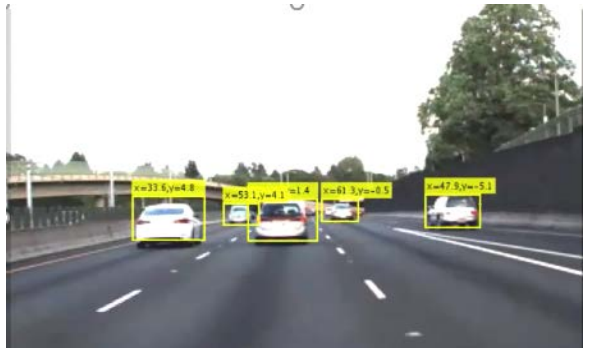
FROM DESKTOP TO CLOUD TO EMBEDDED GPUS

DESIGNING, TRAINING, AND COMPILING VISION AND DEEP LEARNING ALGORITHMS USING MATLAB

Avinash Nehemiah
Joss Knight
Girish Venkataramani

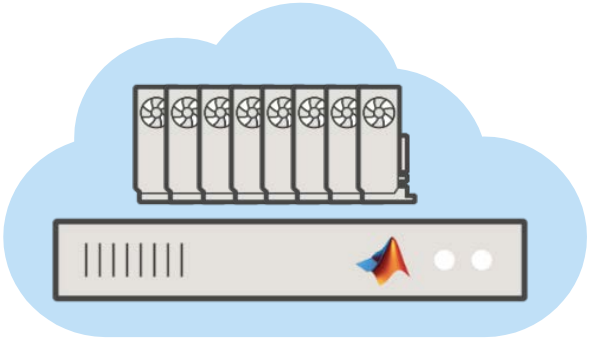
Talk Outline



Design Deep Learning & Vision Algorithms

Highlights

- **Manage** large image sets
- **Automate** image labeling
- **Easy access** to models
- **Pre-built training** frameworks



Accelerate and Scale Training

Highlights

- **Acceleration** with GPU's
- **Scale** to clusters

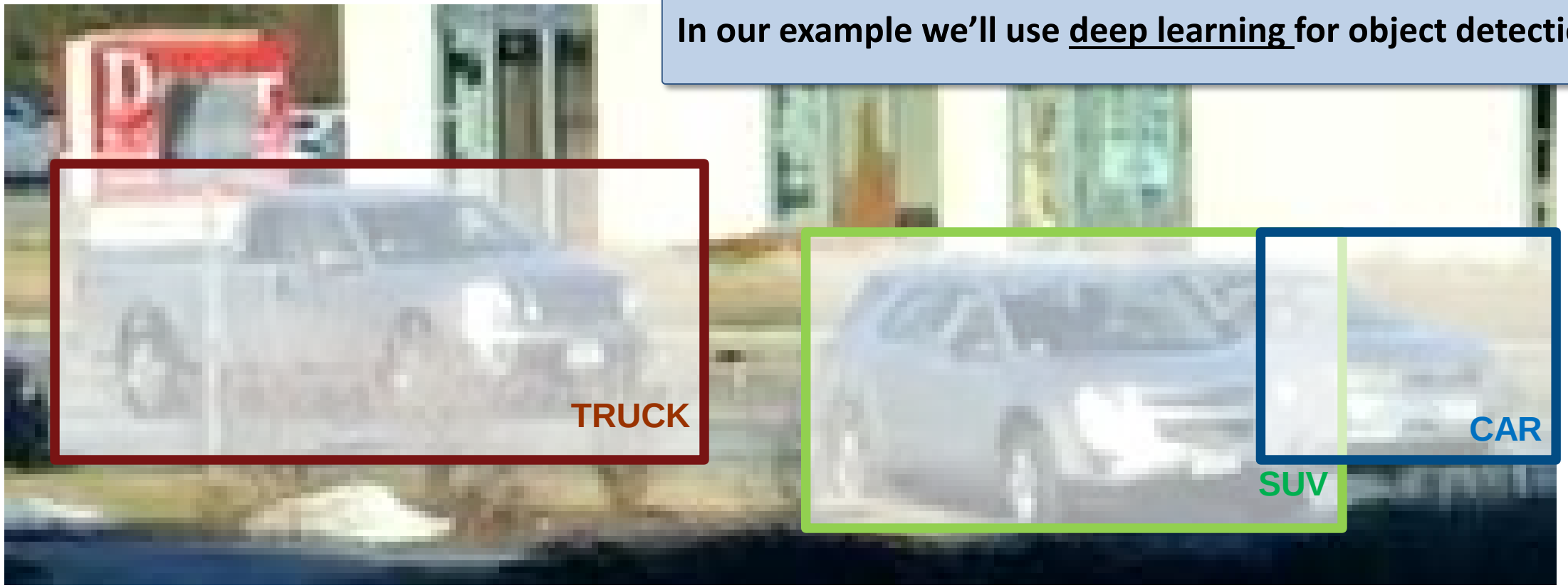
High Performance Embedded Implementation

Highlights

- **Automate compilation** of MATLAB to CUDA
- **14x speedup** over Caffe & **4x speedup** over TensorFlow

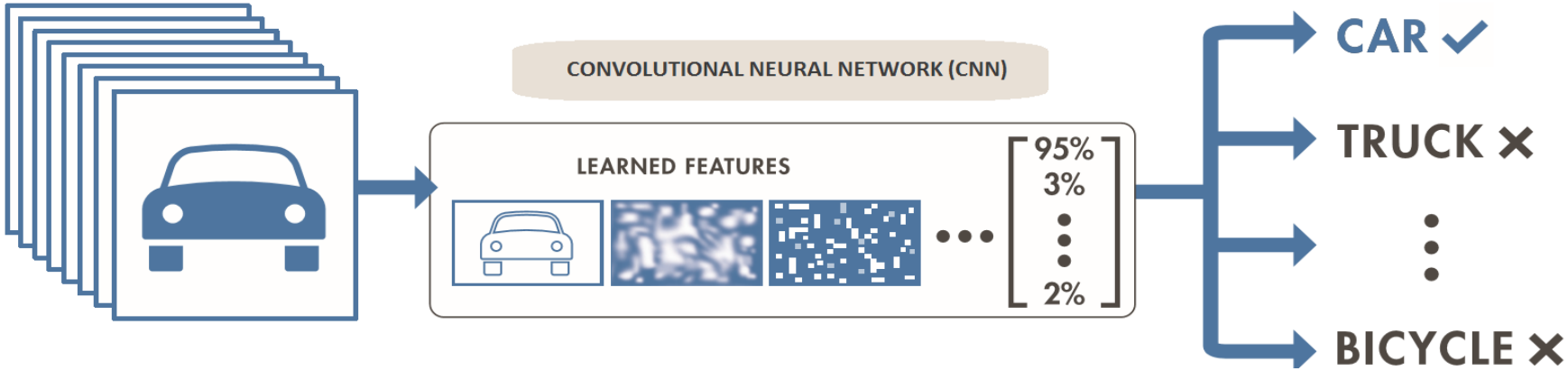
Let’s Use Object Detection as an Example

In our example we’ll use <u>deep learning</u> for object detection.

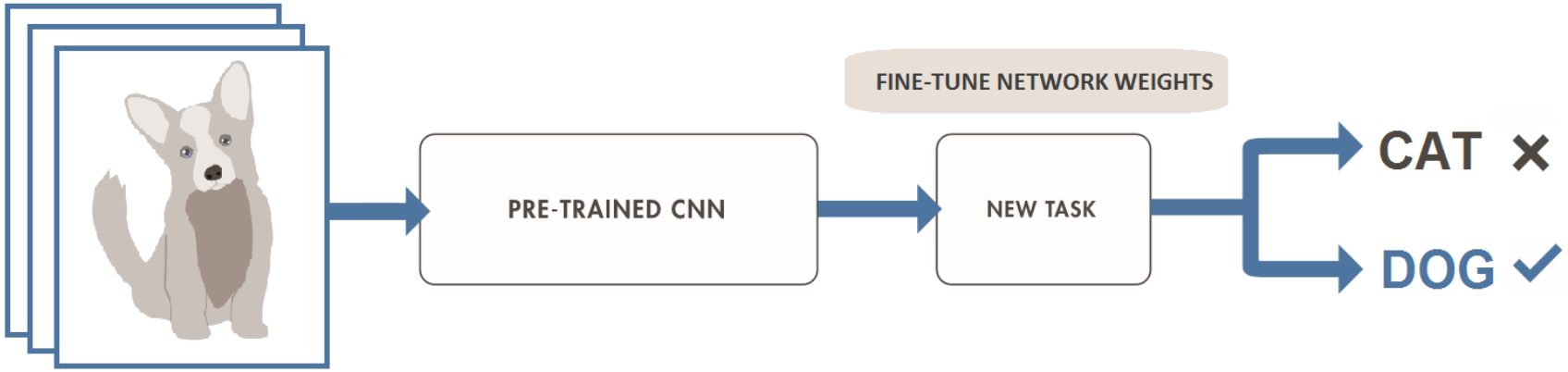


Two Approaches for Deep Learning

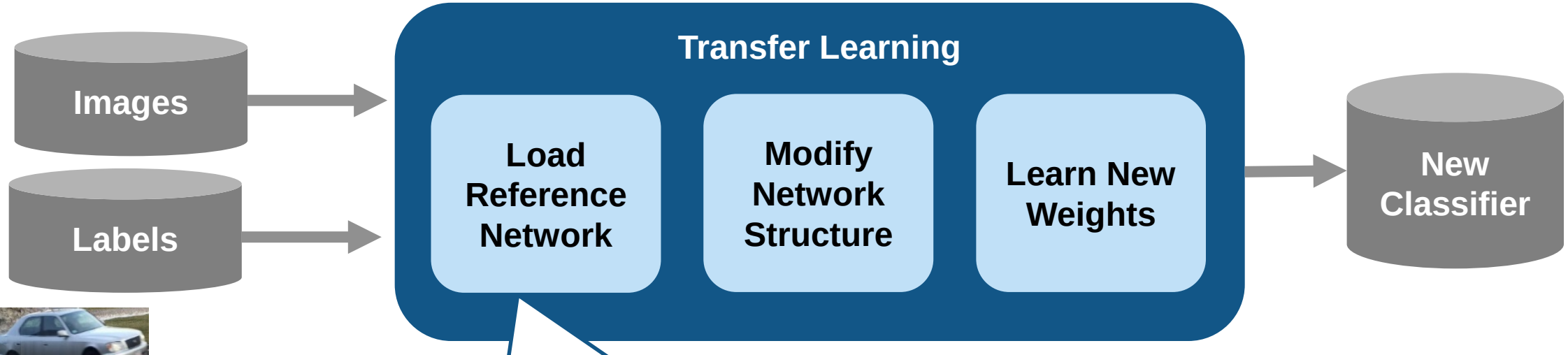
1. Train a Deep Neural Network from Scratch



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



Transfer Learning Workflow

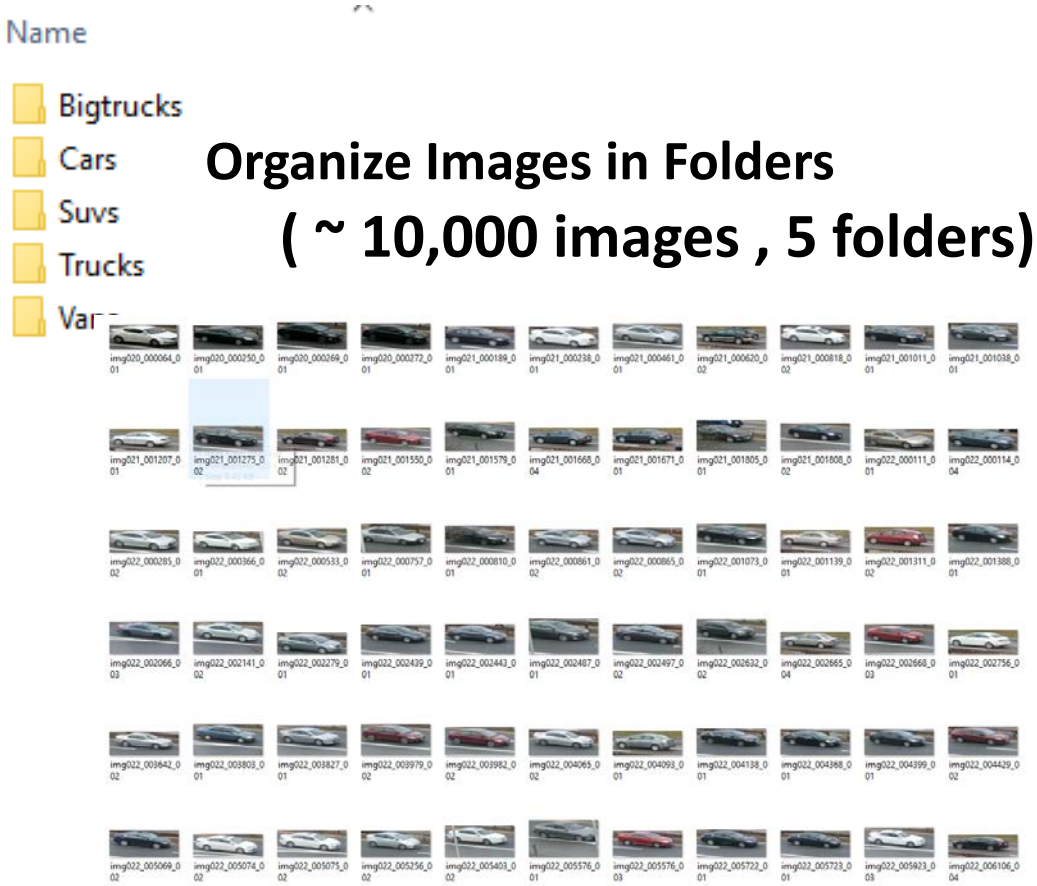


Training Data

Labels: Car, Truck, Large Truck, SUV, Van

Alexnet, VGG-16, VGG-19, GoogLeNet

Manage Large Sets of Images



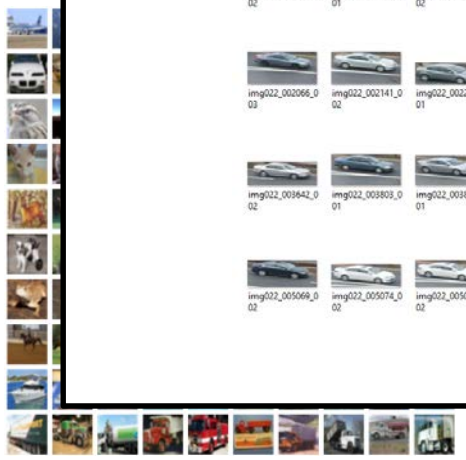
Organize Images in Folders

(~ 10,000 images , 5 folders)

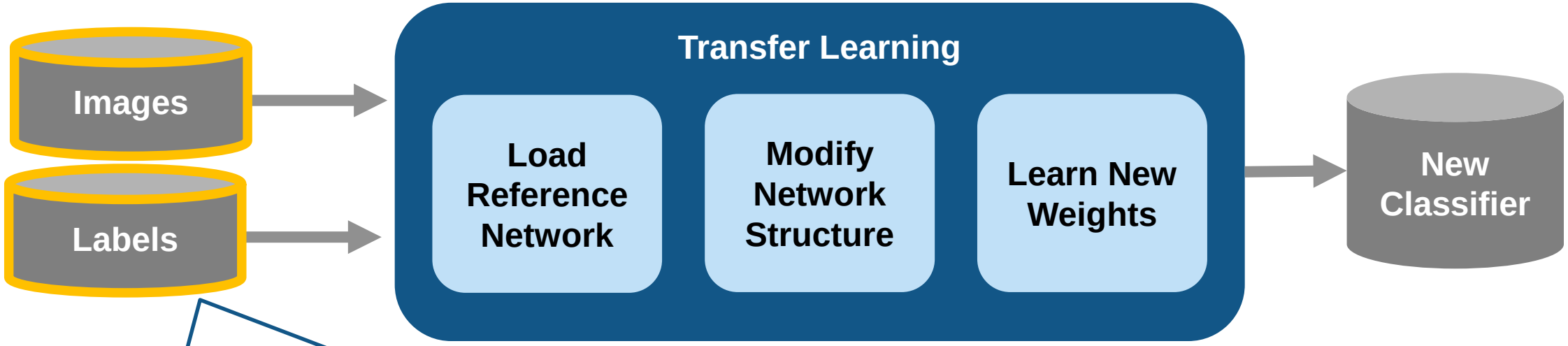
```
imageData = imageDataStore('vehicles')
```

Easily manage large sets of images

- **Single line of code to access images**
- **Operates on disk, database, big-data file system**

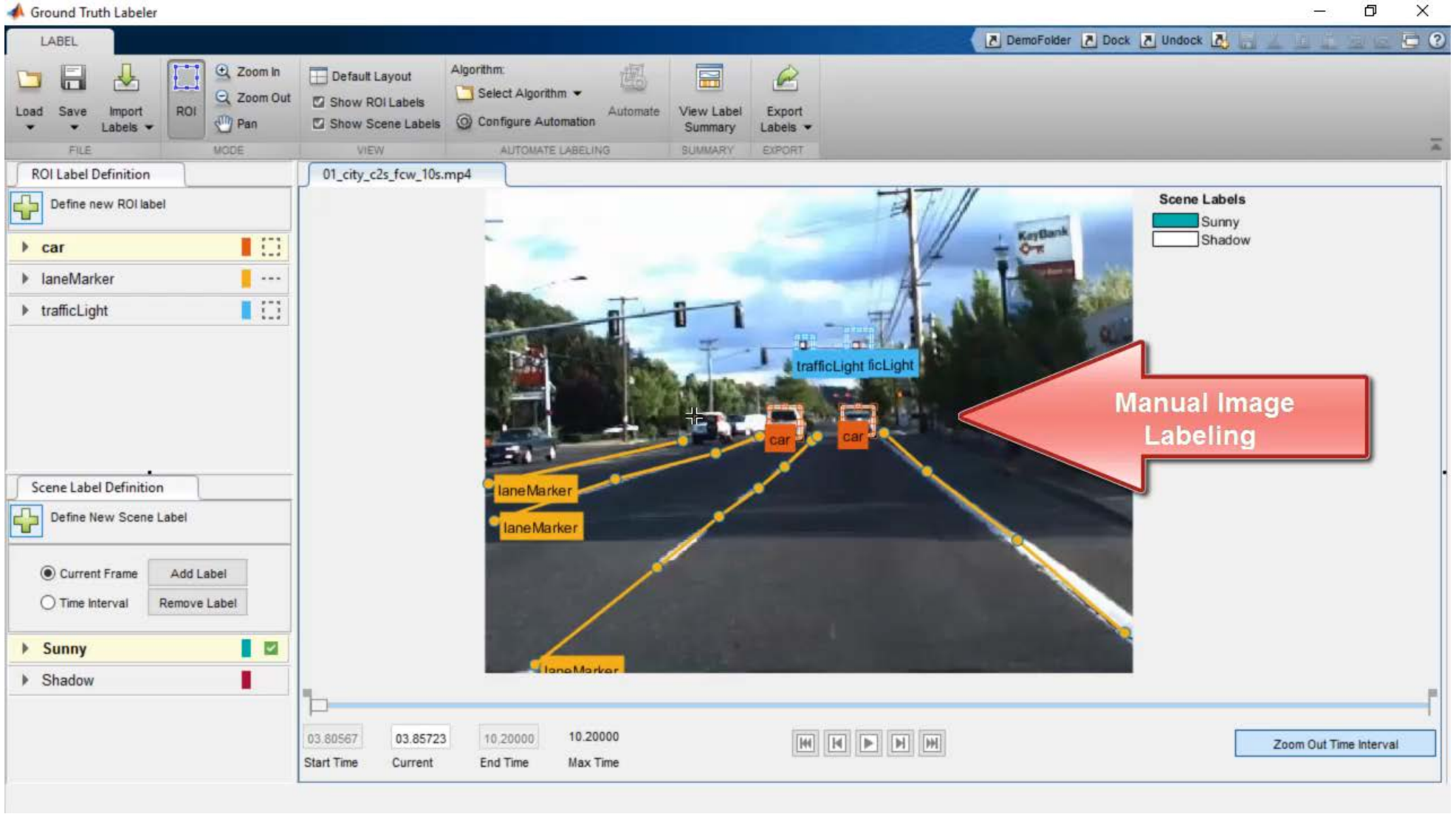


Automate Ground Truth Labeling

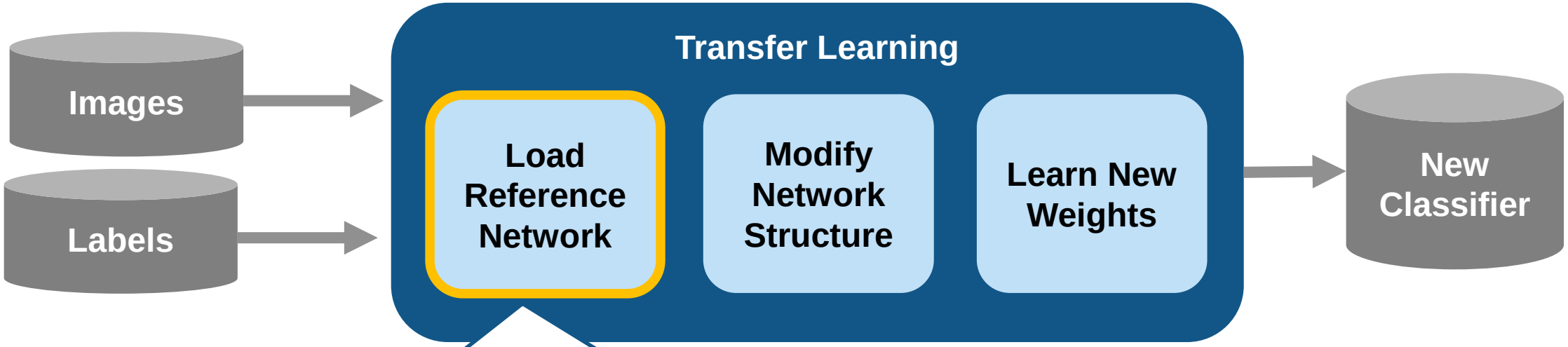


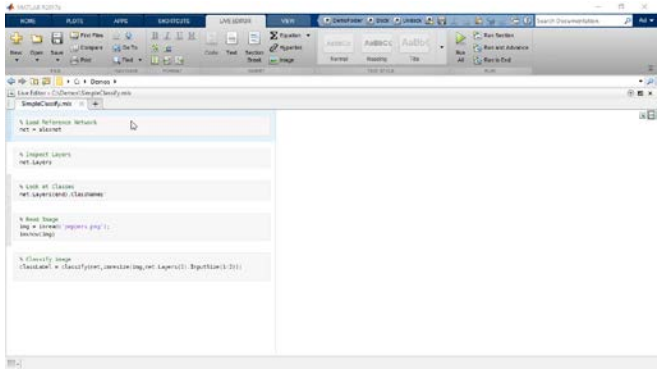
Ground Truth Labeling

Automate Ground Truth Labeling



Access Reference Models in MATLAB



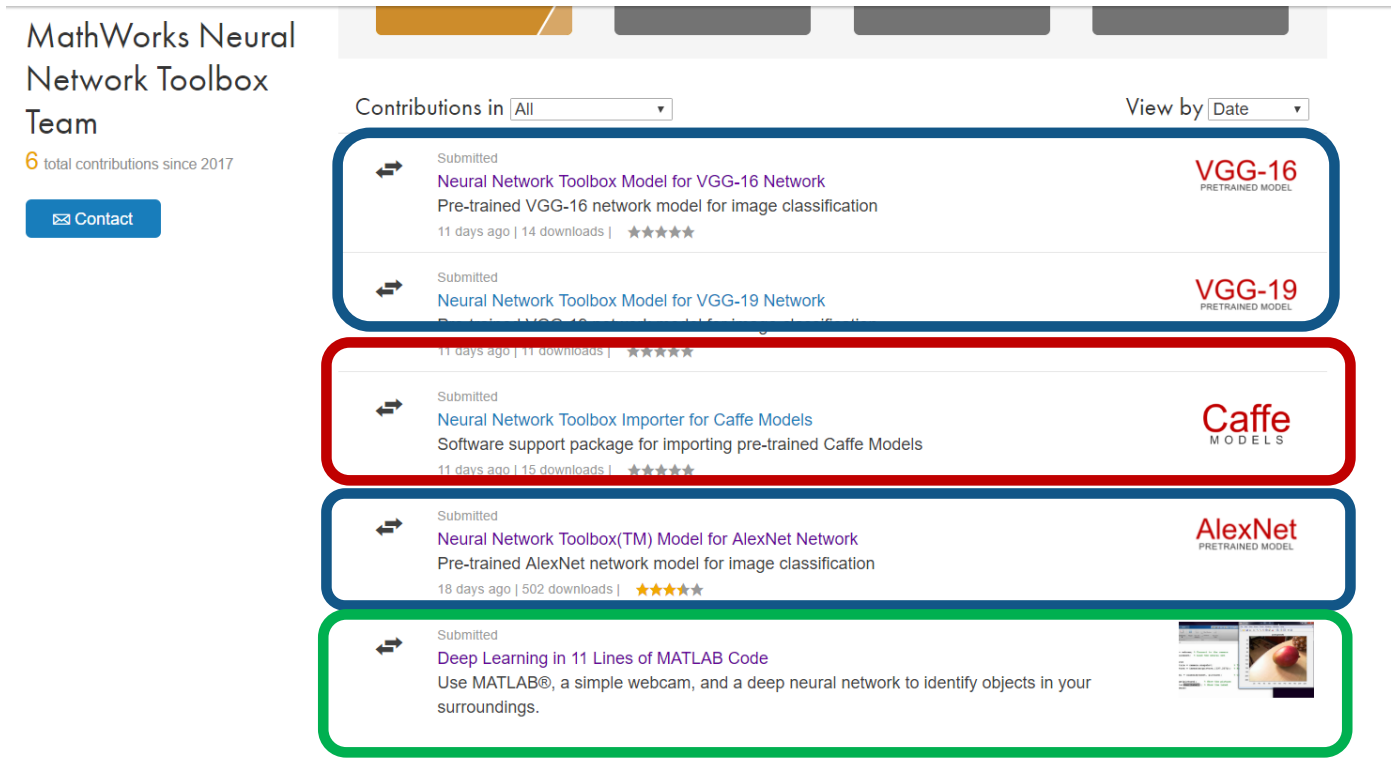


Easily Load Reference Networks

Access Models with 1-line of MATLAB Code

```
Net1 = alexnet
Net2 = vgg16
Net3 = vgg19
```

Access Reference Models in MATLAB

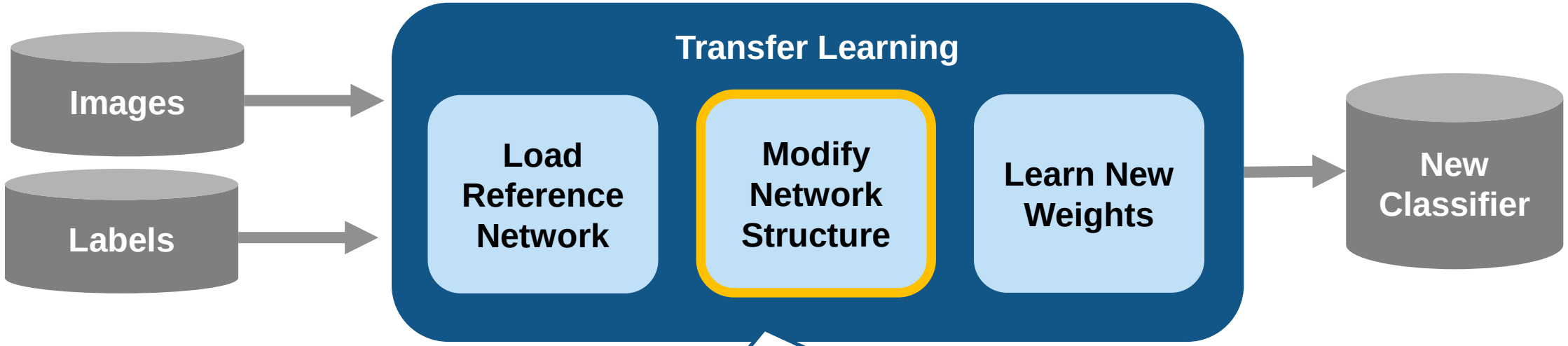


1. Reference Models

2. Model Importer

3. Tutorials

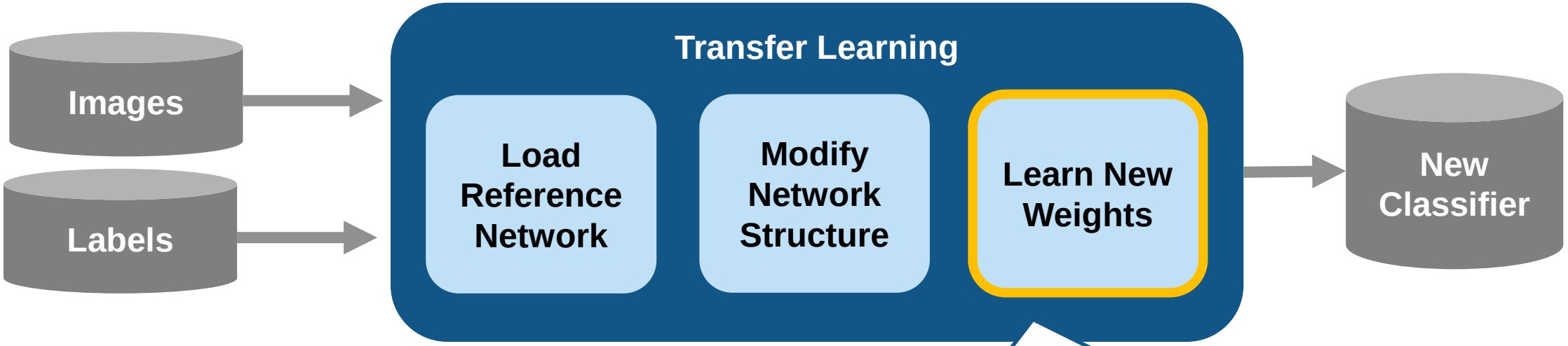
Modify Network Structure



Simple MATLAB API to modify layers:

```
layers(23) = fullyConnectedLayer(5, 'Name','fc8');
layers(25) = classificationLayer('Name',‘VehicleClassifier')
```

Training Object Detectors



Train Any Network

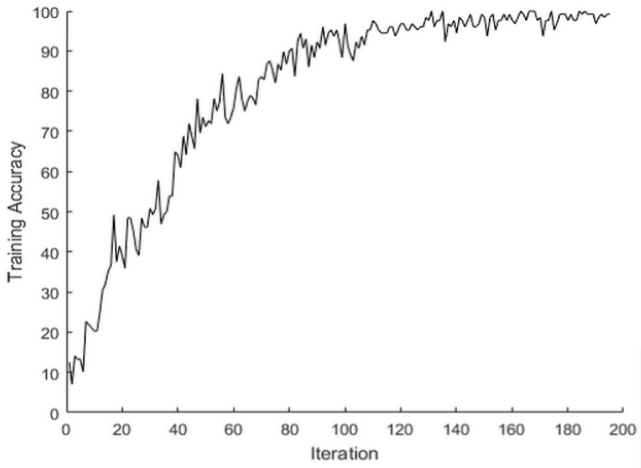
```
trainNetwork(datastore, layers, options)
```

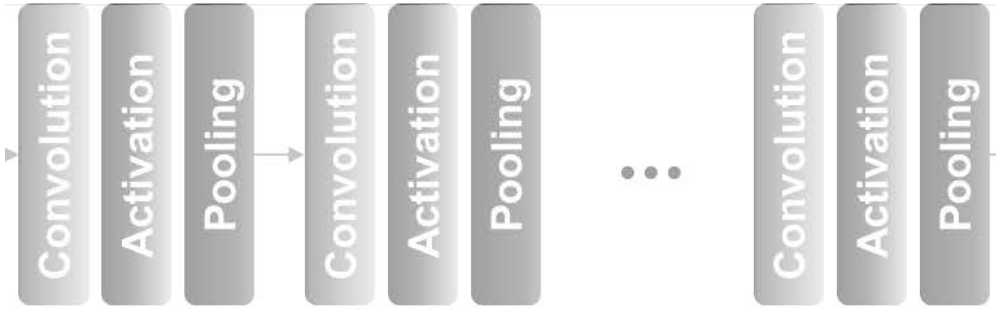
Frameworks for Computer Vision

- **Deep Learning:** R-CNN, Fast R-CNN, Faster R-CNN
- **Machine Learning:** ACF, Cascade Object Detectors

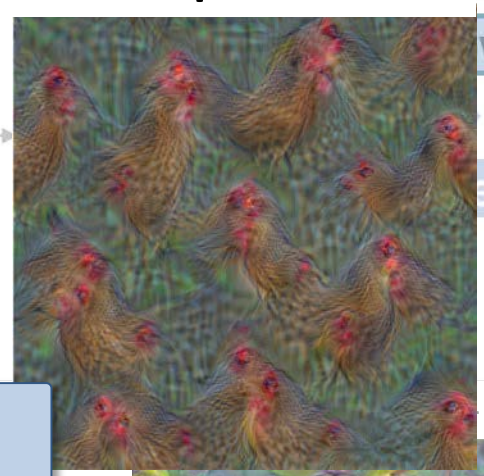
Visualizing and Debugging Intermediate Results

Training Accuracy Visualization



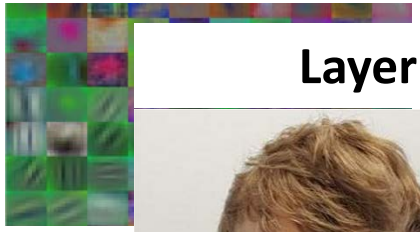


Deep Dream

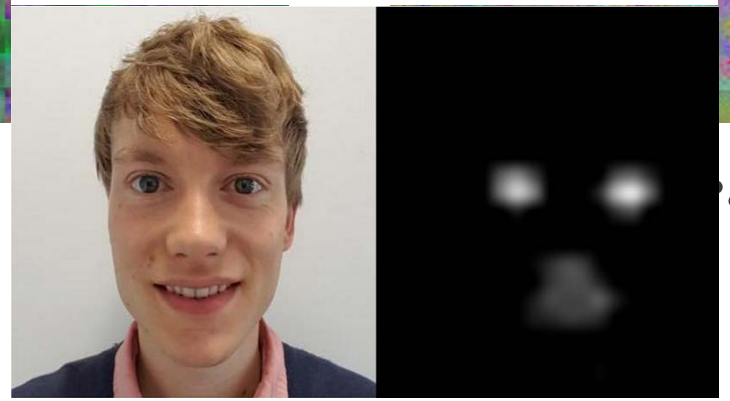


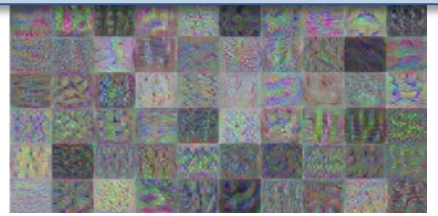
- Many options for visualizations and debugging
- Examples to get started

Filters

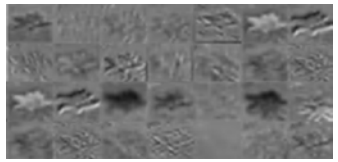


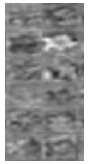
Layer Activations





Activations





Feature Visualization

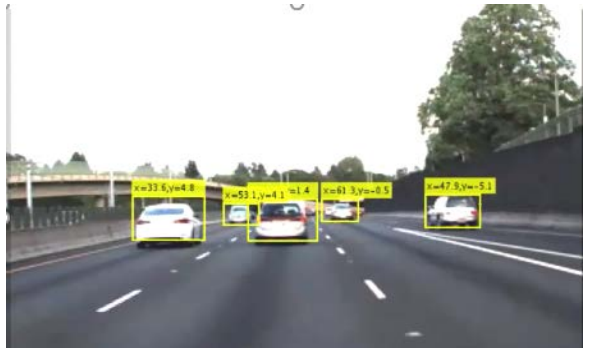
Real World Systems Use More Than Deep Learning

Deep learning vehicle detector performance degraded with environmental effects (fog etc.)

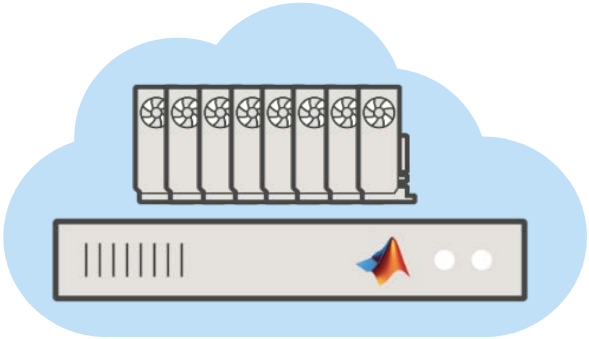
Challenge: Deep learning frameworks do not include “classical” computer vision

Solution: Convert MATLAB code with deep learning and computer vision to embedded implementation

Talk Outline



Design Deep Learning & Vision Algorithms



Accelerate and Scale Training

High Performance Embedded Implementation

- Can you solve “real” problems for production systems with MATLAB ?
- Doesn’t it take hours or days to train ?

Problems of acceleration and scale

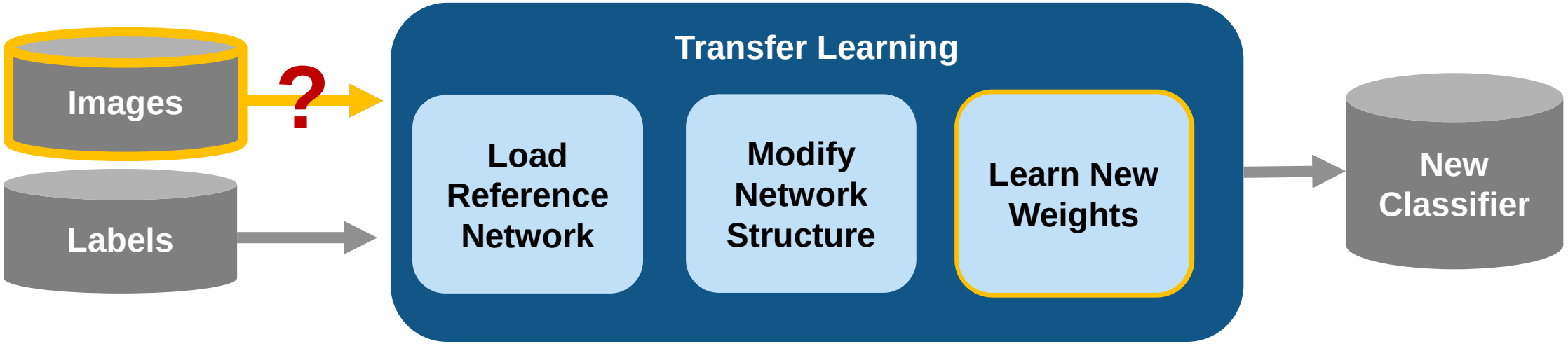
- How can I make my code run faster?
- How can I scale up to bigger problems?

- Will I have to learn new tools?
- Will I have to learn new concepts?

MATLAB and Parallel Computing

- ***Accelerate*** your code on the GPU
  - DEMO: Preprocess your image dataset
- ***Scale*** to multi-GPU and clusters
  - DEMO: Parallel training with an Amazon P2 cluster

Transfer Learning with MATLAB



HOME PLOTS APPS SHORTCUTS EDITOR PUBLISH VIEW Search Documentation Joss

F: ▸ public ▸ Joss_Knight ▸ Presentations ▸ GTC 2017 ▸ Demo1_Preprocess

Editor - F:\public\Joss_Knight\Presentations\GTC 2017\Demo1_Preprocess\LoadAndPreprocessImages.m

LoadAndPreprocessImages.m | preprocessImage.m

This file can be opened as a Live Script. For more information, see Creating Live Scripts.

```
%%
miniBatchSize = 32;
videoReader.CurrentTime = 30;
miniBatch = zeros([imageSize miniBatchSize], 'single' );
framenum = 1;
totalTime = 0;
while hasFrame(videoReader) && framenum <= miniBatchSize
    % Read from video
    I = readFrame(videoReader);

    % Filter, crop, and add to mini-batch
    tic;

    I = preprocessImage(I);

    miniBatch(:,:,:,framenum) = I;

    totalTime = totalTime + toc;

    % Display
    step(videoPlayer, gather(I));
    framenum = framenum + 1;
end
fprintf('Time to create a mini-batch = %.2f secs\n', totalTime);
```

script Ln 37 Col 1

HOME PLOTS APPS SHORTCUTS EDITOR PUBLISH VIEW

Search Documentation

Joss

F: ▸ public ▸ Joss_Knight ▸ Presentations ▸ GTC 2017 ▸ Demo1_Preprocess

Editor - F:\public\Joss_Knight\Presentations\GTC 2017\Demo1_Preprocess\preprocessImage.m

LoadAndPreprocessImages.m | preprocessImage.m

```
function I = preprocessImage(I)
% Example image processing for a classification dataset

% Convert to floating point for accuracy
I = im2single(I);

% Improve contrast using histogram equalization in HSV-space
I = rgb2hsv(I);
I = histeq(I);
I = hsv2rgb(I);

% Enhance horizontal and vertical edges using a Prewitt filter
Ih = imfilter(I, fspecial('prewitt'));
Iv = imfilter(I, fspecial('prewitt')');
I = 3*I + Ih + Iv;

% Renormalize to prevent saturation
I = I/max(I(:));

% Resize and crop to region of interest
I = imresize(I,[720 1280]);
I = imcrop(I, [350 240 700 330]);

end
```

preprocessImage Ln 24 Col 4

HOME PLOTS APPS SHORTCUTS EDITOR PUBLISH VIEW Search Documentation Joss

F: ▸ public ▸ Joss_Knight ▸ Presentations ▸ GTC 2017 ▸ Demo1_Preprocess

Editor - F:\public\Joss_Knight\Presentations\GTC 2017\Demo1_Preprocess\LoadAndPreprocessImages.m

LoadAndPreprocessImages.m | preprocessImage.m

This file can be opened as a Live Script. For more information, see Creating Live Scripts.

```
%%
miniBatchSize = 32;
videoReader.CurrentTime = 30;
miniBatch = zeros([imageSize miniBatchSize], 'single' );
framenum = 1;
totalTime = 0;
while hasFrame(videoReader) && framenum <= miniBatchSize
    % Read from video
    I = readFrame(videoReader);

    % Filter, crop, and add to mini-batch
    tic;

    I = preprocessImage(I);

    miniBatch(:,:,:,framenum) = I;

    totalTime = totalTime + toc;

    % Display
    step(videoPlayer, gather(I));
    framenum = framenum + 1;
end
fprintf('Time to create a mini-batch = %.2f secs\n', totalTime);
```

script Ln 11 Col 1

HOME | PLOTS | APPS | SHORTCUTS | EDITOR | PUBLISH | VIEW

Search Documentation | Joss

F: ▸ public ▸ Joss_Knight ▸ Presentations ▸ GTC 2017 ▸ Demo1_Preprocess

Current Folder | Command History | Scopes - Video Player

Video Player

Processing | RGB:331x701 | 32

Workspace

Name	Value
framenum	33
I	331x701x3 single
imageSize	[331,701,3]
miniBatch	4-D single
miniBatchSize	32
totalTime	8.1672
videoFile	'../holdForTest...
videoPlayer	1x1 VideoPlayer
videoReader	1x1 VideoReader

Editor - F:\public\Joss_Knight\Presentations\GTC 2017\Demo1_Preprocess\LoadAndPreprocessImages.m

LoadAndPreprocessImages.m | preprocessImage.m

This file can be opened as a Live Script. For more information, see Creating Live Scripts.

```
10
11      %%
12      miniBatchSize = 32;
13      videoReader.CurrentTime = 30;
14      miniBatch = zeros([imageSize miniBatchSize], 'single' );
15      framenum = 1;
16      totalTime = 0;
17      while hasFrame(videoReader) && framenum <= miniBatchSize
18          % Read from video
19          I = readFrame(videoReader);
20
21          % Filter, crop, and add to mini-batch
22          tic;
23          I = gpuArray(I);
24          I = preprocessImage(I);
25          I = gather(I);
26          miniBatch(:,:,:,framenum) = I;
27
```

Command Window

New to MATLAB? See resources for Getting Started.

```
Time to create a mini-batch = 8.17 secs
fx >>
```

script | Ln 25 | Col 19

HOME PLOTS APPS SHORTCUTS EDITOR PUBLISH VIEW

Search Documentation

Joss

F: ▸ public ▸ Joss_Knight ▸ Presentations ▸ GTC 2017 ▸ Demo1_Preprocess

Editor - F:\public\Joss_Knight\Presentations\GTC 2017\Demo1_Preprocess\preprocessImage.m

LoadAndPreprocessImages.m | preprocessImage.m

```
function I = preprocessImage(I)
% Example image processing for a classification dataset

% Convert to floating point for accuracy
I = im2single(I);

% Improve contrast using histogram equalization in HSV-space
I = rgb2hsv(I);
I = histeq(I);
I = hsv2rgb(I);

% Enhance horizontal and vertical edges using a Prewitt filter
Ih = imfilter(I, fspecial('prewitt'));
Iv = imfilter(I, fspecial('prewitt')');
I = 3*I + Ih + Iv;

% Renormalize to prevent saturation
I = I/max(I(:));

% Resize and crop to region of interest
I = imresize(I,[720 1280]);
I = imcrop(I, [350 240 700 330]);

end
```

preprocessImage | Ln 24 | Col 4

HOME | PLOTS | APPS | SHORTCUTS | EDITOR | PUBLISH | VIEW

Search Documentation | Joss

F: ▸ public ▸ Joss_Knight ▸ Presentations ▸ GTC 2017 ▸ Demo1_Preprocess

Current Folder | Command History | Scopes - Video Player

Video Player

Processing | RGB:331x701 | 32

Workspace

Name	Value
framenum	33
I	331x701x3 single
imageSize	[331,701,3]
miniBatch	4-D single
miniBatchSize	32
totalTime	8.1672
videoFile	'../holdForTest...
videoPlayer	1x1 VideoPlayer
videoReader	1x1 VideoReader

Editor - F:\public\Joss_Knight\Presentations\GTC 2017\Demo1_Preprocess\LoadAndPreprocessImages.m

LoadAndPreprocessImages.m | preprocessImage.m

This file can be opened as a Live Script. For more information, see Creating Live Scripts.

```
10
11      %%
12 -    miniBatchSize = 32;
13 -    videoReader.CurrentTime = 30;
14 -    miniBatch = zeros([imageSize miniBatchSize], 'single' );
15 -    framenum = 1;
16 -    totalTime = 0;
17 -    while hasFrame(videoReader) && framenum <= miniBatchSize
18          % Read from video
19 -        I = readFrame(videoReader);
20
21          % Filter, crop, and add to mini-batch
22 -        tic;
23 -        I = gpuArray(I);
24 -        I = preprocessImage(I);
25 -        I = gather(I);
26 -        miniBatch(:,:,:,framenum) = I;
27
```

Command Window

New to MATLAB? See resources for Getting Started.

```
Time to create a mini-batch = 8.17 secs
fx >>
```

script | Ln 25 | Col 19

HOME | PLOTS | APPS | SHORTCUTS | EDITOR | PUBLISH | VIEW

Search Documentation | Joss

F: ▸ public ▸ Joss_Knight ▸ Presentations ▸ GTC 2017 ▸ Demo1_Preprocess

Current Folder | Command History | Scopes - Video Player

Video Player

Processing | RGB:331x701 | 64

Workspace

Name	Value
framenum	33
I	331x701x3 single
imageSize	[331,701,3]
miniBatch	4-D single
miniBatchSize	32
totalTime	2.6239
videoFile	'../holdForTest...
videoPlayer	1x1 VideoPlayer
videoReader	1x1 VideoReader

Editor - F:\public\Joss_Knight\Presentations\GTC 2017\Demo1_Preprocess\LoadAndPreprocessImages.m

LoadAndPreprocessImages.m | preprocessImage.m

This file can be opened as a Live Script. For more information, see Creating Live Scripts.

```
%%
miniBatchSize = 32;
videoReader.CurrentTime = 30;
miniBatch = zeros([imageSize miniBatchSize], 'single', 'gpuArray');
framenum = 1;
totalTime = 0;
while hasFrame(videoReader) && framenum <= miniBatchSize
    % Read from video
    I = readFrame(videoReader);

    % Filter, crop, and add to mini-batch
    tic;
    I = gpuArray(I);
    I = preprocessImage(I);

    miniBatch(:,:,:,framenum) = I;
```

Command Window

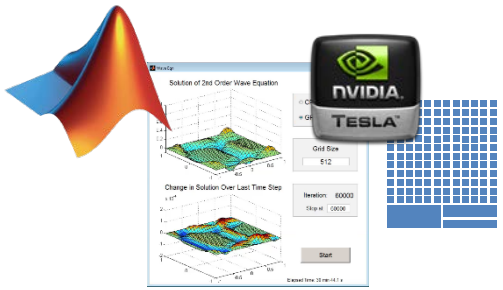
New to MATLAB? See resources for Getting Started.

```
Time to create a mini-batch = 8.17 secs
Time to create a mini-batch = 2.62 secs
>>
```

script | Ln 25 | Col 5

Built-in function support

Parallel Computing



Over 300 core MATLAB functions optimized for GPU

- Elementary math
- Linear algebra
- FFTs and IFFTs
- Convolution and filtering
- Fitting and interpolation
- Reductions and sorting
- Sparse matrix support
- double, single and integer support

Neural Networks

Deep Learning, Neural Network training and simulation

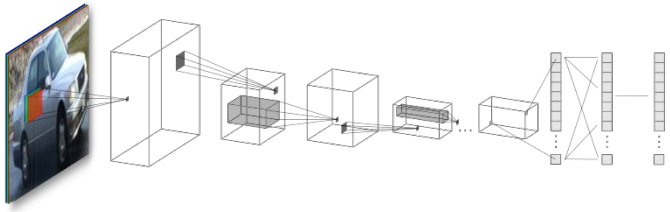
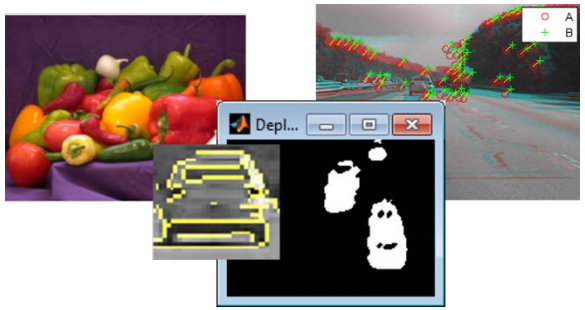


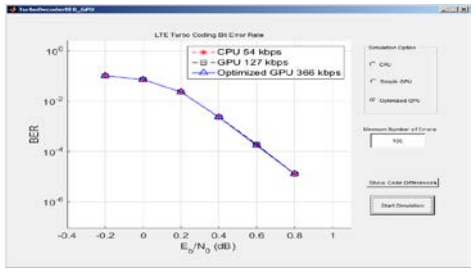
Image Processing and Computer Vision

Feature detection, transformations, filtering, object analysis



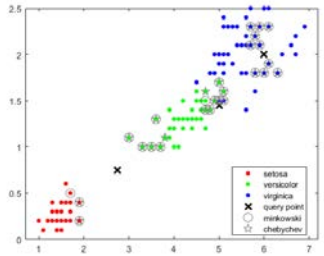
Signal Processing and Communications

FFT filtering, cross correlation, BER simulations



Statistics and Machine Learning

Distributions, hypothesis testing, k-means clustering, nearest neighbour



Programming with GPUs

- GPU-optimized functions
- Simple programming constructs
  - `gpuArray, gather`
- Writing kernels in the MATLAB language
  - `arrayfun`
- Interface with your own CUDA C and C++ code
  - `CUDAKernel, mexcuda`

Prototyping

Test Framework

```
/** The actual Mandelbrot algorithm for a single location */
__device__ unsigned int doIterations( double const realPart0,
                                      double const imagPart0,
                                      unsigned int const maxIters ) {
    // Initialise: z = z0
    double realPart = realPart0;
    double imagPart = imagPart0;
    unsigned int count = 0;
    // Loop until escape
    while ( ( count <= maxIters )
            && ((realPart*realPart + imagPart*imagPart) <= 4.0) ) {
        ++count;
        // Update: z = z*z + z0;
        double const oldRealPart = realPart;
        realPart = realPart*realPart - imagPart*imagPart + realPart0;
        imagPart = 2.0*oldRealPart*imagPart + imagPart0;
    }
    return count;
}
```

```
% Load the kernel
cudaFilename = 'mandelbrotElement.cu';
ptxFilename = ['mandelbrotElement.',parallel.gpu.ptxext];
kernel = parallel.gpu.CUDAKernel( ptxFilename, cudaFilename );

% Make sure we have sufficient blocks to cover all of the locations
numElements = numel( xGrid );
kernel.ThreadBlockSize = [kernel.MaxThreadsPerBlock,1,1];
kernel.GridSize = [ceil(numElements/kernel.MaxThreadsPerBlock),1];

% Call the kernel
count = zeros( size(xGrid), 'gpuArray' );
count = feval( kernel, count, xGrid, yGrid, ...
    maxIterations, numElements );
```

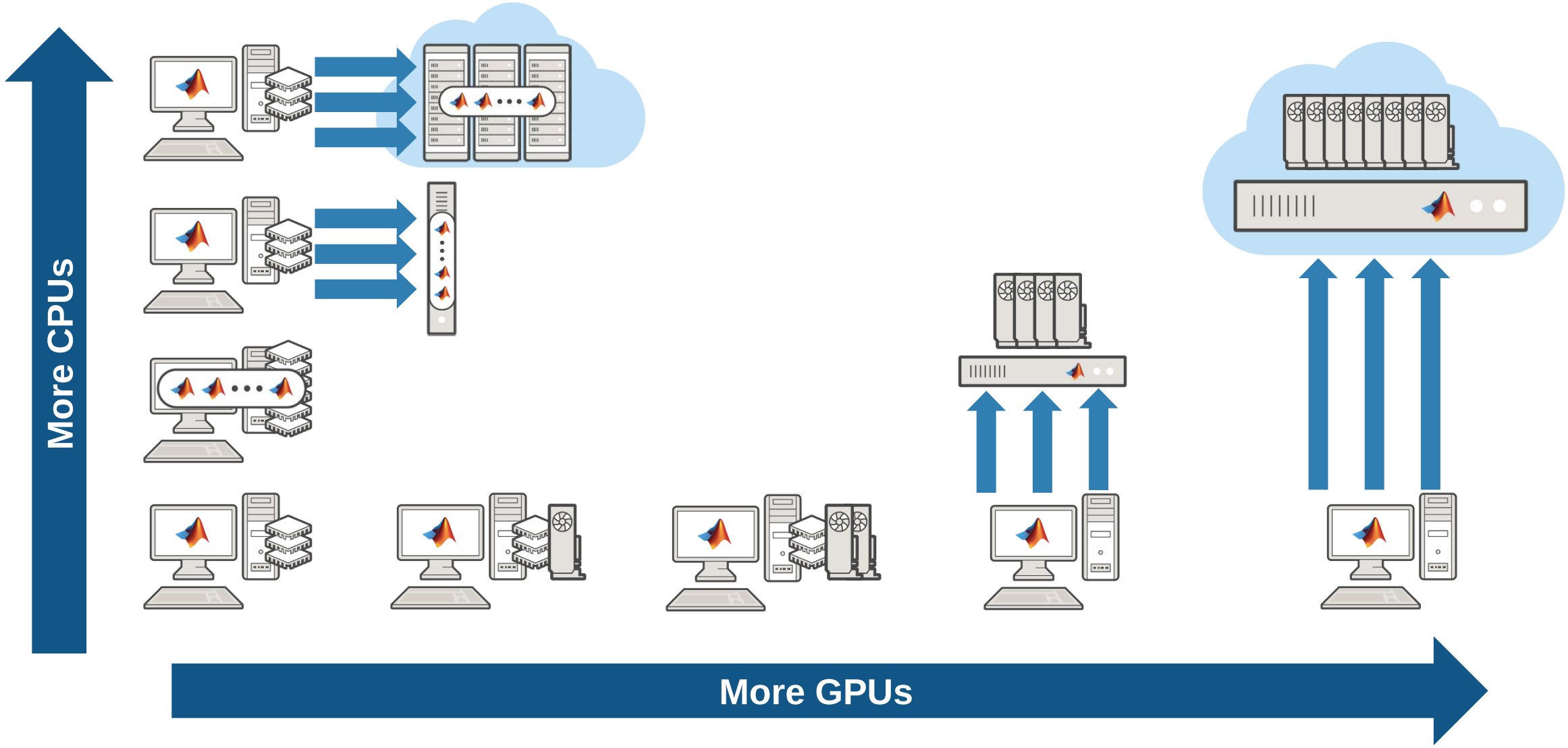
MATLAB and Parallel Computing

- ***Accelerate*** your code on the GPU
  - DEMO: Preprocess your image dataset
- ***Scale*** to multi-GPU and clusters
  - DEMO: Parallel training with an Amazon P2 cluster

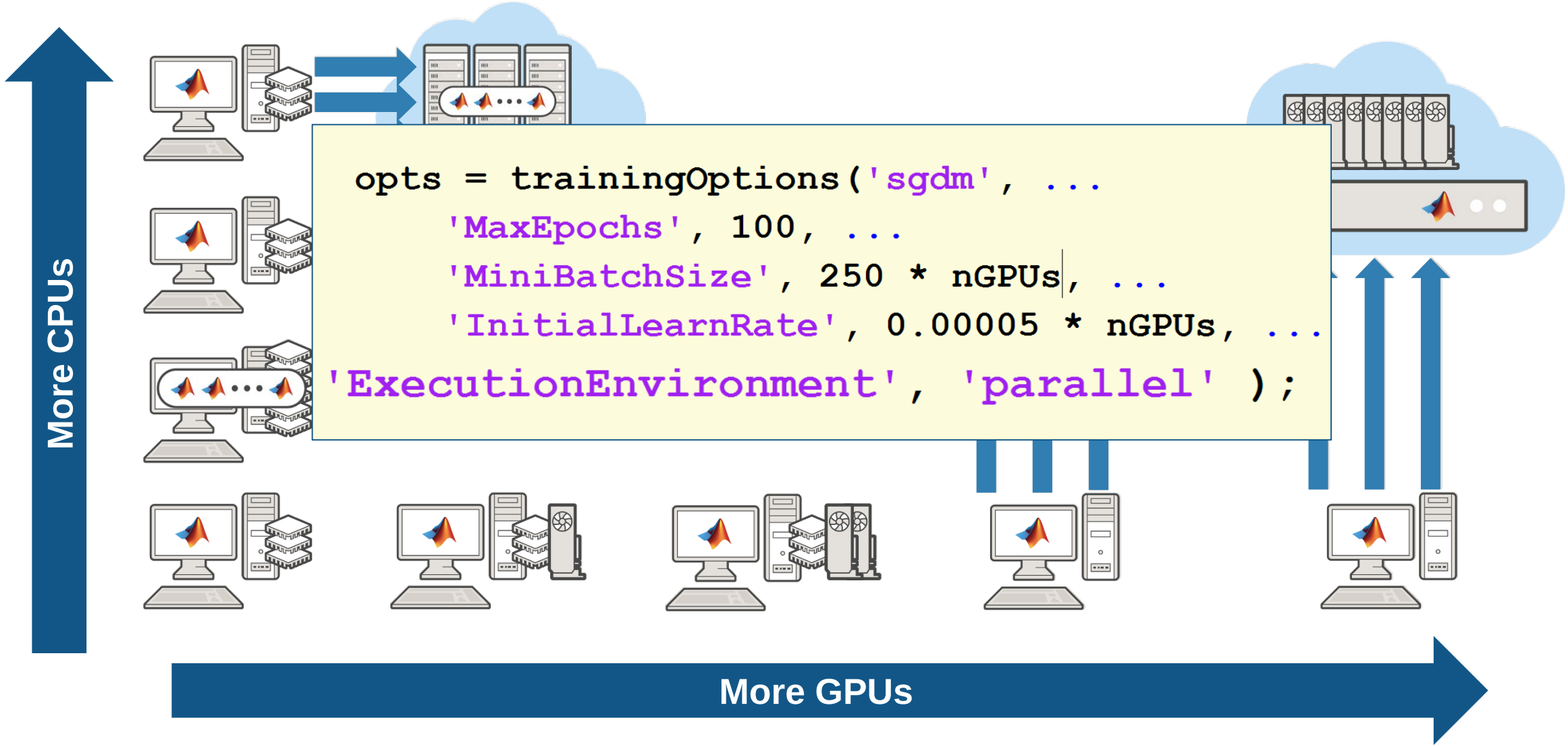
Deep learning on CPU, GPU, multi-GPU and clusters

More GPUs

Deep learning on CPU, GPU, multi-GPU and clusters



Deep learning on CPU, GPU, multi-GPU and clusters




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

MATLAB and Parallel Computing

- ***Accelerate*** your code on the GPU
  - DEMO: Preprocess your image dataset
- ***Scale*** to multi-GPU and clusters
  - DEMO: Parallel training with an Amazon P2 cluster

Transfer learning in 11 lines of MATLAB code

```
%% Set up the datastore
imds = imageDatastore('s3://presentation-eu', ...
    'IncludeSubfolders', true, ...
    'LabelSource', 'FolderNames');

%% Set up the network for transfer learning
net = alexnet;
layers = net.Layers;
num_objects = numel( categories( imds.Labels ) );

% New final Fully Connected Layer and untrained classification layer
layers(23) = fullyConnectedLayer(num_objects, 'Name', 'fc8');
layers(25) = classificationLayer('Name', 'myNewClassifier');

% fc 8 - bump up learning rate for last layers
layers(end-2).WeightLearnRateFactor = 10;
layers(end-2).BiasLearnRateFactor = 20;

%% Fine-tune the Network

miniBatchSize = 250;
maxEpochs = 50;

opts = trainingOptions('sgdm', ...
    'Verbose', true, ...
    'VerboseFrequency', 1, ...
    'LearnRateSchedule', 'none',...
    'InitialLearnRate', 0.00005 * numGPUs,...
    'MaxEpochs', maxEpochs, ...
    'MiniBatchSize', miniBatchSize * numGPUs,...
    'ExecutionEnvironment', 'multi-gpu', ...
    'OutputFcn', @plotTrainingAccuracy );

net = trainNetwork(imds, layers, opts);
```

MathWorks | Accelerating the pace of engineering and science

joss.knight@mathworks.co.uk | Log Out | User's Guide

Cloud Center

- My Clusters
- Create a Cluster
- Preferences
  - User Preferences
  - Cluster Access

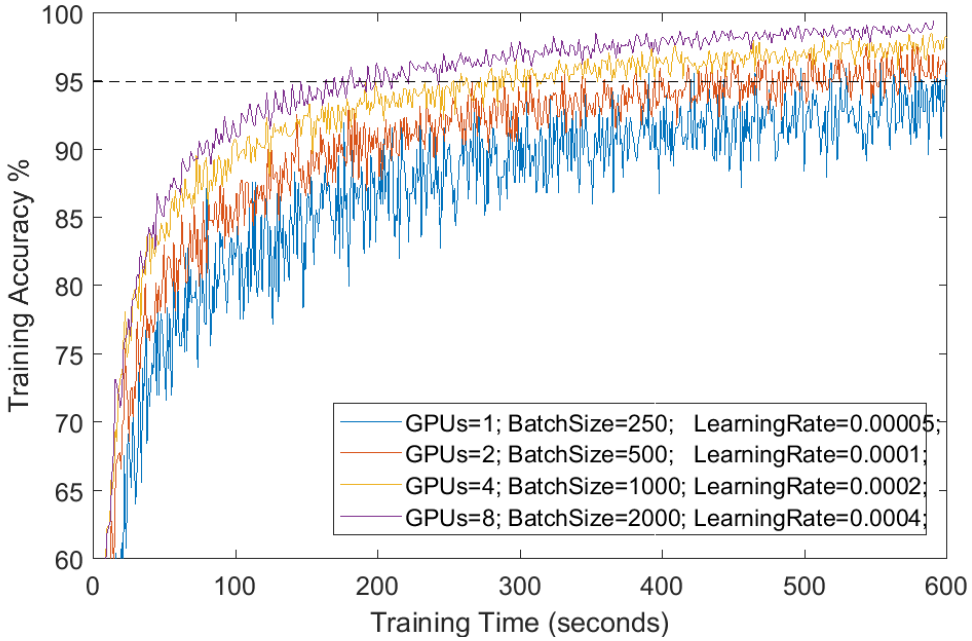
Filter list

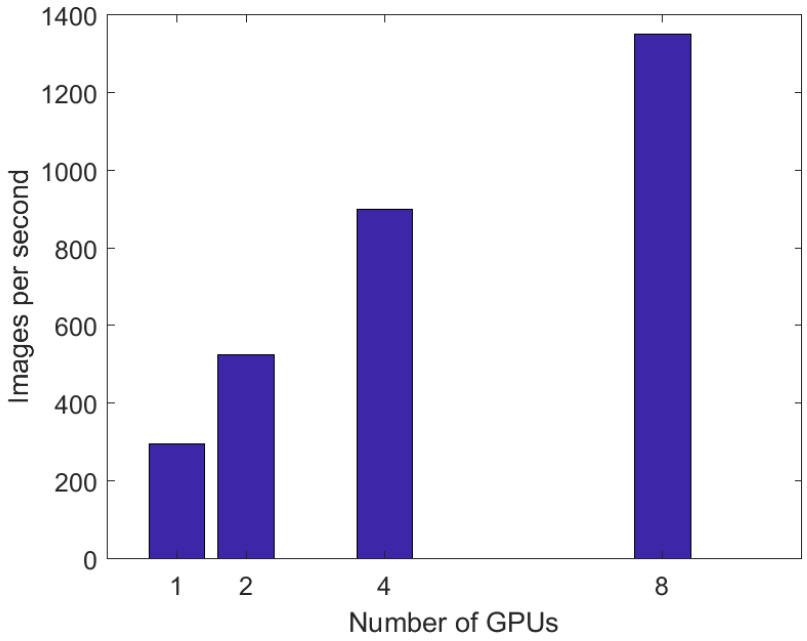
Cluster Name	Region	Workers	Status	Date Created	MATLAB Version	Actions	
Joss Test R2017a		8	Offline	2017-04-04	R2017a	Start Up	Delete
Joss R2017a GTC		8	Offline	2017-04-03	R2017a	Start Up	Delete
Joss SPMD minibash		8	Offline	2017-03-01	R2017b	Start Up	Delete

 Privacy Policy Release Notes dev-SNAPSHOT.b1998

Performance

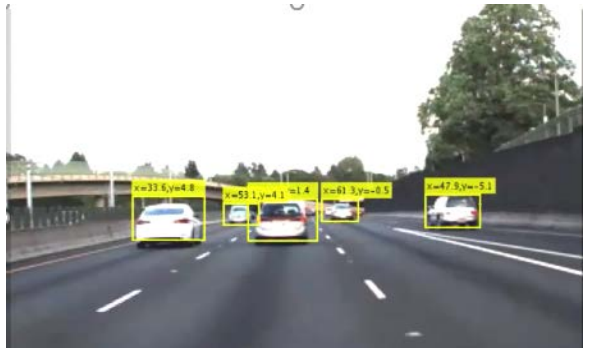
- 30-40% reduction in training time each time you double your GPUs



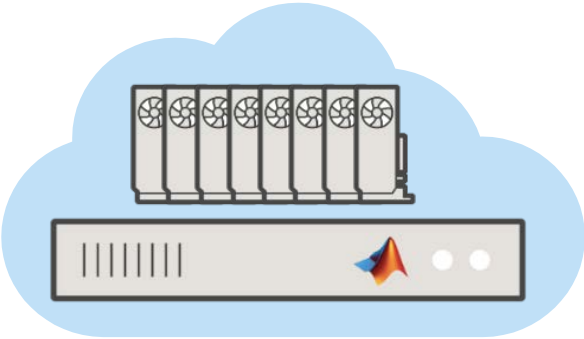


- Communicating with the Cloud costs nothing extra

Talk Outline



Design Deep Learning & Vision Algorithms



Accelerate and Scale Training

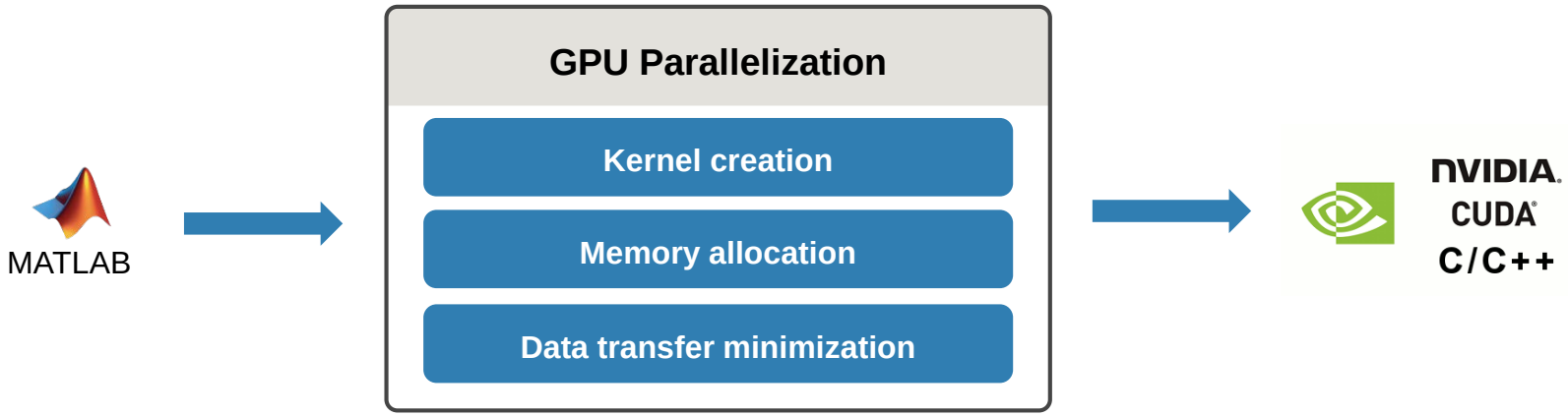
High Performance Embedded Implementation

Can you create high performance implementation from MATLAB code ?

Alexnet inference using MATLAB solution is
~14x faster than pyCaffe and 60% faster than C++ Caffe
~ 4x faster and ~3x less memory-use than TensorFlow

Why?

Presenting the MATLAB to CUDA parallelizing compiler



Sample Generated CUDA Code

MATLAB source code

```
% Time stepping.
for n = 1:Nt,
    % Update H everywhere.
    Hx = Hx+(Dt/mu0)*((Ey(:, :, 2:Nz+1)-Ey(:, :, 1:Nz))*Cz ...
    -(Ez(:, 2:Ny+1, :)-Ez(:, 1:Ny, :))*Cy);
    Hy = Hy+(Dt/mu0)*((Ez(2:Nx+1, :, :)-Ez(1:Nx, :, :))*Cx ...
    -(Ex(:, :, 2:Nz+1)-Ex(:, :, 1:Nz))*Cz);
    Hz = Hz+(Dt/mu0)*((Ex(:, 2:Ny+1, :)-Ex(:, 1:Ny, :))*Cy ...
    -(Ey(2:Nx+1, :, :)-Ey(1:Nx, :, :))*Cx);

    % Update E everywhere except on boundary.
    Ex(:, 2:Ny, 2:Nz) = Ex(:, 2:Ny, 2:Nz)+(Dt/eps0)* ...
    ((Hz(:, 2:Ny, 2:Nz)-Hz(:, 1:Ny-1, 2:Nz))*Cy ...
    -(Hy(:, 2:Ny, 2:Nz)-Hy(:, 2:Ny, 1:Nz-1))*Cz);
    Ey(2:Nx, :, 2:Nz) = Ey(2:Nx, :, 2:Nz)+(Dt/eps0)* ...
    ((Hx(2:Nx, :, 2:Nz)-Hx(2:Nx, :, 1:Nz-1))*Cz ...
    -(Hz(2:Nx, :, 2:Nz)-Hz(1:Nx-1, :, 2:Nz))*Cx);
    Ez(2:Nx, 2:Ny, :) = Ez(2:Nx, 2:Ny, :)+(Dt/eps0)* ...
    ((Hy(2:Nx, 2:Ny, :)-Hy(1:Nx-1, 2:Ny, :))*Cx ...
    -(Hx(2:Nx, 2:Ny, :)-Hx(2:Nx, 1:Ny-1, :))*Cy);

    % Sample the electric field at chosen points.
    Ets(n, :) = [Ex(4, 4, 4) Ey(4, 4, 4) Ez(4, 4, 4)];
end
```

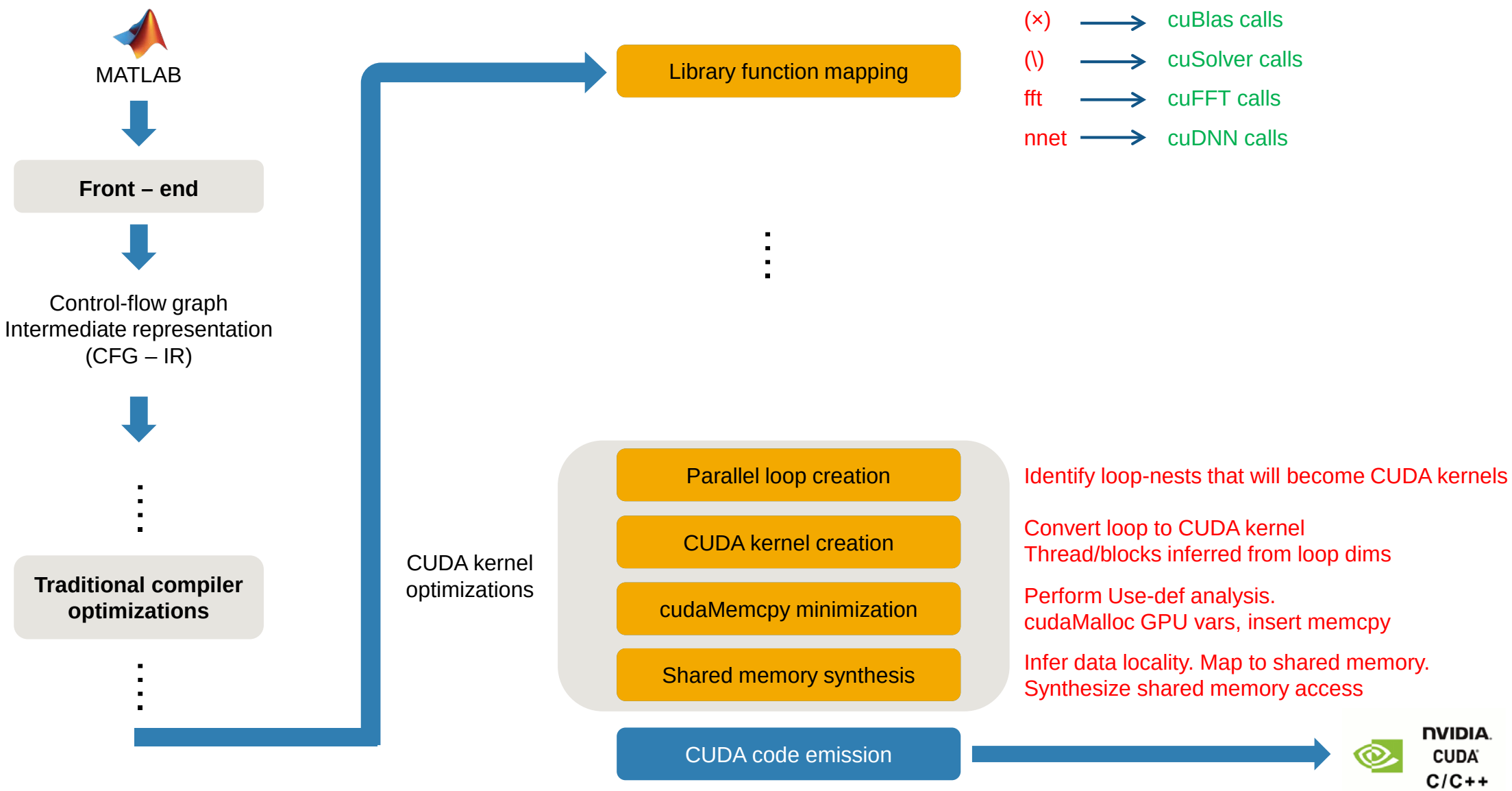
GPU Parallelization

- Kernel creation
- Memory allocation
- Data transfer minimization

Auto-generated CUDA code

```
/*  Time step. */
/*  Allocate field arrays. */
kernel1<<<dim3(17U, 1U, 1U), dim3(512U, 1U, 1U)>>>(gpu_Ex);
kernel2<<<dim3(17U, 1U, 1U), dim3(512U, 1U, 1U)>>>(gpu_Ey);
kernel3<<<dim3(16U, 1U, 1U), dim3(512U, 1U, 1U)>>>(gpu_Ez);
kernel4<<<dim3(16U, 1U, 1U), dim3(512U, 1U, 1U)>>>(gpu_Hx);
kernel5<<<dim3(16U, 1U, 1U), dim3(512U, 1U, 1U)>>>(gpu_Hy);
kernel6<<<dim3(16U, 1U, 1U), dim3(512U, 1U, 1U)>>>(gpu_Hz);

/*  Allocate time signals. */
/*  Initialize fields (near but not on the boundary). */
kernel7<<<dim3(1U, 1U, 1U), dim3(32U, 1U, 1U)>>>(gpu_Ez, gpu_Ey, gpu_Ex);

/*  Time stepping. */
for (n = 0; n < 20000; n++) {
  /*  Update H everywhere. */
  kernel8<<<dim3(16U, 1U, 1U), dim3(512U, 1U, 1U)>>>(gpu_Ez, gpu_Ey, gpu_Hx);
  kernel9<<<dim3(16U, 1U, 1U), dim3(512U, 1U, 1U)>>>(gpu_Ex, gpu_Ez, gpu_Hy);
  kernel10<<<dim3(16U, 1U, 1U), dim3(512U, 1U, 1U)>>>(gpu_Ey, gpu_Ex, gpu_Hz);

  /*  Update E everywhere except on boundary. */
  kernel11<<<dim3(13U, 1U, 1U), dim3(512U, 1U, 1U)>>>(gpu_Hy, gpu_Hz, gpu_Ex);
  kernel12<<<dim3(14U, 1U, 1U), dim3(512U, 1U, 1U)>>>(gpu_Hz, gpu_Hx, gpu_Ey);
  kernel13<<<dim3(14U, 1U, 1U), dim3(512U, 1U, 1U)>>>(gpu_Hx, gpu_Hy, gpu_Ez);

  /*  Sample the electric field at chosen points. */
  kernel14<<<dim3(1U, 1U, 1U), dim3(32U, 1U, 1U)>>>(gpu_Ez, gpu_Ey, gpu_Ex, n,
    gpu_Ets);
  Ets_dirtyOnGpu = true;
}

cudaMemcpy(Ex, gpu_Ex, 67200U, cudaMemcpyDeviceToHost);
cudaMemcpy(Ey, gpu_Ey, 66560U, cudaMemcpyDeviceToHost);
cudaMemcpy(Ez, gpu_Ez, 65520U, cudaMemcpyDeviceToHost);
cudaMemcpy(Hx, gpu_Hx, 62400U, cudaMemcpyDeviceToHost);
cudaMemcpy(Hy, gpu_Hy, 63000U, cudaMemcpyDeviceToHost);
cudaMemcpy(Hz, gpu_Hz, 64000U, cudaMemcpyDeviceToHost);
if (Ets_dirtyOnGpu) {
  cudaMemcpy(Ets, gpu_Ets, 480000U, cudaMemcpyDeviceToHost);
}

cudaFree(gpu_Ex);
cudaFree(gpu_Ey);
```

MATLAB to CUDA compiler flow



MATLAB to CUDA compiler: It's all about big parallel loops!

```
output(:, 1) = (input(:, 1) - diff_im) .* factor;
```

MATLAB

Front – end

Control-flow graph
Intermediate representation
(CFG – IR)

Traditional compiler optimizations

Loop optimizations

- Library function mapping
- Scalarization
- Loop perfectization
- Loop interchange
- Loop fusion
- Scalar replacement

CUDA kernel optimizations

- Parallel loop creation
- CUDA kernel creation
- cudaMemcpy minimization
- Shared memory synthesis

CUDA code emission

```
for i = 1:size(intput, 1)
    difftemp(i) = input(i, 1) - diff_im(i);
end

for i = 1:size(input, 1)
    output(i, 1) = difftemp(i) * factor(i);
end
```

```
for i = 1:size(input, 1)
    difftemp(i) = input(i, 1) - diff_im(i);
    output(i, 1) = difftemp(i) * factor(i);
end
```

```
for i = 1:size(input, 1)
    difftemp_scalar = input(i, 1) - diff_im(i);
    output(i, 1) = difftemp_scalar * factor(i);
end
```

NVIDIA CUDA C/C++

MATLAB to CUDA compiler: It's all about big parallel loops!

MATLAB

Front – end

Control-flow graph
Intermediate representation
(CFG – IR)

Traditional compiler optimizations

Loop optimizations

- Library function mapping
- Scalarization
- Loop perfectization
- Loop interchange
- Loop fusion
- Scalar replacement

CUDA kernel optimizations

- Parallel loop creation
- CUDA kernel creation
- cudaMemcpy minimization
- Shared memory synthesis

CUDA code emission

```
output(:, 1) = (input(:, 1) - diff_im) .* factor;
```

2 kernels (size N), 20*N bytes

1 kernel (size N), 16*N bytes

```
for i = 1:size(input, 1)
    difftemp_scalar = input(i, 1) - diff_im(i);
    output(i, 1) = difftemp_scalar * factor(i);
end
```

NVIDIA CUDA C/C++

cudaMemcpy minimization

```
A(:) = ….
C(:) = ….

for i = 1:N
      ….
      gB = kernel1(gA);
      gA = kernel2(gB);
      if (some_condition)
            gC = kernel3(gA, gB);
      end
      ….
end

…. = C;
```

cudaMemcpy
definitely needed

cudaMemcpy
not needed

cudaMemcpy
may be needed

Assume gA, gB and gC are mapped to GPU memory

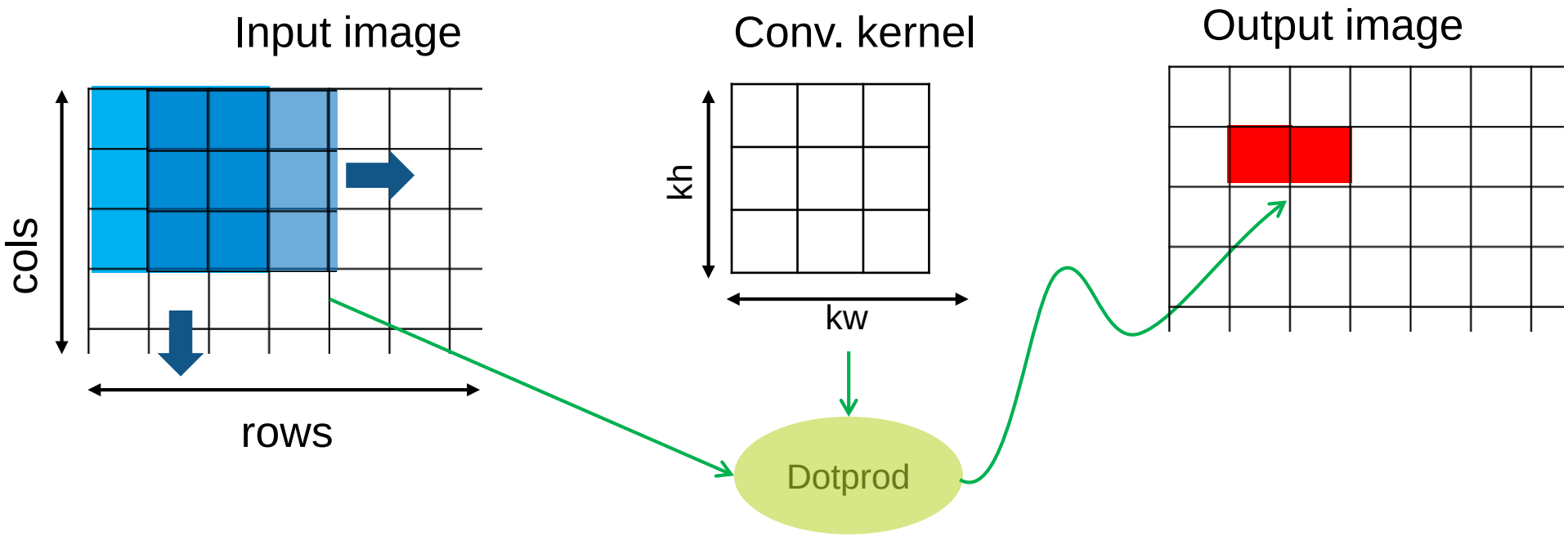
<u>Observations</u>

- Equivalent to Partial redundancy elimination (PRE)
- Dynamic strategy – track memory location with a status flag per variable
- Use-Def to determine where to insert memcpy

Generated (pseudo) code

```
A(:) = …
A_isDirtyOnCpu = true;
…
for i = 1:N
   if (A_isDirtyOnCpu)
      cudaMemcpy(gA, A);
      A_isDirtyOnCpu = false;
   end
   gB = kernel1(gA);
   gA = kernel2(gB);
   if (somecondition)
      gC = kernel3(gA, gB);
      C_isDirtyOnGpu = true;
   end
   …
end
…
if (C_isDirtyOnGpu)
   cudaMemcpy(C, gC);
   C_isDirtyOnGpu = false;
end
… = C;
```

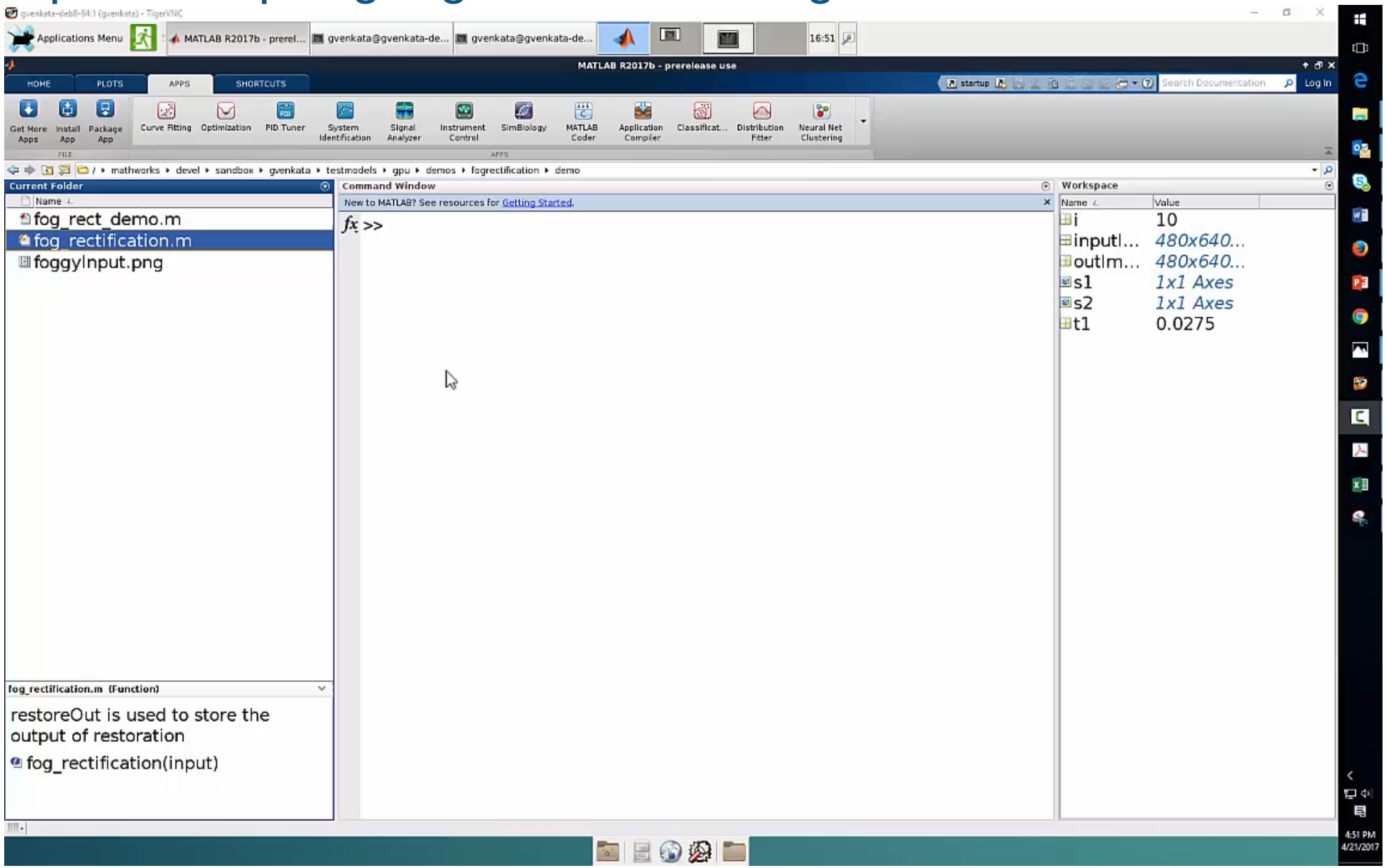
Shared memory synthesis for stencil operations



For stencil operations, the MATLAB to CUDA compiler automatically

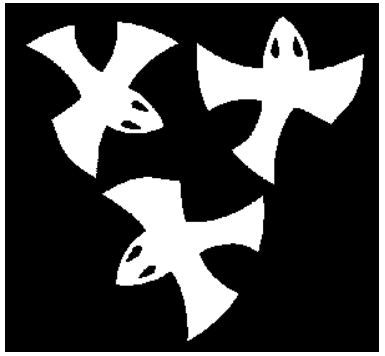
- Infers GPU shared memory
- Automates the collaborative loading in to shared memory block
- Automatically translates access from global variable to shared-mem variable

Example: Compiling fog-rectification algorithm

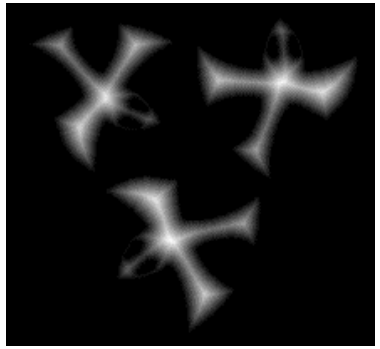


MATLAB to CUDA Compilation in Computer Vision Applications

Fog removal

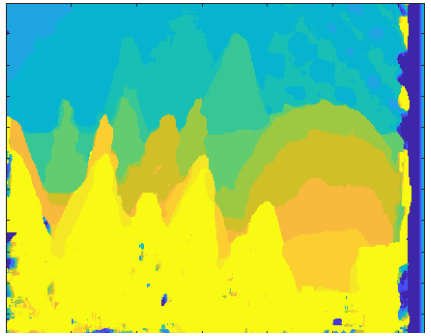


Distance transform

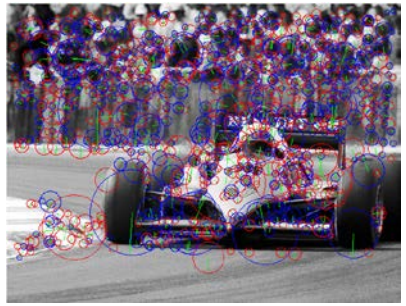


Ray tracing

Stereo disparity

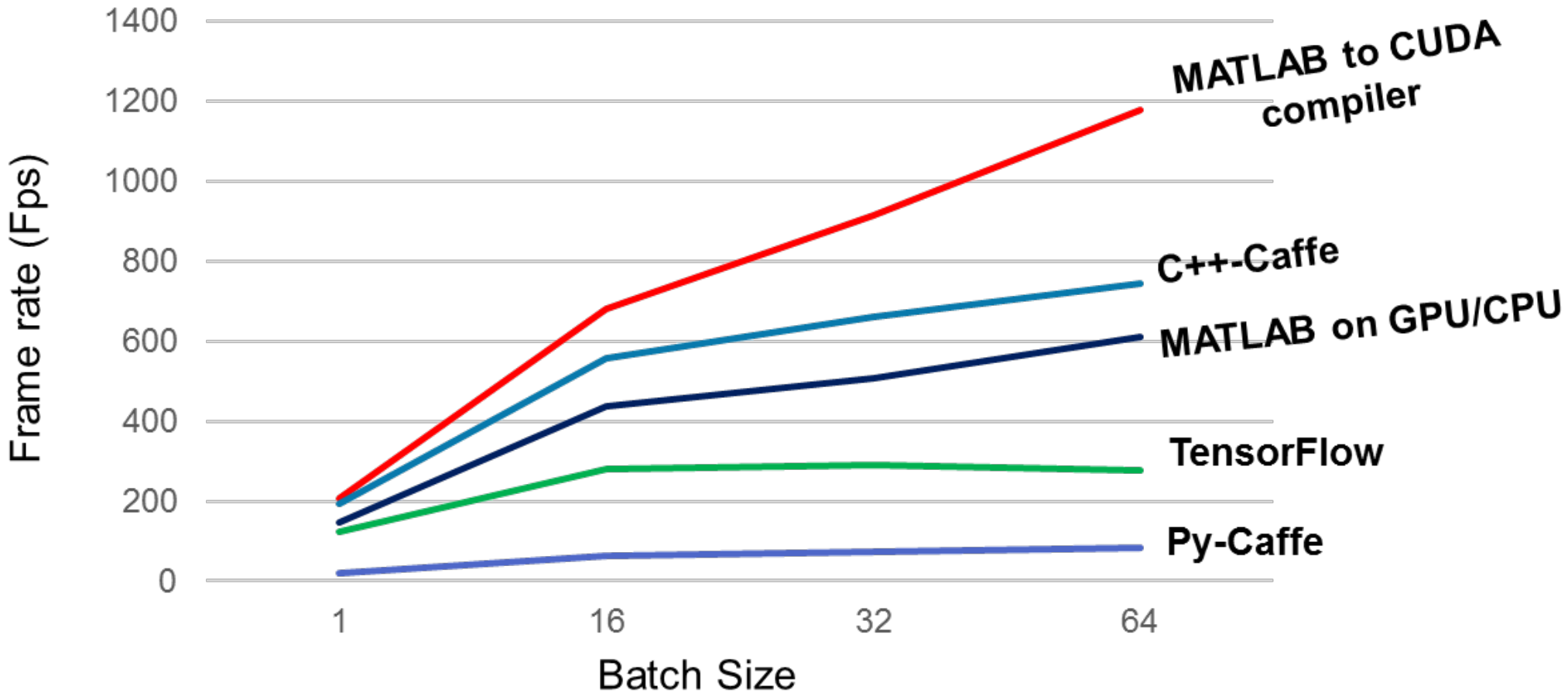


SURF feature extraction



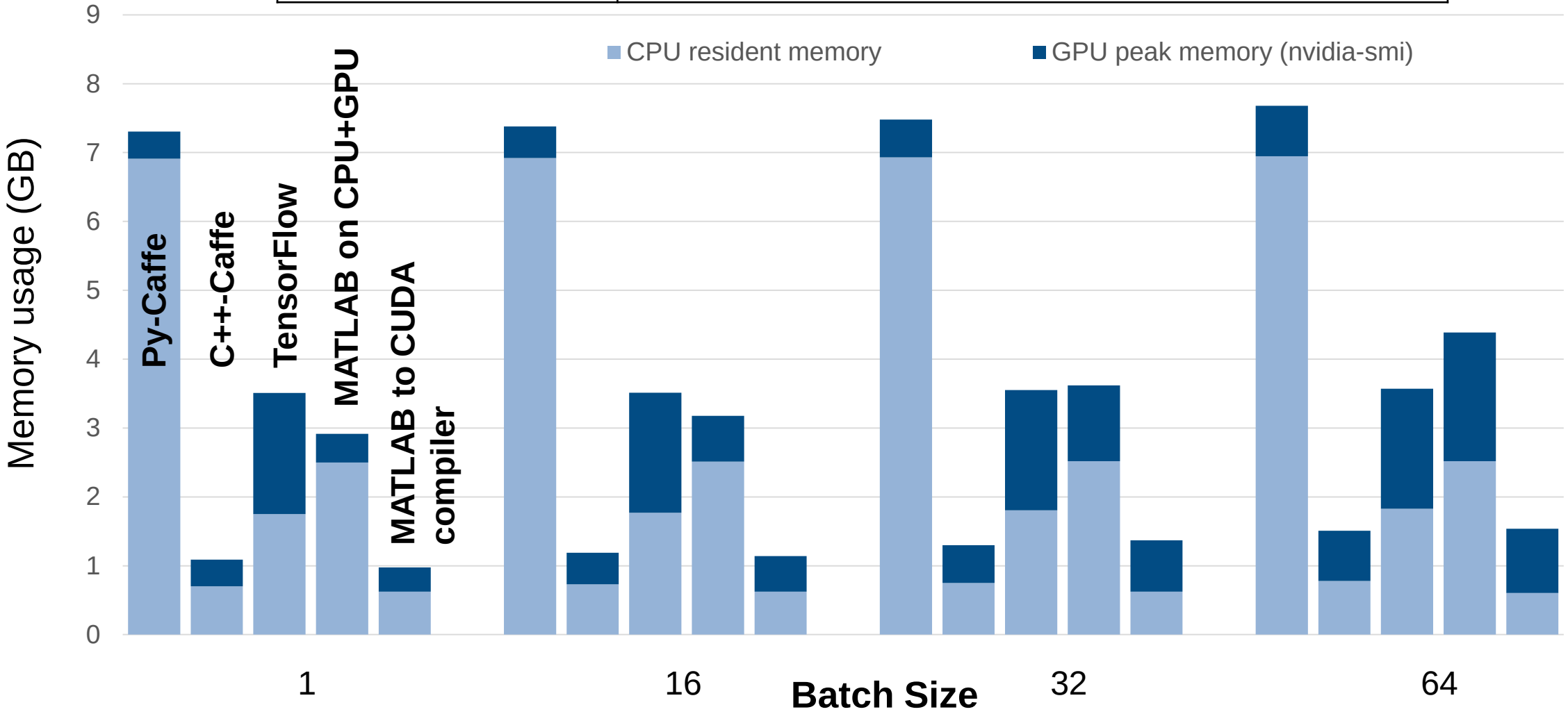
Deep learning prediction performance: Alexnet

CPU	Intel(R) Xeon(R) CPU E5-1650 v3 @ 3.50 GHz
GPU	Tesla K40c

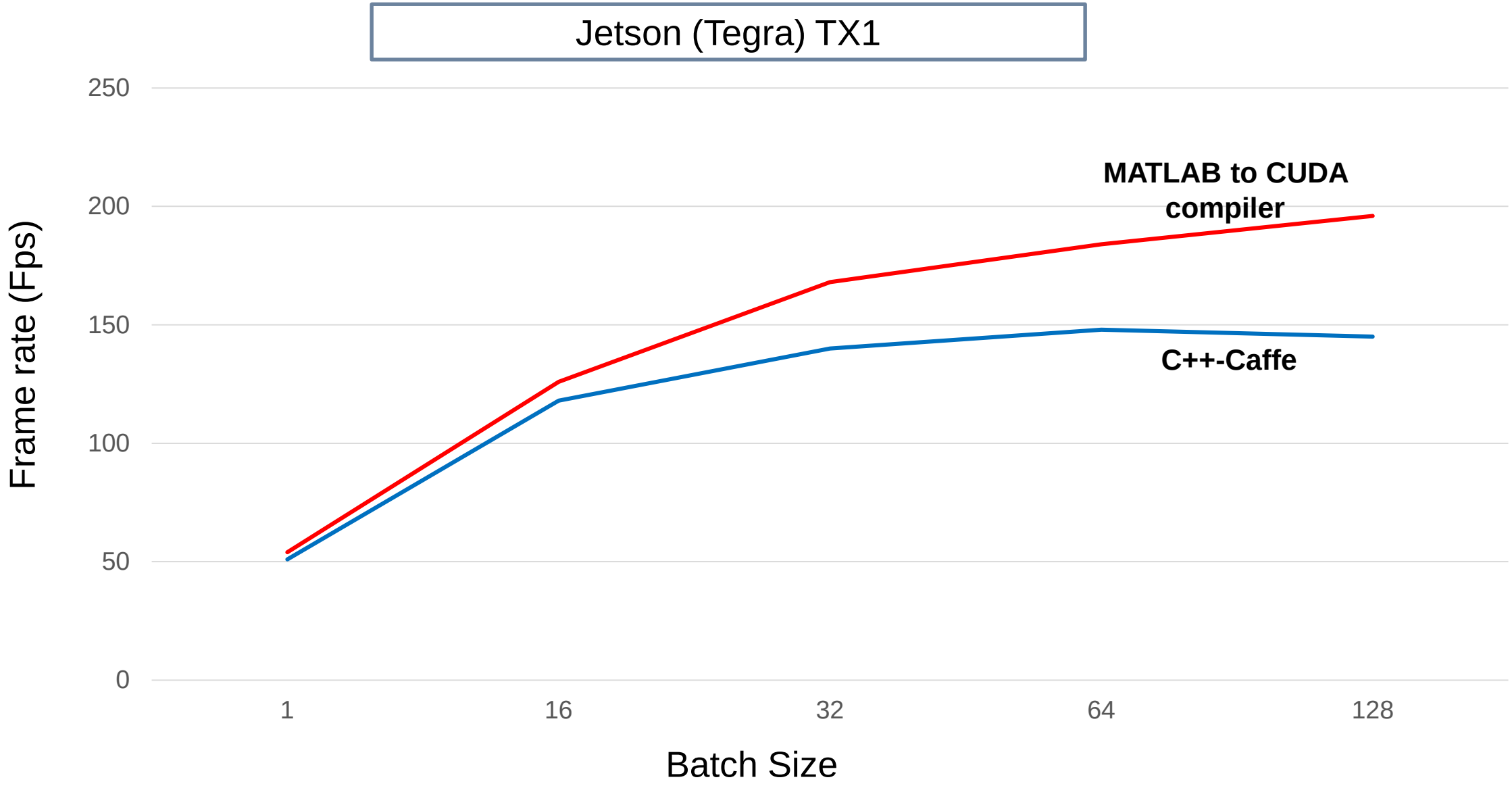


Deep learning prediction performance: Alexnet

CPU	Intel(R) Xeon(R) CPU E5-1650 v3 @ 3.50 GHz
GPU	Tesla K40c

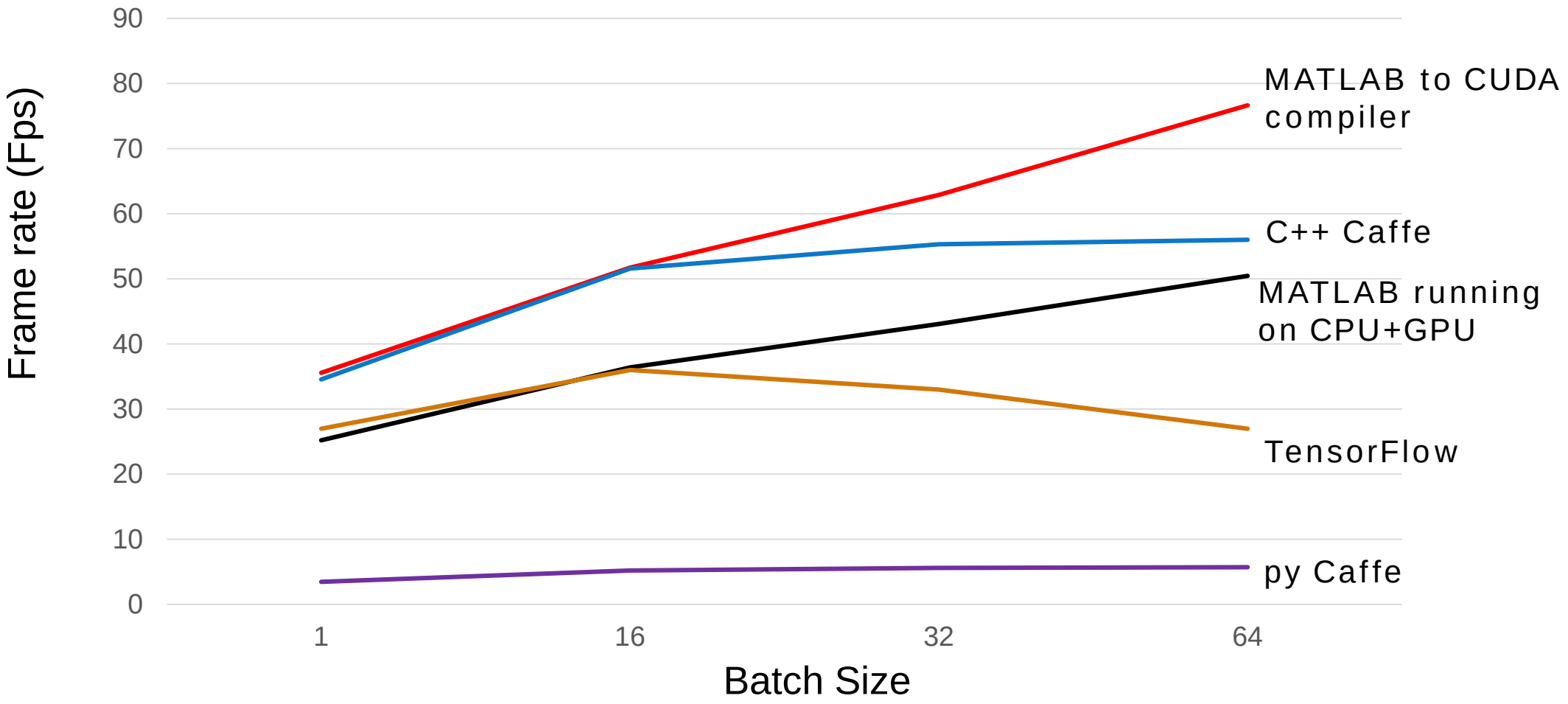


Deep learning prediction performance: Alexnet



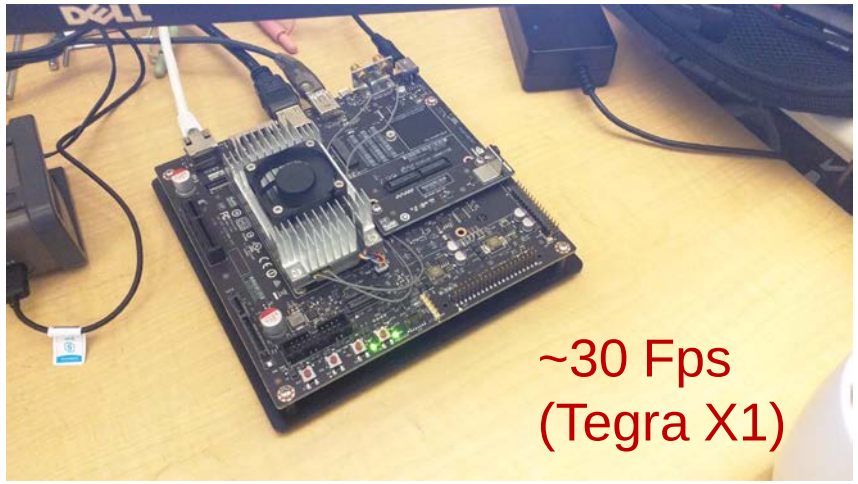
Deep Learning Prediction Performance: VGG-16

CPU	Intel(R) Xeon(R) CPU E5-1650 v3 @ 3.50GHz
GPU	Tesla K40c



Create CNNs with MATLAB, Deploy with MATLAB to CUDA compiler

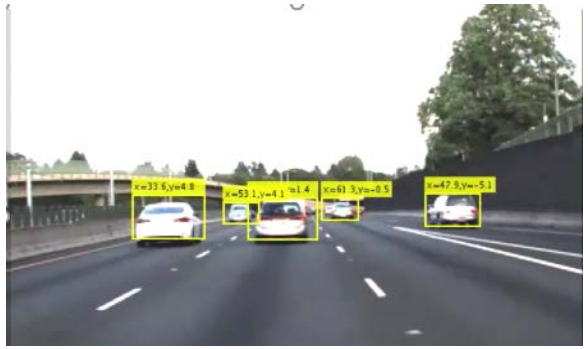
Alexnet



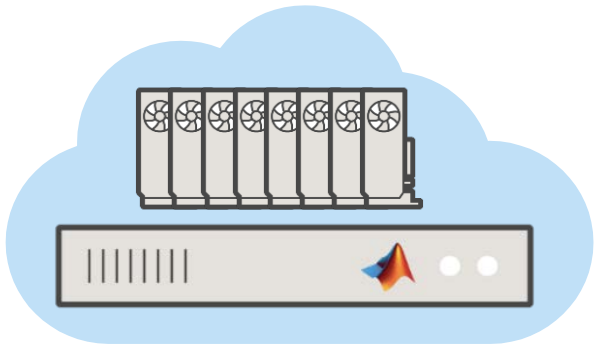
Vehicle Detection

People detection

Lane detection



Design Deep Learning & Vision Algorithm



Accelerate and Scale Training

High Performance Embedded Implementation

Deep learning design is easy in MATLAB

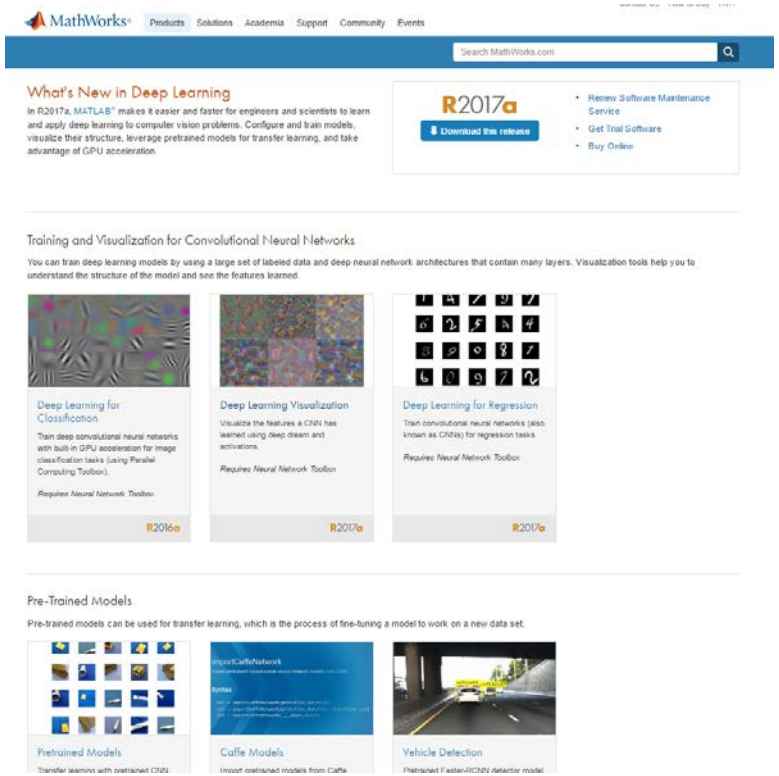
Parallel Computing Toolbox
7x faster than pyCaffe
2x faster than TensorFlow

MATLAB to CUDA compiler
14x faster than pyCaffe
4x faster than TensorFlow
1.6x faster than C++ Caffe

What Next?

Visit our booth, we love to chat: Booth # 804

Try Deep Learning with MATLAB



MATLAB to CUDA compiler: Sign up for our beta program

www.mathworks.com/matlab-cuda-beta