```
addpath('./data');
```

Load the Dataset

```
load('trunk12-patches.mat');

% Extract training and testing data and labels
trainData = trunk.train.data;
trainLabels = trunk.train.labels;
testData = trunk.test.data;
testLabels = trunk.test.labels;

% Normalize the images to the range [0, 1]
trainData = double(trainData) / 255;
testData = double(testData) / 255;
```

Basic CNN Architecture

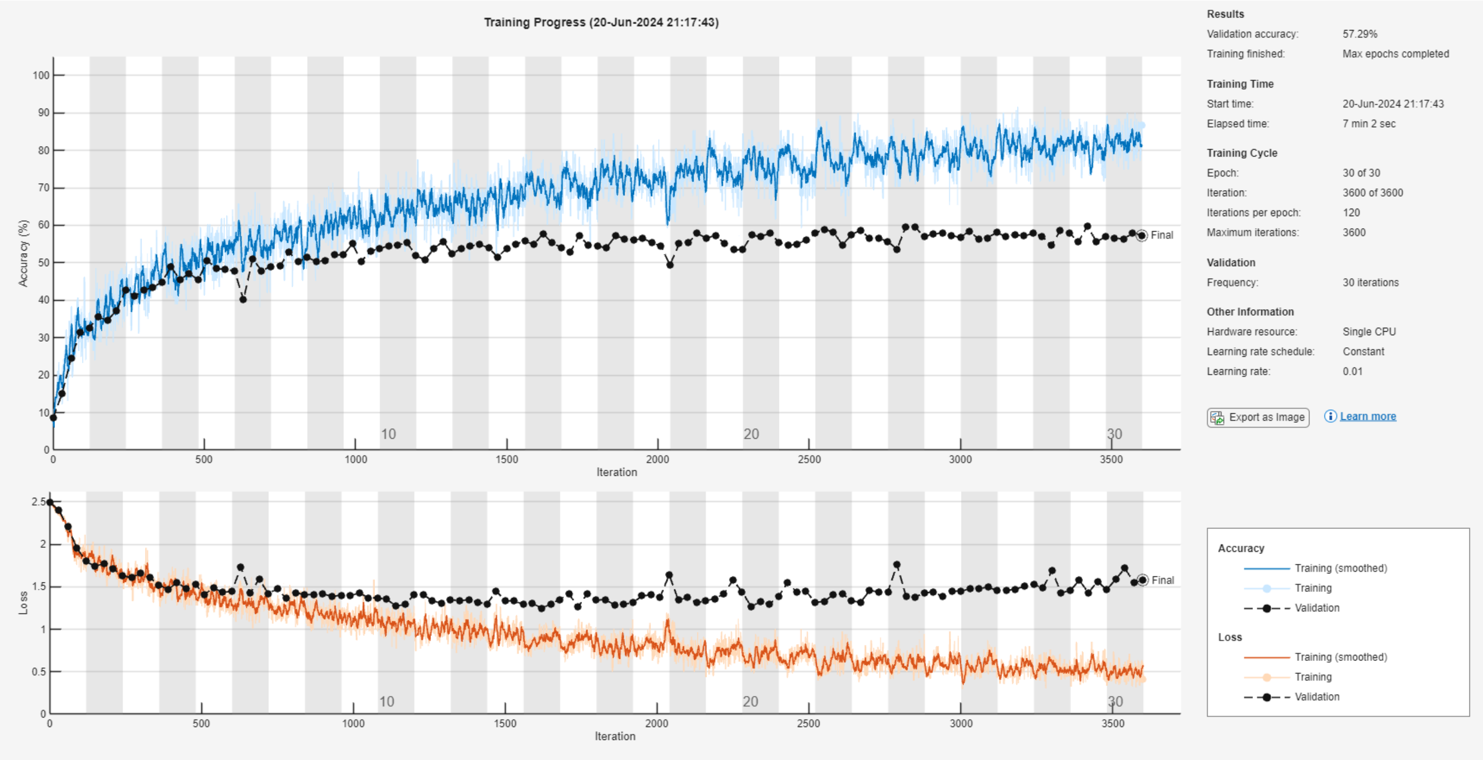
```
layers = [
    imageInputLayer([40 40 3])

    convolution2dLayer(3, 8, 'Padding', 'same')

    maxPooling2dLayer(2, 'Stride', 2)

    convolution2dLayer(3, 16, 'Padding', 'same')

    maxPooling2dLayer(2, 'Stride', 2)

    fullyConnectedLayer(64)

    fullyConnectedLayer(12)
    softmaxLayer
    classificationLayer
];
```

Specify Training Options

```
options = trainingOptions('sgdm', ...
    'InitialLearnRate',0.01, ...
    'MaxEpochs',30, ...
    'Shuffle','every-epoch', ...
    'ValidationData',{testData, testLabels}, ...
    'ValidationFrequency',30, ...
    'Verbose',false, ...
    'Plots','training-progress');
```

## Model Training

```
net = trainNetwork(trainData, trainLabels, layers, options);
```



**Results**

Validation accuracy: 57.29%

Training finished: Max epochs completed

**Training Time**

Start time: 20-Jun-2024 21:17:43

Elapsed time: 7 min 2 sec

**Training Cycle**

Epoch: 30 of 30

Iteration: 3600 of 3600

Iterations per epoch: 120

Maximum iterations: 3600

**Validation**

Frequency: 30 iterations

**Other Information**

Hardware resource: Single CPU

Learning rate schedule: Constant

Learning rate: 0.01

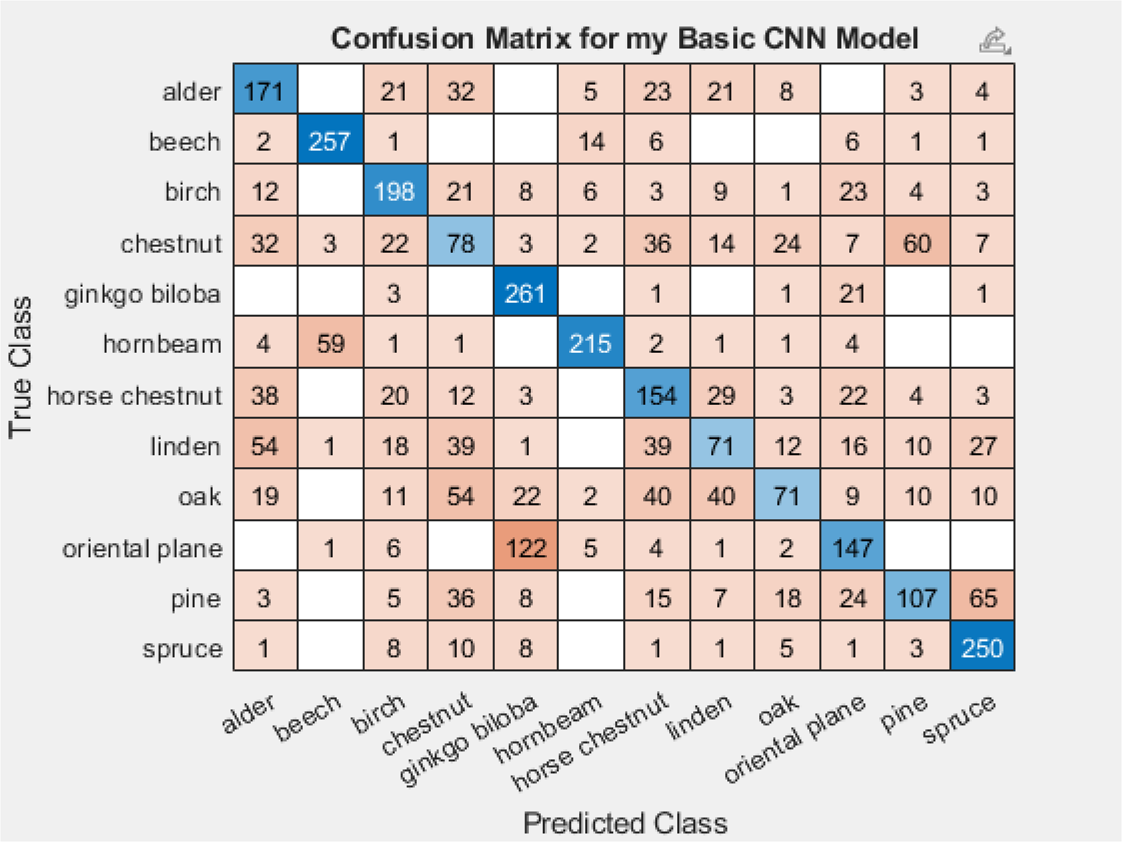
## Evaluate the Model

```
predictedLabels = classify(net, testData);
accuracy = sum(predictedLabels == testLabels) / numel(testLabels);
fprintf('Test accuracy: %.2f%%\n', accuracy * 100);
```

```
Test accuracy: 57.29%
```

## Generate confusion matrix

```
figure('Visible', 'on');
confusionchart(testLabels, predictedLabels);
title('Confusion Matrix for my Basic CNN Model');
```

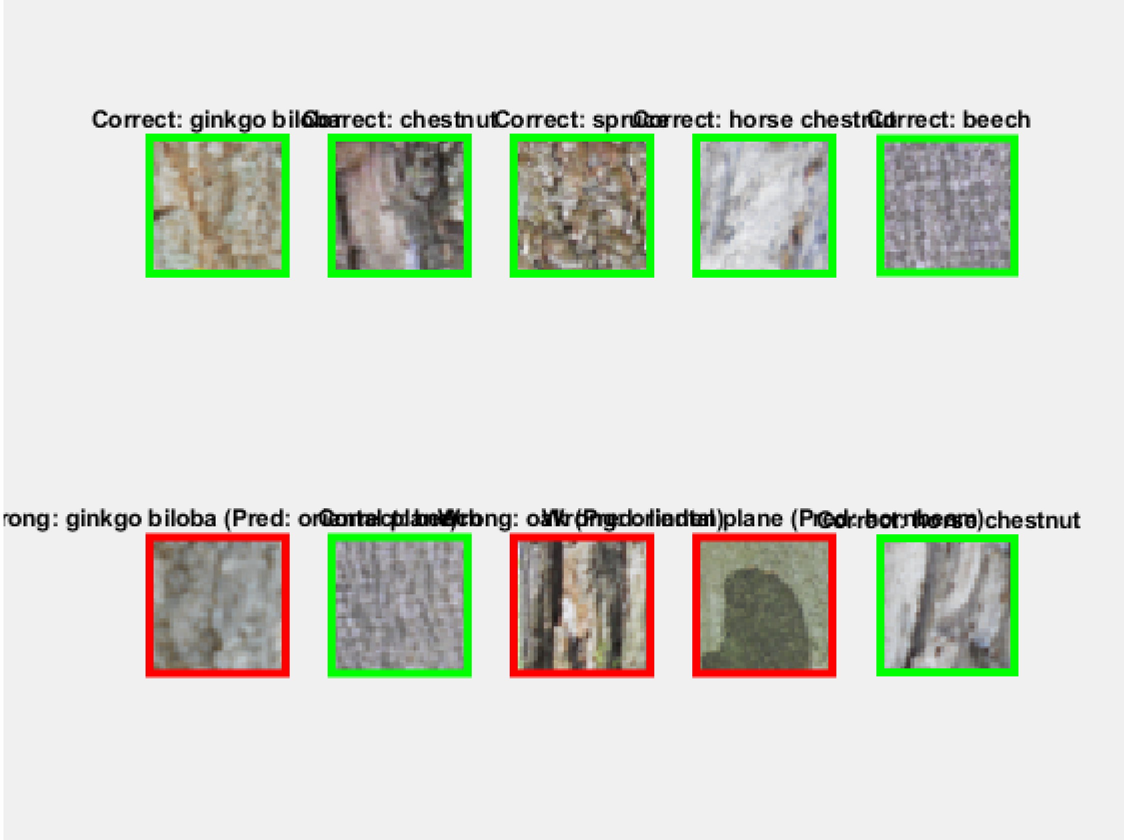


Confusion Matrix for my Basic CNN Model

| True Class \ Predicted Class | alder | beech | birch | chestnut | ginkgo biloba | hornbeam | horse chestnut | linden | oak | oriental plane | pine | spruce |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| alder | 171 |  | 21 | 32 |  | 5 | 23 | 21 | 8 |  | 3 | 4 |
| beech | 2 | 257 | 1 |  |  | 14 | 6 |  |  | 6 | 1 | 1 |
| birch | 12 |  | 198 | 21 | 8 | 6 | 3 | 9 | 1 | 23 | 4 | 3 |
| chestnut | 32 | 3 | 22 | 78 | 3 | 2 | 36 | 14 | 24 | 7 | 60 | 7 |
| ginkgo biloba |  |  | 3 |  | 261 |  | 1 |  | 1 | 21 |  | 1 |
| hornbeam | 4 | 59 | 1 | 1 |  | 215 | 2 | 1 | 1 | 4 |  |  |
| horse chestnut | 38 |  | 20 | 12 | 3 |  | 154 | 29 | 3 | 22 | 4 | 3 |
| linden | 54 | 1 | 18 | 39 | 1 |  | 39 | 71 | 12 | 16 | 10 | 27 |
| oak | 19 |  | 11 | 54 | 22 | 2 | 40 | 40 | 71 | 9 | 10 | 10 |
| oriental plane |  | 1 | 6 |  | 122 | 5 | 4 | 1 | 2 | 147 |  |  |
| pine | 3 |  | 5 | 36 | 8 |  | 15 | 7 | 18 | 24 | 107 | 65 |
| spruce | 1 |  | 8 | 10 | 8 |  | 1 | 1 | 5 | 1 | 3 | 250 |

```
drawnow;
```

```
% Select a subset of test images for visualization
numSamples = 10; % Number of samples to visualize
sampleIndices = randperm(numel(testLabels), numSamples);

figure('Visible', 'on');
for i = 1:numSamples
    index = sampleIndices(i);
    img = testData(:,:,:,index);
    trueLabel = testLabels(index);
    predictedLabel = predictedLabels(index);

    subplot(2, numSamples/2, i);
    imshow(img, 'InitialMagnification', 'fit');

    if trueLabel == predictedLabel
        borderColor = 'g'; % Green for correct classification
        titleText = sprintf('Correct: %s', string(trueLabel));
    else
        borderColor = 'r'; % Red for incorrect classification
        titleText = sprintf('Wrong: %s (Pred: %s)', string(trueLabel),
string(predictedLabel));
    end
    rectangle('Position', [0.5, 0.5, size(img, 2), size(img, 1)], 'EdgeColor',
borderColor, 'LineWidth', 3);
    title(titleText);
end
```


Correct: ginkgo bilCorrect: chestnutCorrect: spruCorrect: horse chestnCorrect: beech
rong: ginkgo biloba (Pred: orCorrect: beechWrong: oak (Pred: orientalWrong: oriental plane (Pred: hornbeam)Correct: horse chestnut

```
drawnow;
```

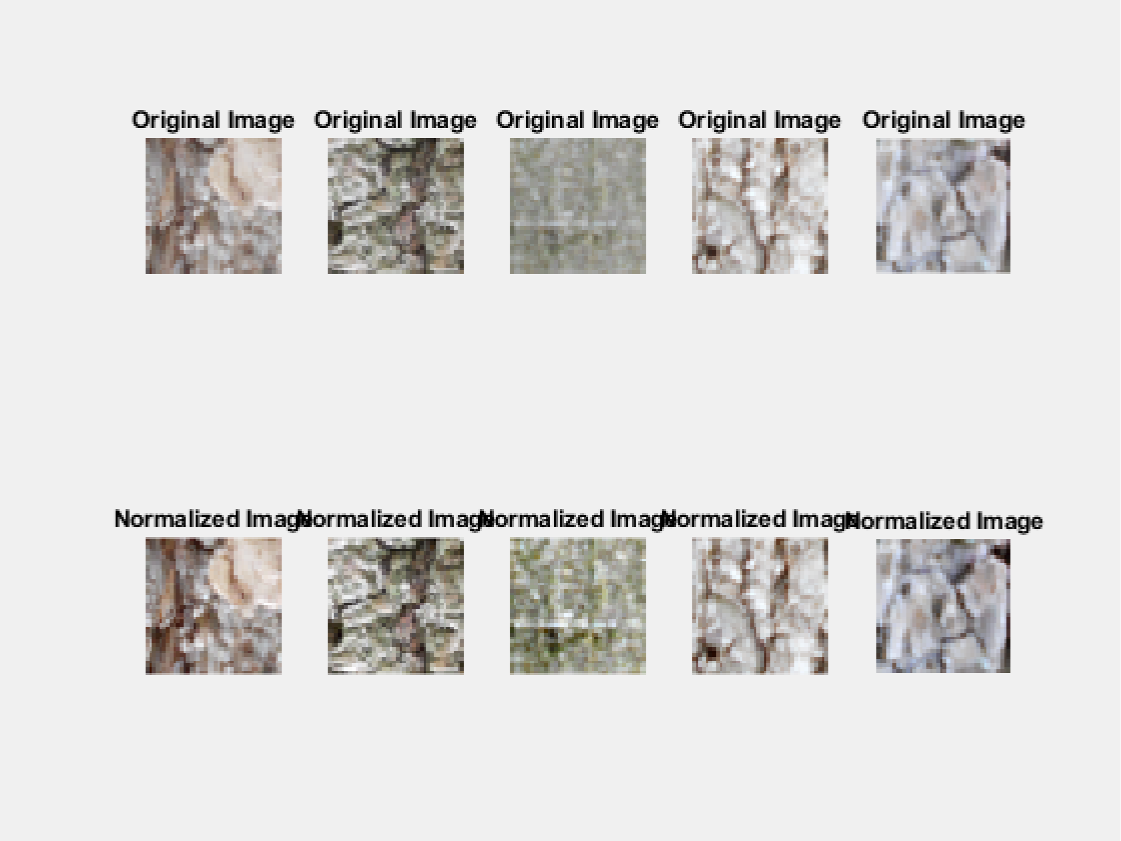
Improving by adding pre-processing (Normalisation)

```
% Normalize the images to mean 0 and variance 1
meanTrain = mean(trainData(:));
stdTrain = std(trainData(:));
trainData = (trainData - meanTrain) / stdTrain;
testData = (testData - meanTrain) / stdTrain; % Use the same mean and std as train
data
```

```
% Visualization of the original and normalized images
numSamples = 5; % Number of samples to visualize
sampleIndices = randperm(size(trunk.train.data, 4), numSamples);
sampleData = trunk.train.data(:,:,:,sampleIndices);
normalizedSampleData = double(sampleData) / 255;
normalizedSampleData = (normalizedSampleData - meanTrain) / stdTrain;
```

```
figure('Visible', 'on');
```

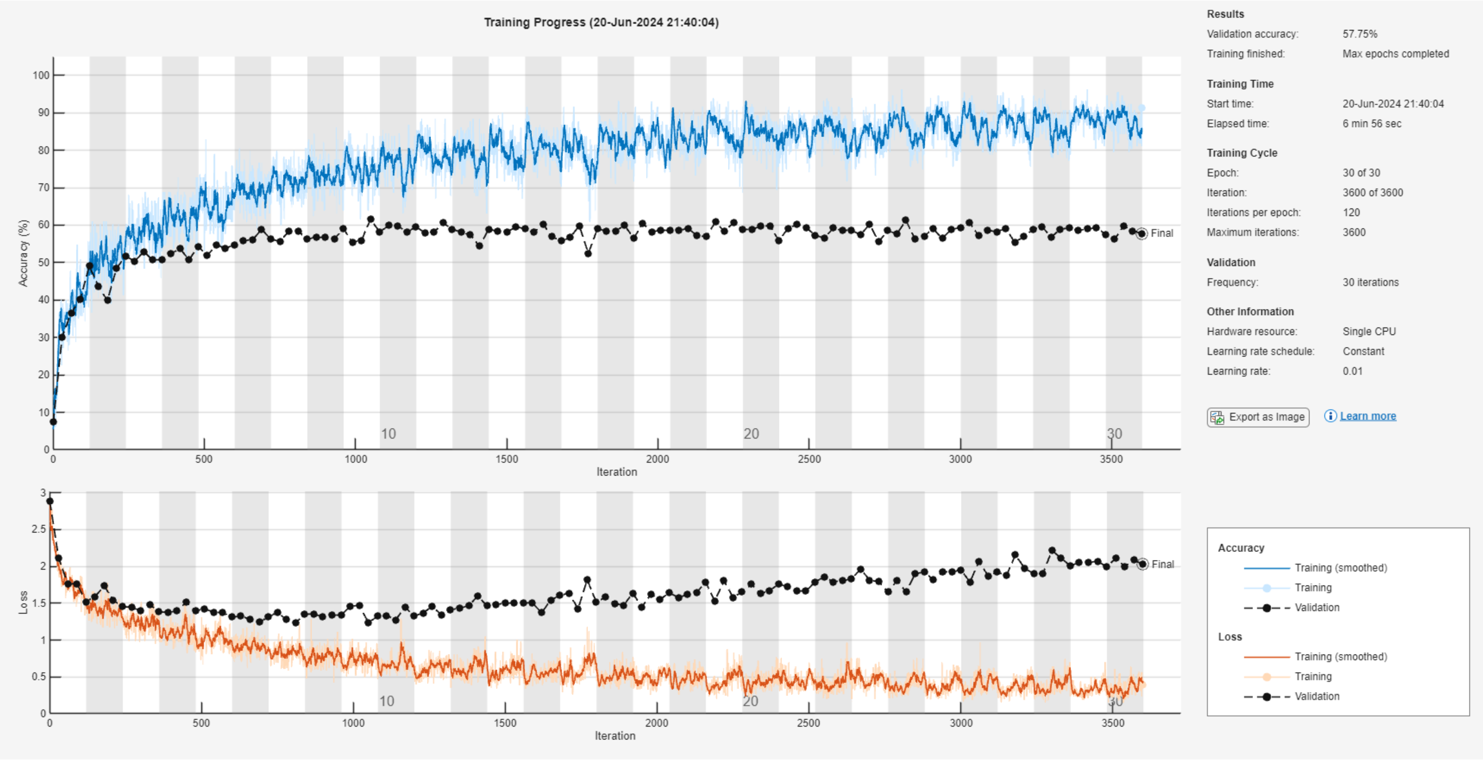
```
for i = 1:numSamples
    % Original image
    subplot(2, numSamples, i);
    imshow(uint8(sampleData(:,:,:,i)));
    title('Original Image');

    % Normalized image
    subplot(2, numSamples, numSamples + i);
    imshow(mat2gray(normalizedSampleData(:,:,:,i))); % mat2gray scales the image to
[0, 1] for display
    title('Normalized Image');
end
```



```
layers = [
    imageInputLayer([40 40 3])

    convolution2dLayer(3, 8, 'Padding', 'same')

    maxPooling2dLayer(2, 'Stride', 2)

    convolution2dLayer(3, 16, 'Padding', 'same')
    maxPooling2dLayer(2, 'Stride', 2)

    fullyConnectedLayer(64)

    fullyConnectedLayer(12)
    softmaxLayer
    classificationLayer
];
```

```
options = trainingOptions('sgdm', ...
    'InitialLearnRate',0.01, ...
    'MaxEpochs',30, ...
    'Shuffle','every-epoch', ...
    'ValidationData',{testData, testLabels}, ...
    'ValidationFrequency',30, ...
    'Verbose',false, ...
    'Plots','training-progress');
```

```
netImproved = trainNetwork(trainData, trainLabels, layers, options);
```



**Results**

Validation accuracy: 57.75%

Training finished: Max epochs completed

**Training Time**

Start time: 20-Jun-2024 21:40:04

Elapsed time: 6 min 56 sec

**Training Cycle**

Epoch: 30 of 30

Iteration: 3600 of 3600

Iterations per epoch: 120

Maximum iterations: 3600

**Validation**

Frequency: 30 iterations

**Other Information**

Hardware resource: Single CPU

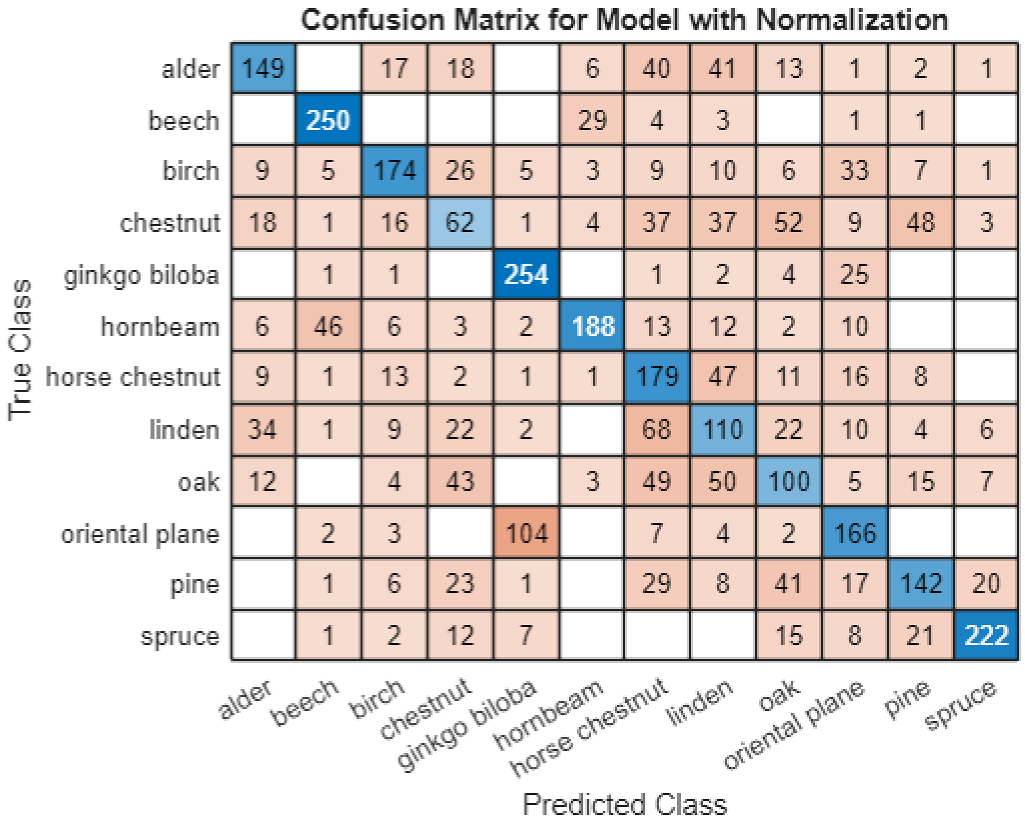
Learning rate schedule: Constant

Learning rate: 0.01

```
predictedLabels = classify(netImproved, testData);
accuracy = sum(predictedLabels == testLabels) / numel(testLabels);
fprintf('Test accuracy: %.2f%%\n', accuracy * 100);
```

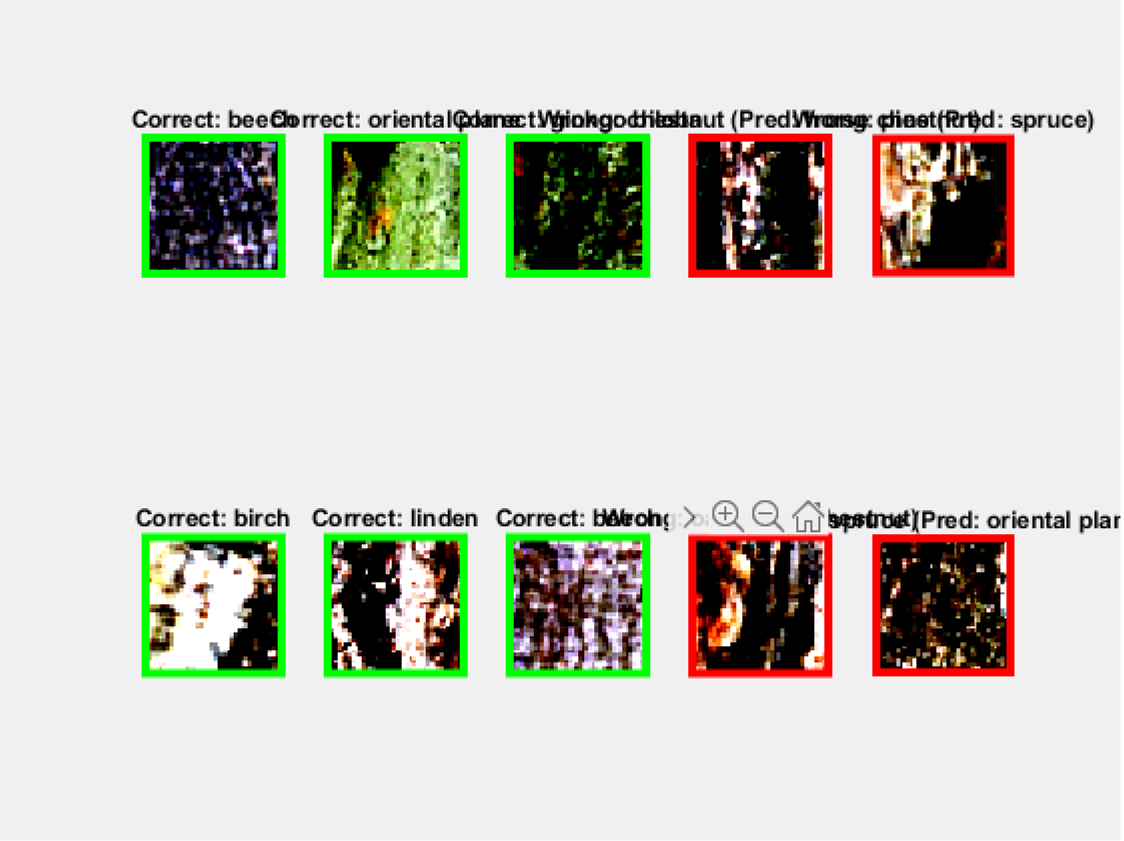
```
Test accuracy: 57.75%
```

```
figure;
confusionchart(testLabels, predictedLabels);
title('Confusion Matrix for Model with Normalization');
```



| True Class \ Predicted Class | alder | beech | birch | chestnut | ginkgo biloba | hornbeam | horse chestnut | linden | oak | oriental plane | pine | spruce |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| alder | 149 |  | 17 | 18 |  | 6 | 40 | 41 | 13 | 1 | 2 | 1 |
| beech |  | 250 |  |  |  | 29 | 4 | 3 |  | 1 | 1 |  |
| birch | 9 | 5 | 174 | 26 | 5 | 3 | 9 | 10 | 6 | 33 | 7 | 1 |
| chestnut | 18 | 1 | 16 | 62 | 1 | 4 | 37 | 37 | 52 | 9 | 48 | 3 |
| ginkgo biloba |  | 1 | 1 |  | 254 |  | 1 | 2 | 4 | 25 |  |  |
| hornbeam | 6 | 46 | 6 | 3 | 2 | 188 | 13 | 12 | 2 | 10 |  |  |
| horse chestnut | 9 | 1 | 13 | 2 | 1 | 1 | 179 | 47 | 11 | 16 | 8 |  |
| linden | 34 | 1 | 9 | 22 | 2 |  | 68 | 110 | 22 | 10 | 4 | 6 |
| oak | 12 |  | 4 | 43 |  | 3 | 49 | 50 | 100 | 5 | 15 | 7 |
| oriental plane |  | 2 | 3 |  | 104 |  | 7 | 4 | 2 | 166 |  |  |
| pine |  | 1 | 6 | 23 | 1 |  | 29 | 8 | 41 | 17 | 142 | 20 |
| spruce |  | 1 | 2 | 12 | 7 |  |  |  | 15 | 8 | 21 | 222 |

```
numSamples = 10;
sampleIndices = randperm(numel(testLabels), numSamples);

figure('Visible', 'on');
for i = 1:numSamples
    index = sampleIndices(i);
    img = testData(:,:,:,index);
    trueLabel = testLabels(index);
    predictedLabel = predictedLabels(index);

    subplot(2, numSamples/2, i);
    imshow(img, 'InitialMagnification', 'fit');

    if trueLabel == predictedLabel
        borderColor = 'g';
        titleText = sprintf('Correct: %s', string(trueLabel));
    else
        borderColor = 'r';
        titleText = sprintf('Wrong: %s (Pred: %s)', string(trueLabel),
string(predictedLabel));
    end
    rectangle('Position', [0.5, 0.5, size(img, 2), size(img, 1)], 'EdgeColor',
borderColor, 'LineWidth', 3);
    title(titleText);
end
```



```
drawnow;
```

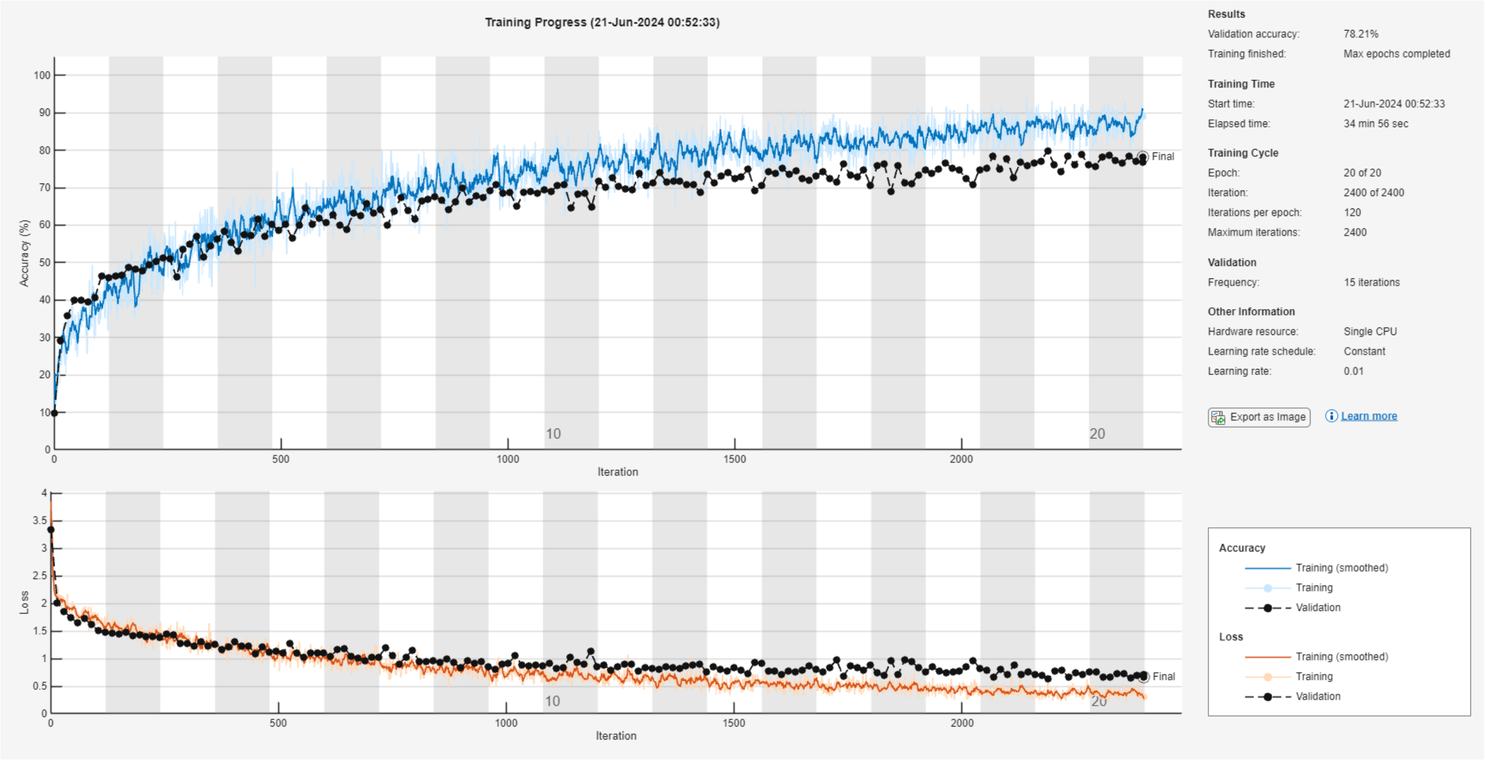
Not Normalised Data with Enhanced CNN model

```
trainData_not_normalised = trunk.train.data;
trainLabels = trunk.train.labels;
testData_not_normalised = trunk.test.data;
testLabels = trunk.test.labels;
```

```
layers = [
    imageInputLayer([40 40 3], 'Normalization', 'none')

    convolution2dLayer(3, 32, 'Padding', 'same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2, 'Stride', 2)
    convolution2dLayer(3, 64, 'Padding', 'same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2, 'Stride', 2)

    convolution2dLayer(3, 128, 'Padding', 'same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2, 'Stride', 2)

    fullyConnectedLayer(256)
    dropoutLayer(0.5) % Add dropout to prevent overfitting
    reluLayer

    fullyConnectedLayer(numel(unique(trainLabels)))
    softmaxLayer
    classificationLayer
];
```

```
options = trainingOptions('sgdm', ...
    'InitialLearnRate',0.01, ...
    'MaxEpochs',20, ...
    'Shuffle','every-epoch', ...
    'ValidationData',{testData_not_normalised, testLabels}, ...
    'ValidationFrequency',15, ...
    'Verbose',false, ...
    'Plots','training-progress');
```

```
% Train the network
netImproved2 = trainNetwork(trainData_not_normalised, trainLabels, layers, options);
```



Results
Validation accuracy: 78.21%
Training finished: Max epochs completed

Training Time
Start time: 21-Jun-2024 00:52:33
Elapsed time: 34 min 56 sec

Training Cycle
Epoch: 20 of 20
Iteration: 2400 of 2400
Iterations per epoch: 120
Maximum iterations: 2400

Validation
Frequency: 15 iterations

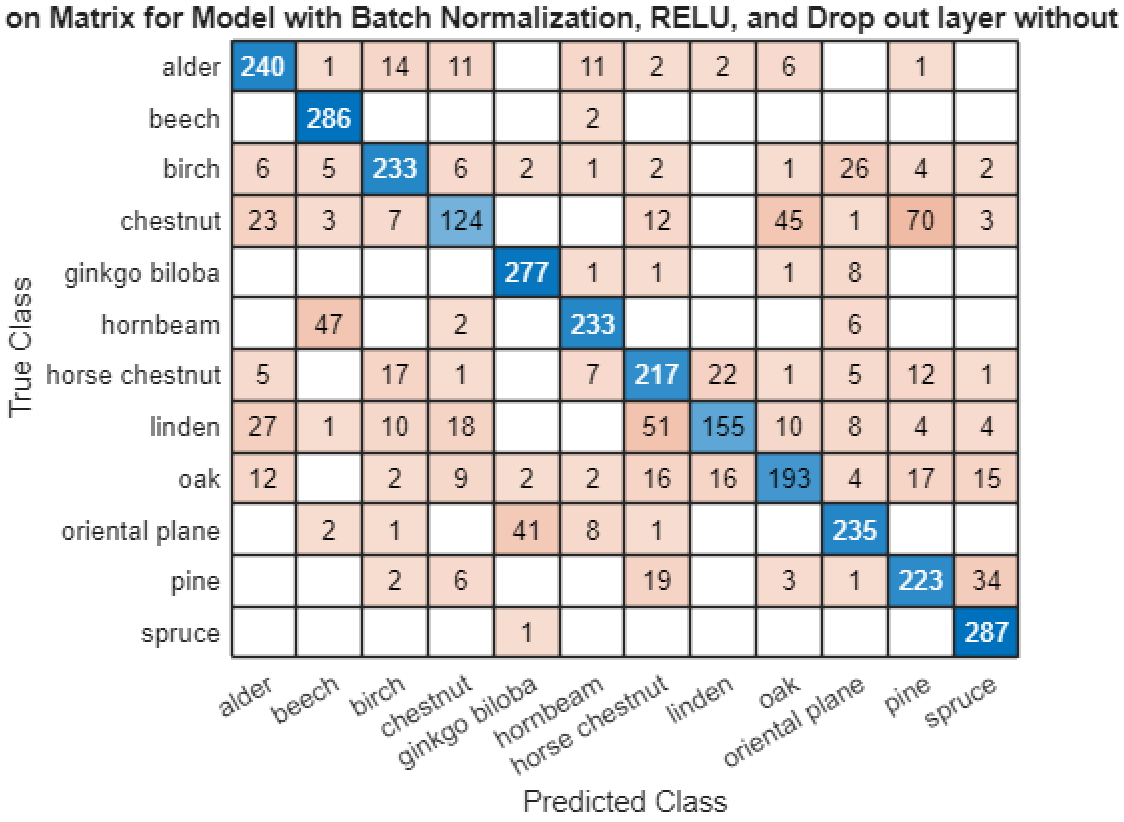
Other Information
Hardware resource: Single CPU
Learning rate schedule: Constant
Learning rate: 0.01

```
predictedLabels = classify(netImproved2, testData_not_normalised);
accuracy = sum(predictedLabels == testLabels) / numel(testLabels);
fprintf('Test accuracy: %.2f%%\n', accuracy * 100);
```

```
Test accuracy: 78.21%
```

```
figure;
confusionchart(testLabels, predictedLabels);
title('Confusion Matrix for Model with Batch Normalization, RELU, and Drop out
layer without normalised Data');
```

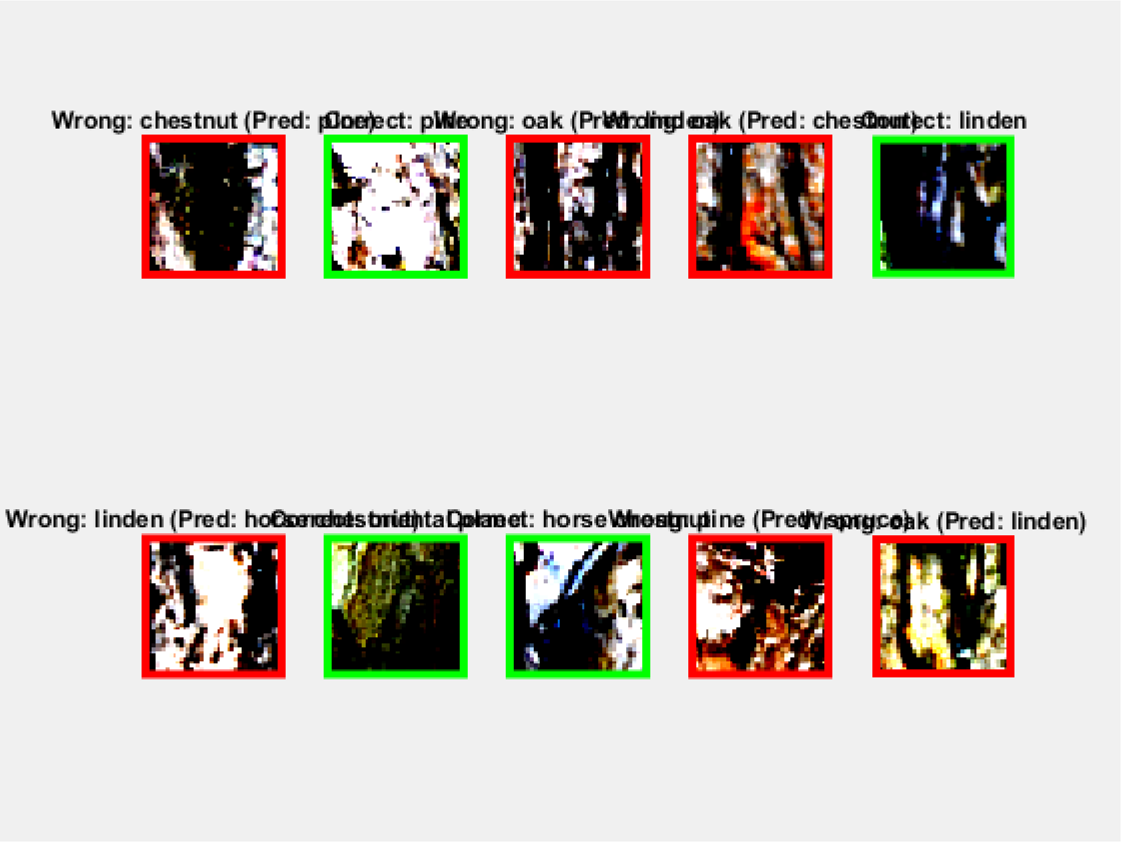
on Matrix for Model with Batch Normalization, RELU, and Drop out layer without



| True Class | alder | beech | birch | chestnut | ginkgo biloba | hornbeam | horse chestnut | linden | oak | oriental plane | pine | spruce |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| alder | 240 | 1 | 14 | 11 |  | 11 | 2 | 2 | 6 |  | 1 |  |
| beech |  | 286 |  |  |  | 2 |  |  |  |  |  |  |
| birch | 6 | 5 | 233 | 6 | 2 | 1 | 2 |  | 1 | 26 | 4 | 2 |
| chestnut | 23 | 3 | 7 | 124 |  |  | 12 |  | 45 | 1 | 70 | 3 |
| ginkgo biloba |  |  |  |  | 277 | 1 | 1 |  | 1 | 8 |  |  |
| hornbeam |  | 47 |  | 2 |  | 233 |  |  |  | 6 |  |  |
| horse chestnut | 5 |  | 17 | 1 |  | 7 | 217 | 22 | 1 | 5 | 12 | 1 |
| linden | 27 | 1 | 10 | 18 |  |  | 51 | 155 | 10 | 8 | 4 | 4 |
| oak | 12 |  | 2 | 9 | 2 | 2 | 16 | 16 | 193 | 4 | 17 | 15 |
| oriental plane |  | 2 | 1 |  | 41 | 8 | 1 |  |  | 235 |  |  |
| pine |  |  | 2 | 6 |  |  | 19 |  | 3 | 1 | 223 | 34 |
| spruce |  |  |  |  | 1 |  |  |  |  |  |  | 287 |

```
numSamples = 10;
sampleIndices = randperm(numel(testLabels), numSamples);

figure('Visible', 'on');
for i = 1:numSamples
    index = sampleIndices(i);
    img = testData(:,:,:,index);
    trueLabel = testLabels(index);
    predictedLabel = predictedLabels(index);

    subplot(2, numSamples/2, i);
    imshow(img, 'InitialMagnification', 'fit');

    if trueLabel == predictedLabel
        borderColor = 'g'; % Green for correct classification
        titleText = sprintf('Correct: %s', string(trueLabel));
    else
        borderColor = 'r'; % Red for incorrect classification
        titleText = sprintf('Wrong: %s (Pred: %s)', string(trueLabel),
string(predictedLabel));
    end

    rectangle('Position', [0.5, 0.5, size(img, 2), size(img, 1)], 'EdgeColor',
borderColor, 'LineWidth', 3);
    title(titleText);
```

```
end
```

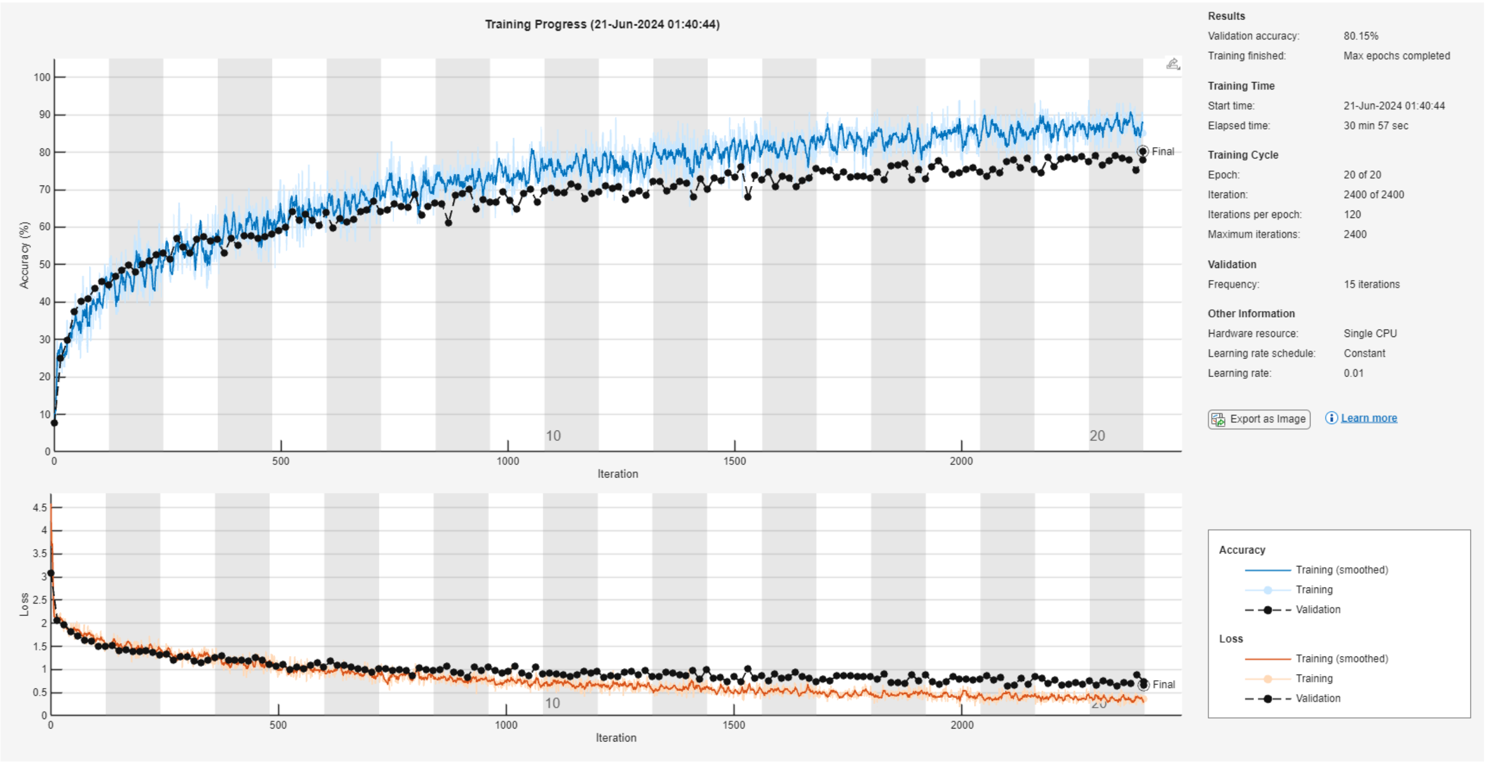

Wrong: chestnut (Pred: pine)
Correct: pine
Wrong: oak (Pred: linden)
Wrong: oak (Pred: chestnut)
Correct: linden
Wrong: linden (Pred: horsechestnut)
Correct: chestnut
Correct: horsechestnut
Wrong: pine (Pred: spruce)
Wrong: oak (Pred: linden)

```
drawnow;
```

Normalised Data with Enhanced CNN model

```
options = trainingOptions('sgdm', ...
    'InitialLearnRate',0.01, ...
    'MaxEpochs',20, ...
    'Shuffle','every-epoch', ...
    'ValidationData',{testData, testLabels}, ...
    'ValidationFrequency',15, ...
    'Verbose',false, ...
    'Plots','training-progress');
```

```
% Train the network
netImproved3 = trainNetwork(trainData, trainLabels, layers, options);
```



**Results**

Validation accuracy: 80.15%

Training finished: Max epochs completed

**Training Time**

Start time: 21-Jun-2024 01:40:44

Elapsed time: 30 min 57 sec

**Training Cycle**

Epoch: 20 of 20

Iteration: 2400 of 2400

Iterations per epoch: 120

Maximum iterations: 2400

**Validation**

Frequency: 15 iterations

**Other Information**

Hardware resource: Single CPU

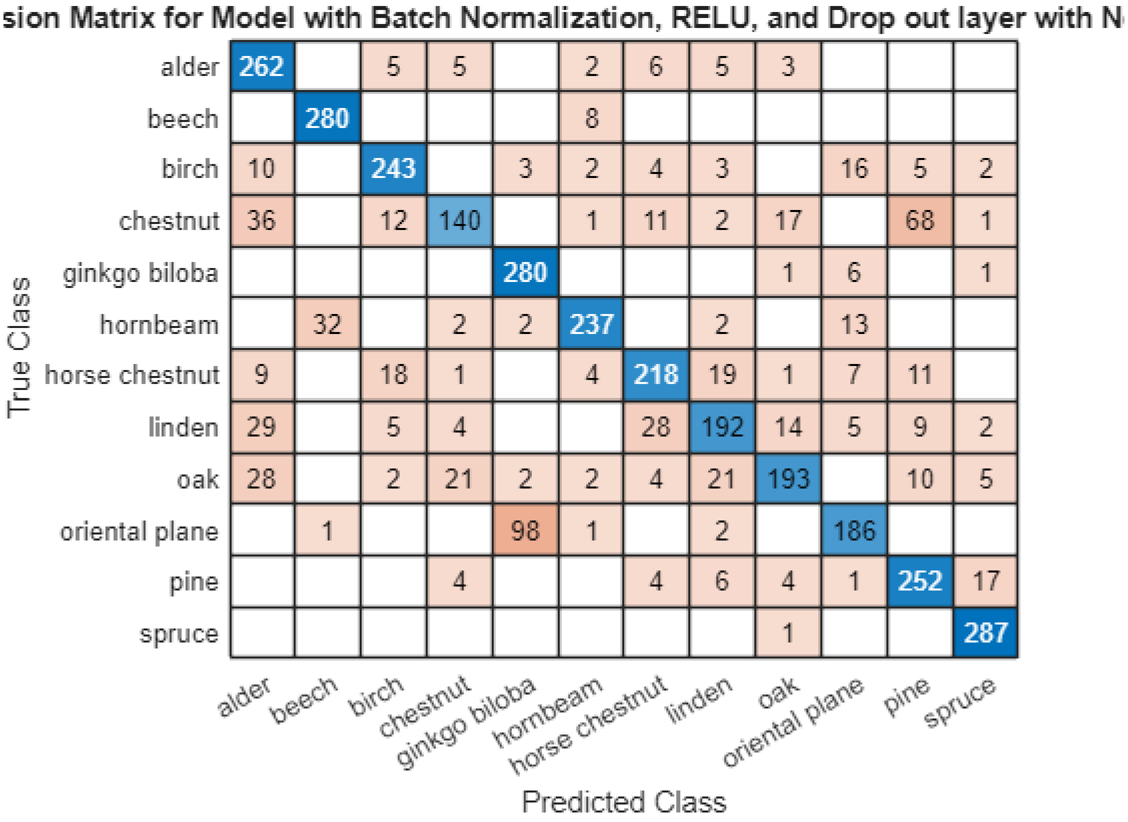
Learning rate schedule: Constant

Learning rate: 0.01

```
predictedLabels = classify(netImproved3, testData);
accuracy = sum(predictedLabels == testLabels) / numel(testLabels);
fprintf('Test accuracy: %.2f%%\n', accuracy * 100);
```

```
Test accuracy: 80.15%
```

```
figure;
confusionchart(testLabels, predictedLabels);
title('Confusion Matrix for Model with Batch Normalization, RELU, and Drop out
layer with Normalised Data');
```



sion Matrix for Model with Batch Normalization, RELU, and Drop out layer with N

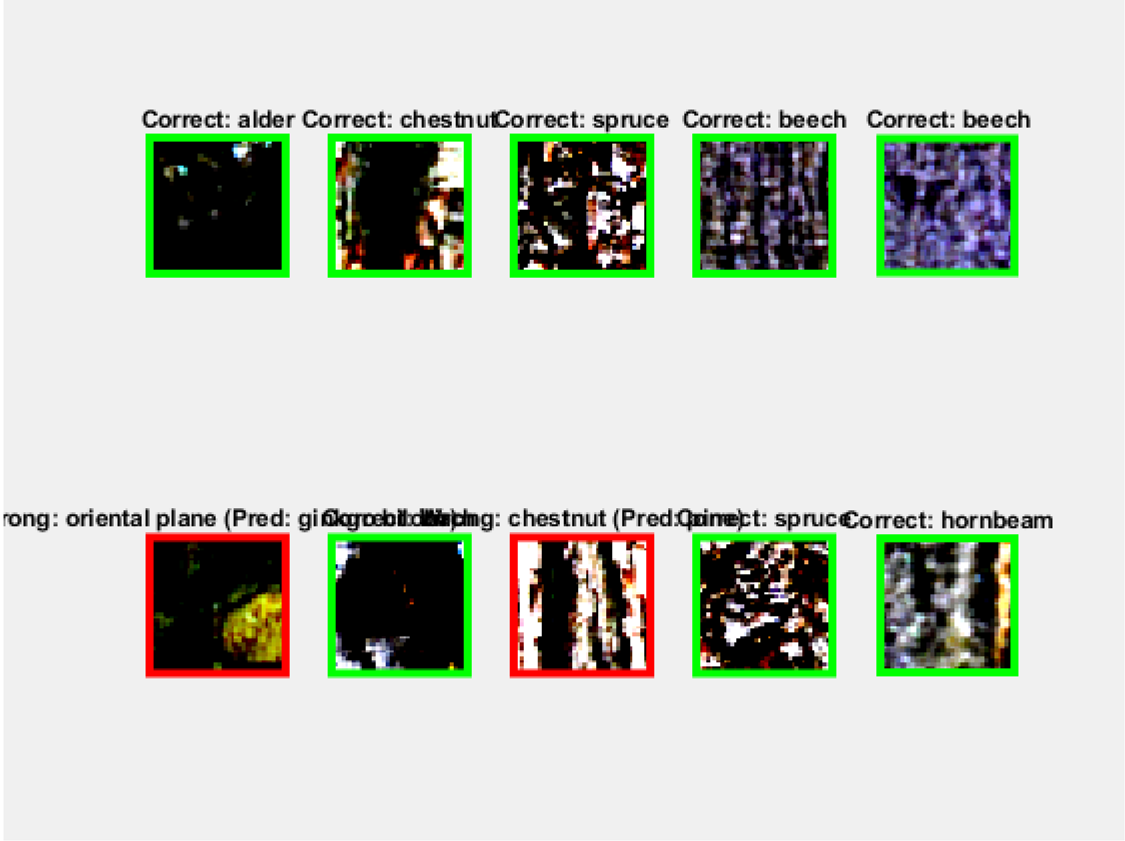
True Class

| | alder | beech | birch | chestnut | ginkgo biloba | hornbeam | horse chestnut | linden | oak | oriental plane | pine | spruce |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| alder | 262 | | 5 | 5 | | 2 | 6 | 5 | 3 | | | |
| beech | | 280 | | | | 8 | | | | | | |
| birch | 10 | | 243 | | 3 | 2 | 4 | 3 | | 16 | 5 | 2 |
| chestnut | 36 | | 12 | 140 | | 1 | 11 | 2 | 17 | | 68 | 1 |
| ginkgo biloba | | | | | 280 | | | | 1 | 6 | | 1 |
| hornbeam | | 32 | | 2 | 2 | 237 | | 2 | | 13 | | |
| horse chestnut | 9 | | 18 | 1 | | 4 | 218 | 19 | 1 | 7 | 11 | |
| linden | 29 | | 5 | 4 | | | 28 | 192 | 14 | 5 | 9 | 2 |
| oak | 28 | | 2 | 21 | 2 | 2 | 4 | 21 | 193 | | 10 | 5 |
| oriental plane | | 1 | | | 98 | 1 | | 2 | | 186 | | |
| pine | | | | 4 | | | 4 | 6 | 4 | 1 | 252 | 17 |
| spruce | | | | | | | | | 1 | | | 287 |

Predicted Class

```
numSamples = 10;
sampleIndices = randperm(numel(testLabels), numSamples);

figure('Visible', 'on');
for i = 1:numSamples
    index = sampleIndices(i);
    img = testData(:,:,:,index);
    trueLabel = testLabels(index);
    predictedLabel = predictedLabels(index);

    subplot(2, numSamples/2, i);
    imshow(img, 'InitialMagnification', 'fit');

    if trueLabel == predictedLabel
        borderColor = 'g'; % Green for correct classification
        titleText = sprintf('Correct: %s', string(trueLabel));
    else
        borderColor = 'r'; % Red for incorrect classification
        titleText = sprintf('Wrong: %s (Pred: %s)', string(trueLabel),
string(predictedLabel));
    end

    rectangle('Position', [0.5, 0.5, size(img, 2), size(img, 1)], 'EdgeColor',
borderColor, 'LineWidth', 3);
    title(titleText);
```

```
end
```



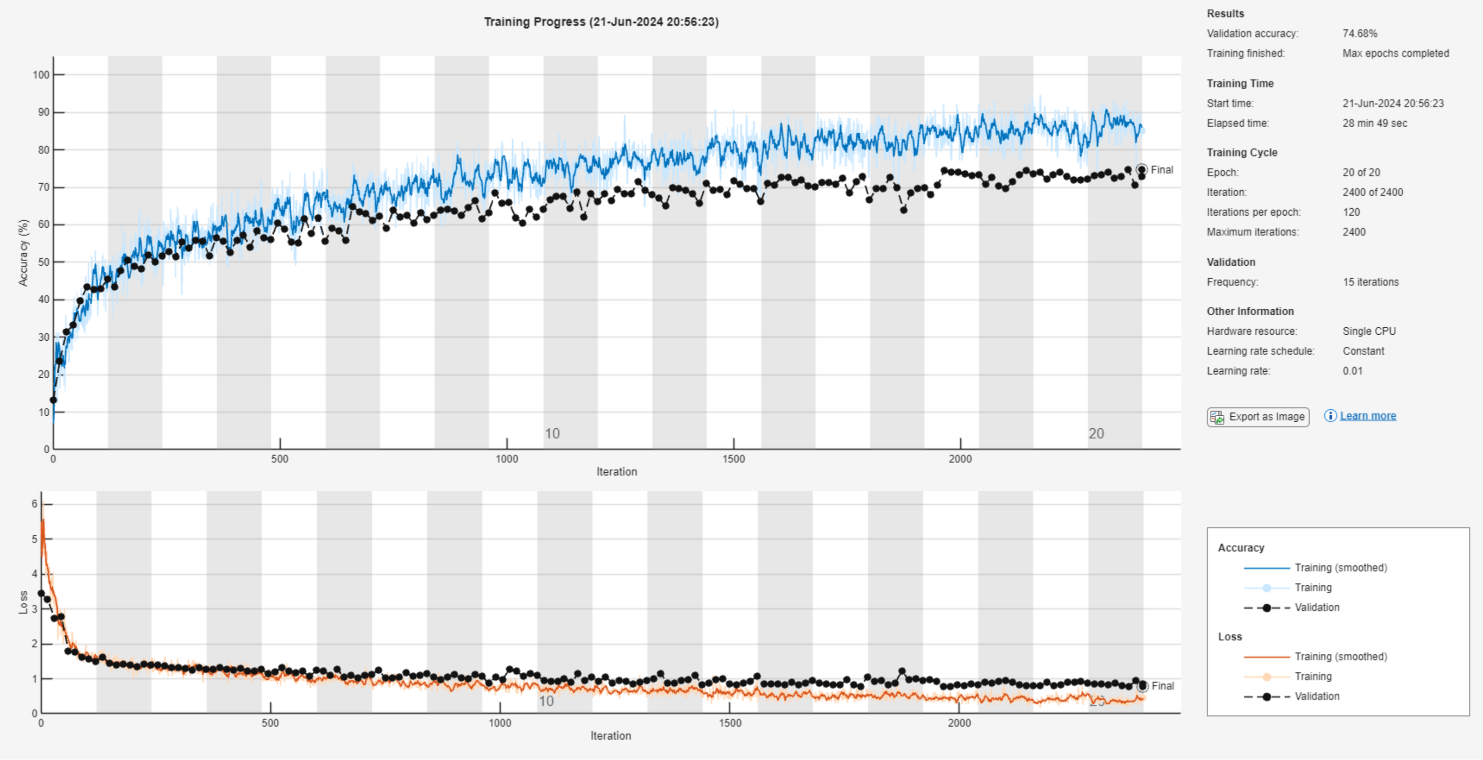
```
drawnow;
```

Remove ReLU Activation from Enhanced Model with Normalised Data

```
layers = [
    imageInputLayer([40 40 3], 'Normalization', 'none')

    convolution2dLayer(3, 32, 'Padding', 'same')
    batchNormalizationLayer
    maxPooling2dLayer(2, 'Stride', 2)

    convolution2dLayer(3, 64, 'Padding', 'same')
    batchNormalizationLayer
    maxPooling2dLayer(2, 'Stride', 2)

    convolution2dLayer(3, 128, 'Padding', 'same')
    batchNormalizationLayer
    maxPooling2dLayer(2, 'Stride', 2)

    fullyConnectedLayer(256)
    dropoutLayer(0.5) % Add dropout to prevent overfitting

    fullyConnectedLayer(numel(unique(trainLabels)))
    softmaxLayer
    classificationLayer
];
```

```
options = trainingOptions('sgdm', ...
    'InitialLearnRate',0.01, ...
    'MaxEpochs',20, ...
    'Shuffle','every-epoch', ...
    'ValidationData',{testData, testLabels}, ...
    'ValidationFrequency',15, ...
    'Verbose',false, ...
    'Plots','training-progress');
```

```
% Train the network
netImproved4 = trainNetwork(trainData, trainLabels, layers, options);
```



Training Progress (21-Jun-2024 20:56:23)

**Results**

Validation accuracy: 74.68%

Training finished: Max epochs completed

**Training Time**

Start time: 21-Jun-2024 20:56:23

Elapsed time: 28 min 49 sec

**Training Cycle**

Epoch: 20 of 20

Iteration: 2400 of 2400

Iterations per epoch: 120

Maximum iterations: 2400

**Validation**

Frequency: 15 iterations

**Other Information**

Hardware resource: Single CPU

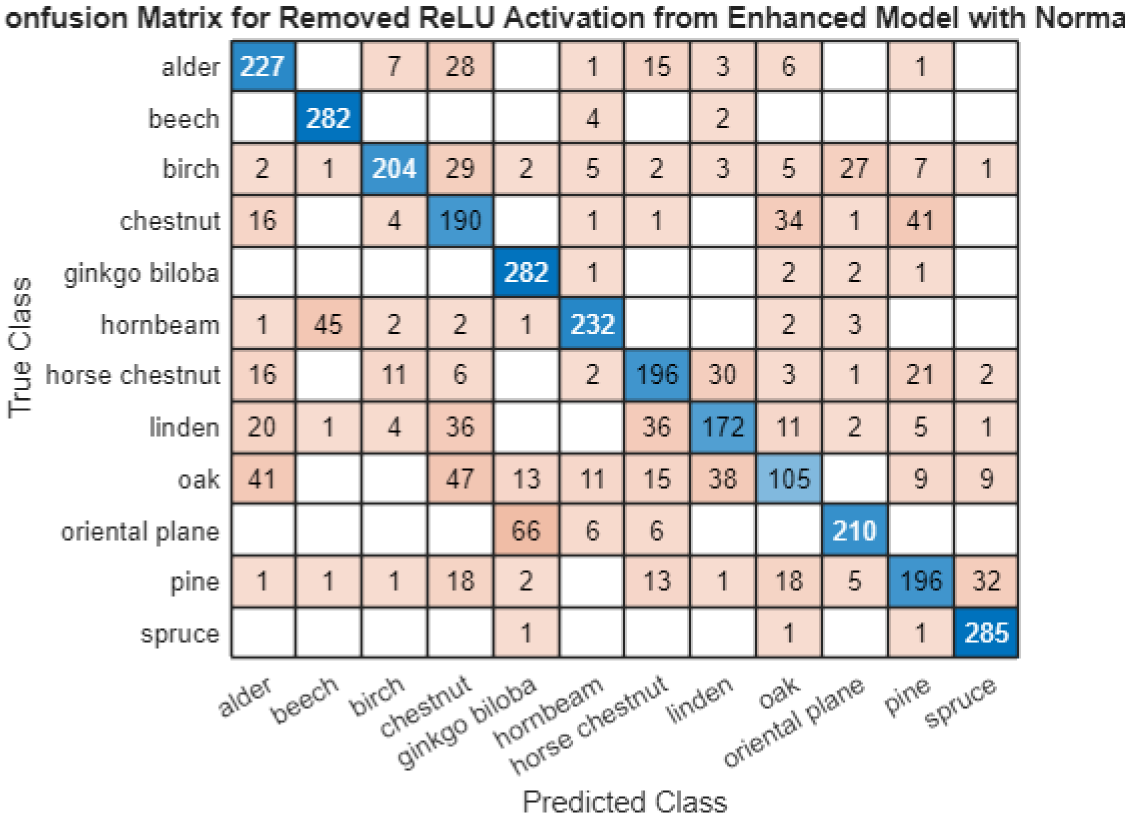
Learning rate schedule: Constant

Learning rate: 0.01

```
predictedLabels = classify(netImproved4, testData);
accuracy = sum(predictedLabels == testLabels) / numel(testLabels);
fprintf('Test accuracy: %.2f%%\n', accuracy * 100);
```

```
Test accuracy: 74.68%
```

```
figure;
confusionchart(testLabels, predictedLabels);
title('Confusion Matrix for Removed ReLU Activation from Enhanced Model with 
Normalised Data');
```



onfusion Matrix for Removed ReLU Activation from Enhanced Model with Norma

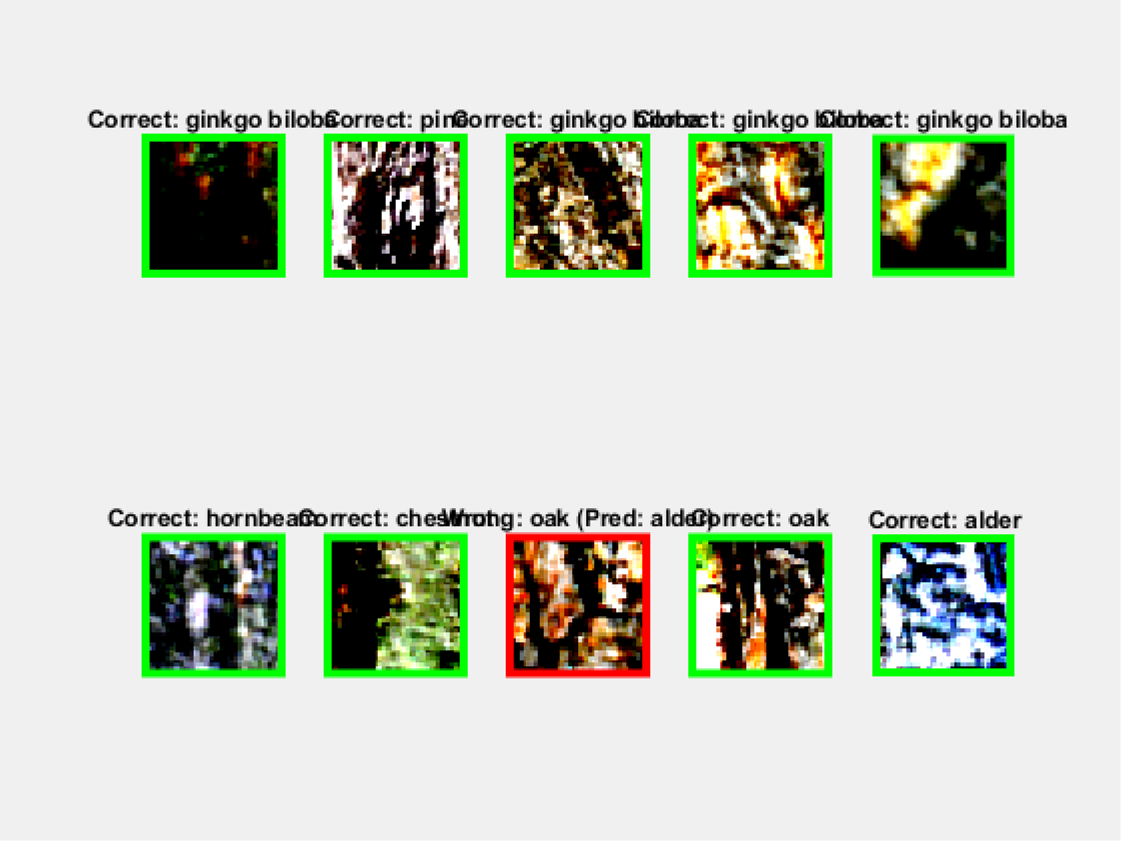
True Class

| | alder | beech | birch | chestnut | ginkgo biloba | hornbeam | horse chestnut | linden | oak | oriental plane | pine | spruce |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| alder | 227 | | 7 | 28 | | 1 | 15 | 3 | 6 | | 1 | |
| beech | | 282 | | | | 4 | | 2 | | | | |
| birch | 2 | 1 | 204 | 29 | 2 | 5 | 2 | 3 | 5 | 27 | 7 | 1 |
| chestnut | 16 | | 4 | 190 | | 1 | 1 | | 34 | 1 | 41 | |
| ginkgo biloba | | | | | 282 | 1 | | | 2 | 2 | 1 | |
| hornbeam | 1 | 45 | 2 | 2 | 1 | 232 | | | 2 | 3 | | |
| horse chestnut | 16 | | 11 | 6 | | 2 | 196 | 30 | 3 | 1 | 21 | 2 |
| linden | 20 | 1 | 4 | 36 | | | 36 | 172 | 11 | 2 | 5 | 1 |
| oak | 41 | | | 47 | 13 | 11 | 15 | 38 | 105 | | 9 | 9 |
| oriental plane | | | | | 66 | 6 | 6 | | | 210 | | |
| pine | 1 | 1 | 1 | 18 | 2 | | 13 | 1 | 18 | 5 | 196 | 32 |
| spruce | | | | | 1 | | | | 1 | | 1 | 285 |

Predicted Class

```
numSamples = 10;
sampleIndices = randperm(numel(testLabels), numSamples);

figure('Visible', 'on');
for i = 1:numSamples
    index = sampleIndices(i);
    img = testData(:,:,:,index);
    trueLabel = testLabels(index);
    predictedLabel = predictedLabels(index);

    subplot(2, numSamples/2, i);
    imshow(img, 'InitialMagnification', 'fit');

    if trueLabel == predictedLabel
        borderColor = 'g';
        titleText = sprintf('Correct: %s', string(trueLabel));
    else
        borderColor = 'r';
        titleText = sprintf('Wrong: %s (Pred: %s)', string(trueLabel),
string(predictedLabel));
    end

    rectangle('Position', [0.5, 0.5, size(img, 2), size(img, 1)], 'EdgeColor',
borderColor, 'LineWidth', 3);
    title(titleText);
```

```
end
```



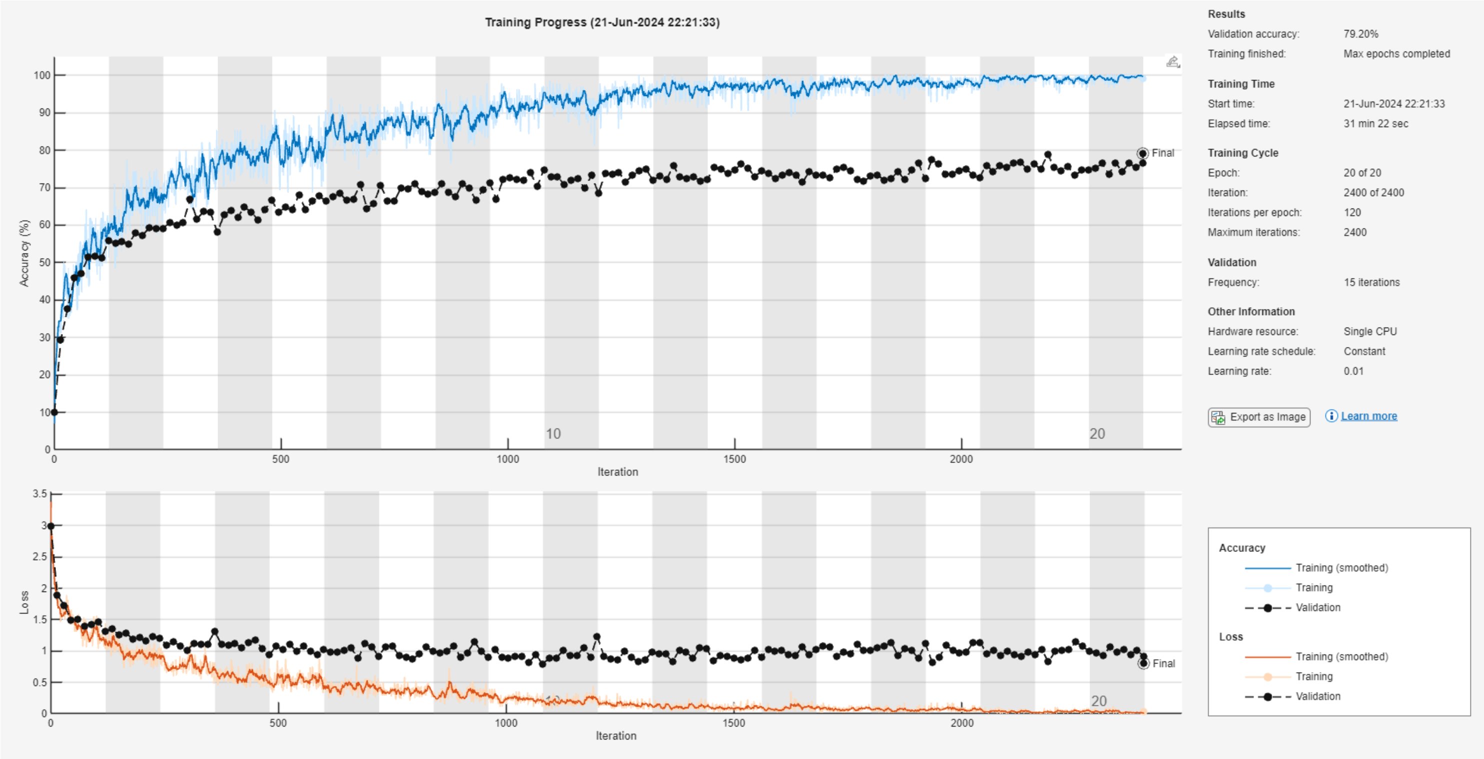
```
drawnow;
```

Removed dropout layer from Enhanced Model with Normalised Data

```
layers = [
    imageInputLayer([40 40 3], 'Normalization', 'none')

    convolution2dLayer(3, 32, 'Padding', 'same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2, 'Stride', 2)

    convolution2dLayer(3, 64, 'Padding', 'same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2, 'Stride', 2)

    convolution2dLayer(3, 128, 'Padding', 'same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2, 'Stride', 2)

    fullyConnectedLayer(256)
    reluLayer
    fullyConnectedLayer(numel(unique(trainLabels)))
    softmaxLayer
    classificationLayer
];
```

```
options = trainingOptions('sgdm', ...
    'InitialLearnRate',0.01, ...
    'MaxEpochs',20, ...
    'Shuffle','every-epoch', ...
    'ValidationData',{testData, testLabels}, ...
    'ValidationFrequency',15, ...
    'Verbose',false, ...
    'Plots','training-progress');
```

```
% Train the network
netImproved5 = trainNetwork(trainData, trainLabels, layers, options);
```



Training Progress (21-Jun-2024 22:21:33)

**Results**
Validation accuracy: 79.20%
Training finished: Max epochs completed

**Training Time**
Start time: 21-Jun-2024 22:21:33
Elapsed time: 31 min 22 sec

**Training Cycle**
Epoch: 20 of 20
Iteration: 2400 of 2400
Iterations per epoch: 120
Maximum iterations: 2400

**Validation**
Frequency: 15 iterations

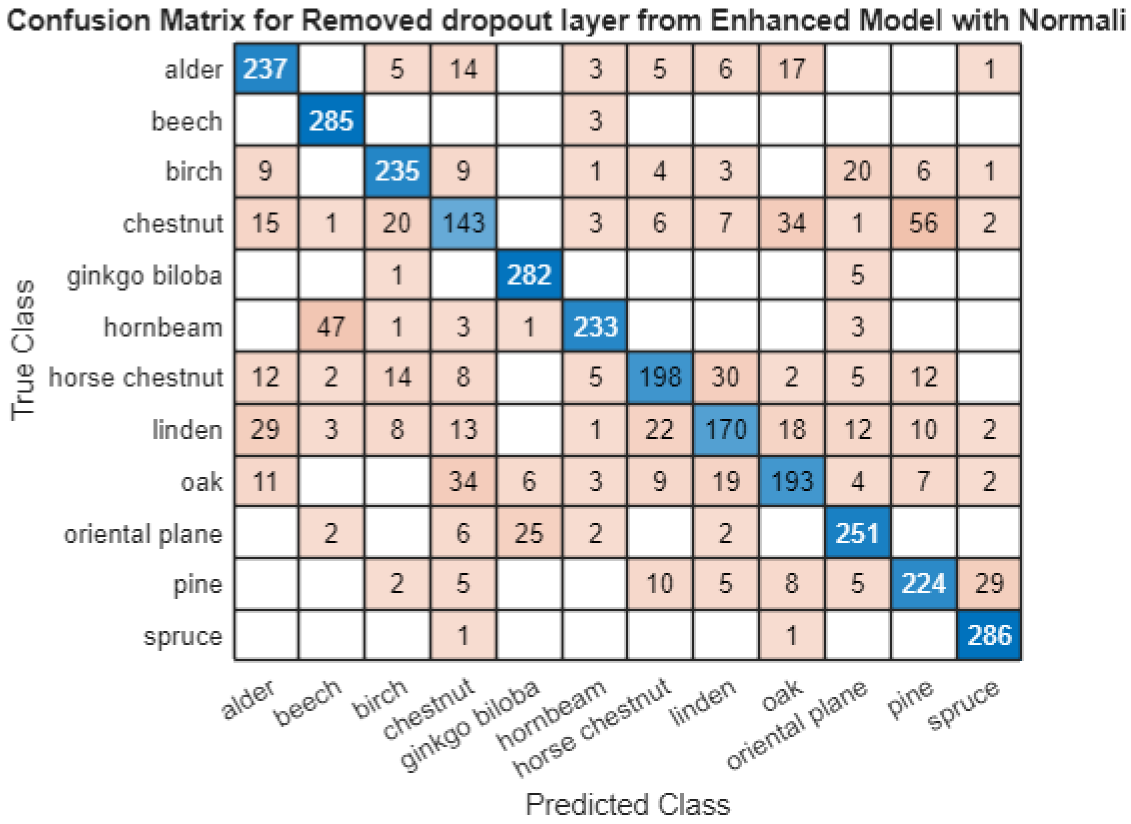
**Other Information**
Hardware resource: Single CPU
Learning rate schedule: Constant
Learning rate: 0.01

```
predictedLabels = classify(netImproved5, testData);
accuracy = sum(predictedLabels == testLabels) / numel(testLabels);
fprintf('Test accuracy: %.2f%%\n', accuracy * 100);
```

```
Test accuracy: 79.20%
```

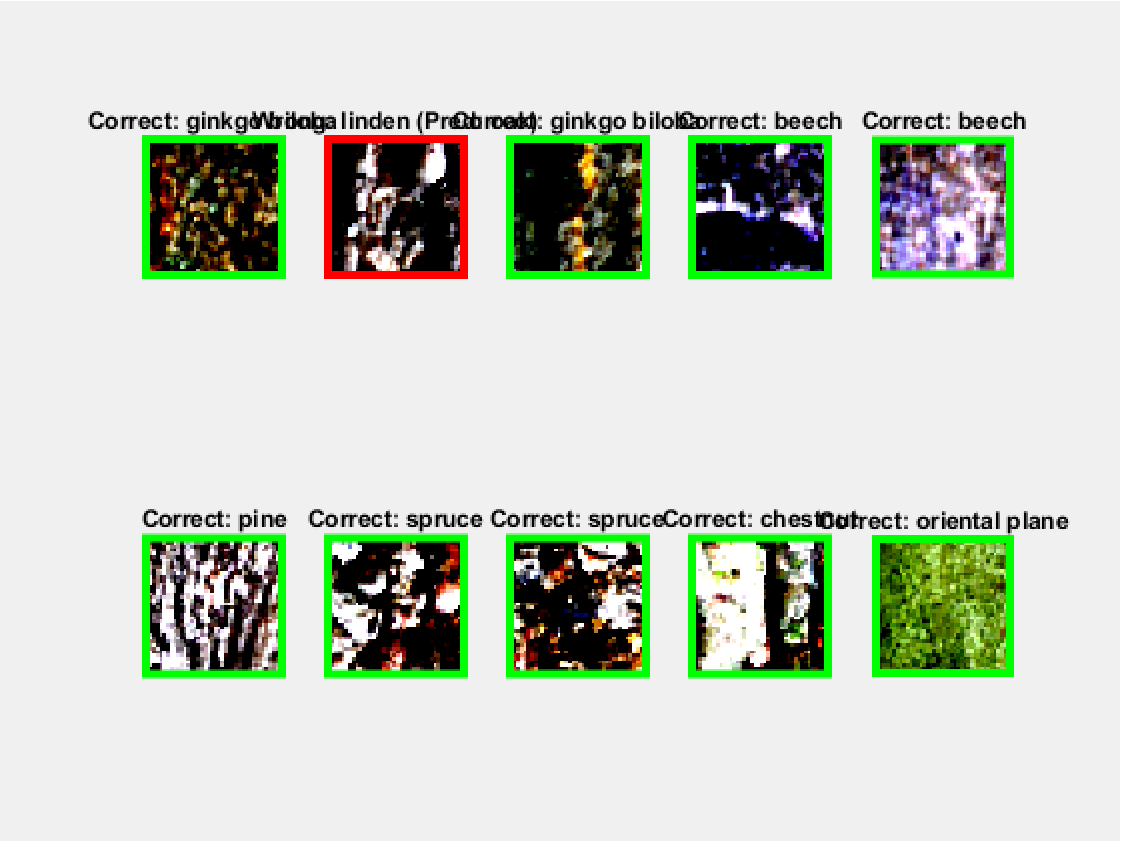
```
figure;
confusionchart(testLabels, predictedLabels);
```

```
title('Confusion Matrix for Removed dropout layer from Enhanced Model with
Normalised Data');
```



| True Class \ Predicted Class | alder | beech | birch | chestnut | ginkgo biloba | hornbeam | horse chestnut | linden | oak | oriental plane | pine | spruce |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| alder | 237 |  | 5 | 14 |  | 3 | 5 | 6 | 17 |  |  | 1 |
| beech |  | 285 |  |  |  | 3 |  |  |  |  |  |  |
| birch | 9 |  | 235 | 9 |  | 1 | 4 | 3 |  | 20 | 6 | 1 |
| chestnut | 15 | 1 | 20 | 143 |  | 3 | 6 | 7 | 34 | 1 | 56 | 2 |
| ginkgo biloba |  |  | 1 |  | 282 |  |  |  |  | 5 |  |  |
| hornbeam |  | 47 | 1 | 3 | 1 | 233 |  |  |  | 3 |  |  |
| horse chestnut | 12 | 2 | 14 | 8 |  | 5 | 198 | 30 | 2 | 5 | 12 |  |
| linden | 29 | 3 | 8 | 13 |  | 1 | 22 | 170 | 18 | 12 | 10 | 2 |
| oak | 11 |  |  | 34 | 6 | 3 | 9 | 19 | 193 | 4 | 7 | 2 |
| oriental plane |  | 2 |  | 6 | 25 | 2 |  | 2 |  | 251 |  |  |
| pine |  |  | 2 | 5 |  |  | 10 | 5 | 8 | 5 | 224 | 29 |
| spruce |  |  |  | 1 |  |  |  |  | 1 |  |  | 286 |

```
numSamples = 10;
sampleIndices = randperm(numel(testLabels), numSamples);

figure('Visible', 'on');
for i = 1:numSamples
    index = sampleIndices(i);
    img = testData(:,:,:,index);
    trueLabel = testLabels(index);
    predictedLabel = predictedLabels(index);

    subplot(2, numSamples/2, i);
    imshow(img, 'InitialMagnification', 'fit');

    if trueLabel == predictedLabel
        borderColor = 'g';
        titleText = sprintf('Correct: %s', string(trueLabel));
    else
        borderColor = 'r';
        titleText = sprintf('Wrong: %s (Pred: %s)', string(trueLabel),
string(predictedLabel));
    end
    rectangle('Position', [0.5, 0.5, size(img, 2), size(img, 1)], 'EdgeColor',
borderColor, 'LineWidth', 3);
    title(titleText);
end
```



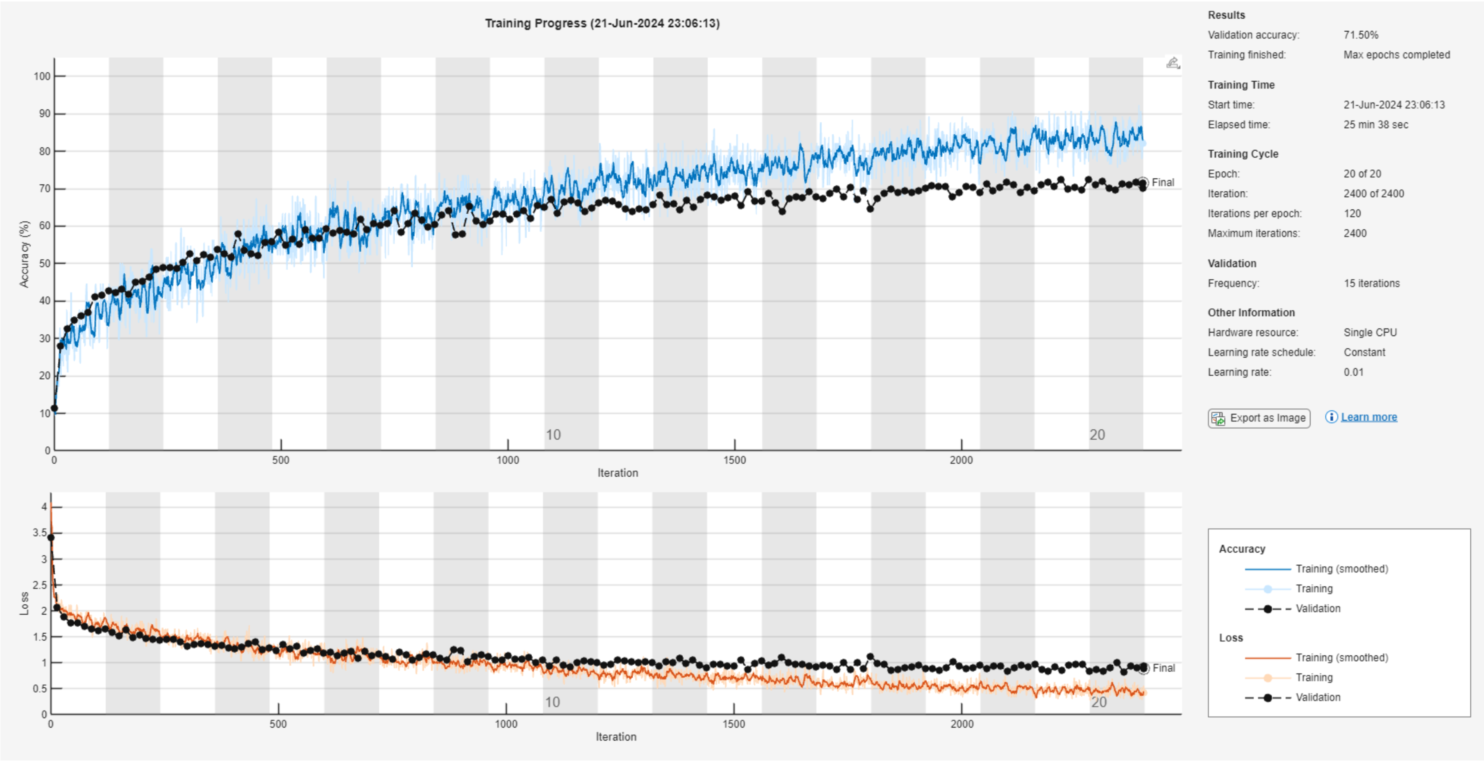
```
drawnow;
```

Remove increased convolutional filters (original filters)

```
layers = [
    imageInputLayer([40 40 3], 'Normalization', 'none')

    convolution2dLayer(3, 32, 'Padding', 'same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2, 'Stride', 2)

    convolution2dLayer(3, 64, 'Padding', 'same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2, 'Stride', 2)

    fullyConnectedLayer(256)
    dropoutLayer(0.5) % Add dropout to prevent overfitting
    reluLayer
    fullyConnectedLayer(numel(unique(trainLabels)))
    softmaxLayer
    classificationLayer
];
```

```
options = trainingOptions('sgdm', ...
    'InitialLearnRate',0.01, ...
    'MaxEpochs',20, ...
    'Shuffle','every-epoch', ...
    'ValidationData',{testData, testLabels}, ...
    'ValidationFrequency',15, ...
    'Verbose',false, ...
    'Plots','training-progress');
```

```
% Train the network
netImproved6 = trainNetwork(trainData, trainLabels, layers, options);
```



**Results**
Validation accuracy: 71.50%
Training finished: Max epochs completed

**Training Time**
Start time: 21-Jun-2024 23:06:13
Elapsed time: 25 min 38 sec

**Training Cycle**
Epoch: 20 of 20
Iteration: 2400 of 2400
Iterations per epoch: 120
Maximum iterations: 2400

**Validation**
Frequency: 15 iterations

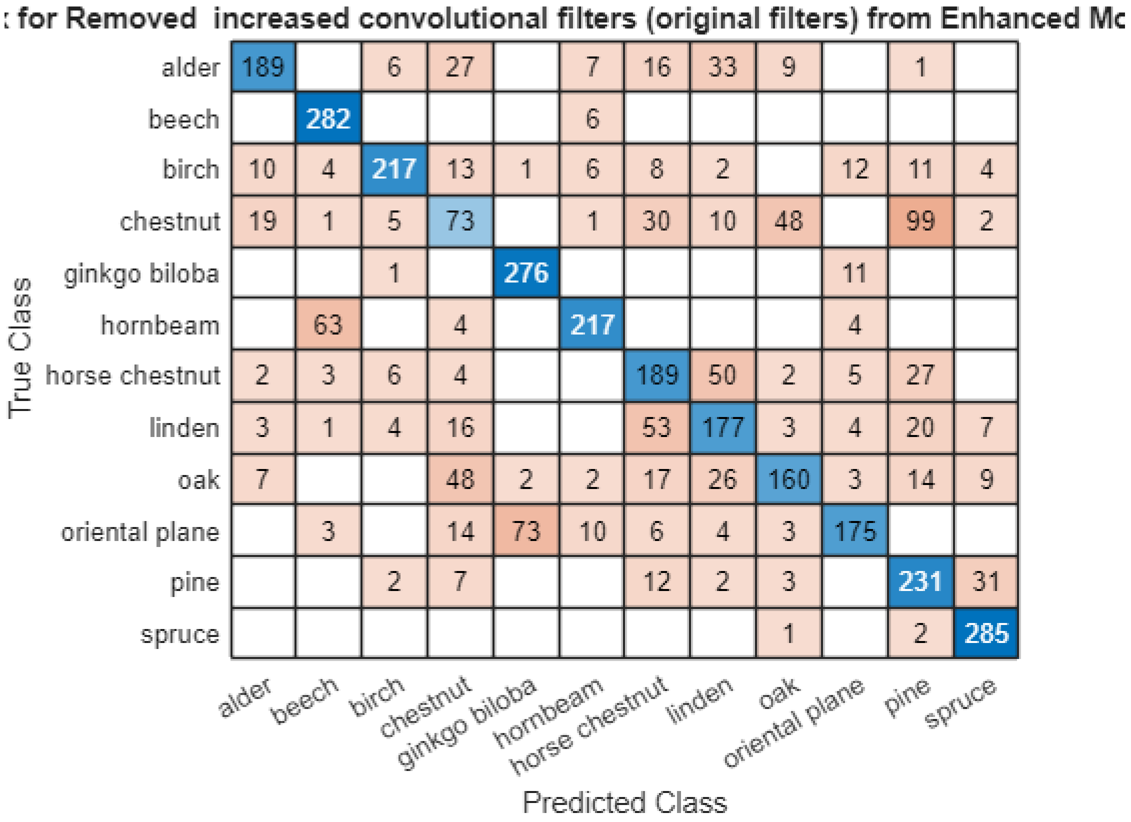
**Other Information**
Hardware resource: Single CPU
Learning rate schedule: Constant
Learning rate: 0.01

```
predictedLabels = classify(netImproved6, testData);
accuracy = sum(predictedLabels == testLabels) / numel(testLabels);
fprintf('Test accuracy: %.2f%%\n', accuracy * 100);
```

```
Test accuracy: 71.50%
```

```
figure;
confusionchart(testLabels, predictedLabels);
```

```
title('Confusion Matrix for Removed  increased convolutional filters (original
filters) from Enhanced Model with Normalised Data');
```



| | alder | beech | birch | chestnut | ginkgo biloba | hornbeam | horse chestnut | linden | oak | oriental plane | pine | spruce |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| alder | 189 | | 6 | 27 | | 7 | 16 | 33 | 9 | | 1 | |
| beech | | 282 | | | | 6 | | | | | | |
| birch | 10 | 4 | 217 | 13 | 1 | 6 | 8 | 2 | | 12 | 11 | 4 |
| chestnut | 19 | 1 | 5 | 73 | | 1 | 30 | 10 | 48 | | 99 | 2 |
| ginkgo biloba | | | 1 | | 276 | | | | | 11 | | |
| hornbeam | | 63 | | 4 | | 217 | | | | 4 | | |
| horse chestnut | 2 | 3 | 6 | 4 | | | 189 | 50 | 2 | 5 | 27 | |
| linden | 3 | 1 | 4 | 16 | | | 53 | 177 | 3 | 4 | 20 | 7 |
| oak | 7 | | | 48 | 2 | 2 | 17 | 26 | 160 | 3 | 14 | 9 |
| oriental plane | | 3 | | 14 | 73 | 10 | 6 | 4 | 3 | 175 | | |
| pine | | | 2 | 7 | | | 12 | 2 | 3 | | 231 | 31 |
| spruce | | | | | | | | | 1 | | 2 | 285 |

```
numSamples = 10;
sampleIndices = randperm(numel(testLabels), numSamples);

figure('Visible', 'on');
for i = 1:numSamples
    index = sampleIndices(i);
    img = testData(:,:,:,index);
    trueLabel = testLabels(index);
    predictedLabel = predictedLabels(index);

    subplot(2, numSamples/2, i);
    imshow(img, 'InitialMagnification', 'fit');

    if trueLabel == predictedLabel
        borderColor = 'g';
        titleText = sprintf('Correct: %s', string(trueLabel));
    else
        borderColor = 'r';
        titleText = sprintf('Wrong: %s (Pred: %s)', string(trueLabel),
string(predictedLabel));
    end
    rectangle('Position', [0.5, 0.5, size(img, 2), size(img, 1)], 'EdgeColor',
borderColor, 'LineWidth', 3);
    title(titleText);
end

drawnow;
```