MATLAB and NVIDIA Docker: A Complete AI Solution, Where You Need It, In an Instant

Jos Martin, Joss Knight
Motivating Examples

*Build a scalable web-application that can classify the genre of a short sample of music*

*Build a model to detect tumors in 3-D MRI brain scans*
Deep Learning Workflow

Access Data → Preprocess → Select Network → Train/Test → Deploy
Problem: Music Genre Classification

- Can we classify a (short) portion of audio into one of a number of genres?

- (Self imposed) constraint
  - *Try to assume as little as possible about how to do this*
  - optimize hyper-parameters, network architecture, etc.
  - Try out various different pre-processing techniques

REF: Music Popcorn – A visualization of the music genre space. (http://static.echonest.com/popcorn/)
Problem: Music Genre Classification

- Some fundamental assumptions
  - Convert audio to image-like data
    - Want to use CNN-like architecture
  - Not going to use recurrent network (RNN)
    - We have a classification problem
    - This is not an obvious choice and might be worth revisiting
  - Work on a 5s sample
Everything Starts with Data

- GTZAN Genre Collection (2002)
- Labelled set of 1000 tracks
  - ~30s long in 10 categories

REF: “Musical genre classification of audio signals” by G. Tzanetakis and P. Cook in IEEE Transactions on Audio and Speech Processing 2002
DOWNLOAD: http://opihi.cs.uvic.ca/sound/genres.tar.gz
Exploring My New Data

Look at the first 5 waveforms from the dataset

```matlab
ds = audioDatastore("L:\data\genres", 'IncludeSubfolders', true);
for i = 1:5
    y = ds.read;
    x = linspace(0, 30, numel(y));
    subplot(5, 1, i)
    plot(x, y)
end```

Sound 2 Image – MEL Spectrograms

```matlab
winLen = 2048;
y = log(melSpectrogram(audioData,f,...
  'WindowLength',winLen,...
  'OverlapLength',winLen-512,...
  'FFTLength',2*winLen,...
  'NumBands',256));
```

How big is the spectrogram and what does it look like?

```matlab
size(y)
```

ans = 1×2

256   212
Sound 2 Image – Wavelet Scattering

```matlab
sf = waveletScattering('SignalLength', sampleTime*f, ...
    'SamplingFrequency', f, ...
    'InvarianceScale', 0.1);
y = featureMatrix(sf, audioData, 'Transform', 'log');
```

How big is the feature matrix and what does it look like?

```matlab
size(y)
```

```matlab
ans = 1×2
    208   216
```

```matlab
h = pcolor(y);
h.EdgeColor = 'none';
```
tic

while myds.hasdata
    [x, info] = myds.read;
    x = x./max(x);
    for ii = find(sampleEnd <= numel(x))
        audioSample = x(sampleStart(ii):sampleEnd(ii));
        info.SampleIndex = ii;
        melProcessThisSignal(audioSample, info, roots);
        waveletProcessThisSignal(audioSample, info, roots);
    end
end

toc

Elapsed time is 5600.002248 seconds.
Lots of Sound 2 Image – Just Faster

tic

def spmd
    myds = partition(ds, numlabs, labindex);
    while myds.hasdata
        [x, info] = myds.read;
        x = x./max(x);
        for ii = find(sampleEnd <= numel(x))
            audioSample = x(sampleStart(ii):sampleEnd(ii));
            info.SampleIndex = ii;
            melProcessThisSignal(audioSample, info, roots);
            waveletProcessThisSignal(audioSample, info, roots);
        end
    end
end

toc

Elapsed time is 1492.312001 seconds. 3.75x
What Might My Network Look Like?
That Irksome (Self imposed) Constraint

*Try to assume as little as possible about how to do this*

- Can I know *a priori* the best
  - Network architecture,
  - Training parameters
  - etc.

REF: Auguste Rodin 'The Thinker'
Letting the Algorithm Choose For You

- From Wikipedia

*Bayesian optimization is a sequential design strategy for global optimization of black-box functions that doesn’t require derivatives.*

- Sounds good because:
Letting the Algorithm Choose For You

▪ From Wikipedia

**Bayesian optimization** is a sequential design strategy for **global optimization** of black-box functions that doesn’t require derivatives.

▪ Sounds good because:
  – We want the best pre-processing strategy, network architecture and hyper-parameters for genre recognition (none of which we know *a priori*)
Letting the Algorithm Choose For You

- From Wikipedia

**Bayesian optimization** is a sequential design strategy for **global optimization** of **black-box functions** that doesn’t require derivatives.

- Sounds good because:
  - We want the best pre-processing strategy, network architecture and hyper-parameters for genre recognition (none of which we know *a priori*)
  - We have no idea how changing any of those things might affect the loss
Letting the Algorithm Choose For You

- From Wikipedia

**Bayesian optimization** is a sequential design strategy for global optimization of black-box functions that doesn’t require derivatives.

- Sounds good because:
  - We want the best pre-processing strategy, network architecture and hyper-parameters for genre recognition (none of which we know *a priori*)
  - We have no idea how changing any of those things might affect the loss
  - We want to get to the best solution as quickly as possible
What Should We Let It Choose?

1. Pooling & conv size
2. Number of channels in conv
3. Final pooling size

Dimensions:
- 256x212
- 256x212x32
- 84x70x96
- 27x22x96
What Should We Let It Choose?

4. Replicate this convolutional block a variable number of times

5. Dropout Probability

Training parameters:

6. Initial learn rate

7. Learn rate drop period
What Should We Let It Choose?

Training parameters:
6. Initial learn rate
7. Learn rate drop period
Make Sure It Works (once)

```matlab
imds = imageDatastore('your_dataset_folder', 'IncludeSubfolders', true);
[imdsTrain, imdsVal] = splitEachLabel(imds, 0.7, 'random');

options = trainingOptions('adam', ...'
    'InitialLearnRate', 0.001, ...'
    'MiniBatchSize', 64, ...'
    'L2Regularization', 1e-4, ...'
    'MaxEpochs', 16, ...'
    'Shuffle', 'every-epoch', ...'
    'Plots', true, ...'
    'training-progress', true, ...'
    'Verbose', false, ...'
    'ValidationData', imdsVal, ...'
    'ValidationFrequency', 1,...'
    'LearnRateSchedule', 'constant', ...'
    'LearnRateDropPeriod', 1,...'
    'LearnRateDropFactor', 0.5,...
);  
trainedNet = trainNetwork(imdsTrain, layers, options);```

![Training Progress Chart](chart.png)
When Will It Give Me an Answer?

- How many times should Bayesian Optimization evaluate my cost function?
  - Rule of thumb … 10-15x number of variables being optimized
  - So about 100 times

\[
\text{disp}(2 \times 100 \times 1500 / 3600 / 24) = 3.4722 \text{ (days!)}
\]
“We’re Gonna Need a Bigger Machine”

docker login nvcr.io

docker pull nvcr.io/partners/matlab:r2019a

rsync -rave ssh /data/genres/ dgx:/tmp/genres

nvidia-docker run --rm -p 6080:6080 --shm-size=512M -v /tmp/genres:/data nvcr.io/partners/matlab:r2019a
“We’re Gonna Need a Bigger Machine”

- Desktop access using browser
- Or VNC
- Or at the docker command prompt
Getting Started

- Start MATLAB & login
>> gpuDevice

ans =

CUDADevice with properties:

  Name: 'Tesla V100-DGXS-15GB'
  Index: 1
  ComputeCapability: '7.0'
  SupportsDouble: 1
  DriverVersion: 10
  ToolkitVersion: 10
  MaxThreadsPerBlock: 1024
  MaxShmemPerBlock: 49152
  MaxThreadBlockSize: [1024 1024 64]
  MaxGridSize: [2.1475e+09 65535 65535]
  SIMDWidth: 32
  TotalMemory: 1.6908e+10
  AvailableMemory: 1.6130e+10
  MultiprocessorCount: 80
  ClockRateKHz: 1530000
  ComputeMode: 'Default'
  GPUOverlapsTransfers: 1
  KernelExecutionTimeout: 1
  CanMapHostMemory: 1
  DeviceSupported: 1
  DeviceSelected: 1

>> gpuDeviceCount

ans =

4
Putting Everything Together

```matlab
imds = imageDatastore('/tmp/processed-genres-wave/', ...  
    'IncludeSubfolders', true, 'LabelSource', 'foldernames');  
[training,validation] = splitEachLabel(imds, 0.8, 'randomized');
```

```matlab
v(1) = optimizableVariable('InitialLearnRate', [1e-7 1e-1], 'Transform', 'log');  
v(2) = optimizableVariable('LearnRateDropPeriod', [1 20], 'Type', 'integer');  
v(3) = optimizableVariable('NumFilters', [5 64], 'Type', 'integer');  
v(4) = optimizableVariable('NumConvolutionalBlocks', [2 7], 'Type', 'integer');  
v(5) = optimizableVariable('PoolingSize', [2 6], 'Type', 'integer');  
v(6) = optimizableVariable('FinalPoolingSize', [2 8], 'Type', 'integer');  
v(7) = optimizableVariable('DropoutProbability', [0.2 0.8]);
```

```matlab
results = bayesopt(makeCostFunction(training, validation), ...  
    v, ...  
    'MaxObjectiveEvaluations',100, ...  
    'AcquisitionFunctionName','expected-improvement-plus', ...  
    'UseParallel', true)
```
Sit back, relax, ...
How Did Wavelet Decomposition Do?

Total function evaluations: 100
Total elapsed time: 31143.0405 seconds.
Total objective function evaluation time: 117231.7899

3.76x
And the Spectrogram?

Total function evaluations: 100
Total elapsed time: 38106.4788 seconds.
Total objective function evaluation time: 150863.0503

Best observed feasible point:

<table>
<thead>
<tr>
<th>InitialLearnRate</th>
<th>LearnRateDropPeriod</th>
<th>NumFilters</th>
<th>NumConvolutionalBlocks</th>
<th>PoolingSize</th>
<th>FinalPoolingSize</th>
<th>DropoutProbability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00010071</td>
<td>18</td>
<td>44</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>0.22672</td>
</tr>
</tbody>
</table>

Observed objective function value = 0.049958
Estimated objective function value = 0.050476
Function evaluation time = 1016.8068
docker login nvcr.io

docker pull nvcr.io/partners/matlab:r2019a

rsync -rave ssh /data/genres/ dgx:/tmp/genres/

nvidia-docker run --it --rm -p 6080:6080 --shm-size=512M -v /tmp/genres/:/data nvcr.io/partners/matlab:r2019a

Optimization completed.
MaxObjectiveEvaluations of 100 reached.
Total function evaluations: 100
Total elapsed time: 21365.4506 seconds.
Total objective function evaluation time: 164366.5305

Time’s A-wasting – I Haven’t Got a Whole Day!

7.7x
Overall Results

- Wavelet pre-processing had a slightly lower loss than mel spectrogram
  - 0.044 opposed to 0.049

- We did 3.1 days work in 19h on the DGX (and in ~9h on a p3.16xlarge)

- We know the best way to train our system is wavelet + model with:

<table>
<thead>
<tr>
<th>Best observed feasible point:</th>
<th>InitialLearnRate</th>
<th>LearnRateDropPeriod</th>
<th>NumFilters</th>
<th>NumConvolutionalBlocks</th>
<th>PoolingSize</th>
<th>FinalPoolingSize</th>
<th>DropoutProbability</th>
</tr>
</thead>
<tbody>
<tr>
<td>InitialLearnRate</td>
<td>0.0011456</td>
<td>16</td>
<td>39</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>0.30633</td>
</tr>
</tbody>
</table>
Is the Model Any Good?

```
imds = imageDatastore('/tmp/data', 'IncludeSubfolders', true);
[trainning, validation] = splitEachLabel(imds, 0.7);

% Training happened here!

predictedLabels = classify(net, validation.Files);
trueLabels = validation.Labels;

confusionchart(trueLabels, predictedLabels, 'title', 'Validation Accuracy - Genre Recognition');
```
Detecting Tumors in 3-D Brain Images

- 240-by-240-by-155-by-4 Brain MRI data
- Label every voxel as tumor or not-tumor
- This is *Semantic Segmentation*
- Uses 3-D features
Detecting Tumors in 3-D Brain Images

- 240-by-240-by-155-by-4 Brain MRI data
- Label every voxel as tumor or not-tumor
- This is *Semantic Segmentation*
- Uses 3-D features

```matlab
for i = 1:size(volume,3)
    vol{i} = labeloverlay(volume(:,,:,i), label(:,,:,i));
end
vol = cat(4, labelledVol{:});
montage(vol)
```
Detecting Tumors in 3-D Brain Images

```matlab
h = labelvolshow(label, volume(:, :, :, 2));
h.LabelVisibility(1) = 0;
h.VolumeThreshold = 0.8;
```
Pre-process to Reduce Data Size

while hasdata(pxls)
    label = readNumeric(pxls);
    volume = read(volxls);

    % Crop a box around the tumor
    reg = regionprops3(label > 0,'BoundingBox');
    tol = 64;
    ROI = ceil(reg.BoundingBox(1,:));
    ROIst = ROI(1:3) - tol;
    ROIend = ROI(1:3) + ROI(4:6) + tol;
    ...

    % Dataset with a valid size for 3-D U-Net
    ind = floor(size(tcropVol)/8)*8;
    incropVol = tcropVol(1:ind(1),1:ind(2),1:ind(3),:);
    cropVol = channelWisePreProcess(incropVol);
    ...
end

function out = channelWisePreProcess(in)
    in = gpuArray(in);
    chn_Mean = mean(in,[1 2 3]);
    chn Std = std(in,0,[1 2 3]);
    out = (in - chn_Mean)./chn Std;
    ...

    % rescale the data in the range [0,1]
    out = (out - rangeMin) / (rangeMax - rangeMin);
    out = gather(out);
end
Pre-process to Reduce Data Size

montage({origSlice;croppedSlice},'Size',[1 2],'BorderSize',5);
Define Data Iterators

```matlab
volReader = @(x) matRead(x);
volLoc = fullfile(preprocessDataLoc,'imagesTr');
volds = imageDatastore(volLoc, ...
    'FileExtensions','mat','ReadFcn',volReader);

labelReader = @(x) matRead(x);
lblLoc = fullfile(preprocessDataLoc,'labelsTr');
classNames = {'background','tumor'};
pixelLabelID = [0 1];
pxds = pixellabelDatastore(lblLoc,classNames,pixelLabelID, ...
    'FileExtensions','mat','ReadFcn',labelReader);

patchSize = [64 64 64];
patchPerImage = 16;
miniBatchSize = 8;
patchds = randomPatchExtractionDatastore(volds,pxds,patchSize, ...
    'PatchesPerImage',patchPerImage);
patchds.MiniBatchSize = miniBatchSize;
```
Define data augmentation

```matlab
augpatchds = transform(patchds,@augment3dPatch);

function patchOut = augment3dPatch(patchIn)
    % 5 augmentations: nil, rot90, fliplr, flipud, rot90(fliplr)
    fliprot = @(x) rot90(fliplr(x));
    augType = {@rot90, @fliplr, @flipud, fliprot};

    for id=1:size(patchIn,1)
        rndIdx = randi(8,1);
        tmpImg = gpuArray(patchIn.InputImage{id});
        tmpResp = gpuArray(patchIn.ResponsePixelLabelImage{id});
        if rndIdx < 5
            out = augType{rndIdx}(tmpImg);
            respOut = augType{rndIdx}(tmpResp);
        else
            out = tmpImg;
            respOut = tmpResp;
        end
        inpVol{id}=out;
        inpResponse{id}=respOut;
    end

    patchOut = table(inpVol,inpResponse);
end
```
Define Model
Define Model

Create Deep Learning Network Architecture

Script for creating the layers for a deep learning network with:

Number of layers: 44
Number of connections: 46

Run the script to create the layers in the workspace variable `lgraph`.

To learn more, see Generate MATLAB Code From Deep Network Designer.

Auto-generated by MATLAB on 08-Mar-2019 10:28:04

Create the Layer Graph

Create the layer graph variable to contain the network's layers.

\[ lgraph = layerGraph(); \]

Add the Layer Branches

Add the branches of the network to the layer graph. Each branch is a linear array of layers.

\[
\text{templayers} = [
    \text{image3dInputLayer([64 64 64 4]),"Name","input","Normalization","none"},
    \text{convolution3dLayer([3 3 3],32,"Name","en1_conv1","Padding","same"},
    \text{batchNormalizationLayer("Name","en1_bn1"},
    \text{reluLayer("Name","en1_relu1"}),
    \text{convolution3dLayer([3 3 3],64,"Name","en1_conv2","Padding","same"},
    \text{reluLayer("Name","en1_relu2"});
\]
\[ lgraph = addLayers(lgraph,templayers); \]

\[
\text{templayers} = [
    \text{maxPooling3dLayer([2 2 2],"Name","en1_maxpool","Padding","same","Stride","[2 2 2]"},
    \text{batchNormalizationLayer("Name","en2_bn1"},
    \text{reluLayer("Name","en2_relu1"}),
    \text{convolution3dLayer([3 3 3],128,"Name","en2_conv2","Padding","same"},
    \text{reluLayer("Name","en2_relu2"});
\]
\[ lgraph = addLayers(lgraph,templayers); \]

Connect the Layer Branches

Connect all the branches of the network to create the network's graph.

\[ lgraph = connectLayers(lgraph,\text{en1_relu2},\text{en1_maxpool}); \]
\[ lgraph = connectLayers(lgraph,\text{en2_relu2},\text{en2_maxpool}); \]
\[ lgraph = connectLayers(lgraph,\text{en1_relui},\text{conc1/lin1}); \]
\[ lgraph = connectLayers(lgraph,\text{en2_relui},\text{conc2/lin2}); \]
\[ lgraph = connectLayers(lgraph,\text{de4_transconv},\text{conc3/lin2}); \]
\[ lgraph = connectLayers(lgraph,\text{de3_transconv},\text{conc1/lin2}); \]
\[ lgraph = connectLayers(lgraph,\text{de2_transconv},\text{conc1/lin2}); \]
Define Model

analyzeNetwork(Igraph)
Define Model
Train Model

```matlab
options = trainingOptions('adam','MaxEpochs',100, ...
    'InitialLearnRate',5e-4, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropPeriod',5, ...
    'LearnRateDropFactor',0.95, ...
    'ValidationData',patchdsVal, ...
    'ValidationFrequency',400, ...
    'Plots','training-progress', ...
    'MiniBatchSize',miniBatchSize, ...
    'DispatchInBackground',false, ...
    'ExecutionEnvironment', 'gpu');

[net,info] = trainNetwork(augpatchds,lgraph,options);
```
Train Model
“We’re Gonna Need a Bigger Machine”
profile on
[net,info] = trainNetwork(augpatchds,lgraph,options);
profile viewer

Train Model
Train Model

```matlab
options = trainingOptions('adam','MaxEpochs',100, ...
    'InitialLearnRate',5e-4, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropPeriod',5, ...
    'LearnRateDropFactor',0.95, ...
    'ValidationData',patchdsVal, ...
    'ValidationFrequency',400, ...
    'Plots','training-progress', ...
    'MiniBatchSize' miniBatchSize
    'DispatchInBackground', true, ...
    'ExecutionEnvironment', 'gpu'

[net,info] = trainNetwork(augpatchds,lgraph,options);
```
Multi-GPU Scaling

![Bar chart showing speed-up comparisons for different GPU configurations.](chart.png)
Segment Test Data

tic

id=1;
prediction = {};
while hasdata(voldsTest)
    tempVol = read(voldsTest);

    tempSeg = semanticseg(tempVol,net);

    % Get the non-brain region mask from the test image.
    volMask = tempVol(:,:,1)==0;
    % Set the non-brain region of the predicted label as background.
    tempSeg(volMask) = className(1);
    % Perform median filtering on the predicted label.
    tempSeg = medfilt3(uint8(tempSeg)-1);
    % Cast the filtered label to categorical.
    tempSeg = categorical(tempSeg,pixelLabelID,className);
    prediction(id) = tempSeg;

    id=id+1;
end
predictedLabels = prediction;

toc

Elapsed time is 97.429421 seconds.
Segment Test Data

tic
pool = gcp;
N = pool.NumWorkers;
parfor i = 1:N
    volPart = partition(voldsTest,N,i);
    id=1;
    prediction = {};
    while hasdata(volPart)
        tempVol = read(volPart);
        tempSeg = semanticseg(tempVol,net);

        % Get the non-brain region mask from the test image.
        volMask = tempVol(:,:,1)==0;
        % Set the non-brain region of the predicted label as background.
        tempSeg(volMask) = classNames(1);
        % Perform median filtering on the predicted label.
        tempSeg = medfilt3(uint8(tempSeg)-1);
        % Cast the filtered label to categorical.
        tempSeg = categorical(tempSeg,pixelLabelID,classNames);
        prediction(id) = tempSeg;
        id=id+1;
    end
    predictedLabels = [predictedLabels prediction];
end
toc

Elapsed time is 18.603188 seconds.
Visualize Results

```matlab
for zID = 1:size(volume,3)
    sliceGround = labeloverlay(volume(:,:,zID),labelGT(:,:,zID));
    slicePred = labeloverlay(volume(:,:,zID),labelPred(:,:,zID));
    montage({sliceGround,slicePred},'Size',[1 2],'BorderSize',5);
    drawnow
    M(i) = getframe;
end
movie(M)
```
Deep Learning Workflow

Access Data → Preprocess → Select Network → Train/Test → Deploy
Deploying an AI System

See - Deploying AI on Jetson Xavier/DRIVE Xavier with TensorRT and MATLAB
Jaya Shankar & Avi Nehemiah
result = classifyGenre(signal)
result = segment3DScan(scan)

POST-request
{"function":"classifyGenre",
"nargout":1,
"rhs":[signaldata]"}
In Summary

- Simple and extensive analysis, pre-processing and visualization
- Flexible training environment
- Easy to move your workflow anywhere with NGC
- Verification and validation tools all across the workflow
- Supports your design to deployment workflow

Thank you

Deep Learning for Image Classification and Time-Series Forecasting in MATLAB
Pitambar Dayal - Thursday, Mar 21, 10:00 AM - 12:00 PM – SJCC Room LL21E