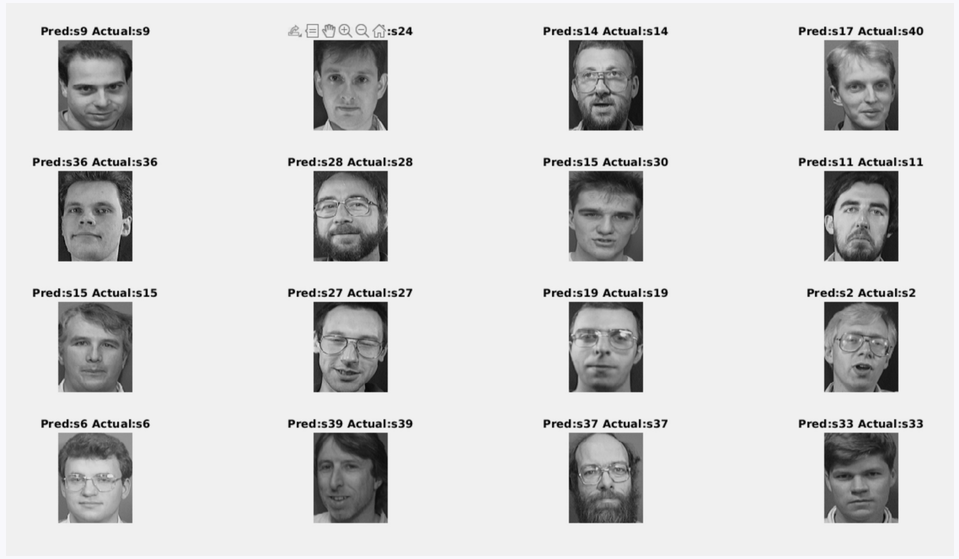
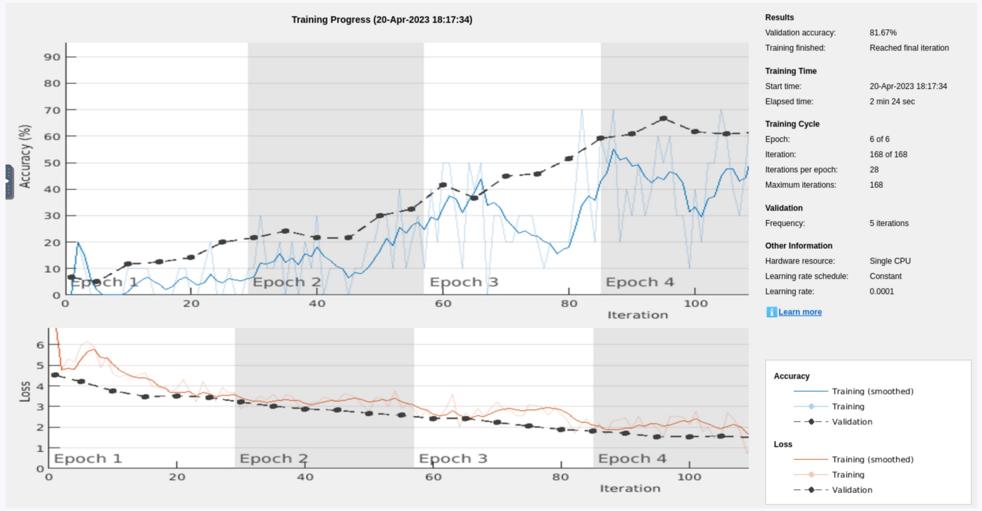
Proj 3





Results

Validation accuracy:	81.67%
Training finished:	Reached final iteration

Training Time

Start time:	20-Apr-2023 18:17:34
Elapsed time:	2 min 24 sec

Training Cycle

Epoch:	6 of 6
Iteration:	168 of 168
Iterations per epoch:	28
Maximum iterations:	168

Validation

Frequency:	5 iterations

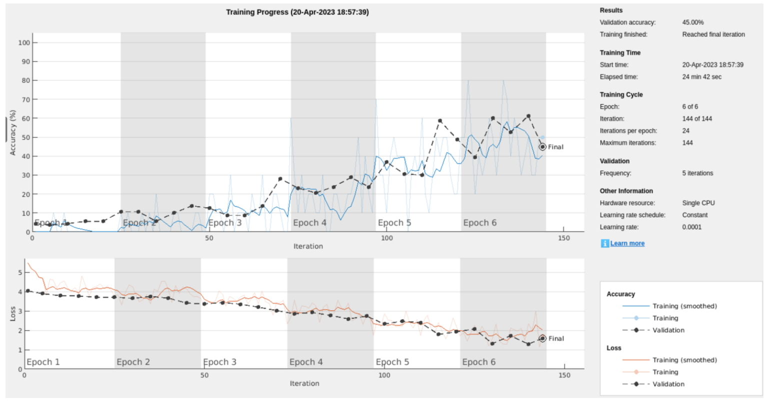
Other Information

Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.0001

Learn more

Alexnet

VGG19



Results

Validation accuracy:	45.00%
Training finished:	Reached final iteration

Training Time

Start time:	20-Apr-2023 18:57:39
Elapsed time:	24 min 42 sec

Training Cycle

Epoch:	6 of 6
Iteration:	144 of 144
Iterations per epoch:	24
Maximum iterations:	144

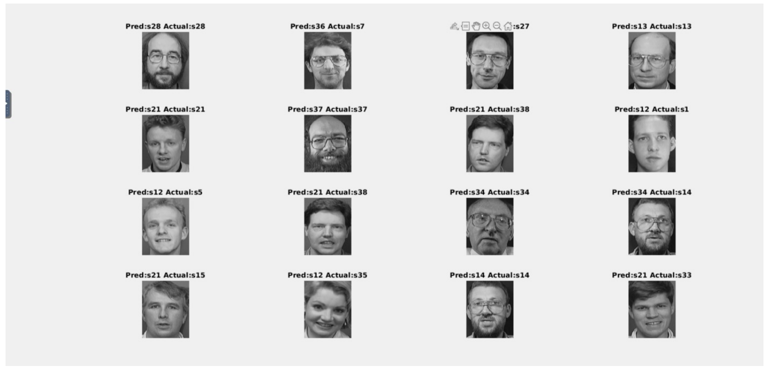
Validation

Frequency:	5 iterations

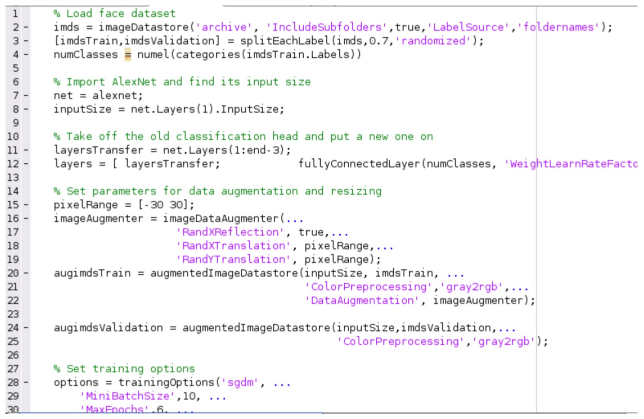
Other Information

Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.0001

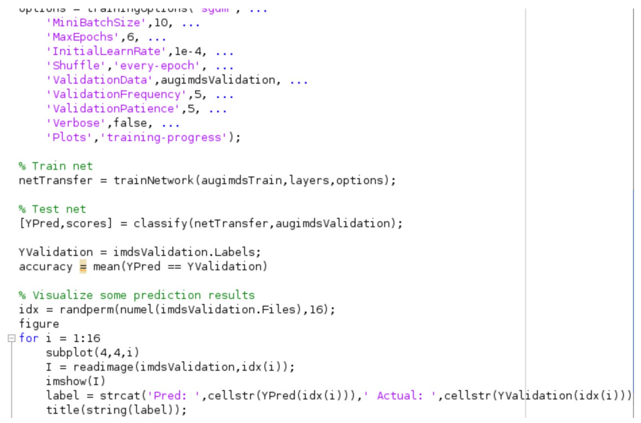
Learn more



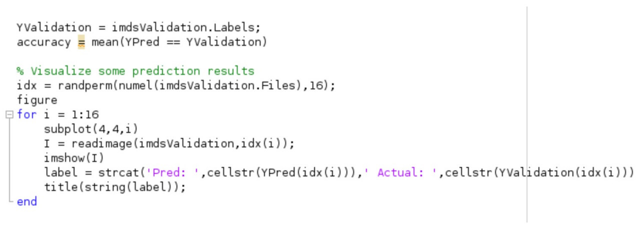
Alexnet code



```matlab
% Load face dataset
imds = imageDatastore('archive', 'IncludeSubfolders',true,'LabelSource','foldernames');
[imdsTrain,imdsValidation] = splitEachLabel(imds,0.7,'randomized');
numClasses = numel(categories(imdsTrain.Labels))

% Import AlexNet and find its input size
net = alexnet;
inputSize = net.Layers(1).InputSize;

% Take off the old classification head and put a new one on
layersTransfer = net.Layers(1:end-3);
layers = [ layersTransfer;           fullyConnectedLayer(numClasses, 'WeightLearnRateFacto

% Set parameters for data augmentation and resizing
pixelRange = [-30 30];
imageAugmenter = imageDataAugmenter(...
                   'RandXReflection', true,...
                   'RandXTranslation', pixelRange,...
                   'RandYTranslation', pixelRange);
augimdsTrain = augmentedImageDatastore(inputSize, imdsTrain, ...
                                       'ColorPreprocessing','gray2rgb',...
                                       'DataAugmentation', imageAugmenter);

augimdsValidation = augmentedImageDatastore(inputSize,imdsValidation,...
                                            'ColorPreprocessing','gray2rgb');

% Set training options
options = trainingOptions('sgdm', ...
    'MiniBatchSize',10, ...
    'MaxEpochs',6, ...
```

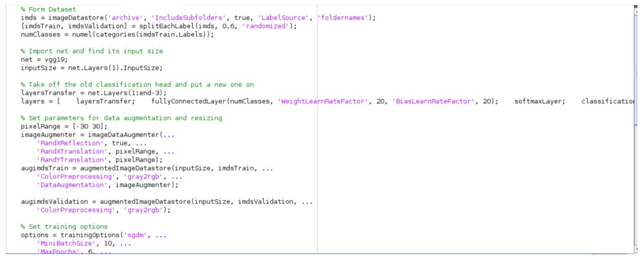


```matlab
options = trainingOptions('sgdm', ...
    'MiniBatchSize',10, ...
    'MaxEpochs',6, ...
    'InitialLearnRate',1e-4, ...
    'Shuffle','every-epoch', ...
    'ValidationData',augimdsValidation, ...
    'ValidationFrequency',5, ...
    'ValidationPatience',5, ...
    'Verbose',false, ...
    'Plots','training-progress');

% Train net
netTransfer = trainNetwork(augimdsTrain,layers,options);

% Test net
[YPred,scores] = classify(netTransfer,augimdsValidation);

YValidation = imdsValidation.Labels;
accuracy = mean(YPred == YValidation)

% Visualize some prediction results
idx = randperm(numel(imdsValidation.Files),16);
figure
for i = 1:16
    subplot(4,4,i)
    I = readimage(imdsValidation,idx(i));
    imshow(I)
    label = strcat('Pred: ',cellstr(YPred(idx(i))),' Actual: ',cellstr(YValidation(idx(i)))
    title(string(label));
```

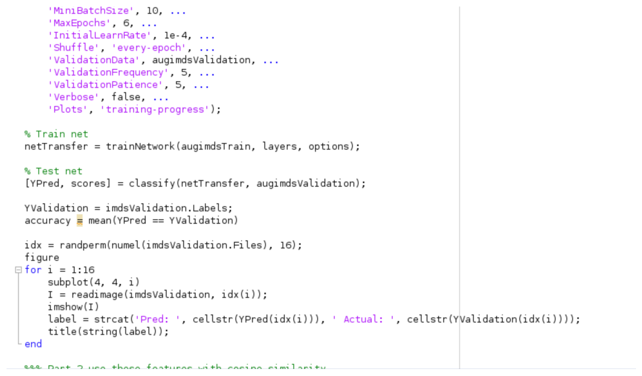


```matlab
YValidation = imdsValidation.Labels;
accuracy = mean(YPred == YValidation)

% Visualize some prediction results
idx = randperm(numel(imdsValidation.Files),16);
figure
for i = 1:16
    subplot(4,4,i)
    I = readimage(imdsValidation,idx(i));
    imshow(I)
    label = strcat('Pred: ',cellstr(YPred(idx(i))),' Actual: ',cellstr(YValidation(idx(i)))
    title(string(label));
end
```

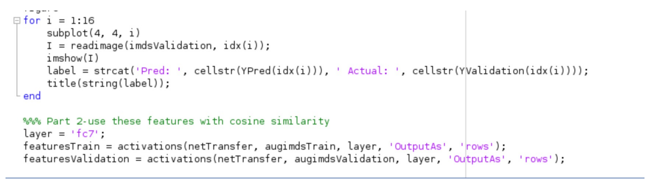
VGG19 code



```
% Form Dataset
imds = imageDatastore('archive', 'IncludeSubfolders', true, 'LabelSource', 'foldernames');
[imdsTrain, imdsValidation] = splitEachLabel(imds, 0.6, 'randomized');
numClasses = numel(categories(imdsTrain.Labels));

% Import net and find its input size
net = vgg19;
inputSize = net.Layers(1).InputSize;

% Take off the old classification head and put a new one on
layersTransfer = net.Layers(1:end-3);
layers = [    layersTransfer;    fullyConnectedLayer(numClasses, 'WeightLearnRateFactor', 20, 'BiasLearnRateFactor', 20);    softmaxLayer;    classificatio

% Set parameters for data augmentation and resizing
pixelRange = [-30 30];
imageAugmenter = imageDataAugmenter(...
    'RandXReflection', true, ...
    'RandXTranslation', pixelRange, ...
    'RandYTranslation', pixelRange);
augimdsTrain = augmentedImageDatastore(inputSize, imdsTrain, ...
    'ColorPreprocessing', 'gray2rgb', ...
    'DataAugmentation', imageAugmenter);

augimdsValidation = augmentedImageDatastore(inputSize, imdsValidation, ...
    'ColorPreprocessing', 'gray2rgb');

% Set training options
options = trainingOptions('sgdm', ...
    'MiniBatchSize', 10, ...
    'MaxEpochs', 6, ...
```



```
    'MiniBatchSize', 10, ...
    'MaxEpochs', 6, ...
    'InitialLearnRate', 1e-4, ...
    'Shuffle', 'every-epoch', ...
    'ValidationData', augimdsValidation, ...
    'ValidationFrequency', 5, ...
    'ValidationPatience', 5, ...
    'Verbose', false, ...
    'Plots', 'training-progress');

% Train net
netTransfer = trainNetwork(augimdsTrain, layers, options);

% Test net
[YPred, scores] = classify(netTransfer, augimdsValidation);

YValidation = imdsValidation.Labels;
accuracy = mean(YPred == YValidation)

idx = randperm(numel(imdsValidation.Files), 16);
figure
for i = 1:16
    subplot(4, 4, i)
    I = readimage(imdsValidation, idx(i));
    imshow(I)
    label = strcat('Pred: ', cellstr(YPred(idx(i))), ' Actual: ', cellstr(YValidation(idx(i))));
    title(string(label));
end

%%% Part 2-use these features with cosine similarity
```



```
for i = 1:16
    subplot(4, 4, i)
    I = readimage(imdsValidation, idx(i));
    imshow(I)
    label = strcat('Pred: ', cellstr(YPred(idx(i))), ' Actual: ', cellstr(YValidation(idx(i))));
    title(string(label));
end

%%% Part 2-use these features with cosine similarity
layer = 'fc7';
featuresTrain = activations(netTransfer, augimdsTrain, layer, 'OutputAs', 'rows');
featuresValidation = activations(netTransfer, augimdsValidation, layer, 'OutputAs', 'rows');
```

Analysis

- AlexNet was fairly accurate, able to get up to 81% validation accuracy, which was much better than the VGG19, which only got to 45%. VGG19 also took considerably longer, from 2.5 to 25, an order of magnitude higher than the AlexNet so it did not perform as well as the AlexNet