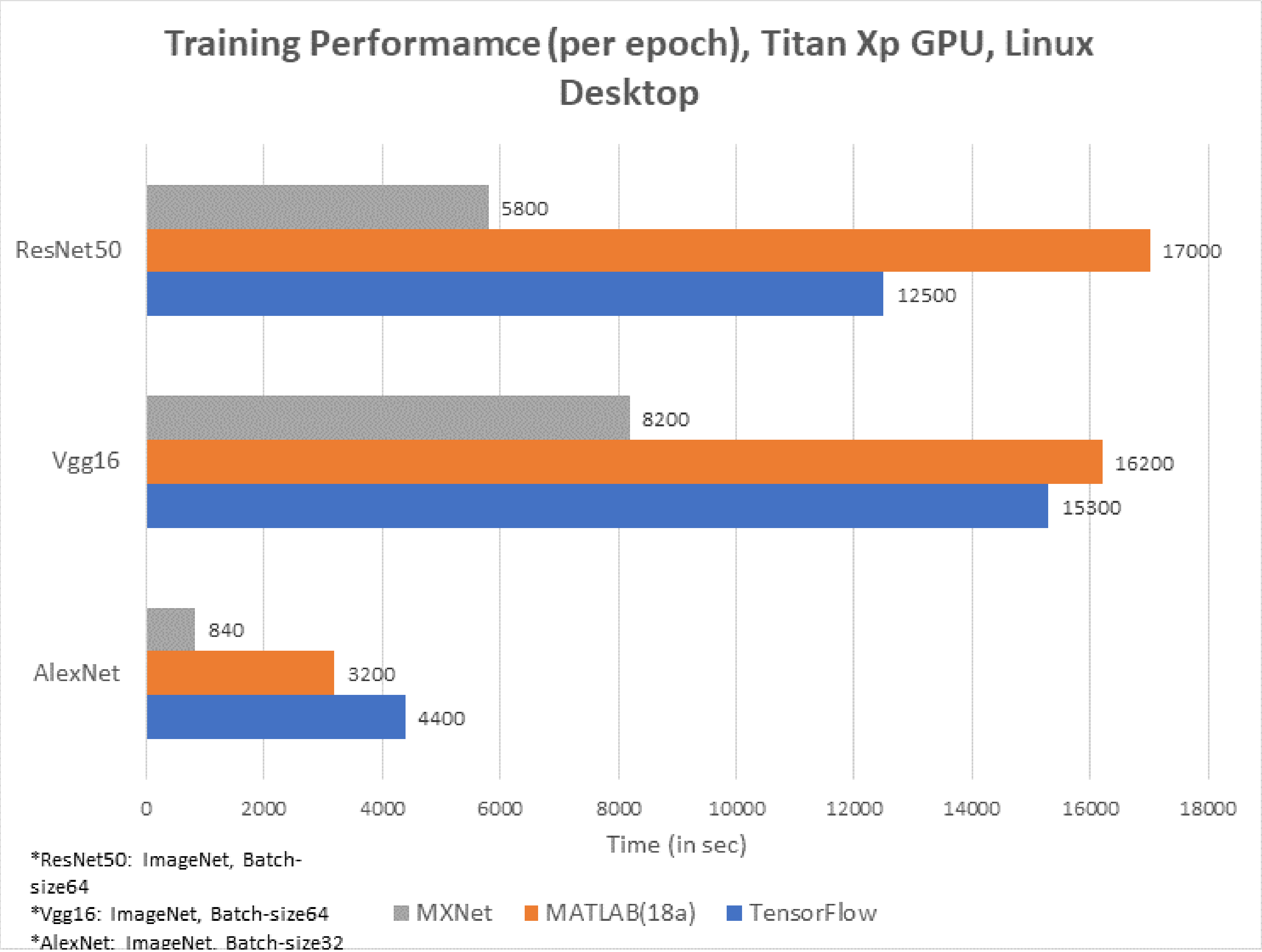
Open Script

Deep Learning Using Bayesian Optimization

Apply Bayesian optimization to deep learning and find optimal network parameters and training options for convolutional neural networks.

Open Live Script

Training Performance

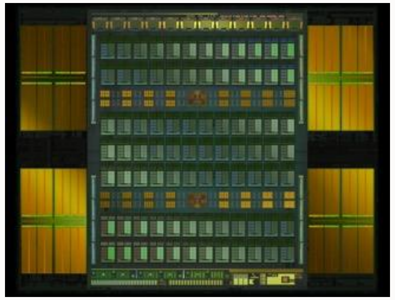


TensorFlow
MATLAB
MXNet

NVIDIA Tesla V100 32GB

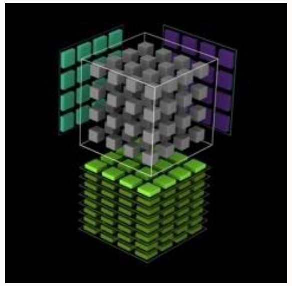
The Fastest and Most Productive GPU for AI and HPC

Volta Architecture



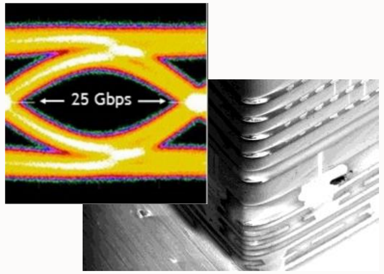
Most Productive GPU

Tensor Core



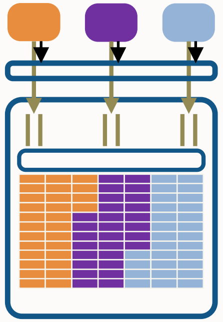
125 Programmable
TFLOPS Deep Learning

Improved NVLink & HBM2



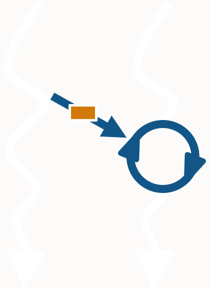
Efficient Bandwidth

Volta MPS

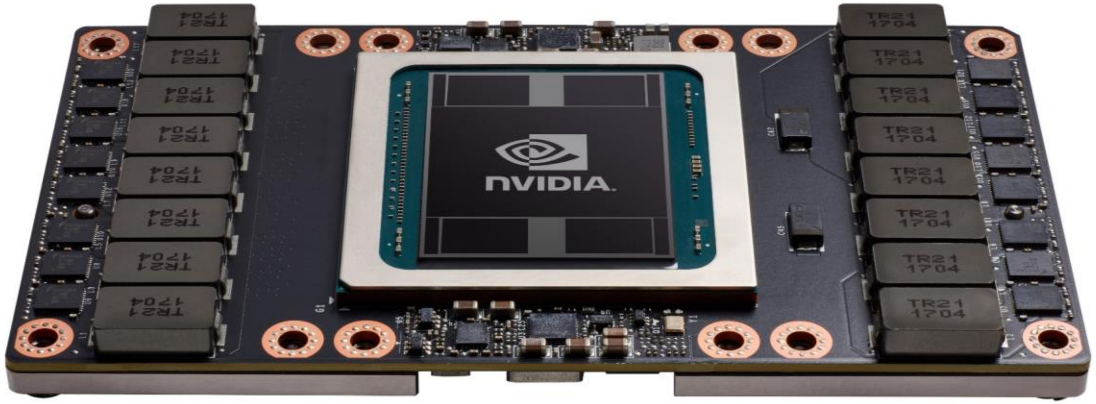


Inference Utilization

Improved SIMT Model



New Algorithms



Core	5120 CUDA cores, 640 Tensor cores
Compute	7.8 TF DP · 15.7 TF SP · 125 TF DL
Memory	HBM2: 900 GB/s · 32 GB/16 GB
Interconnect	NVLink (up to 300 GB/s) + PCIe Gen3 (up to 32 GB/s)

Visit NVIDIA booth
to learn more

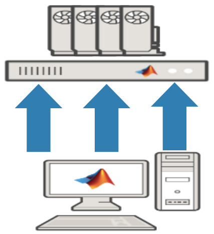
NVIDIA

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

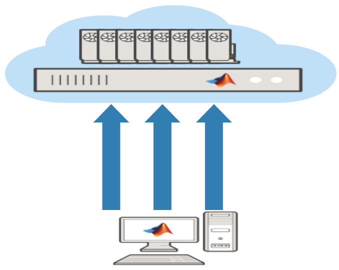
Single
CPU

Single CPU
Single GPU

Single CPU, Multiple GPUs



On-prem server with
GPUs



Cloud GPUs
(AWS)

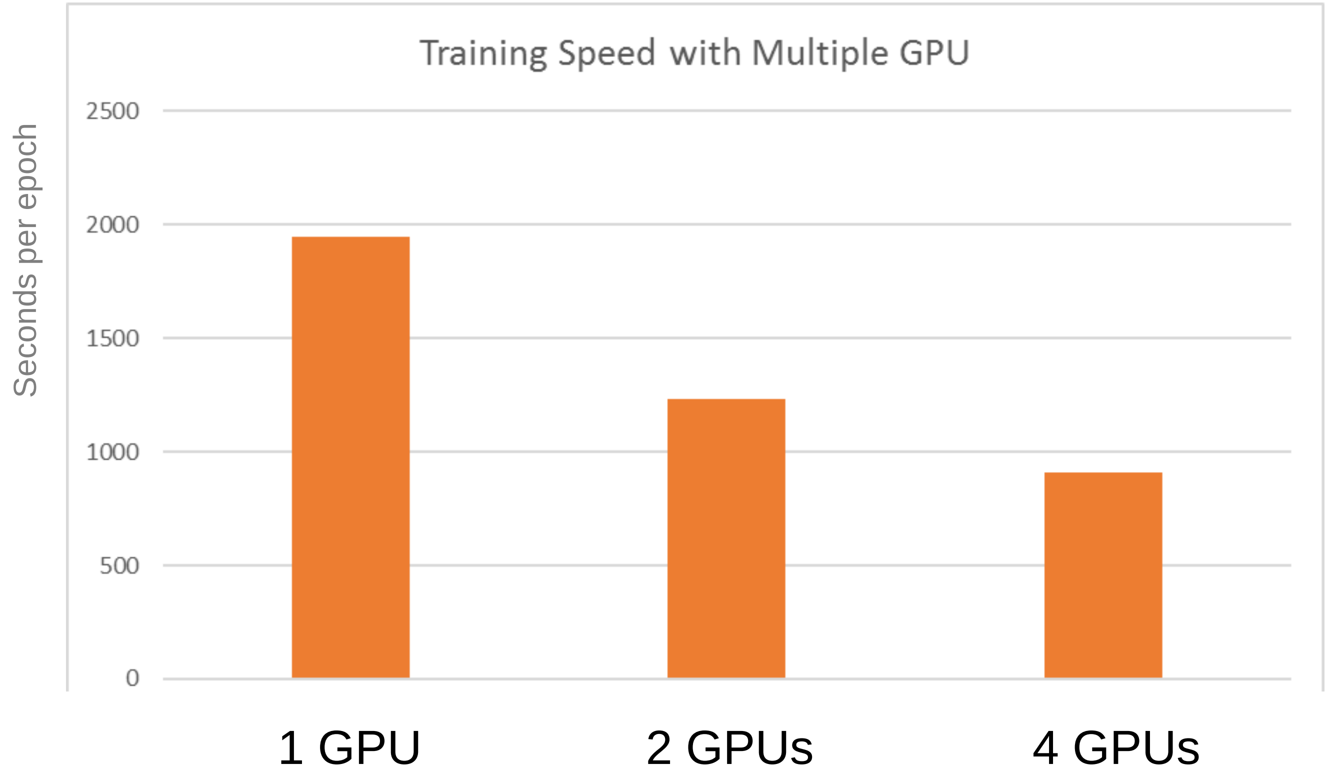
How to target?

```
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' );
```

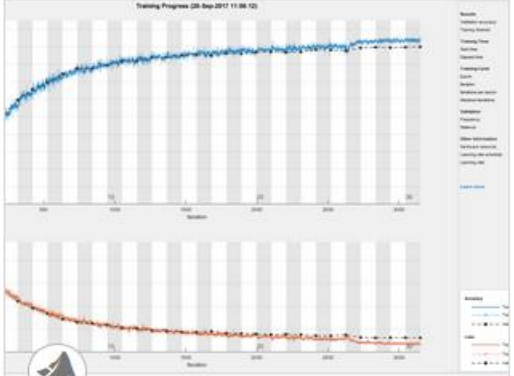
Multi-GPU Performance Scaling



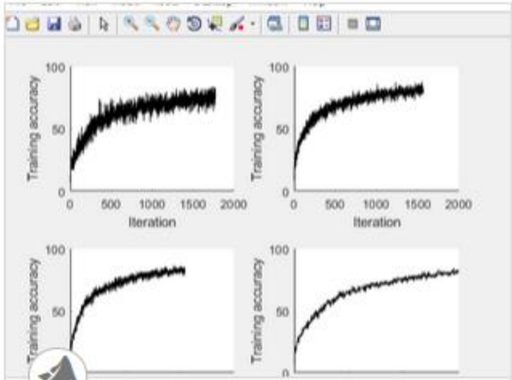
Ease of scaling

- MATLAB “transparently” scales to multiple GPUs
- Runs on Windows!

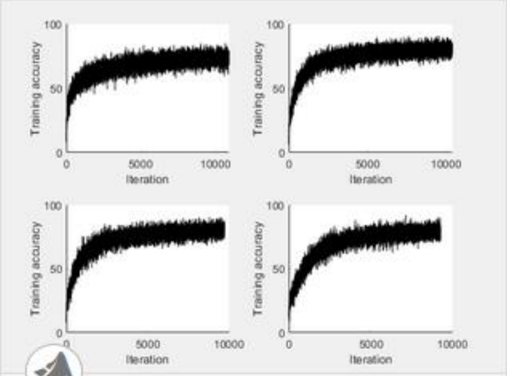
Examples to Learn More



Train Network in the Cloud Using Built-in Parallel Support

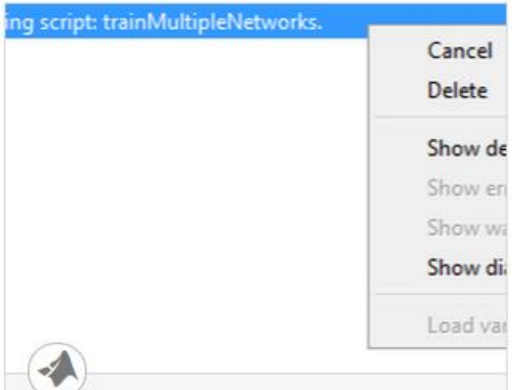


Use parfor to Train Multiple Deep Learning Networks



Use parfeval to Train Multiple Deep Learning Networks

Upload Deep Learning Data to the Cloud



Send Deep Learning Batch Job To Cluster

Example – Semantic Segmentation

Available Here

Examples

Search Help

Semantic Segmentation Using Deep Learning

This example shows how to train a semantic segmentation network using deep learning.

A semantic segmentation network classifies every pixel in an image, resulting in an image that is segmented by class. Applications for semantic segmentation include road segmentation for autonomous driving and cancer cell segmentation for medical diagnosis. To learn more, see Semantic Segmentation Basics.

To illustrate the training procedure, this example trains SegNet [1], one type of convolutional neural network (CNN) designed for semantic image segmentation. Other types networks for semantic segmentation include fully convolutional networks (FCN) and U-Net. The training procedure shown here can be applied to those networks too.

This example also uses:
Neural Network Toolbox
vgg16

Open Script

This example uses the CamVid dataset [2] from the University of Cambridge for training. This dataset is a collection of images containing street-level views obtained while driving. The dataset provides pixel-level labels for 32 semantic classes including car, pedestrian, and road.

Learn more about

Setup

This example creates the SegNet network with weights initialized from the VGG-16 network. To get VGG-16, install Neural Network Toolbox™ Model for VGG-16 Network. After installation is complete, run the following code to verify that the installation is correct.

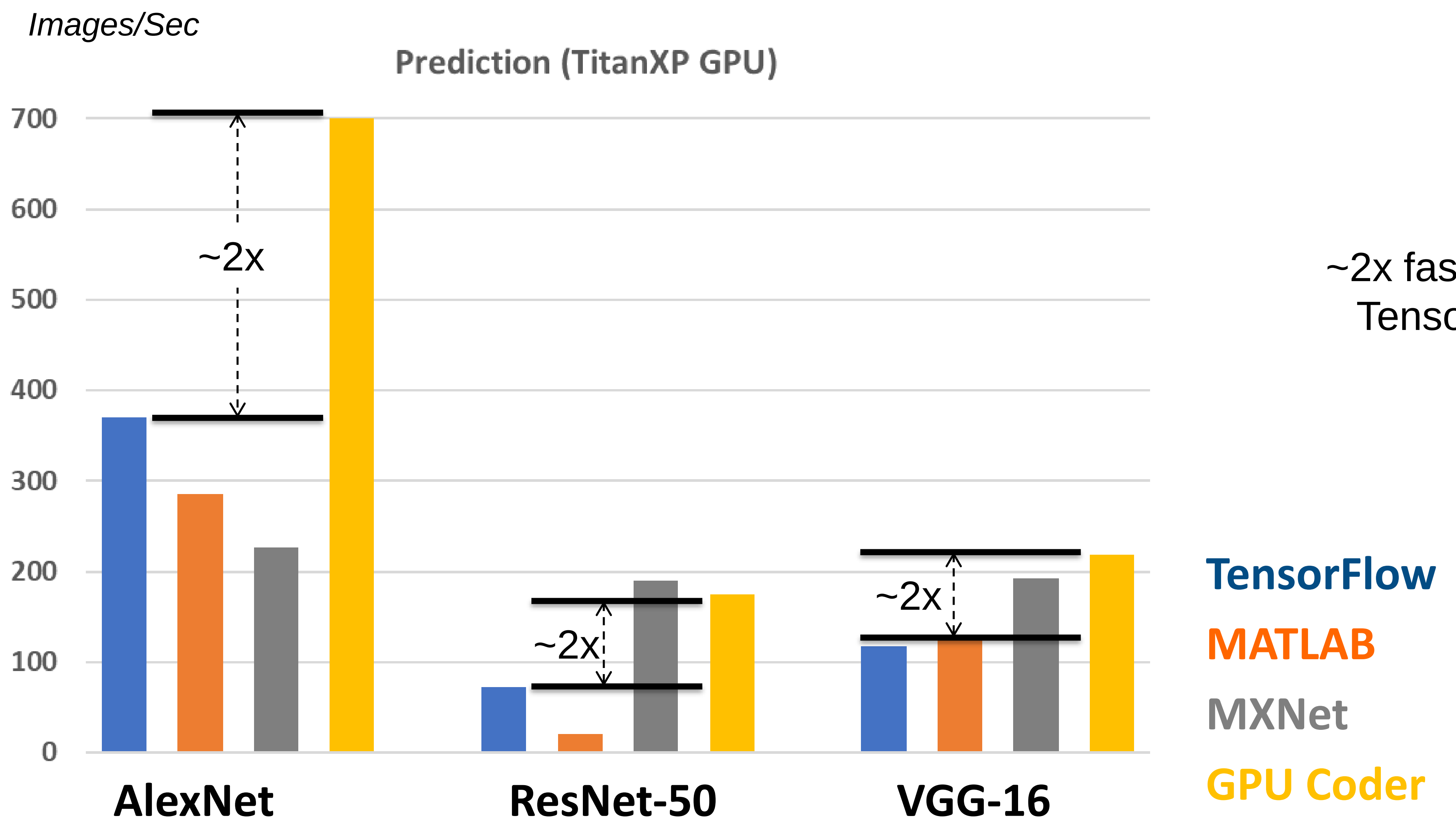
```
vgg16();
```

Accelerate Using GPU Coder

Running in MATLAB

Generated Code from GPU Coder

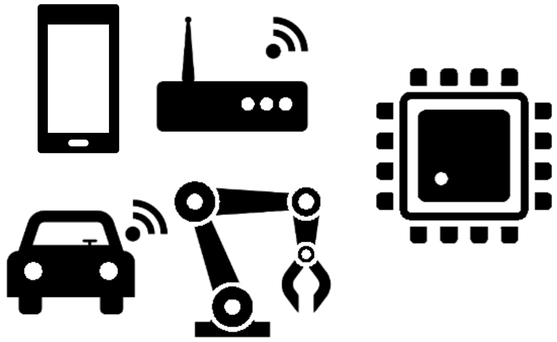
Prediction Performance: Fast with GPU Coder



~2x faster than TensorFlow

Deploying Deep Learning Application

Embedded Hardware

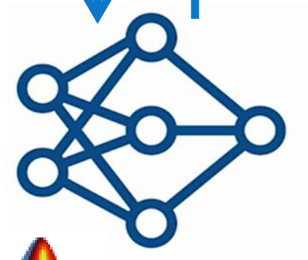


ARM

intel

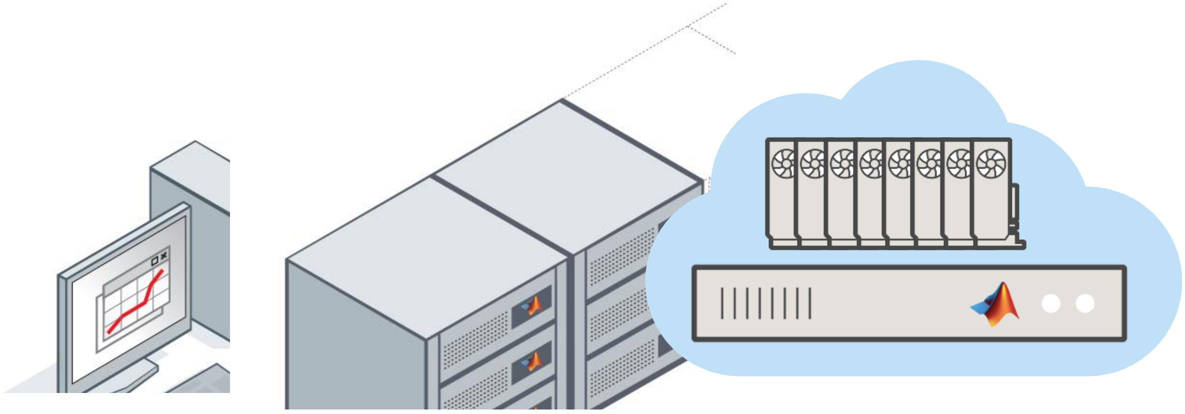
NVIDIA

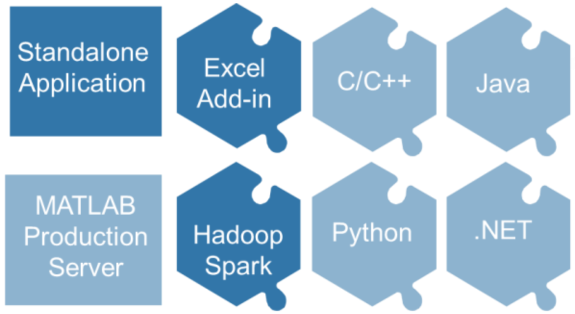
Application logic



MATLAB

Desktop, Web, Cloud





Code Generation

Application Deployment

Next Session

Deploying Deep Neural Networks to Embedded GPUs and CPUs

15:30–16:15

Designing and deploying deep learning and computer vision applications to embedded CPU and GPU platforms is challenging because of resource constraints inherent in embedded devices. A MATLAB® based workflow facilitates the design of these applications, and automatically generated C or CUDA® code can be deployed on boards like the Jetson TX2 and DRIVE™ PX to achieve very fast inference.

The presentation illustrates how MATLAB supports all major phases of this workflow. Starting with algorithm design, the algorithm may employ deep neural networks augmented with traditional computer vision techniques and can be tested and verified within MATLAB. Next, these networks are trained using GPU and parallel computing support for MATLAB either on the desktop, cluster, or the cloud. Finally, GPU Coder™ generates portable and optimized C/C++ and/or CUDA® code from the MATLAB algorithm, which is then cross-compiled and deployed to CPUs and/or a Tegra® board. Benchmarks show that performance of the auto-generated CUDA code is ~2.5x faster than MXNet, ~5x faster than Caffe2, ~7x faster than TensorFlow®, and on par with TensorRT™ implementation.

Rishu Gupta, Ph.D., Senior Application Engineer, MathWorks India

Addressing Deep Learning Challenges

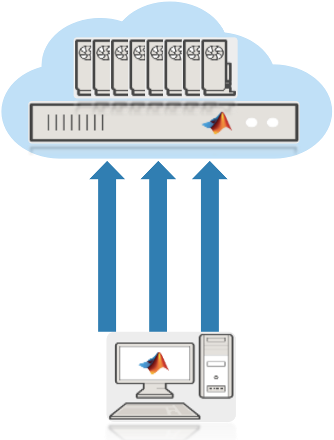
✓Perform deep learning without being an expert

AlexNet
PRETRAINED MODEL

✓Automate ground truth labeling

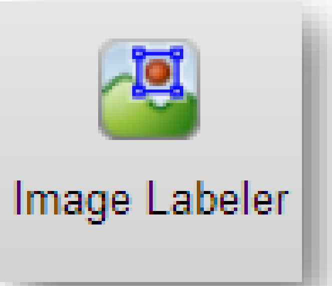
Inception-v3
Pretrained Model

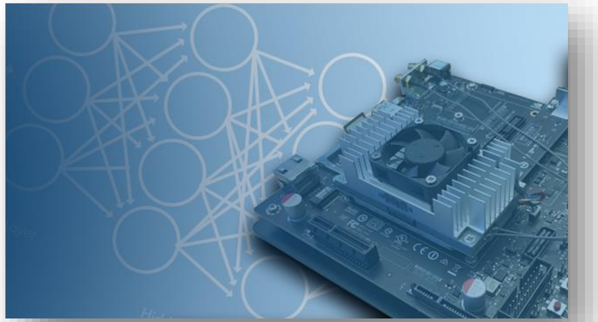
✓Create and visualize models with just a few lines of code



✓Seamless scale training to GPUs, clusters and cloud

✓Integrate & deploy deep learning in a single workflow





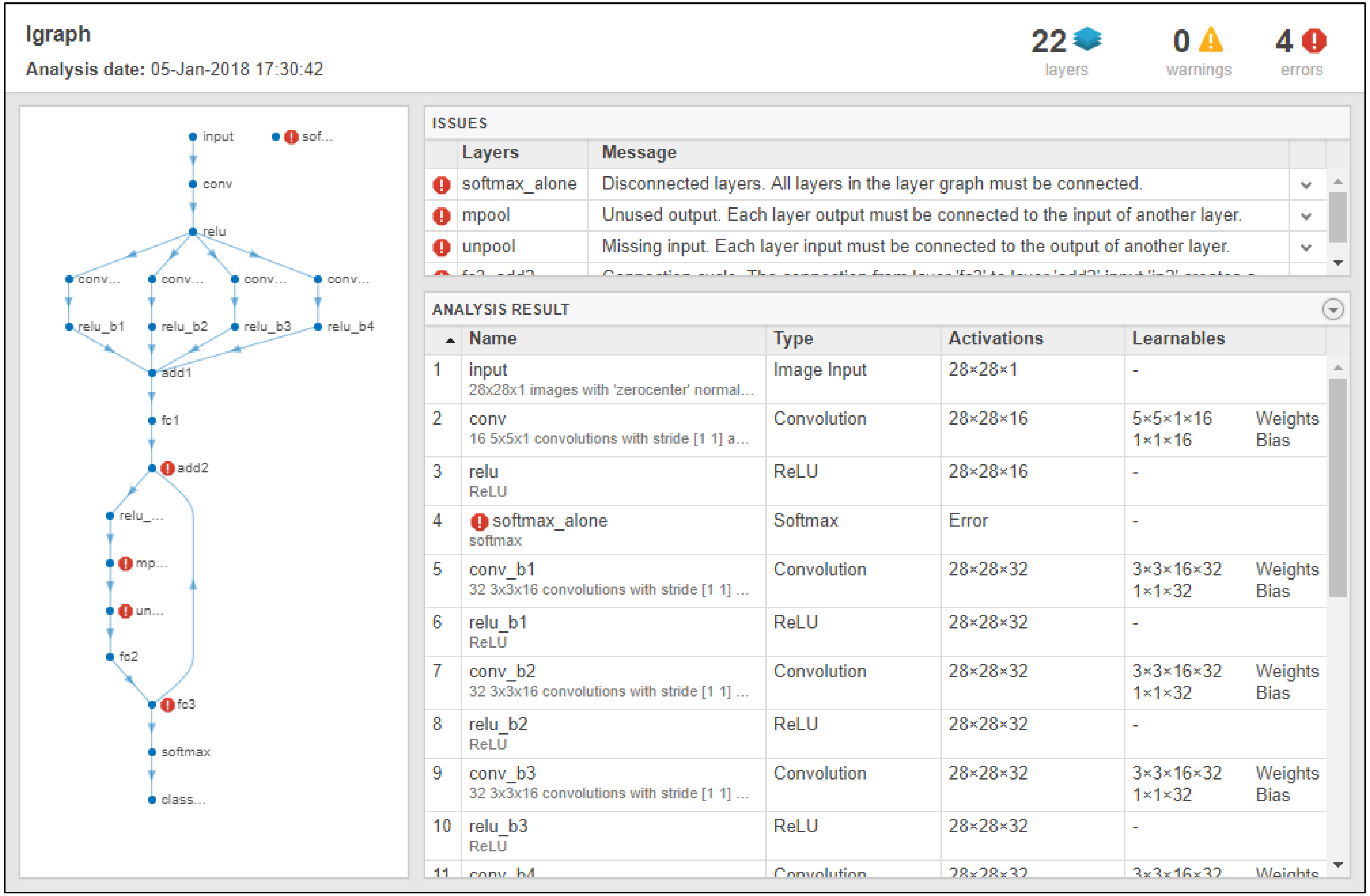
Framework Improvements

R2018a

- Architectures / layers
  - Regression LSTMs
  - Bidirectional LSTMs
  - Multi-spectral images
  - Custom layer validation
- Data pre-processing
  - Custom Mini-Batch Datastores
- Performance
  - CPU performance optimizations
  - Optimizations for zero learning-rate
- Network training
  - ADAM & RMSProp optimizers
  - Gradient clipping
  - Multi-GPU DAG network training
  - DAG network activations

Deep Learning Network Analyzer

R2018a



Creating Custom Architectures

MATLAB provides a simple programmatic interface to create layers and form a network

- **`addLayers`**
- **`removeLayers`**
- **`connectLayers`**
- **`disconnectLayers`**

… but are all these layers compatible?

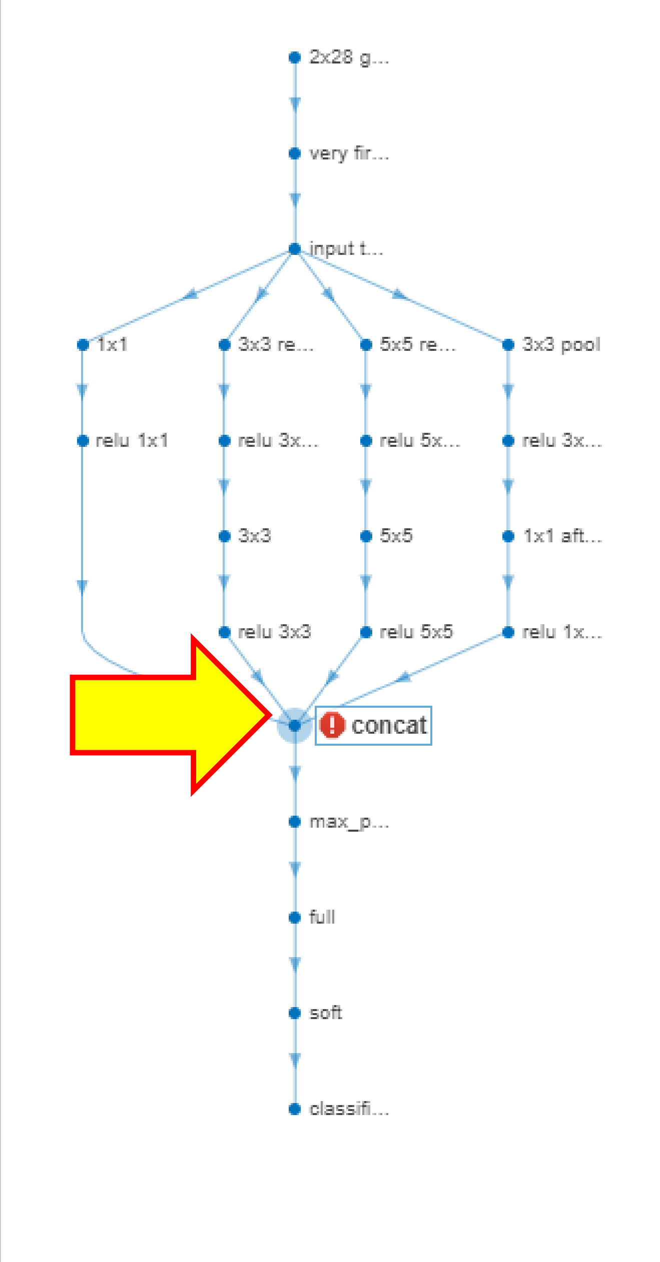
Create the layers

```
previous = reluLayer( 'Name', 'input to inception' );

oneByOne = [
    convolution2dLayer( 1, 64, 'Stride', 3, 'Name', '1x1' )
    reluLayer( 'Name', 'relu 1x1')
    ];

threeByThree = [
    convolution2dLayer( 1, 96, 'Name', '3x3 reduce' )
    reluLayer( 'Name', 'relu 3x3_reduce')
    convolution2dLayer( 3, 128, 'Stride', 3, 'Name', '3x3' )
    reluLayer( 'Name', 'relu 3x3')
    ];

fivebyFive = [
    convolution2dLayer( 1, 16, 'Name', '5x5 reduce' )
    reluLayer( 'Name', 'relu 5x5_reduce')
    convolution2dLayer( 5, 32, 'Stride', 3, 'Padding', 0, 'Name', '5x5' )
    reluLayer( 'Name', 'relu 5x5')
    ];

threeMaxPool = [
    maxPooling2dLayer( 3, 'Stride', 3, 'Name', '3x3 pool' )
    reluLayer( 'Name', 'relu 3x3_pool')
    convolution2dLayer( 1, 32, 'Name', '1x1 after pool' )
    reluLayer( 'Name', 'relu 1x1 after pool')
    ];

concat = depthConcatenationLayer( 4, 'Name', 'concat' );
```

myLayerGraph

Analysis date: 05-Jan-2018 17:20:01

22 layers | 0 warnings | 1 errors



ISSUES

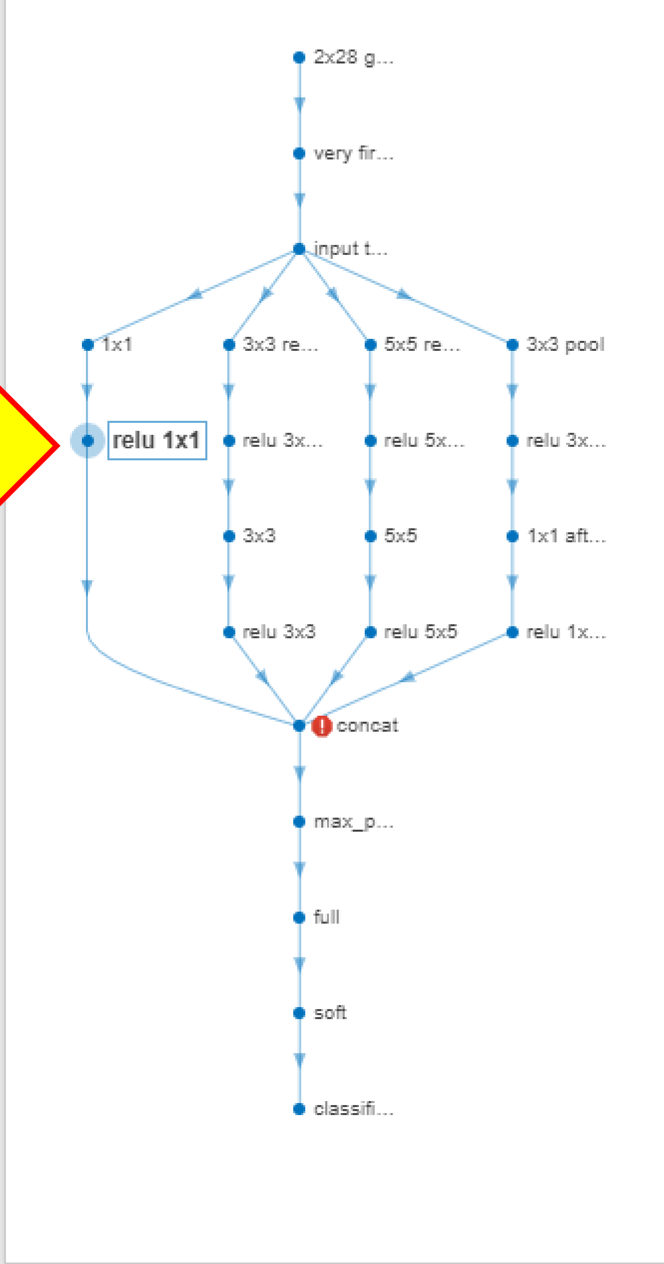
	Layers	Message
!	concat	Input size mismatch. Size of input to this layer is different from the expected input size.

ANALYSIS RESULT

	Name	Type	Activations	Learnables	
11	relu 5x5_reduce ReLU	ReLU	24×36×16	-	
12	5x5 32 5x5x16 convolutions with stride [3 3] and padding [0 0 0 0]	Convolution	7×11×32	5×5×16×32 1×1×32	Weights Bias
13	relu 5x5 ReLU	ReLU	7×11×32	-	
14	3x3 pool 3x3 max pooling with stride [3 3] and padding [0 0 0 0]	Max Pooling	8×12×20	-	
15	relu 3x3_pool ReLU	ReLU	8×12×20	-	
16	1x1 after pool 32 1x1x20 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	8×12×32	1×1×20×32 1×1×32	Weights Bias
17	relu 1x1 after pool ReLU	ReLU	8×12×32	-	
18	! concat Depth concatenation of 4 inputs	Depth concatenation	Error	-	
19	max_pool 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	Error	-	
20	full 10 fully connected layer	Fully Connected	1×1×1	Error 10×1	Weights Bias
21	soft softmax	Softmax	1×1×1	-	
22	classification crossentropyex	Classification Output	-	-	

myLayerGraph
Analysis date: 05-Jan-2018 17:20:01

22 layers | 0 warnings | 1 errors



ISSUES

	Layers	Message
!	concat	Input size mismatch. Size of input to this layer is different from the expected input size.

ANALYSIS RESULT

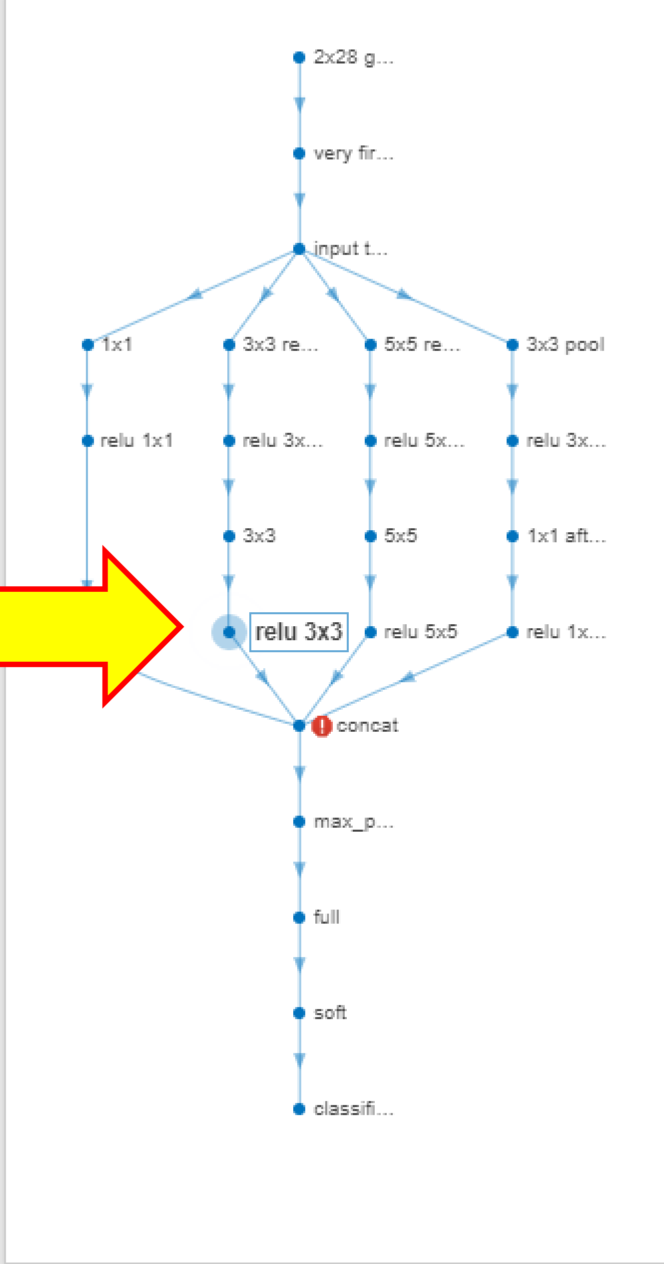
	Name	Type	Activations	Learnables	
1	2x28 grey 28x40x1 images with 'zerocenter' normalization	Image Input	28×40×1	-	
2	very first conv 20 5x5x1 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	24×36×20	5×5×1×20 1×1×20	Weights Bias
3	input to inception ReLU	ReLU	24×36×20	-	
4	1x1 64 1x1x20 convolutions with stride [3 3] and padding [0 0 0 0]	Convolution	8×12×64	1×1×20×64 1×1×64	Weights Bias
5	relu 1x1 ReLU	ReLU			
6	3x3 reduce 96 1x1x20 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	[illegible]	1×1×20×96 1×1×96	Weights Bias
7	relu 3x3_reduce ReLU	ReLU	24[illegible]	-	
8	3x3 128 3x3x96 convolutions with stride [3 3] and padding [0 0 0 0]	Convolution	8×[illegible]	3×3×96×128 1×1×128	Weights Bias
9	relu 3x3 ReLU	ReLU	8×12×128	-	
10	5x5 reduce 16 1x1x20 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	24×36×16	1×1×20×16 1×1×16	Weights Bias
11	relu 5x5_reduce ReLU	ReLU	24×36×16	-	
12	5x5	Convolution	7×11×32	5×5×16×32	Weights

8 x 12 x 64

myLayerGraph

Analysis date: 05-Jan-2018 17:20:01

22 layers | 0 warnings | 1 errors



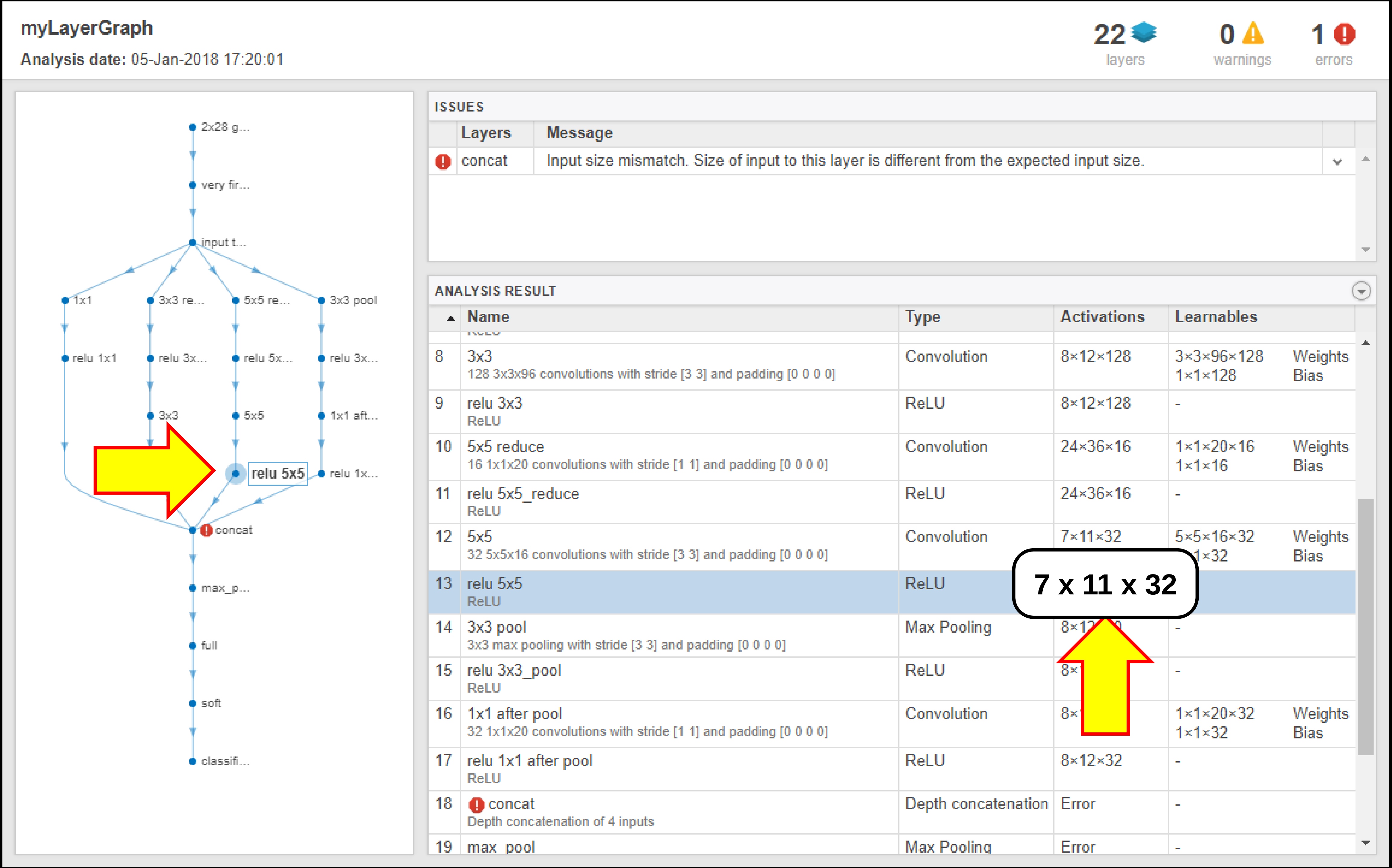
ISSUES

	Layers	Message
!	concat	Input size mismatch. Size of input to this layer is different from the expected input size.

ANALYSIS RESULT

	Name	Type	Activations	Learnables	
1	2x28 grey 28x40x1 images with 'zerocenter' normalization	Image Input	28×40×1	-	
2	very first conv 20 5x5x1 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	24×36×20	5×5×1×20 1×1×20	Weights Bias
3	input to inception ReLU	ReLU	24×36×20	-	
4	1x1 64 1x1x20 convolutions with stride [3 3] and padding [0 0 0 0]	Convolution	8×12×64	1×1×20×64 1×1×64	Weights Bias
5	relu 1x1 ReLU	ReLU	8×12×64	-	
6	3x3 reduce 96 1x1x20 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	24[illegible]	1×1×20×96 1×1×96	Weights Bias
7	relu 3x3_reduce ReLU	ReLU	24[illegible]	-	
8	3x3 128 3x3x96 convolutions with stride [3 3] and padding [0 0 0 0]	Convolution	8×[illegible]	3×3×96×128 1×1×128	Weights Bias
9	relu 3x3 ReLU	ReLU	[illegible]		
10	5x5 reduce 16 1x1x20 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	[illegible]	[illegible]×20×16 1×1×16	Weights Bias
11	relu 5x5_reduce ReLU	ReLU	24×36×16	-	
12	5x5	Convolution	7×11×32	5×5×16×32	Weights

8 x 12 x 128


myLayerGraph
Analysis date: 05-Jan-2018 17:20:01
22 layers
0 warnings
1 errors
2x28 g...
very fir...
input t...
1x1
3x3 re...
5x5 re...
3x3 pool
relu 1x1
relu 3x...
relu 5x...
relu 3x...
3x3
5x5
1x1 aft...
relu 5x5
relu 1x...
concat
max_p...
full
soft
classifi...
ISSUES
Layers
Message
concat
Input size mismatch. Size of input to this layer is different from the expected input size.
ANALYSIS RESULT
Name
Type
Activations
Learnables
8 3x3 Convolution 8×12×128 3×3×96×128 Weights 1×1×128 Bias
128 3x3x96 convolutions with stride [3 3] and padding [0 0 0 0]
9 relu 3x3 ReLU 8×12×128 -
ReLU
10 5x5 reduce Convolution 24×36×16 1×1×20×16 Weights 1×1×16 Bias
16 1x1x20 convolutions with stride [1 1] and padding [0 0 0 0]
11 relu 5x5_reduce ReLU 24×36×16 -
ReLU
12 5x5 Convolution 7×11×32 5×5×16×32 Weights Bias
32 5x5x16 convolutions with stride [3 3] and padding [0 0 0 0]
13 relu 5x5 ReLU
ReLU
7 x 11 x 32
14 3x3 pool Max Pooling -
3x3 max pooling with stride [3 3] and padding [0 0 0 0]
15 relu 3x3_pool ReLU -
ReLU
16 1x1 after pool Convolution 1×1×20×32 Weights 1×1×32 Bias
32 1x1x20 convolutions with stride [1 1] and padding [0 0 0 0]
17 relu 1x1 after pool ReLU 8×12×32 -
ReLU
18 concat Depth concatenation Error -
Depth concatenation of 4 inputs
19 max_pool Max Pooling Error -

Create the layers

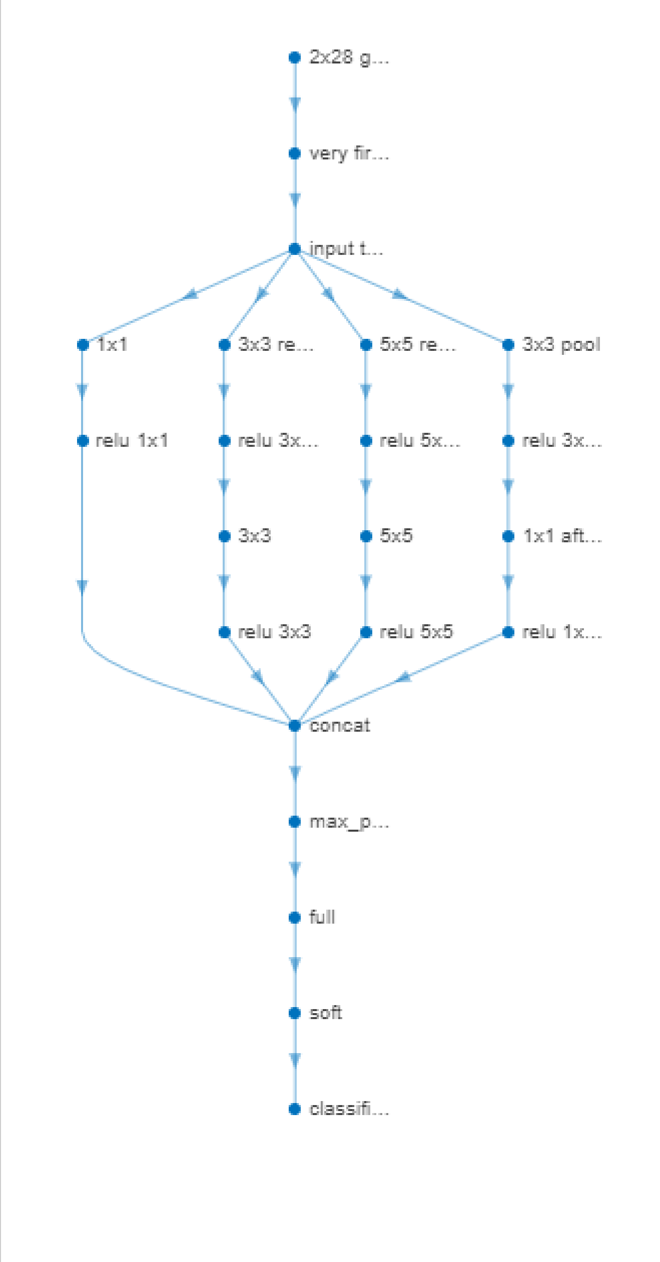
```
previous = reluLayer( 'Name', 'input to inception' );

oneByOne = [
    convolution2dLayer( 1, 64, 'Stride', 3, 'Name', '1x1' )
    reluLayer( 'Name', 'relu 1x1')
    ];

threeByThree = [
    convolution2dLayer( 1, 96, 'Name', '3x3 reduce' )
    reluLayer( 'Name', 'relu 3x3_reduce')
    convolution2dLayer( 3, 128, 'Stride', 3, 'Name', '3x3' )
    reluLayer( 'Name', 'relu 3x3')
    ];

fivebyFive = [
    convolution2dLayer( 1, 16, 'Name', '5x5 reduce' )
    reluLayer( 'Name', 'relu 5x5_reduce')
    convolution2dLayer( 5, 32, 'Stride', 3, 'Padding', 1, 'Name', '5x5' )
    reluLayer( 'Name', 'relu 5x5')
    ];

threeMaxPool = [
    maxPooling2dLayer( 3, 'Stride', 3, 'Name', '3    ol' )
    reluLayer( 'Name', 'relu 3x3_pool')
    convolution2dLayer( 1, 32, 'Name', '1x1 after pool' )
    reluLayer( 'Name', 'relu 1x1 after pool')
    ];

concat = depthConcatenationLayer( 4, 'Name', 'concat' );
```

Change the padding from zero to one

myLayerGraph

Analysis date: 05-Jan-2018 17:23:00

22 layers | 0 warnings | 0 errors



ANALYSIS RESULT

	Name	Type	Activations	Learnables	
1	2x28 grey 28x40x1 images with 'zerocenter' normalization	Image Input	28×40×1	-	
2	very first conv 20 5x5x1 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	24×36×20	5×5×1×20 1×1×20	Weights Bias
3	input to inception ReLU	ReLU	24×36×20	-	
4	1x1 64 1x1x20 convolutions with stride [3 3] and padding [0 0 0 0]	Convolution	8×12×64	1×1×20×64 1×1×64	Weights Bias
5	relu 1x1 ReLU	ReLU	8×12×64	-	
6	3x3 reduce 96 1x1x20 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	24×36×96	1×1×20×96 1×1×96	Weights Bias
7	relu 3x3_reduce ReLU	ReLU	24×36×96	-	
8	3x3 128 3x3x96 convolutions with stride [3 3] and padding [0 0 0 0]	Convolution	8×12×128	3×3×96×128 1×1×128	Weights Bias
9	relu 3x3 ReLU	ReLU	8×12×128	-	
10	5x5 reduce 16 1x1x20 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	24×36×16	1×1×20×16 1×1×16	Weights Bias
11	relu 5x5_reduce ReLU	ReLU	24×36×16	-	
12	5x5 32 5x5x16 convolutions with stride [3 3] and padding [1 1 1 1]	Convolution	8×12×32	5×5×16×32 1×1×32	Weights Bias
13	relu 5x5 ReLU	ReLU	8×12×32	-	
14	3x3 pool 3x3 max pooling with stride [3 3] and padding [0 0 0 0]	Max Pooling	8×12×20	-	
15	relu 3x3_pool ReLU	ReLU	8×12×20	-	
16	1x1 after pool 32 1x1x20 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	8×12×32	1×1×20×32 1×1×32	Weights Bias

Call to Action – Deep Learning Onramp

Free Introductory Course

Deep Learning Onramp

This free self-paced course provides an interactive introduction to practical deep learning. It focuses on using MATLAB® to apply deep learning methods to perform image recognition. The course consists of hands-on exercises and short videos. In the exercises, you will enter commands in an online version of MATLAB and receive contextual feedback that will help you correct common mistakes. Topics include:

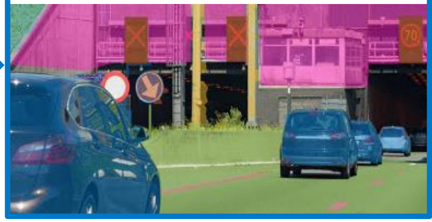
- Convolutional neural networks
- Preprocessing images
- Using pretrained networks
- Transfer learning
- Evaluating network performance

Available Here

MATLAB Deep Learning Framework

Access Data

Design + Train



Deploy

- **Manage** large image sets
- **Automate** image labeling

- **Easy access** to models
- **Acceleration** with GPU's
- **Scale** to GPUs & clusters

- **Optimized** inference deployment
- Target NVIDIA, Intel, ARM platforms

Speaker Details

Email: Amod.Anandkumar@mathworks.in

LinkedIn: https://in.linkedin.com/in/ajga2

Twitter: @_Dr_Amod

Contact MathWorks India

Products/Training Enquiry Booth

Call: 080-6632-6000

Email: info@mathworks.in

- **Share your experience with MATLAB & Simulink on Social Media**
  - Use #MATLABEXPO

- **Share your session feedback:**

  Please fill in your feedback for this session in the feedback form