```
% Define the function trainAndEvaluateNetwork
function loss = trainNetworkForDeeplearning(params, dsTrain, dsVal, imdsVal,
num_classes, filter_size, image_size, timePoolSize, dropout_rate, Maxpool_value)

    num_layers = params.num_layers;
    num_filters = params.num_filters;

    % define the network layers
    layers1 = [
        imageInputLayer([image_size 3])
    ];


    % add the convolutional layers
    for i = 1:num_layers
        layers1 = [
            layers1
            convolution2dLayer(filter_size, num_filters, Padding="same")
            batchNormalizationLayer
            reluLayer
        ];


        if i <= Maxpool_value
        layers1 = [
                layers1
                maxPooling2dLayer(filter_size, Stride=2, Padding="same")
            ];
        end

        num_filters = num_filters * 2;  % double the number of filters for the next
layer
    end


    % add the rest of the layers
    layers1 = [
        layers1
        maxPooling2dLayer([timePoolSize,1])
        dropoutLayer(dropout_rate)
        fullyConnectedLayer(num_classes)
        softmaxLayer
        classificationLayer
    ];

    % training options
    options = trainingOptions('adam', ...
        "MiniBatchSize",30, ...
        'InitialLearnRate',0.001, ...
        'MaxEpochs',15, ...
```

```
        'Shuffle','every-epoch', ...
        'ValidationData',dsVal, ...
        'ValidationFrequency',10, ...
        'Verbose',false, ...  % turn off text output
        'Plots','none', ...  % turn off plots
        'ExecutionEnvironment','gpu');

    % train the network
    net = trainNetwork(dsTrain,layers1,options);

    % classify the validation output using the trained network
    [YPred,probs] = classify(net,dsVal);

    % extract ground truth labels
    YVal = imdsVal.Labels;

    % calculate the loss
    loss = 1 - mean(YPred == YVal);  % loss is 1 - accuracy
end
```