# Deep Learning Using Bayesian Optimization as Hyperparameter Tuning

This exercise shows how to apply Bayesian optimization to deep learning and find optimal network hyperparameters and training options for convolutional neural networks.

## Choose Variables to Optimize

Choose which variables to optimize using Bayesian optimization, and specify the ranges to search in. Also, specify whether the variables are integers and whether to search the interval in logarithmic space. Optimize the following variables:

```
optimVars = [
    optimizableVariable('SectionDepth',[1 3],'Type','integer')
    optimizableVariable('InitialLearnRate',[1e-2 1],'Transform','log')
    optimizableVariable('Momentum',[0.8 0.98])];
    %optimizableVariable('L2Regularization',[1e-10 1e-2],'Transform','log')];
```

## Perform Bayesian Optimization

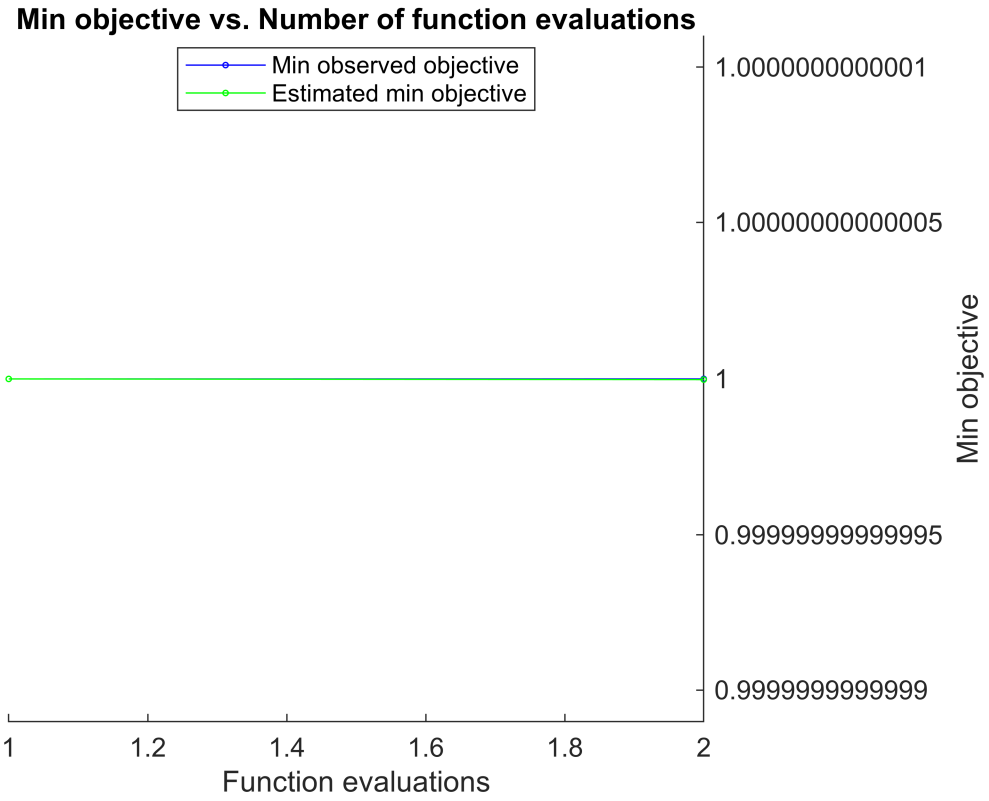
Create the objective function for the Bayesian optimizer, using the training and validation data as inputs. The objective function trains a convolutional neural network and returns the classification error on the validation set. This function is defined at the end of this script. Because `bayesopt` uses the error rate on the validation set to choose the best model, it is possible that the final network overfits on the validation set. The final chosen model is then tested on the independent test set to estimate the generalization error. Data is from the previous exercise.

```
ObjFcn = makeObjFcn(xTrain,xTest,label_Train,label_Test);
```

Perform Bayesian optimization by minimizing the classification error on the validation set.

```
BayesObject = bayesopt(ObjFcn,optimVars, ...
    'MaxTime',14*60*60, ...
    'IsObjectiveDeterministic',false, ...
    'UseParallel',false);
```

```
|=====================================================================================================================
| Iter | Eval   | Objective   | Objective   | BestSoFar   | BestSoFar   | SectionDepth | InitialLearn-|     Momentu
|      | result |             | runtime     | (observed)  | (estim.)    |              | Rate         |
|=====================================================================================================================
|    1 | Best   |           1 |      101.61 |           1 |           1 |            2 |      0.52871 |      0.8073
|    2 | Accept |           1 |      92.822 |           1 |           1 |            2 |     0.086061 |      0.9440
```




## Evaluate Final Network

Load the best network found in the optimization and its validation accuracy.

```
bestIdx = BayesObject.IndexOfMinimumTrace(end);
fileName = BayesObject.UserDataTrace{bestIdx};
savedStruct = load(fileName);
valError = savedStruct.valError
```

```
valError = 1
```

```
scoresTest = minibatchpredict(savedStruct.trainedNet,xTest);
YPredicted = onehotdecode(scoresTest,classNames,2);

accuracy = mean(YPredicted == label_Test)
```

```
accuracy = 0.9812
```

## Export best selection to ONNX

```
exportONNXNetwork(savedStruct.trainedNet, "hpo_1.onnx");
```

### Objective Function for Optimization

Define the objective function for optimization.

```
function ObjFcn = makeObjFcn(XTrain,YTrain,XValidation,YValidation)
ObjFcn = @valErrorFun;
    function [valError,cons,fileName] = valErrorFun(optVars)
        imageSize = [8 8 1];
        numClasses = numel(unique(YValidation));
        classnames = unique(YValidation);
        numF = round(16/sqrt(optVars.SectionDepth));
        layers = [
            imageInputLayer(imageSize)

            % The spatial input and output sizes of these convolutional
            % layers are 8-by-8, and the following max pooling layer
            % reduces this to 4-by-4.
            convBlock(3,numF,optVars.SectionDepth)
            maxPooling2dLayer(1,'Stride',2,'Padding','same')

            % The spatial input and output sizes of these convolutional
            % layers are 16-by-16, and the following max pooling layer
            % reduces this to 8-by-8.
            convBlock(3,2*numF,optVars.SectionDepth)
            maxPooling2dLayer(1,'Stride',2,'Padding','same')

            % The spatial input and output sizes of these convolutional
            % layers are 8-by-8. The global average pooling layer averages
            % over the 8-by-8 inputs, giving an output of size
            % 1-by-1-by-4*initialNumFilters. With a global average
            % pooling layer, the final classification output is only
            % sensitive to the total amount of each feature present in the
            % input image, but insensitive to the spatial positions of the
            % features.
            convBlock(3,4*numF,optVars.SectionDepth)
            averagePooling2dLayer(2)

            % Add the fully connected layer and the final softmax and
            % classification layers.
            fullyConnectedLayer(numClasses)
            softmaxLayer];

        options = trainingOptions("sgdm", ...
            ExecutionEnvironment="cpu", ...
            InitialLearnRate = optVars.InitialLearnRate, ...
            Momentum = optVars.Momentum, ...
            ValidationData={YTrain,YValidation}, ...
            MaxEpochs = 60, ...
            LearnRateSchedule='piecewise', ...
            LearnRateDropPeriod = 40, ...
            LearnRateDropFactor = 0.1, ...
            Verbose=false);
```

```
        imageAugmenter = imageDataAugmenter('RandRotation',[-90,90]);
        datasource =
augmentedImageDatastore(imageSize,XTrain,XValidation,'DataAugmentation',imageAugment
er);

        trainedNet = trainnet(datasource,layers,"crossentropy", options);
        close(findall(groot,'Tag','NNET_CNN_TRAININGPLOT_UIFIGURE'))
        scoresTest = minibatchpredict(trainedNet,YTrain);
        YPredicted = onehotdecode(scoresTest,int8(classnames),2);

        valError = 1 - mean(YPredicted == YValidation);
        fileName = num2str(valError) + ".mat";
        save(fileName,'trainedNet','valError','options')
        cons = [];

    end
end
function layers = convBlock(filterSize,numFilters,numConvLayers)
layers = [
    convolution2dLayer(filterSize,numFilters,'Padding','same')
    batchNormalizationLayer
    reluLayer];
layers = repmat(layers,numConvLayers,1);
end
```