```
clc; clear;
dsFolder = "p_dataset_26";
subFolder = ["0", "4", "7", "8", "A", "D", "H"];
categories = ["0", "4", "7", "8", "A", "D", "H"];
```

```
allData = table('Size', [0, 2], 'VariableTypes', {'cell', 'cell'}, 'VariableNames',
{'Image', 'Label'});

desiredSize = [224, 224]; % Set your desired image size

for i = 1:length(subFolder)
    folderPath = fullfile(dsFolder, subFolder(i));
    pngFiles = dir(fullfile(folderPath, "*.png"));

    for j = 1:length(pngFiles)
        pngFilePath = fullfile(folderPath, pngFiles(j).name);
        image = imread(pngFilePath);

        % Resize or pad the image to the desired size
        % resizedImage = imresize(image, desiredSize);
        % Alternatively, use padarray to add padding
        % paddedImage = padarray(image, [desiredSize(1)-size(image,1),
desiredSize(2)-size(image,2)], 0, 'post');

        % Store the resized/padded image and label
        allData = [allData; {image, categories(i)}];

%           allData = [allData; {image, categories(i)}];
    end
end
```

```
% define images & labels
numImages = size(allData, 1);
imageSize = size(allData.Image{1});
images = zeros(imageSize(1), imageSize(2), 1, numImages);

for i = 1:numImages
    images(:, :, 1, i) = allData.Image{i};
end

labels = categorical(allData.Label);
```

```
% split data into train and test set
cv = cvpartition(labels, 'HoldOut', 0.25);
trainIdx = training(cv);
testIdx = test(cv);
trainImages = images(:, :, :, trainIdx);
trainLabels = labels(trainIdx);
```

```
testImages = images(:, :, :, testIdx);
testLabels = labels(testIdx);
```

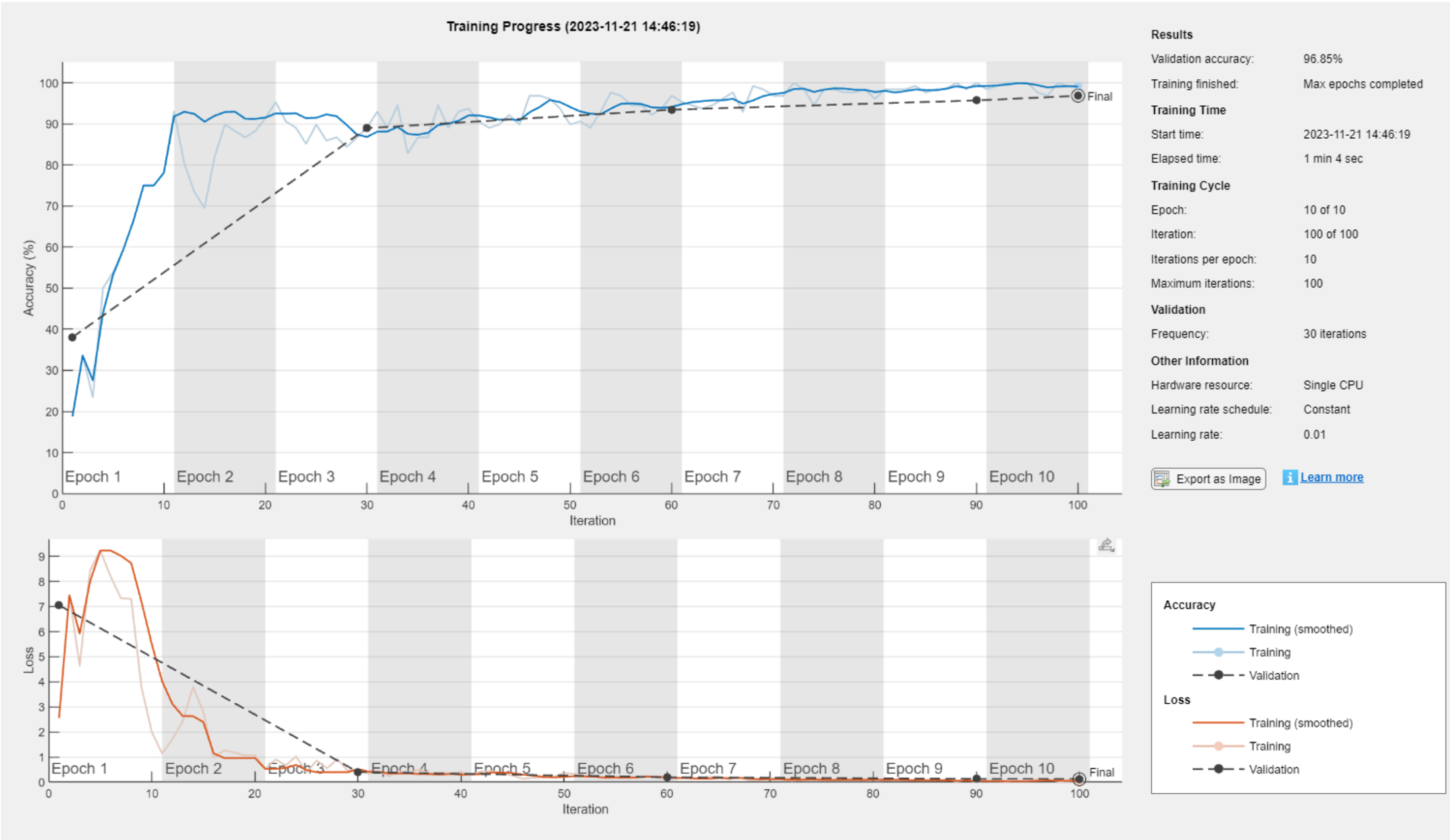
```
% Define the layers of the CNN
layers = [
    imageInputLayer([imageSize 1])

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

    maxPooling2dLayer(2, 'Stride', 2)

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

    maxPooling2dLayer(2, 'Stride', 2)

    fullyConnectedLayer(numel(categories))
    softmaxLayer
    classificationLayer
    ];
```

```
% Training options
options = trainingOptions( ...
    'sgdm', ... % Stochastic Gradient Descent with Momentum
    'InitialLearnRate', 0.01, ...
    'MaxEpochs', 10, ...
    'Shuffle', 'every-epoch', ...
    'ValidationData', {testImages, testLabels}, ...
    'ValidationFrequency', 30, ...
    'Verbose', true, ...
    'Plots', 'training-progress' ...
    );
```

```
% Train the CNN

trainingStartTime = tic;

% Train the neural network
net = trainNetwork(trainImages, trainLabels, layers, options);
```

```
Training on single CPU.
Initializing input data normalization.
|==========================================================================================================
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learni
|         |             |   (hh:mm:ss)   |   Accuracy   |   Accuracy   |     Loss     |     Loss     |      Rate
|==========================================================================================================
```

```
|       1 |        1 |    00:00:07 |    18.75% |    38.06% |    2.5563 |    7.0497 |          0.01
|       3 |       30 |    00:00:23 |    89.06% |    88.96% |    0.5235 |    0.4020 |          0.01
|       5 |       50 |    00:00:33 |    89.84% |           |    0.3650 |           |          0.01
|       6 |       60 |    00:00:41 |    96.88% |    93.47% |    0.1193 |    0.1956 |          0.01
|       9 |       90 |    00:00:58 |   100.00% |    95.72% |    0.0296 |    0.1304 |          0.01
|      10 |      100 |    00:01:03 |    99.22% |    96.85% |    0.0337 |    0.1285 |          0.01
|==================================================================================================
Training finished: Max epochs completed.
```




```
% Stop the timer
trainingTime = toc(trainingStartTime);

% Display the training time
fprintf('Training time: %.2f seconds\n', trainingTime);
```

```
Training time: 78.36 seconds
```

```
% save CNN model
save('trainedCNN.mat', 'net');
```

```
% Classify Test Images
predictedLabels = classify(net, testImages);

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

```
Test Accuracy: 96.85%
```

```
% Calculate precision, recall, and F1-score
confusionMatrix = confusionmat(testLabels, predictedLabels)
```

```
confusionMatrix = 7×7
    62     0     0     0     0     1     0
     0    61     0     0     2     0     0
     0     0    64     0     0     0     0
     1     0     0    62     0     0     0
     0     0     0     0    60     3     0
     0     0     0     1     0    63     0
     0     0     0     2     1     3    58
```

```


truePositive = confusionMatrix(1, 1);
falsePositive = confusionMatrix(2, 1);
falseNegative = confusionMatrix(1, 2);

precision = truePositive / (truePositive + falsePositive);
recall = truePositive / (truePositive + falseNegative);
f1Score = 2 * (precision * recall) / (precision + recall);

fprintf('Precision: %.2f\n', precision);
```

```
Precision: 1.00
```

```
fprintf('Recall: %.2f\n', recall);
```

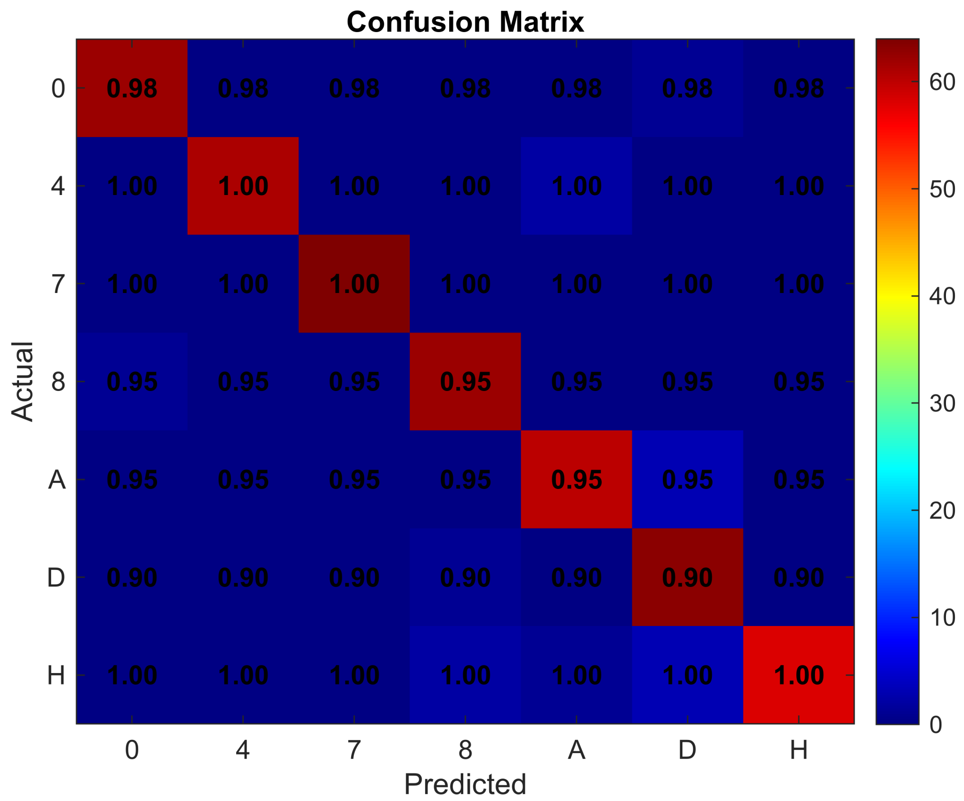
```
Recall: 1.00
```

```
fprintf('F1-Score: %.2f\n', f1Score);
```

```
F1-Score: 1.00
```

```
% Visualize the confusion matrix using imagesc and annotate precision
figure;
imagesc(confusionMatrix);
colorbar;
colormap('jet'); % You can change the colormap as needed
title('Confusion Matrix');
xlabel('Predicted');
ylabel('Actual');
xticks(1:numel(categories));
xticklabels(categories);
yticks(1:numel(categories));
yticklabels(categories);

% Calculate and annotate precision for each class
for i = 1:numel(categories)
    for j = 1:numel(categories)
        precision = confusionMatrix(i, i) / sum(confusionMatrix(:, i));
```

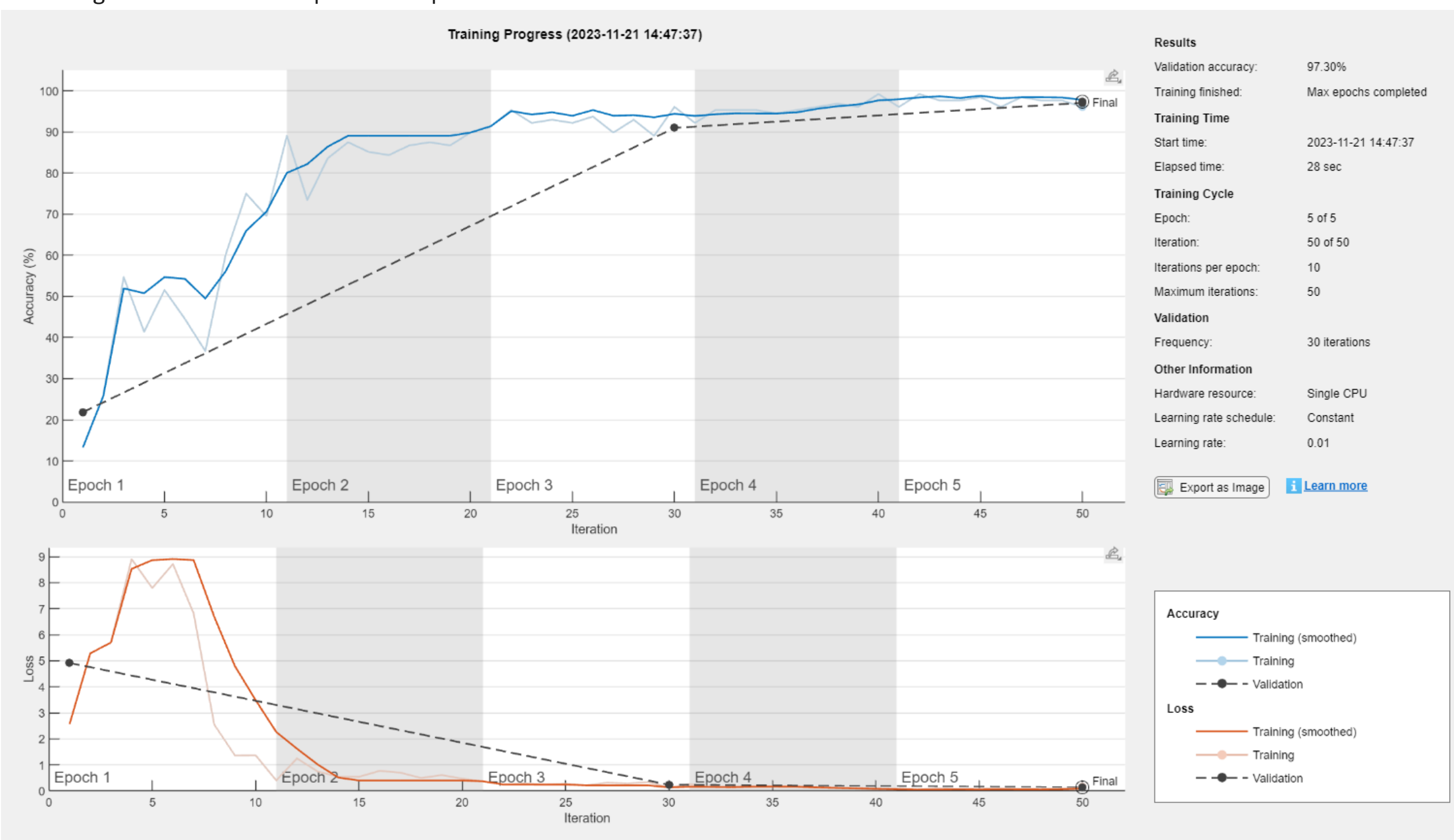
```
        text(j, i, sprintf('%.2f', precision), 'HorizontalAlignment', 'center',
'VerticalAlignment', 'middle', 'Color', 'k', 'FontWeight', 'bold');
    end
end
```




```
% Hyperparameter tuning for the number of epochs
numEpochsList = [5, 10, 15, 20]; % You can modify this list according to your
requirements

% Initialize a matrix to store accuracy for each number of epochs
EpochsAccuracyMatrix = zeros(length(numEpochsList), 1);
% Initialize a matrix to store training for each number of epochs
EpochsTime = zeros(length(numEpochsList), 1);

for idx = 1:length(numEpochsList)
    currentNumEpochs = numEpochsList(idx);

    % Update the MaxEpochs parameter in trainingOptions
    options.MaxEpochs = currentNumEpochs;

    trainingStartTime = tic;

    % Train the CNN
    net = trainNetwork(trainImages, trainLabels, layers, options);

    trainingTime = toc(trainingStartTime);
```

```
    % Classify Test Images
    predictedLabels = classify(net, testImages);

    % Calculate the Accuracy
    accuracy = sum(predictedLabels == testLabels) / numel(testLabels);
    EpochsAccuracyMatrix(idx) = accuracy;
    EpochsTime(idx) = trainingTime;

    fprintf('Test Accuracy for %d Epochs: %.2f%%\n', currentNumEpochs, accuracy *
100);
end
```

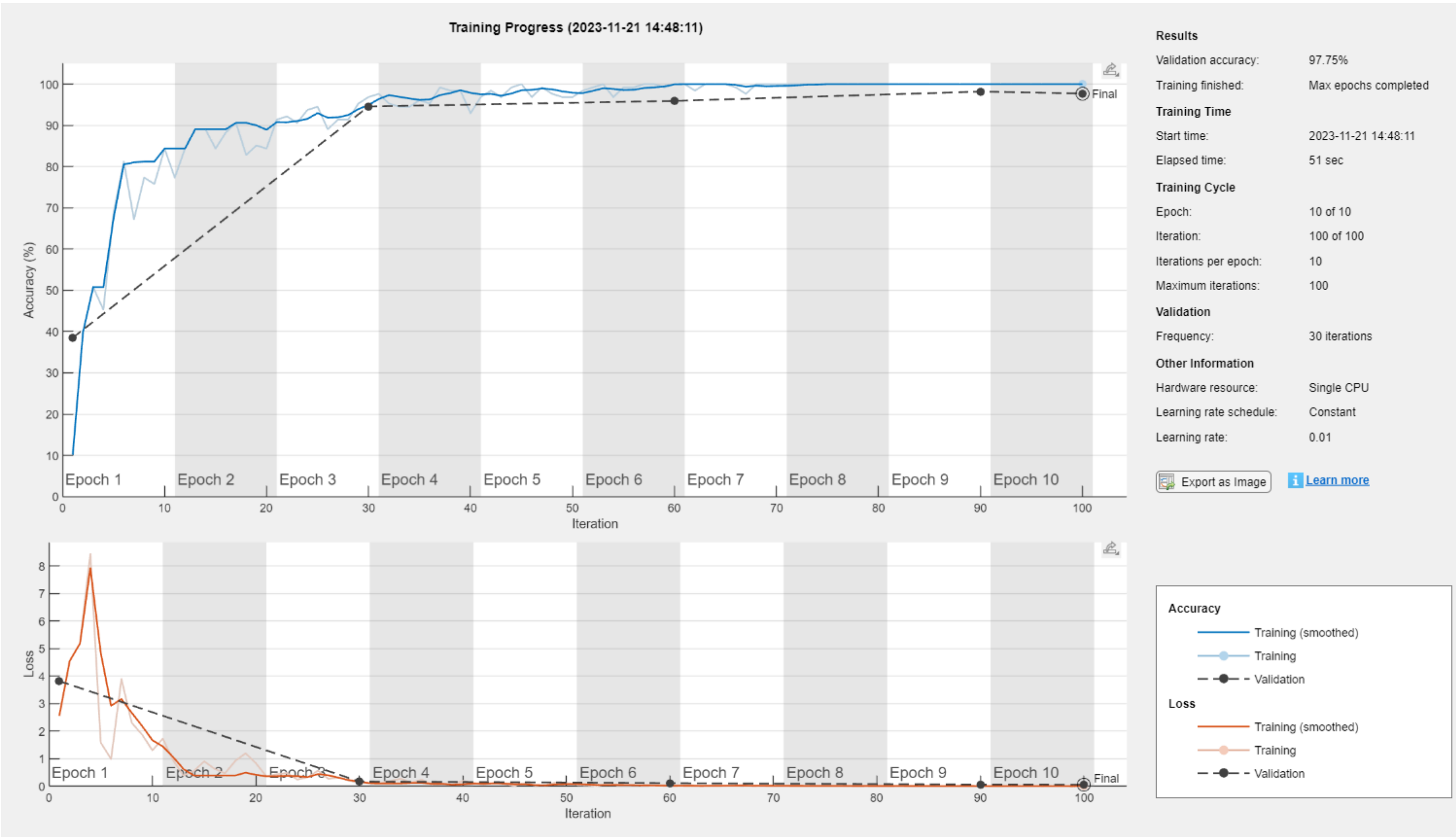
```
Training on single CPU.
Initializing input data normalization.
|=================================================================================================================
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learni
|         |             |   (hh:mm:ss)   |   Accuracy   |   Accuracy   |     Loss     |     Loss     |     Rate
|=================================================================================================================
|       1 |           1 |       00:00:02 |       13.28% |       21.85% |       2.5653 |       4.9162 |          0.01
|       3 |          30 |       00:00:18 |       96.09% |       90.99% |       0.1231 |       0.2368 |          0.01
|       5 |          50 |       00:00:28 |       96.09% |       97.07% |       0.1541 |       0.1341 |          0.01
|=================================================================================================================
Training finished: Max epochs completed.
```




```
Test Accuracy for 5 Epochs: 97.30%
Training on single CPU.
Initializing input data normalization.
|=================================================================================================================
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learni
|         |             |   (hh:mm:ss)   |   Accuracy   |   Accuracy   |     Loss     |     Loss     |     Rate
|=================================================================================================================
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 00:00:02 | 10.16% | 38.51% | 2.5570 | 3.8204 | 0.0 |
| 3 | 30 | 00:00:16 | 96.88% | 94.59% | 0.1199 | 0.1750 | 0.0 |
| 5 | 50 | 00:00:26 | 96.88% | | 0.1212 | | 0.0 |
| 6 | 60 | 00:00:31 | 100.00% | 95.95% | 0.0264 | 0.1167 | 0.0 |
| 9 | 90 | 00:00:45 | 100.00% | 98.20% | 0.0046 | 0.0624 | 0.0 |
| 10 | 100 | 00:00:50 | 100.00% | 97.75% | 0.0090 | 0.0599 | 0.0 |

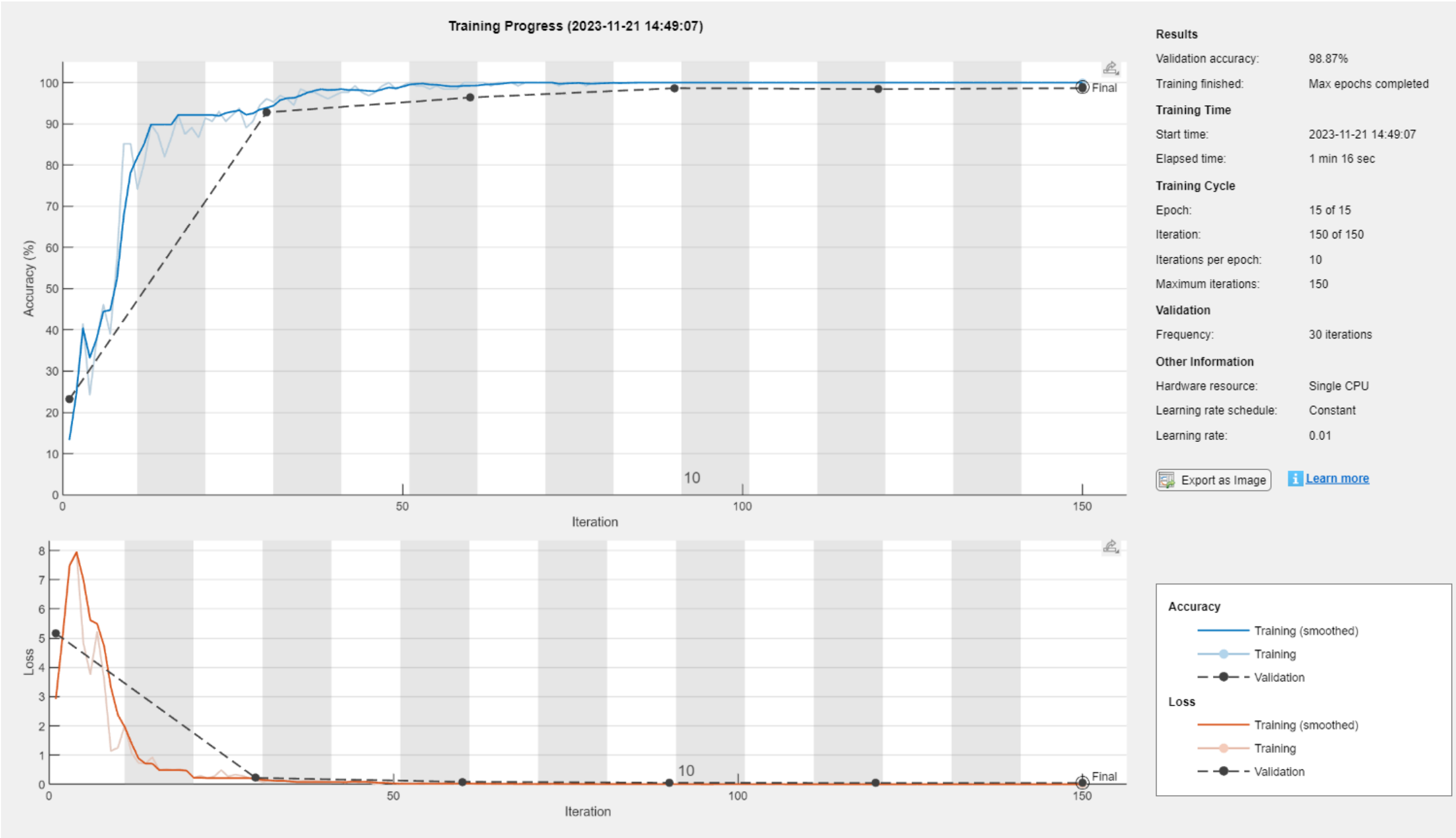
Training finished: Max epochs completed.



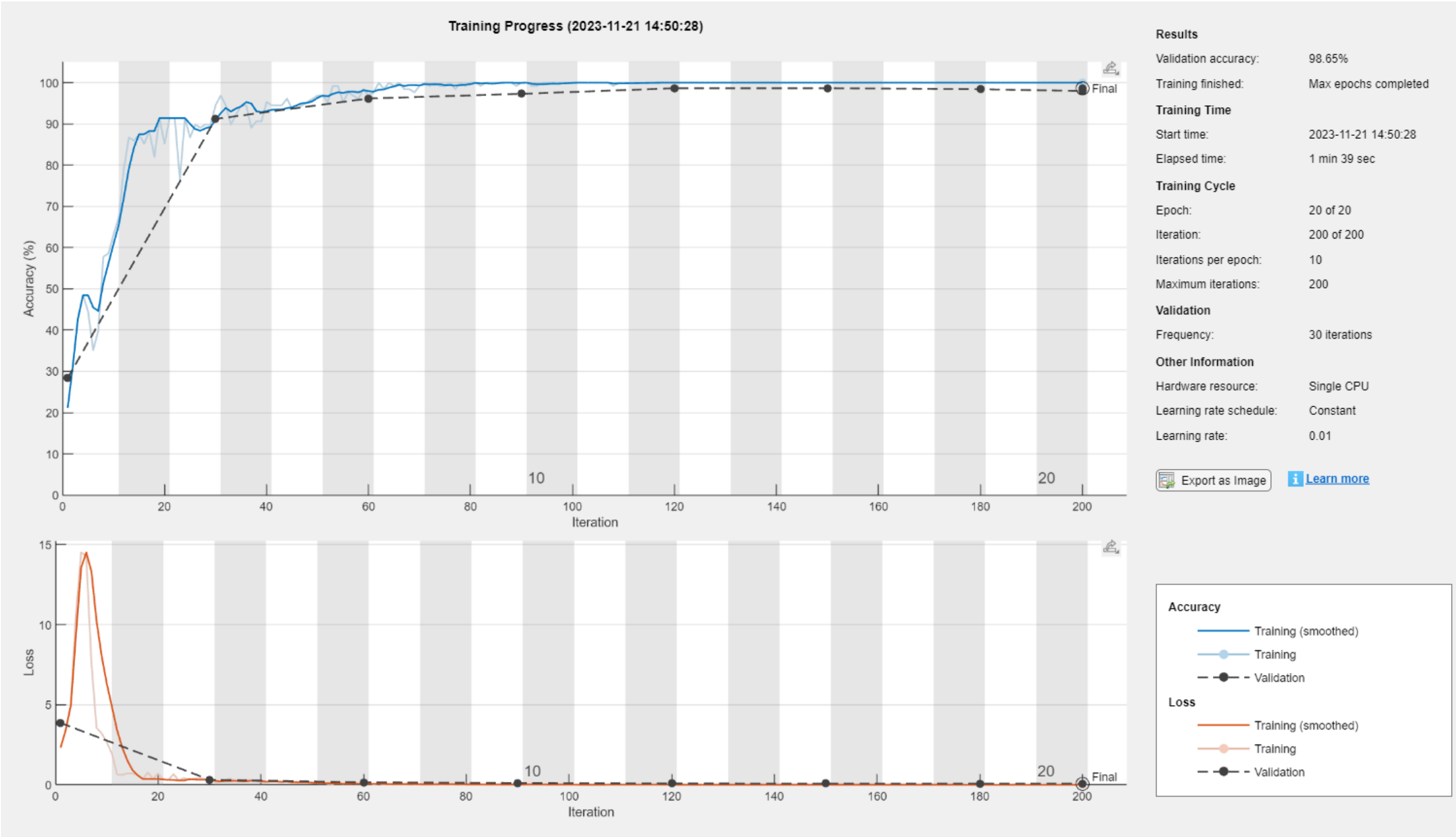

Test Accuracy for 10 Epochs: 97.75%
Training on single CPU.
Initializing input data normalization.

| Epoch | Iteration | Time Elapsed (hh:mm:ss) | Mini-batch Accuracy | Validation Accuracy | Mini-batch Loss | Validation Loss | Base Learni Rate |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 00:00:02 | 13.28% | 23.20% | 2.9228 | 5.1634 | 0.0 |
| 3 | 30 | 00:00:16 | 96.09% | 92.79% | 0.1380 | 0.2214 | 0.0 |
| 5 | 50 | 00:00:26 | 99.22% | | 0.0356 | | 0.0 |
| 6 | 60 | 00:00:31 | 100.00% | 96.40% | 0.0211 | 0.0879 | 0.0 |
| 9 | 90 | 00:00:47 | 100.00% | 98.65% | 0.0035 | 0.0544 | 0.0 |
| 10 | 100 | 00:00:52 | 100.00% | | 0.0045 | | 0.0 |
| 12 | 120 | 00:01:02 | 100.00% | 98.42% | 0.0021 | 0.0500 | 0.0 |
| 15 | 150 | 00:01:16 | 100.00% | 98.65% | 0.0024 | 0.0463 | 0.0 |

Training finished: Max epochs completed.




```
Test Accuracy for 15 Epochs: 98.87%
Training on single CPU.
Initializing input data normalization.
|==========================================================================================================
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learni
|         |             |   (hh:mm:ss)   |   Accuracy   |   Accuracy   |     Loss     |     Loss     |      Rate
|==========================================================================================================
|       1 |           1 |       00:00:02 |       21.09% |       28.38% |       2.3195 |       3.8634 |          0.0
|       3 |          30 |       00:00:16 |       94.53% |       91.22% |       0.2595 |       0.3194 |          0.0
|       5 |          50 |       00:00:25 |       96.88% |              |       0.1023 |              |          0.0
|       6 |          60 |       00:00:31 |       96.88% |       96.17% |       0.0839 |       0.1556 |          0.0
|       9 |          90 |       00:00:45 |      100.00% |       97.30% |       0.0135 |       0.1099 |          0.0
|      10 |         100 |       00:00:50 |      100.00% |              |       0.0122 |              |          0.0
|      12 |         120 |       00:01:00 |      100.00% |       98.65% |       0.0081 |       0.0860 |          0.0
|      15 |         150 |       00:01:14 |      100.00% |       98.65% |       0.0038 |       0.0809 |          0.0
|      18 |         180 |       00:01:29 |      100.00% |       98.42% |       0.0043 |       0.0801 |          0.0
|      20 |         200 |       00:01:38 |      100.00% |       97.97% |       0.0023 |       0.0791 |          0.0
|==========================================================================================================
Training finished: Max epochs completed.
```

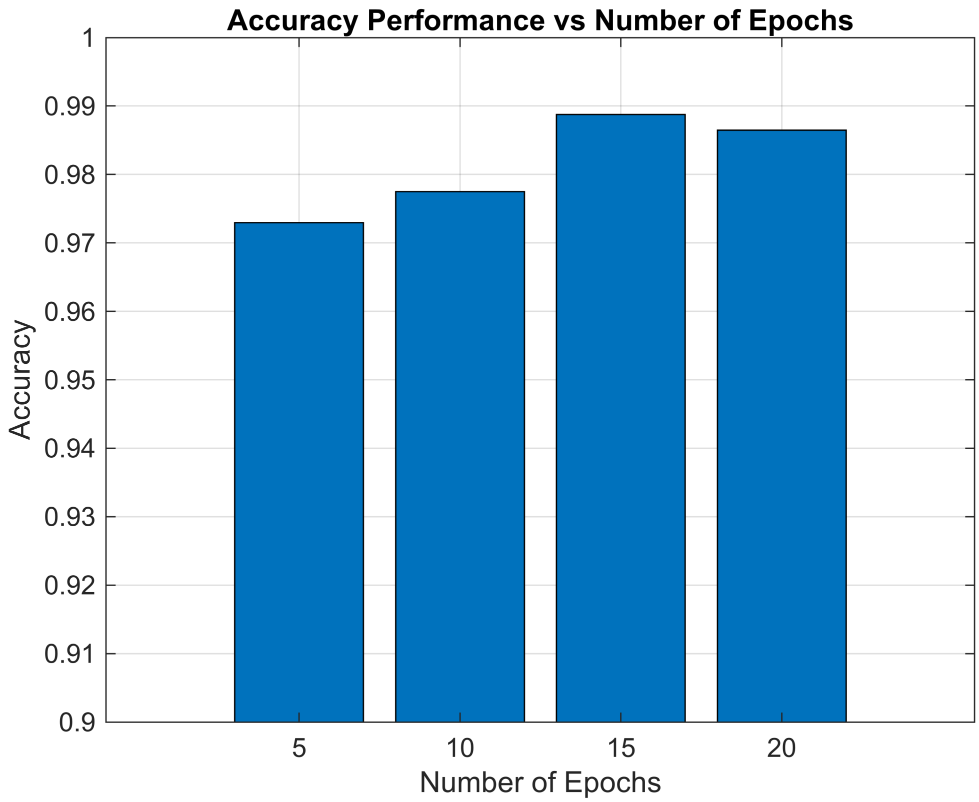



```
Test Accuracy for 20 Epochs: 98.65%
```

```
% Plot the accuracy performance
figure;

bar(numEpochsList, EpochsAccuracyMatrix);

ylim([0.9, 1]);
xlabel('Number of Epochs');
ylabel('Accuracy');
title('Accuracy Performance vs Number of Epochs');
grid on;

% Save the plot as an EPS file
eps_filename = 'results/epochs.eps'
```

```
eps_filename = 
'results/epochs.eps'
```

```
saveas(gcf, eps_filename, 'epsc');
```



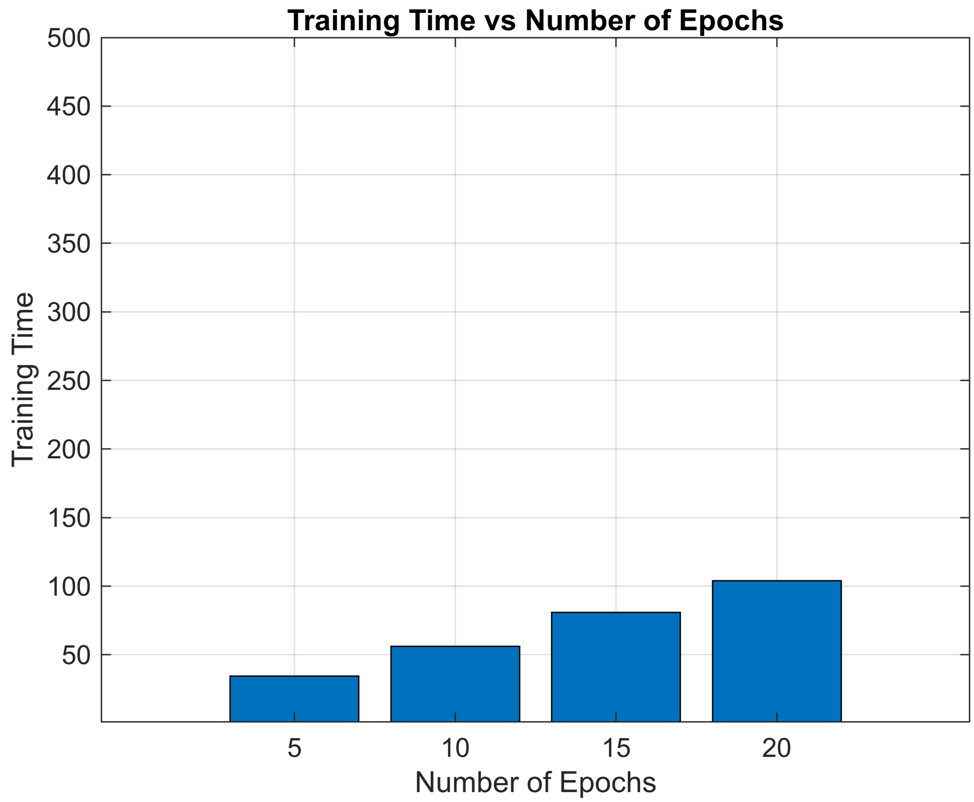

```
fprintf('Saved %s\n', eps_filename);
```

```
Saved results/epochs.eps
```

```
% Plot the training time
figure;

bar(numEpochsList, EpochsTime);

xlabel('Number of Epochs');
ylabel('Training Time');
ylim([1, 500]);

title('Training Time vs Number of Epochs');
grid on;

% Save the plot as an EPS file
eps_filename = 'results/epochs_trainingtime.eps'
```

```
eps_filename =
'results/epochs_trainingtime.eps'
```

```
saveas(gcf, eps_filename, 'epsc');
```



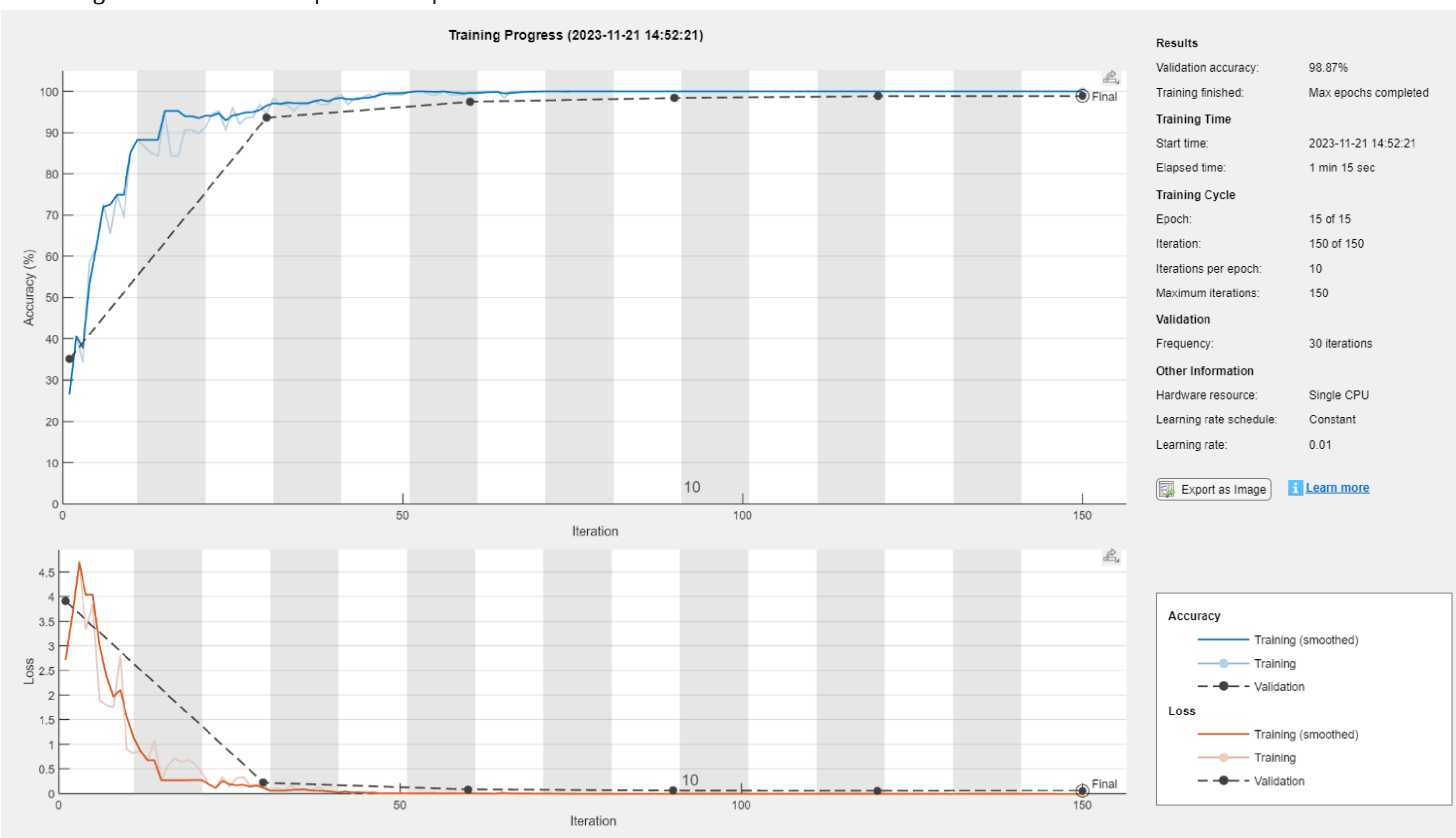

```
fprintf('Saved %s\n', eps_filename);
```

```
Saved results/epochs_trainingtime.eps
```

```
% Hyperparameter tuning for the number of learning rates
learningRates = [0.01, 0.001, 0.0001]; % You can modify this list according to your
requirements

% Initialize a matrix to store accuracy for each number of epochs
LRaccuracyMatrix = zeros(length(learningRates), 1);
LRTrainingTime = zeros(length(learningRates), 1);

for idx = 1:length(learningRates)
    currentlearningRate = learningRates(idx);

    % Update the MaxEpochs parameter in trainingOptions
    options.InitialLearnRate = currentlearningRate;
    options.MaxEpochs = 15;

    trainingStartTime = tic;

    % Train the CNN
    net = trainNetwork(trainImages, trainLabels, layers, options);

    trainingTime = toc(trainingStartTime);
```

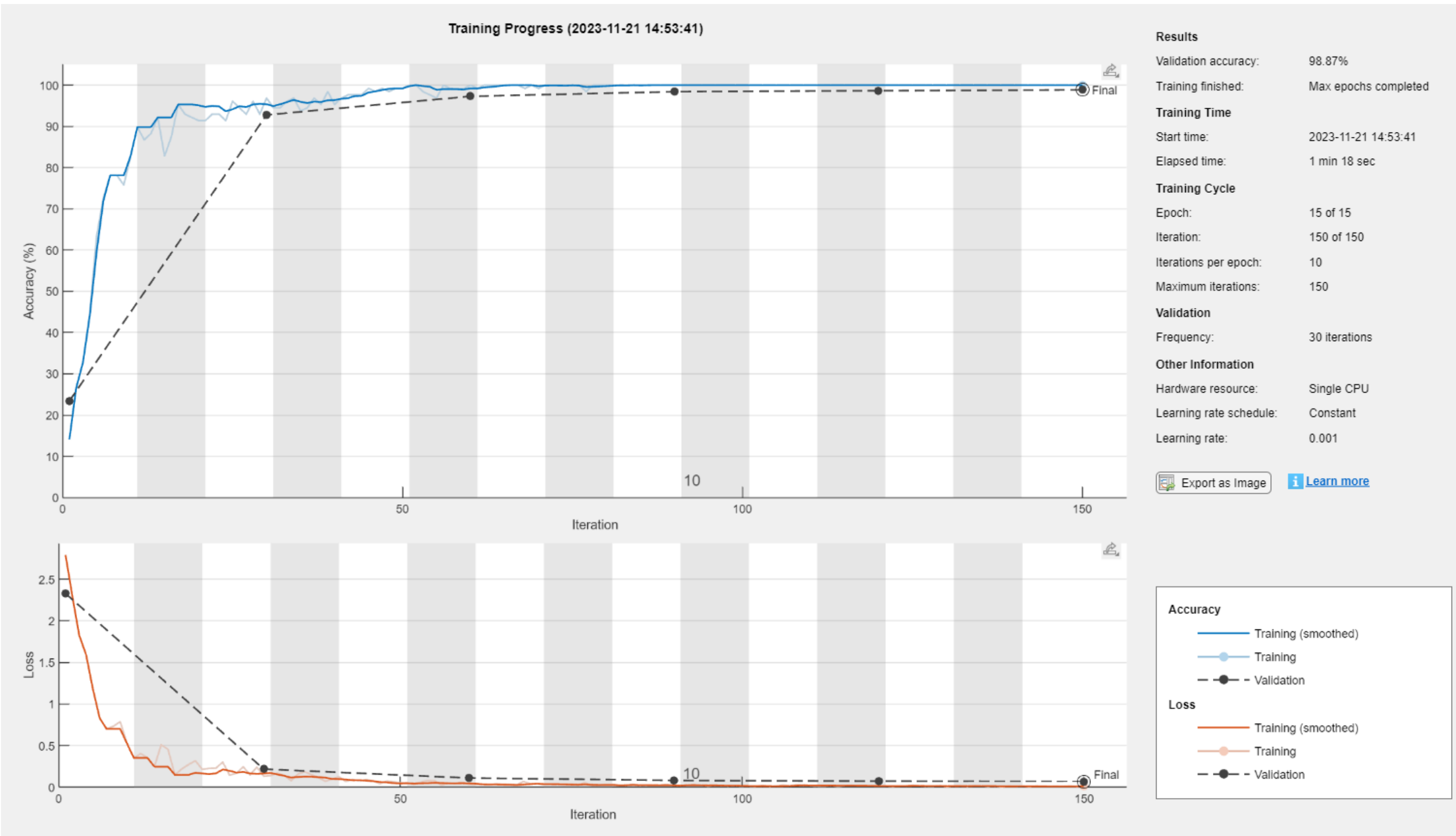
```
    % Classify Test Images
    predictedLabels = classify(net, testImages);

    % Calculate the Accuracy
    accuracy = sum(predictedLabels == testLabels) / numel(testLabels);
    LRaccuracyMatrix(idx) = accuracy;
    LRTrainingTime(idx) = trainingTime;

    fprintf('Test Accuracy for %d Learning Rate: %.2f%%\n', currentlearningRate,
accuracy * 100);
end
```

```
Training on single CPU.
Initializing input data normalization.
|===================================================================================================================
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learnin
|         |             |   (hh:mm:ss)   |   Accuracy   |   Accuracy   |     Loss     |     Loss     |      Rate
|===================================================================================================================
|       1 |           1 |       00:00:02 |       26.56% |       35.14% |       2.7204 |       3.9127 |          0.0
|       3 |          30 |       00:00:17 |       94.53% |       93.69% |       0.1843 |       0.2265 |          0.0
|       5 |          50 |       00:00:26 |       99.22% |              |       0.0151 |              |          0.0
|       6 |          60 |       00:00:31 |      100.00% |       97.52% |       0.0144 |       0.0916 |          0.0
|       9 |          90 |       00:00:46 |      100.00% |       98.42% |       0.0037 |       0.0682 |          0.0
|      10 |         100 |       00:00:50 |      100.00% |              |       0.0023 |              |          0.0
|      12 |         120 |       00:01:00 |      100.00% |       98.87% |       0.0012 |       0.0648 |          0.0
|      15 |         150 |       00:01:14 |      100.00% |       98.87% |       0.0011 |       0.0645 |          0.0
|===================================================================================================================
Training finished: Max epochs completed.
```

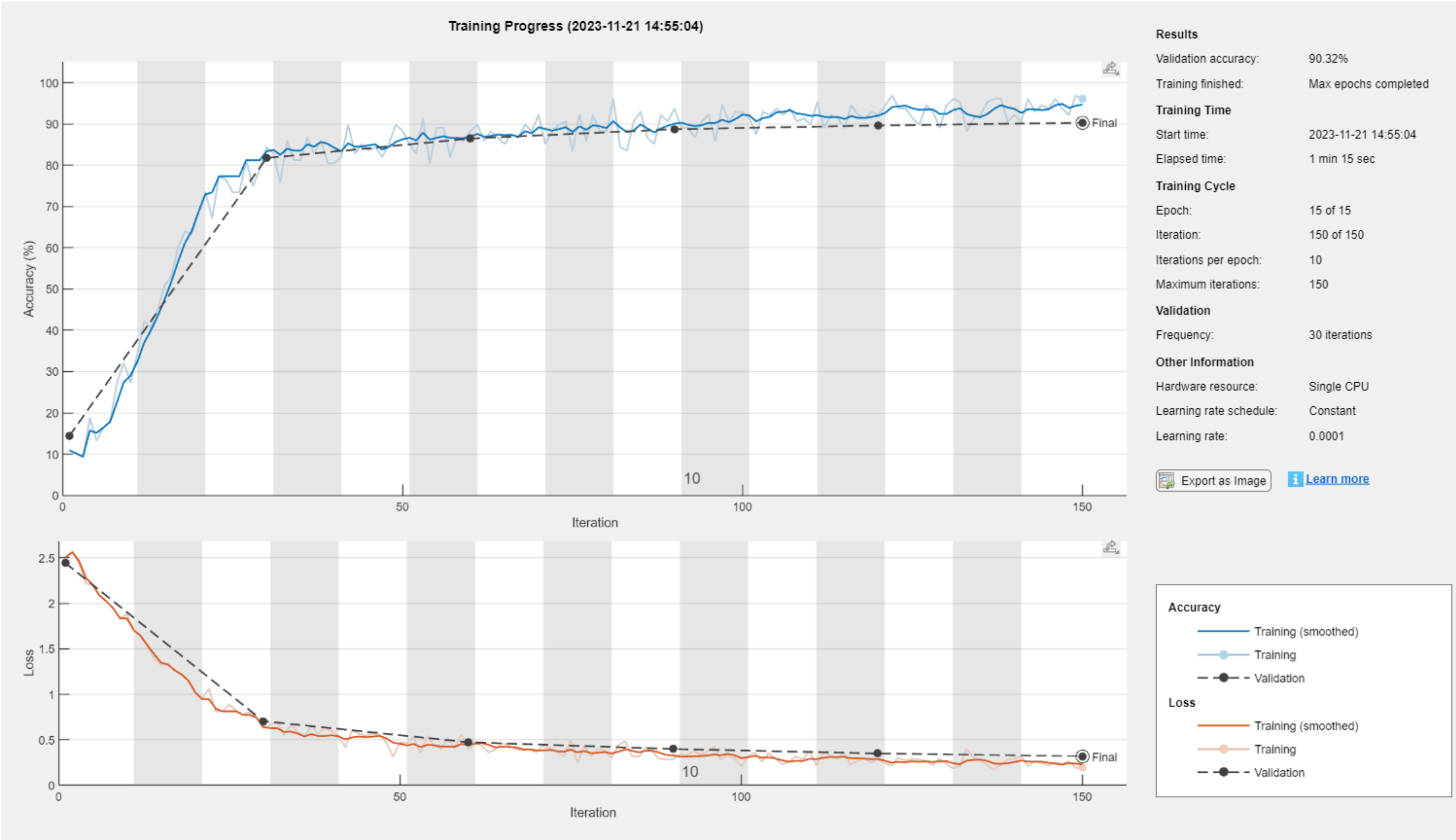



```
Test Accuracy for 1.000000e-02 Learning Rate: 98.87%
```

```
Training on single CPU.
Initializing input data normalization.
|=========================================================================================================================
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learni
|         |             |   (hh:mm:ss)   |   Accuracy   |   Accuracy   |     Loss     |     Loss     |      Rate
|=========================================================================================================================
|       1 |           1 |       00:00:02 |       14.06% |       23.42% |       2.7943 |       2.3296 |          0.00
|       3 |          30 |       00:00:18 |       96.88% |       92.79% |       0.1328 |       0.2186 |          0.00
|       5 |          50 |       00:00:27 |       99.22% |              |       0.0538 |              |          0.00
|       6 |          60 |       00:00:33 |      100.00% |       97.30% |       0.0349 |       0.1119 |          0.00
|       9 |          90 |       00:00:48 |      100.00% |       98.42% |       0.0194 |       0.0830 |          0.00
|      10 |         100 |       00:00:53 |      100.00% |              |       0.0154 |              |          0.00
|      12 |         120 |       00:01:03 |      100.00% |       98.65% |       0.0167 |       0.0734 |          0.00
|      15 |         150 |       00:01:18 |      100.00% |       98.87% |       0.0151 |       0.0678 |          0.00
|=========================================================================================================================
Training finished: Max epochs completed.
```




```
Test Accuracy for 1.000000e-03 Learning Rate: 98.87%
Training on single CPU.
Initializing input data normalization.
|=========================================================================================================================
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learni
|         |             |   (hh:mm:ss)   |   Accuracy   |   Accuracy   |     Loss     |     Loss     |      Rate
|=========================================================================================================================
|       1 |           1 |       00:00:02 |       10.94% |       14.41% |       2.5042 |       2.4422 |      1.0000e-
|       3 |          30 |       00:00:16 |       84.38% |       81.76% |       0.6292 |       0.7046 |      1.0000e-
|       5 |          50 |       00:00:26 |       88.28% |              |       0.4736 |              |      1.0000e-
|       6 |          60 |       00:00:31 |       88.28% |       86.49% |       0.3918 |       0.4739 |      1.0000e-
|       9 |          90 |       00:00:45 |       93.75% |       88.74% |       0.2836 |       0.3995 |      1.0000e-
|      10 |         100 |       00:00:50 |       92.97% |              |       0.2103 |              |      1.0000e-
|      12 |         120 |       00:01:00 |       92.19% |       89.64% |       0.2906 |       0.3502 |      1.0000e-
|      15 |         150 |       00:01:14 |       96.09% |       90.32% |       0.1974 |       0.3169 |      1.0000e-
|=========================================================================================================================
Training finished: Max epochs completed.
```

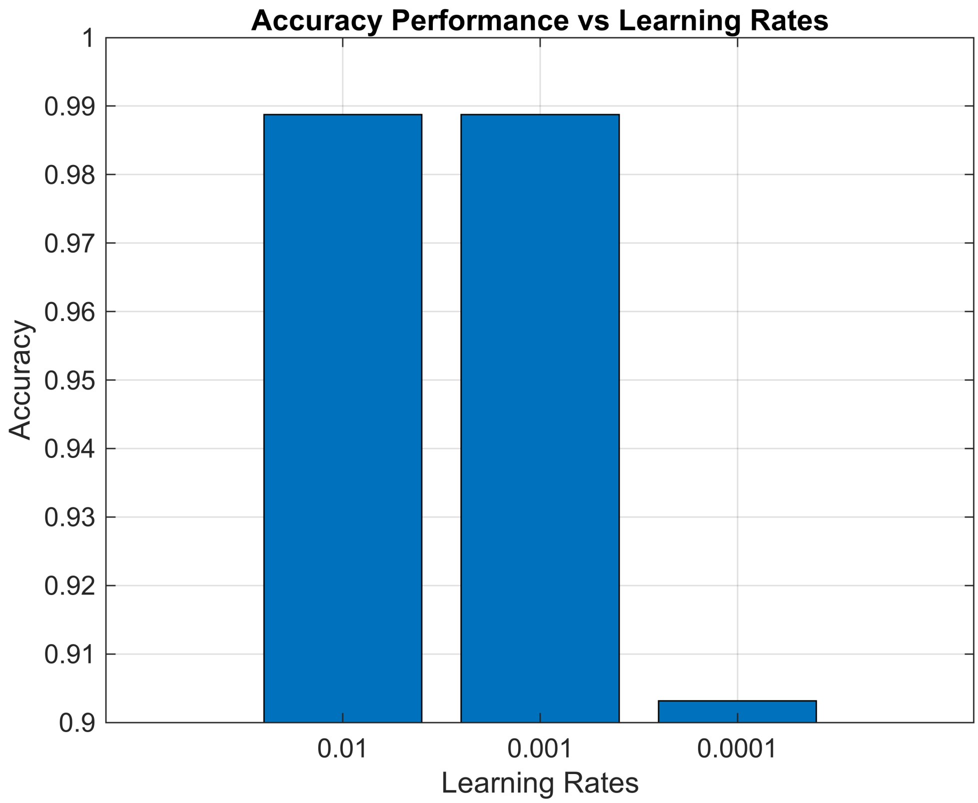



```
Test Accuracy for 1.000000e-04 Learning Rate: 90.32%
```

```
% Plot the accuracy performance
learningRates = {'0.01', '0.001','0.0001'};
learningRateValues = [1, 2, 3];
figure;

% bar(learningRates, LRaccuracyMatrix);
bar(learningRateValues, LRaccuracyMatrix);
xticks(learningRateValues);
xticklabels(learningRates);

xlabel('Learning Rates');
ylabel('Accuracy');
ylim([0.9, 1]);

title('Accuracy Performance vs Learning Rates');
grid on;

% Save the plot as an EPS file
eps_filename = 'results/learning_rate_accuracy.eps'
```

```
eps_filename = 
'results/learning_rate_accuracy.eps'
```

```
saveas(gcf, eps_filename, 'epsc');
```




```
fprintf('Saved %s\n', eps_filename);
```

```
Saved results/learning_rate_accuracy.eps
```

```
% Plot the accuracy performance
learningRates = {'0.01', '0.001','0.0001'};
learningRateValues = [1, 2, 3];
figure;

% bar(learningRates, LRaccuracyMatrix);
bar(learningRateValues, LRTrainingTime);
xticks(learningRateValues);
xticklabels(learningRates);

xlabel('Learning Rates');
ylabel('Training Time');
ylim([1, 500]);

title('Training Time vs Learning Rates');
grid on;

% Save the plot as an EPS file
eps_filename = 'results/learning_rate_trainingtime.eps'
```

```
eps_filename =
'results/learning_rate_trainingtime.eps'
```

```
saveas(gcf, eps_filename, 'epsc');
```

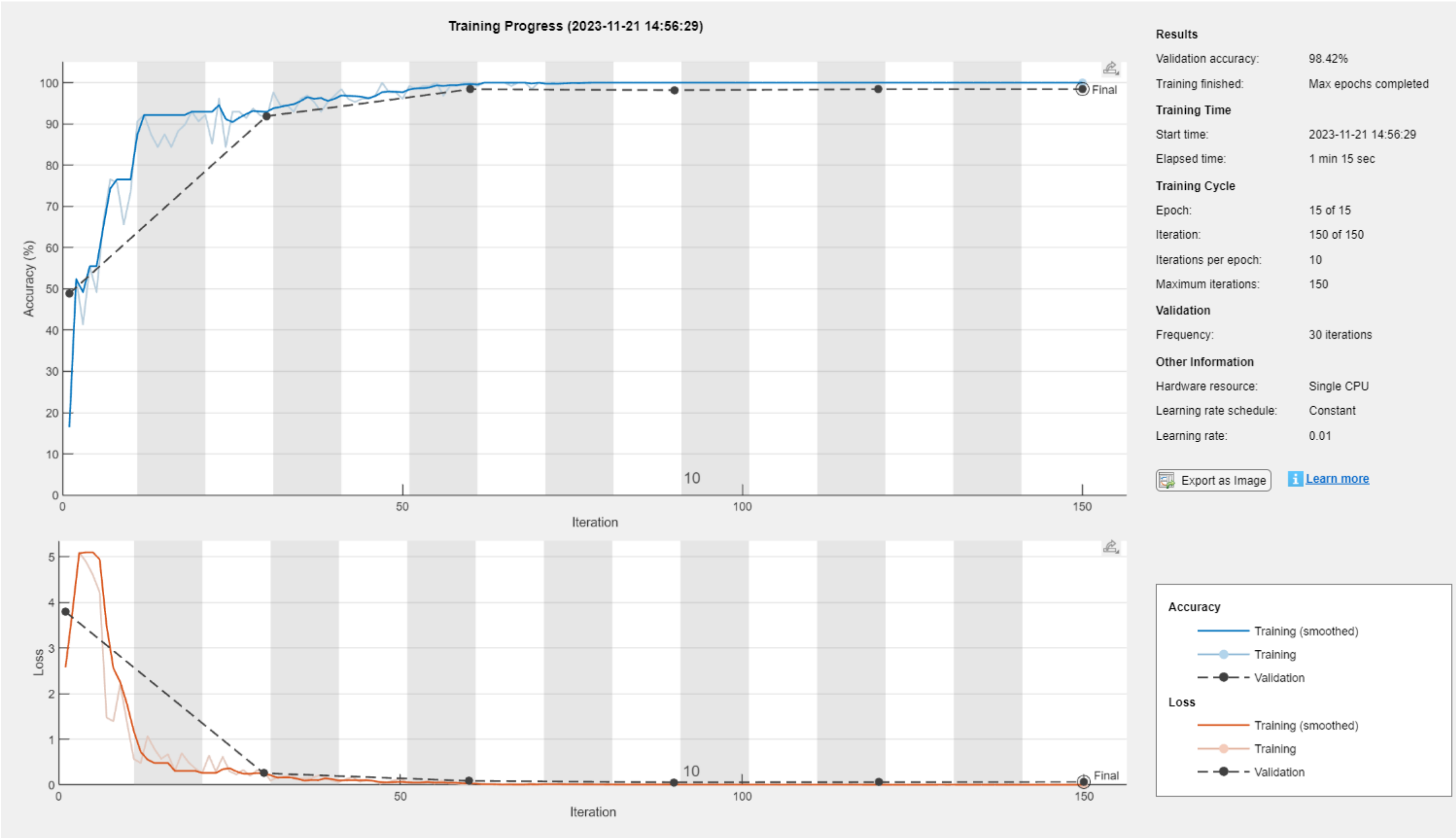

```
fprintf('Saved %s\n', eps_filename);
```

```
Saved results/learning_rate_trainingtime.eps
```

```
% Hyperparameter tuning for the number of epochs
optimizers = {'sgdm', 'adam', 'rmsprop'}; % You can modify this list according to
your requirements

% Initialize a matrix to store accuracy for each number of epochs
OaccuracyMatrix = zeros(length(optimizers), 1);
OTrainingTime = zeros(length(optimizers), 1);

for idx = 1:length(optimizers)
    currentoptimizer = optimizers(idx);

    options = trainingOptions( ...
    currentoptimizer, ... % Stochastic Gradient Descent with Momentum
    'InitialLearnRate', 0.01, ...
    'MaxEpochs', 15, ...
    'Shuffle', 'every-epoch', ...
    'ValidationData', {testImages, testLabels}, ...
    'ValidationFrequency', 30, ...
    'Verbose', true, ...
    'Plots', 'training-progress' ...
```

```
    );

    % Update the MaxEpochs parameter in trainingOptions
    % options.InitialLearnRate = currentlearningRate;
    % options.MaxEpochs = 15;
    % options.InitialLearnRate = 0.01;

    trainingStartTime = tic;

    % Train the CNN
    net = trainNetwork(trainImages, trainLabels, layers, options);

    trainingTime = toc(trainingStartTime);

    % Classify Test Images
    predictedLabels = classify(net, testImages);

    % Calculate the Accuracy
    accuracy = sum(predictedLabels == testLabels) / numel(testLabels);
    OaccuracyMatrix(idx) = accuracy;
    OTrainingTime(idx) = trainingTime;

    % fprintf('Test Accuracy for %s Learning Rate: %.2f%%\n', currentoptimizer,
accuracy * 100);
end
```

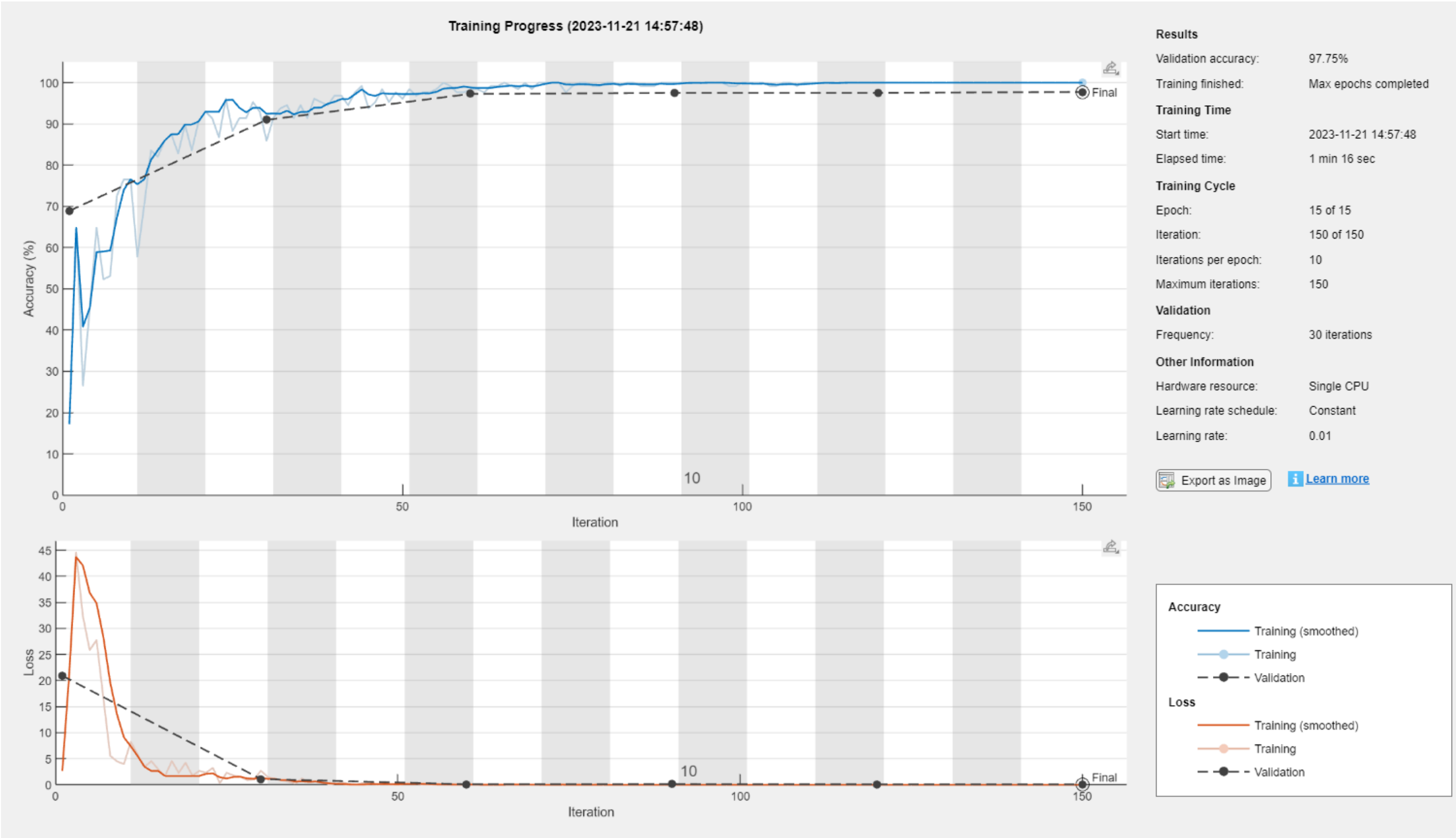
```
Training on single CPU.
Initializing input data normalization.
|==========================================================================================================
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learni
|         |             |   (hh:mm:ss)   |   Accuracy   |   Accuracy   |     Loss     |     Loss     |      Rate
|==========================================================================================================
|       1 |           1 |       00:00:02 |       16.41% |       48.87% |       2.5717 |       3.7912 |          0.01
|       3 |          30 |       00:00:16 |       91.41% |       91.89% |       0.3008 |       0.2554 |          0.01
|       5 |          50 |       00:00:26 |       96.09% |              |       0.1272 |              |          0.01
|       6 |          60 |       00:00:31 |      100.00% |       98.42% |       0.0192 |       0.0862 |          0.01
|       9 |          90 |       00:00:45 |      100.00% |       98.20% |       0.0057 |       0.0566 |          0.01
|      10 |         100 |       00:00:50 |      100.00% |              |       0.0040 |              |          0.01
|      12 |         120 |       00:01:00 |      100.00% |       98.42% |       0.0019 |       0.0596 |          0.01
|      15 |         150 |       00:01:14 |      100.00% |       98.42% |       0.0022 |       0.0590 |          0.01
|==========================================================================================================
Training finished: Max epochs completed.
```




Training on single CPU.
Initializing input data normalization.

| Epoch | Iteration | Time Elapsed (hh:mm:ss) | Mini-batch Accuracy | Validation Accuracy | Mini-batch Loss | Validation Loss | Base Learni Rate |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 00:00:02 | 17.19% | 68.92% | 2.6595 | 20.9126 | 0.0 |
| 3 | 30 | 00:00:16 | 85.94% | 90.99% | 2.7117 | 1.1174 | 0.0 |
| 5 | 50 | 00:00:25 | 96.09% | | 0.3663 | | 0.0 |
| 6 | 60 | 00:00:30 | 98.44% | 97.30% | 0.0345 | 0.1154 | 0.0 |
| 9 | 90 | 00:00:46 | 100.00% | 97.52% | 0.0032 | 0.1385 | 0.0 |
| 10 | 100 | 00:00:51 | 100.00% | | 0.0010 | | 0.0 |
| 12 | 120 | 00:01:01 | 100.00% | 97.52% | 0.0001 | 0.0996 | 0.0 |
| 15 | 150 | 00:01:15 | 100.00% | 97.75% | 0.0003 | 0.0959 | 0.0 |

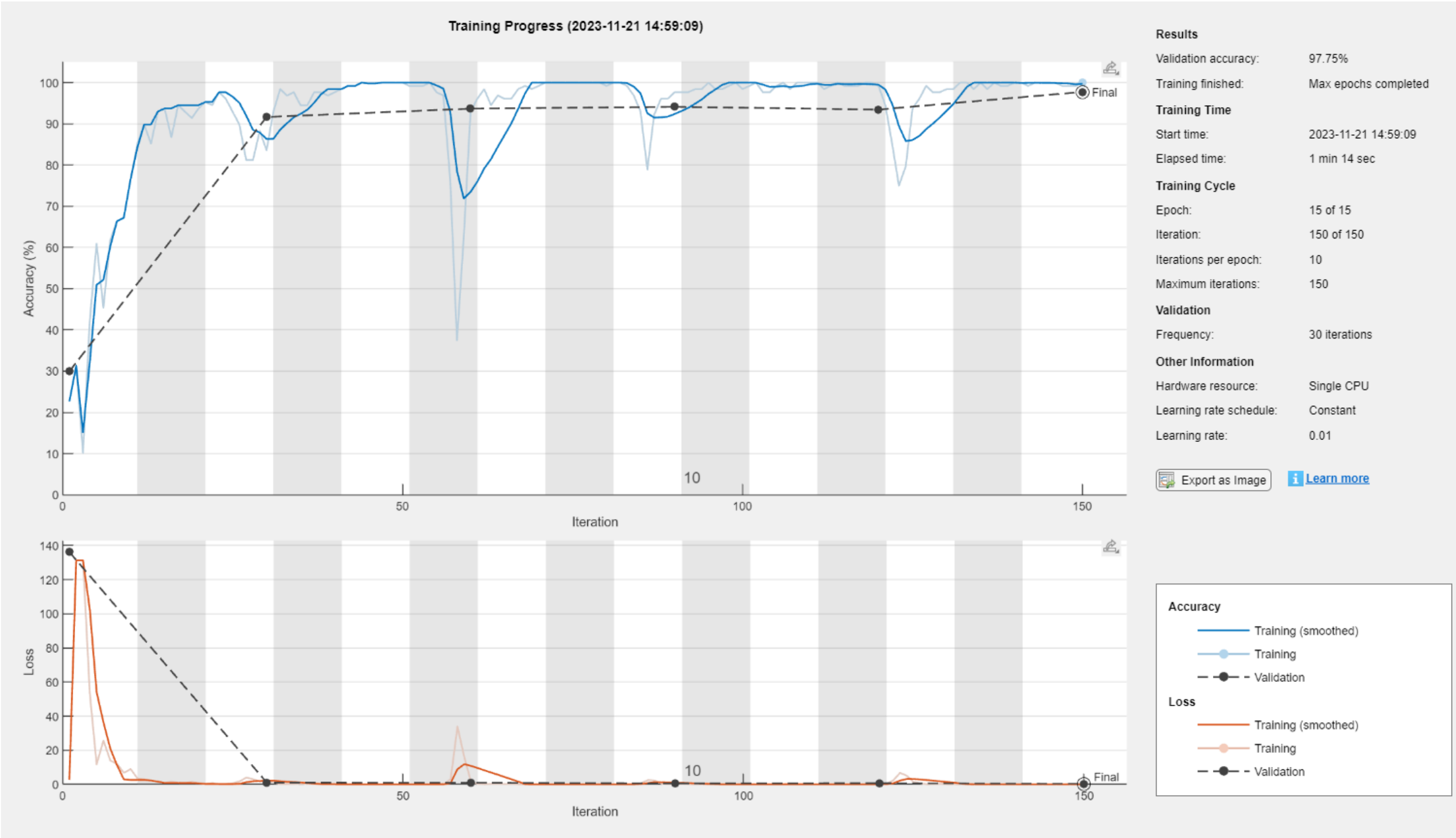
Training finished: Max epochs completed.




Training on single CPU.
Initializing input data normalization.

| Epoch | Iteration | Time Elapsed (hh:mm:ss) | Mini-batch Accuracy | Validation Accuracy | Mini-batch Loss | Validation Loss | Base Learni Rate |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 00:00:02 | 22.66% | 29.95% | 2.6184 | 136.2048 | 0.0 |
| 3 | 30 | 00:00:16 | 83.59% | 91.67% | 2.2580 | 0.9953 | 0.0 |
| 5 | 50 | 00:00:26 | 100.00% | | 0.0037 | | 0.0 |
| 6 | 60 | 00:00:31 | 92.97% | 93.69% | 1.2950 | 0.9489 | 0.0 |
| 9 | 90 | 00:00:45 | 97.66% | 94.14% | 0.2456 | 0.6065 | 0.0 |
| 10 | 100 | 00:00:50 | 98.44% | | 0.0765 | | 0.0 |
| 12 | 120 | 00:00:59 | 99.22% | 93.47% | 0.0608 | 0.7252 | 0.0 |
| 15 | 150 | 00:01:14 | 100.00% | 97.75% | 0.0006 | 0.2003 | 0.0 |

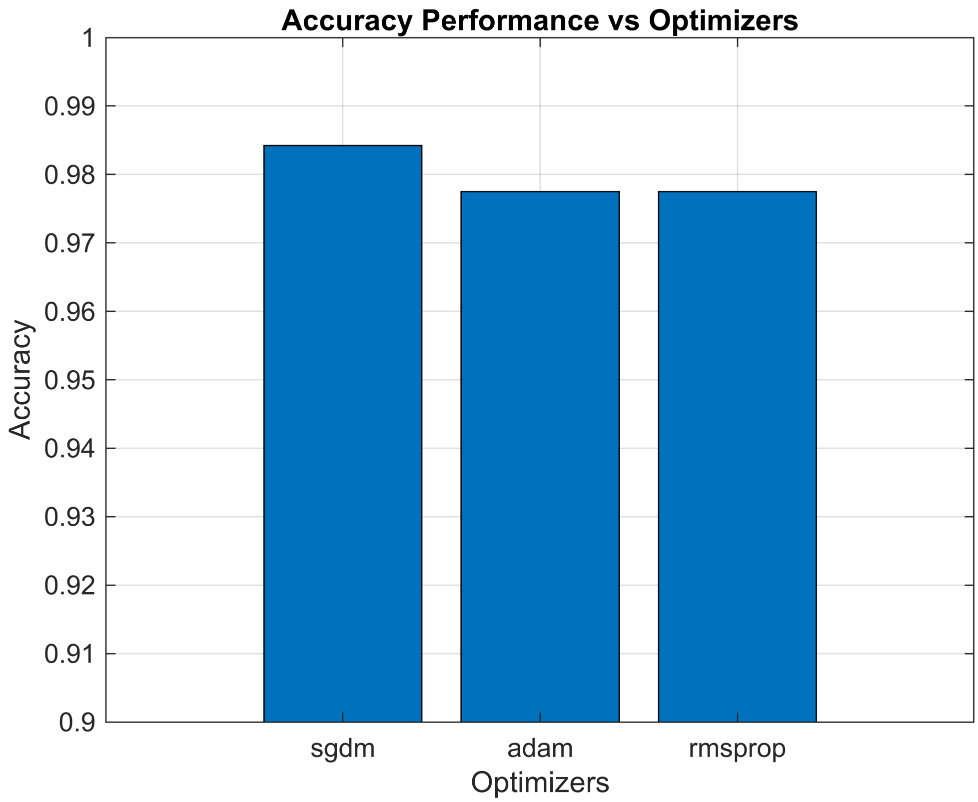
Training finished: Max epochs completed.




```
% Plot the accuracy performance
figure;
bar(OaccuracyMatrix);
xticks([1 2 3]);
xticklabels(optimizers);
xlabel('Optimizers');
ylabel('Accuracy');
ylim([0.9, 1]);
title('Accuracy Performance vs Optimizers');
grid on;

% Save the plot as an EPS file
eps_filename = 'results/optimiser.eps'
```

```
eps_filename = 
'results/optimiser.eps'
```

```
saveas(gcf, eps_filename, 'epsc');
```



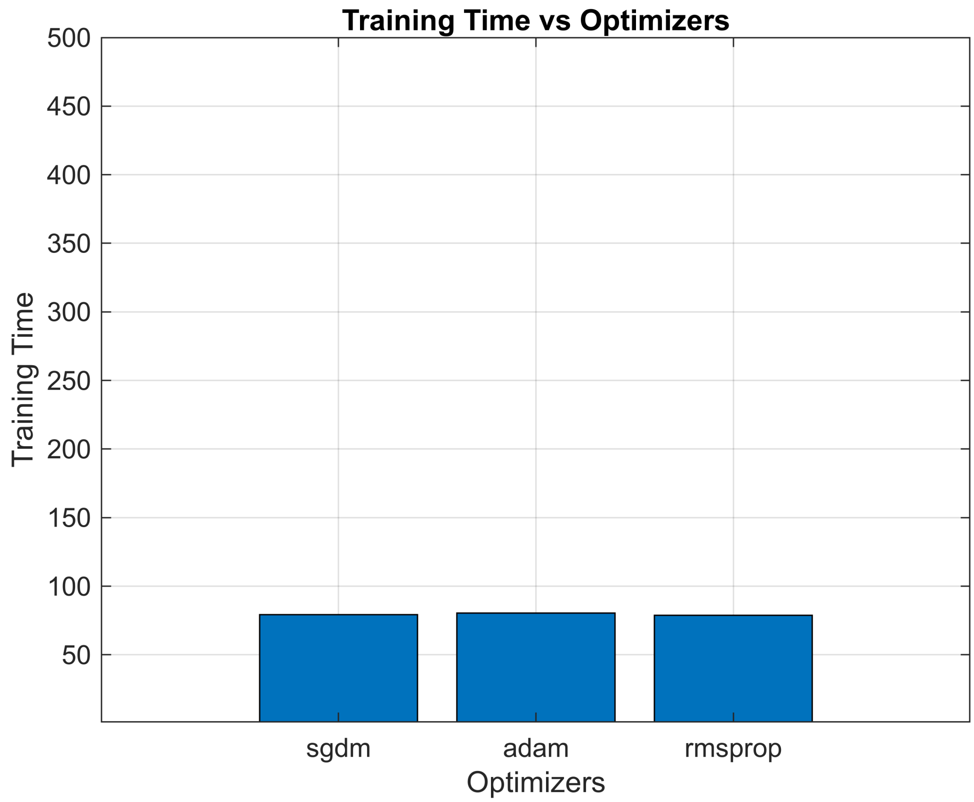

```
fprintf('Saved %s\n', eps_filename);
```

```
Saved results/optimiser.eps
```

```
% Plot the training time
figure;

bar(OTrainingTime);
xticks([1 2 3]);
xticklabels(optimizers);

xlabel('Optimizers');
ylabel('Training Time');
ylim([1, 500]);

title('Training Time vs Optimizers');
grid on;

% Save the plot as an EPS file
eps_filename = 'results/optimiser_trainingtime.eps'
```

```
eps_filename = 
'results/optimiser_trainingtime.eps'
```

```
saveas(gcf, eps_filename, 'epsc');
```




```
fprintf('Saved %s\n', eps_filename);
```

```
Saved results/optimiser_trainingtime.eps
```

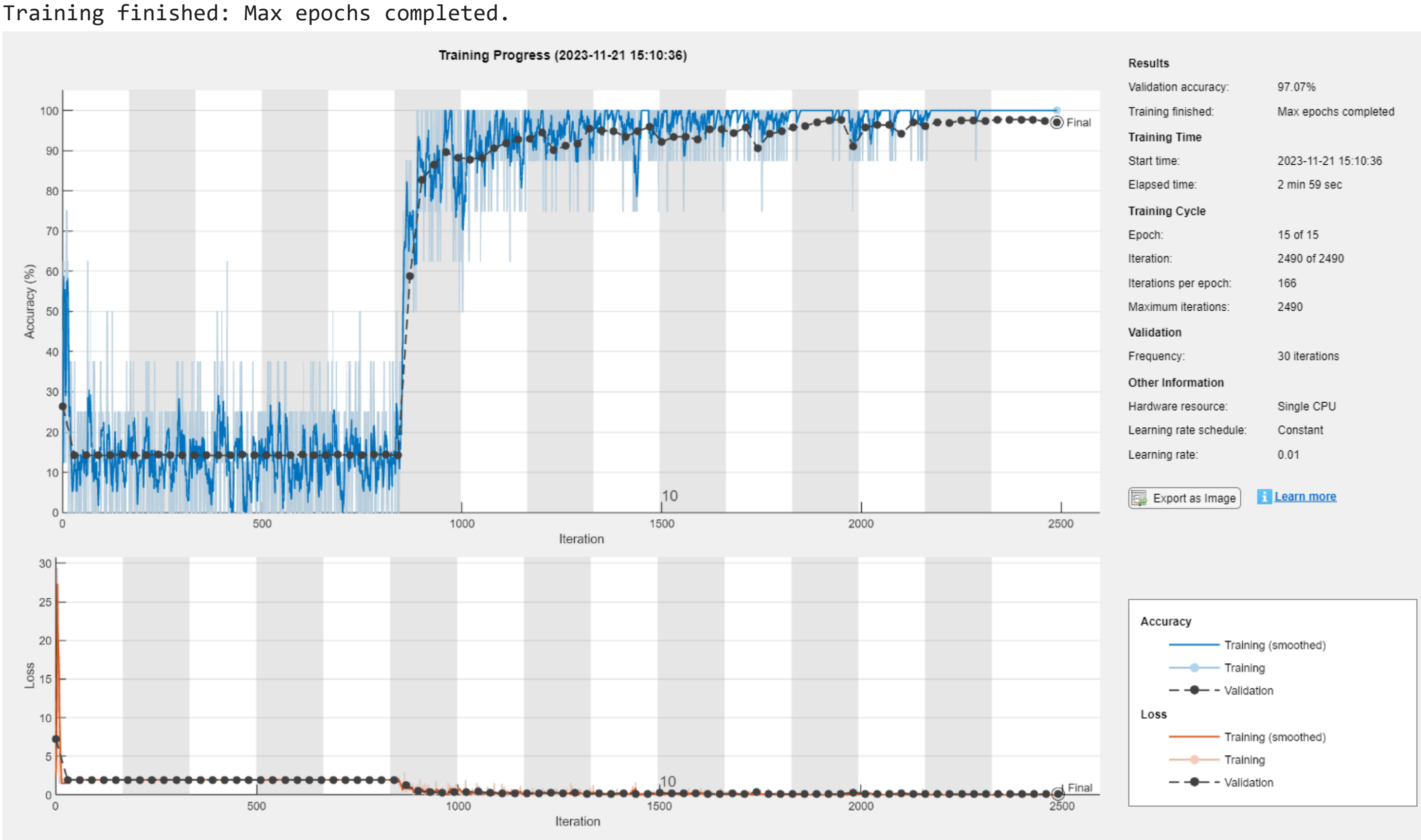
```
% Hyperparameter tuning for the number of epochs
batch_size = [8,16,32]; % You can modify this list according to your requirements

% Initialize a matrix to store accuracy for each number of epochs
batchaccuracyMatrix = zeros(length(batch_size), 1);
batchTrainingTime = zeros(length(batch_size), 1);

for idx = 1:length(batch_size)
    currentbatch_size = batch_size(idx);

    options = trainingOptions( ...
    'sgdm', ... % Stochastic Gradient Descent with Momentum
    'MiniBatchSize', currentbatch_size, ...
    'InitialLearnRate', 0.01, ...
    'MaxEpochs', 15, ...
    'Shuffle', 'every-epoch', ...
    'ValidationData', {testImages, testLabels}, ...
    'ValidationFrequency', 30, ...
    'Verbose', true, ...
    'Plots', 'training-progress' ...
```

```
    );

    % Update the MaxEpochs parameter in trainingOptions
    % options.InitialLearnRate = currentlearningRate;
    % options.MaxEpochs = 15;
    % options.InitialLearnRate = 0.01;

    trainingStartTime = tic;

    % Train the CNN
    net = trainNetwork(trainImages, trainLabels, layers, options);

    trainingTime = toc(trainingStartTime);

    % Classify Test Images
    predictedLabels = classify(net, testImages);

    % Calculate the Accuracy
    accuracy = sum(predictedLabels == testLabels) / numel(testLabels);
    batchaccuracyMatrix(idx) = accuracy;
    batchTrainingTime(idx) = trainingTime;

    % fprintf('Test Accuracy for %s Learning Rate: %.2f%%\n', currentoptimizer,
accuracy * 100);
end
```

```
Training on single CPU.
Initializing input data normalization.
|==================================================================================================================
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learnin
|         |             |   (hh:mm:ss)   |   Accuracy   |   Accuracy   |     Loss     |     Loss     |      Rate
|==================================================================================================================
|       1 |           1 |       00:00:02 |       12.50% |       26.35% |       2.4753 |       7.2069 |          0.01
|       1 |          30 |       00:00:05 |       12.50% |       14.19% |       1.9501 |       1.9469 |          0.01
|       1 |          50 |       00:00:06 |       12.50% |              |       1.9639 |              |          0.01
|       1 |          60 |       00:00:07 |        0.00% |       14.19% |       1.9655 |       1.9464 |          0.01
|       1 |          90 |       00:00:09 |        0.00% |       14.19% |       1.9779 |       1.9475 |          0.01
|       1 |         100 |       00:00:10 |       12.50% |              |       1.9556 |              |          0.01
|       1 |         120 |       00:00:11 |       12.50% |       14.19% |       1.9555 |       1.9472 |          0.01
|       1 |         150 |       00:00:13 |        0.00% |       14.41% |       1.9696 |       1.9463 |          0.01
|       2 |         180 |       00:00:15 |       25.00% |       14.19% |       1.9227 |       1.9473 |          0.01
|       2 |         200 |       00:00:16 |       12.50% |              |       1.9468 |              |          0.01
|       2 |         210 |       00:00:17 |       37.50% |       14.19% |       1.9500 |       1.9465 |          0.01
|       2 |         240 |       00:00:19 |       12.50% |       14.41% |       1.9585 |       1.9476 |          0.01
|       2 |         250 |       00:00:20 |        0.00% |              |       1.9256 |              |          0.01
|       2 |         270 |       00:00:21 |        0.00% |       14.19% |       1.9577 |       1.9465 |          0.01
|       2 |         300 |       00:00:23 |       12.50% |       14.19% |       1.9568 |       1.9475 |          0.01
|       2 |         330 |       00:00:25 |        0.00% |       14.19% |       1.9534 |       1.9476 |          0.01
|       3 |         350 |       00:00:26 |       25.00% |              |       1.9256 |              |          0.01
|       3 |         360 |       00:00:27 |       12.50% |       14.19% |       1.9202 |       1.9475 |          0.01
|       3 |         390 |       00:00:29 |       37.50% |       14.19% |       1.9254 |       1.9470 |          0.01
|       3 |         400 |       00:00:30 |       12.50% |              |       1.9659 |              |          0.01
|       3 |         420 |       00:00:31 |        0.00% |       14.19% |       1.9708 |       1.9493 |          0.01
|       3 |         450 |       00:00:33 |        0.00% |       14.41% |       1.9872 |       1.9476 |          0.01
|       3 |         480 |       00:00:35 |       12.50% |       14.19% |       1.9368 |       1.9467 |          0.01
|       4 |         500 |       00:00:36 |        0.00% |              |       1.9676 |              |          0.01
|       4 |         510 |       00:00:37 |       25.00% |       14.19% |       1.9614 |       1.9471 |          0.01
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 4 | 540 | 00:00:39 | 12.50% | 14.19% | 1.9251 | 1.9482 | 0.01 |
| 4 | 550 | 00:00:39 | 12.50% | | 1.9883 | | 0.01 |
| 4 | 570 | 00:00:41 | 0.00% | 14.19% | 1.9632 | 1.9469 | 0.01 |
| 4 | 600 | 00:00:43 | 0.00% | 14.41% | 1.9495 | 1.9462 | 0.01 |
| 4 | 630 | 00:00:45 | 25.00% | 14.19% | 1.9453 | 1.9463 | 0.01 |
| 4 | 650 | 00:00:46 | 25.00% | | 1.9438 | | 0.01 |
| 4 | 660 | 00:00:47 | 12.50% | 14.19% | 1.9417 | 1.9462 | 0.01 |
| 5 | 690 | 00:00:48 | 25.00% | 14.41% | 1.9353 | 1.9467 | 0.01 |
| 5 | 700 | 00:00:49 | 0.00% | | 1.9722 | | 0.01 |
| 5 | 720 | 00:00:51 | 0.00% | 14.19% | 1.9564 | 1.9467 | 0.01 |
| 5 | 750 | 00:00:52 | 12.50% | 14.19% | 1.9676 | 1.9478 | 0.01 |
| 5 | 780 | 00:00:55 | 25.00% | 14.41% | 1.9270 | 1.9466 | 0.01 |
| 5 | 800 | 00:00:56 | 25.00% | | 1.9321 | | 0.01 |
| 5 | 810 | 00:00:57 | 25.00% | 14.41% | 1.9332 | 1.9462 | 0.01 |
| 6 | 840 | 00:00:59 | 25.00% | 14.19% | 1.9434 | 1.9493 | 0.01 |
| 6 | 850 | 00:00:59 | 37.50% | | 2.2898 | | 0.01 |
| 6 | 870 | 00:01:00 | 75.00% | 58.78% | 0.7304 | 1.2387 | 0.01 |
| 6 | 900 | 00:01:02 | 87.50% | 82.66% | 0.5822 | 0.5406 | 0.01 |
| 6 | 930 | 00:01:04 | 87.50% | 86.49% | 0.4220 | 0.3655 | 0.01 |
| 6 | 950 | 00:01:05 | 87.50% | | 0.3591 | | 0.01 |
| 6 | 960 | 00:01:06 | 87.50% | 89.64% | 0.1335 | 0.3191 | 0.01 |
| 6 | 990 | 00:01:08 | 100.00% | 88.29% | 0.1146 | 0.3555 | 0.01 |
| 7 | 1000 | 00:01:09 | 75.00% | | 0.6976 | | 0.01 |
| 7 | 1020 | 00:01:10 | 100.00% | 87.84% | 0.1262 | 0.3999 | 0.01 |
| 7 | 1050 | 00:01:13 | 75.00% | 88.29% | 0.8298 | 0.4097 | 0.01 |
| 7 | 1080 | 00:01:15 | 100.00% | 90.54% | 0.0558 | 0.2684 | 0.01 |
| 7 | 1100 | 00:01:17 | 87.50% | | 0.2942 | | 0.01 |
| 7 | 1110 | 00:01:18 | 100.00% | 91.89% | 0.0093 | 0.1997 | 0.01 |
| 7 | 1140 | 00:01:20 | 100.00% | 92.79% | 0.1100 | 0.1875 | 0.01 |
| 7 | 1150 | 00:01:21 | 87.50% | | 0.1808 | | 0.01 |
| 8 | 1170 | 00:01:23 | 87.50% | 93.02% | 0.2432 | 0.1868 | 0.01 |
| 8 | 1200 | 00:01:25 | 100.00% | 94.59% | 0.2964 | 0.1947 | 0.01 |
| 8 | 1230 | 00:01:28 | 100.00% | 90.09% | 0.1618 | 0.2923 | 0.01 |
| 8 | 1250 | 00:01:29 | 100.00% | | 0.0030 | | 0.01 |
| 8 | 1260 | 00:01:30 | 100.00% | 91.22% | 0.2060 | 0.2388 | 0.01 |
| 8 | 1290 | 00:01:32 | 75.00% | 91.67% | 0.5371 | 0.2547 | 0.01 |
| 8 | 1300 | 00:01:33 | 100.00% | | 0.0309 | | 0.01 |
| 8 | 1320 | 00:01:35 | 75.00% | 95.50% | 0.6313 | 0.1345 | 0.01 |
| 9 | 1350 | 00:01:37 | 100.00% | 95.05% | 0.0858 | 0.1732 | 0.01 |
| 9 | 1380 | 00:01:39 | 100.00% | 94.82% | 0.0408 | 0.1538 | 0.01 |
| 9 | 1400 | 00:01:40 | 87.50% | | 0.2466 | | 0.01 |
| 9 | 1410 | 00:01:42 | 87.50% | 93.47% | 0.1636 | 0.2286 | 0.01 |
| 9 | 1440 | 00:01:44 | 87.50% | 94.82% | 1.5713 | 0.1497 | 0.01 |
| 9 | 1450 | 00:01:44 | 100.00% | | 0.1542 | | 0.01 |
| 9 | 1470 | 00:01:46 | 100.00% | 95.95% | 0.0013 | 0.1406 | 0.01 |
| 10 | 1500 | 00:01:48 | 100.00% | 92.12% | 0.0281 | 0.2554 | 0.01 |
| 10 | 1530 | 00:01:50 | 100.00% | 93.47% | 0.0142 | 0.1967 | 0.01 |
| 10 | 1550 | 00:01:51 | 87.50% | | 0.2747 | | 0.01 |
| 10 | 1560 | 00:01:52 | 100.00% | 93.47% | 0.0063 | 0.1903 | 0.01 |
| 10 | 1590 | 00:01:54 | 87.50% | 92.79% | 0.2971 | 0.2061 | 0.01 |
| 10 | 1600 | 00:01:55 | 100.00% | | 0.0232 | | 0.01 |
| 10 | 1620 | 00:01:56 | 100.00% | 95.27% | 0.0084 | 0.1154 | 0.01 |
| 10 | 1650 | 00:01:59 | 100.00% | 95.27% | 0.0130 | 0.1500 | 0.01 |
| 11 | 1680 | 00:02:01 | 100.00% | 94.37% | 0.0092 | 0.1797 | 0.01 |
| 11 | 1700 | 00:02:02 | 100.00% | | 0.0116 | | 0.01 |
| 11 | 1710 | 00:02:03 | 100.00% | 95.72% | 0.0066 | 0.1216 | 0.01 |
| 11 | 1740 | 00:02:05 | 100.00% | 90.54% | 0.1149 | 0.3291 | 0.01 |
| 11 | 1750 | 00:02:05 | 100.00% | | 0.0475 | | 0.01 |
| 11 | 1770 | 00:02:07 | 100.00% | 94.14% | 0.0048 | 0.1527 | 0.01 |
| 11 | 1800 | 00:02:09 | 87.50% | 94.82% | 0.3181 | 0.1424 | 0.01 |
| 12 | 1830 | 00:02:11 | 100.00% | 95.72% | 0.0731 | 0.1391 | 0.01 |
| 12 | 1850 | 00:02:12 | 100.00% | | 0.0003 | | 0.01 |
| 12 | 1860 | 00:02:13 | 100.00% | 96.17% | 0.0006 | 0.1226 | 0.01 |
| 12 | 1890 | 00:02:15 | 100.00% | 97.07% | 0.0006 | 0.1056 | 0.01 |

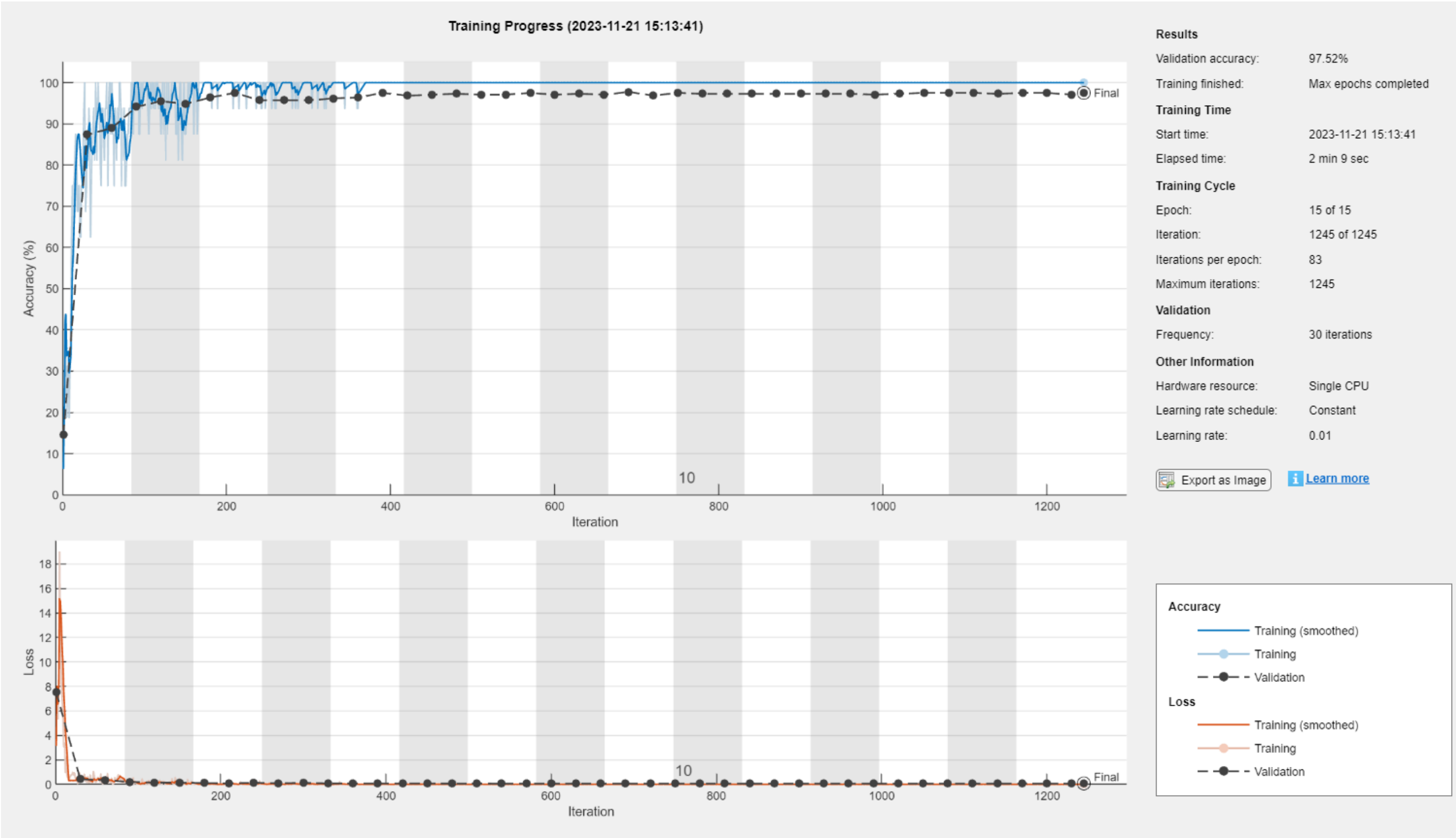
```
|      12 |        1900 |       00:02:15 |     100.00% |             |       0.0146 |             |          0.0
|      12 |        1920 |       00:02:17 |     100.00% |      97.52% |       0.0004 |      0.1076 |          0.0
|      12 |        1950 |       00:02:19 |     100.00% |      97.75% |   1.2114e-05 |      0.1121 |          0.0
|      12 |        1980 |       00:02:21 |     100.00% |      90.99% |       0.0029 |      0.3241 |          0.0
|      13 |        2000 |       00:02:22 |     100.00% |             |       0.0686 |             |          0.0
|      13 |        2010 |       00:02:23 |     100.00% |      95.72% |       0.0132 |      0.1416 |          0.0
|      13 |        2040 |       00:02:25 |      87.50% |      96.40% |       0.4183 |      0.1166 |          0.0
|      13 |        2050 |       00:02:26 |     100.00% |             |       0.0001 |             |          0.0
|      13 |        2070 |       00:02:28 |     100.00% |      96.40% |       0.0209 |      0.0986 |          0.0
|      13 |        2100 |       00:02:30 |     100.00% |      94.14% |       0.0018 |      0.2173 |          0.0
|      13 |        2130 |       00:02:32 |     100.00% |      97.07% |       0.0017 |      0.1219 |          0.0
|      13 |        2150 |       00:02:33 |     100.00% |             |       0.0483 |             |          0.0
|      14 |        2160 |       00:02:34 |     100.00% |      96.17% |   4.5187e-05 |      0.1271 |          0.0
|      14 |        2190 |       00:02:36 |     100.00% |      97.07% |       0.0007 |      0.1073 |          0.0
|      14 |        2200 |       00:02:37 |     100.00% |             |       0.0061 |             |          0.0
|      14 |        2220 |       00:02:39 |     100.00% |      96.85% |       0.0906 |      0.0937 |          0.0
|      14 |        2250 |       00:02:41 |     100.00% |      97.52% |   3.3676e-06 |      0.0969 |          0.0
|      14 |        2280 |       00:02:43 |     100.00% |      97.52% |   1.3262e-06 |      0.1035 |          0.0
|      14 |        2300 |       00:02:44 |     100.00% |             |       0.0003 |             |          0.0
|      14 |        2310 |       00:02:45 |     100.00% |      97.30% |       0.0162 |      0.0952 |          0.0
|      15 |        2340 |       00:02:47 |     100.00% |      97.75% |       0.0001 |      0.0960 |          0.0
|      15 |        2350 |       00:02:48 |     100.00% |             |       0.0019 |             |          0.0
|      15 |        2370 |       00:02:50 |     100.00% |      97.75% |   2.1069e-05 |      0.0899 |          0.0
|      15 |        2400 |       00:02:52 |     100.00% |      97.75% |       0.0014 |      0.0908 |          0.0
|      15 |        2430 |       00:02:54 |     100.00% |      97.75% |       0.0007 |      0.0956 |          0.0
|      15 |        2450 |       00:02:55 |     100.00% |             |       0.0133 |             |          0.0
|      15 |        2460 |       00:02:56 |     100.00% |      97.30% |       0.0007 |      0.0979 |          0.0
|      15 |        2490 |       00:02:58 |     100.00% |      97.07% |       0.0004 |      0.0997 |          0.0
|==========================================================================================================
Training finished: Max epochs completed.
```




```
Training on single CPU.
Initializing input data normalization.
|==========================================================================================================
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learni
```

| | | (hh:mm:ss) | Accuracy | Accuracy | Loss | Loss | Rate |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 00:00:02 | 6.25% | 14.64% | 3.1439 | 7.5239 | 0.01 |
| 1 | 30 | 00:00:07 | 87.50% | 87.39% | 0.3432 | 0.4231 | 0.01 |
| 1 | 50 | 00:00:09 | 87.50% | | 0.2728 | | 0.01 |
| 1 | 60 | 00:00:11 | 100.00% | 88.96% | 0.1807 | 0.3496 | 0.01 |
| 2 | 90 | 00:00:15 | 100.00% | 94.14% | 0.0819 | 0.1983 | 0.01 |
| 2 | 100 | 00:00:16 | 100.00% | | 0.0337 | | 0.01 |
| 2 | 120 | 00:00:18 | 87.50% | 95.50% | 0.3813 | 0.1387 | 0.01 |
| 2 | 150 | 00:00:23 | 87.50% | 94.82% | 0.4499 | 0.1407 | 0.01 |
| 3 | 180 | 00:00:27 | 100.00% | 96.40% | 0.0105 | 0.1103 | 0.01 |
| 3 | 200 | 00:00:29 | 100.00% | | 0.0034 | | 0.01 |
| 3 | 210 | 00:00:31 | 100.00% | 97.52% | 0.0084 | 0.0896 | 0.01 |
| 3 | 240 | 00:00:34 | 100.00% | 95.72% | 0.0063 | 0.1309 | 0.01 |
| 4 | 250 | 00:00:35 | 100.00% | | 0.0321 | | 0.01 |
| 4 | 270 | 00:00:37 | 100.00% | 95.72% | 0.0017 | 0.0962 | 0.01 |
| 4 | 300 | 00:00:40 | 100.00% | 95.72% | 0.0010 | 0.1198 | 0.01 |
| 4 | 330 | 00:00:43 | 100.00% | 96.17% | 0.0013 | 0.0895 | 0.01 |
| 5 | 350 | 00:00:44 | 100.00% | | 0.0034 | | 0.01 |
| 5 | 360 | 00:00:45 | 93.75% | 96.40% | 0.0470 | 0.0838 | 0.01 |
| 5 | 390 | 00:00:48 | 100.00% | 97.52% | 0.0004 | 0.0765 | 0.01 |
| 5 | 400 | 00:00:49 | 100.00% | | 0.0011 | | 0.01 |
| 6 | 420 | 00:00:51 | 100.00% | 96.85% | 4.8148e-05 | 0.0767 | 0.01 |
| 6 | 450 | 00:00:54 | 100.00% | 97.07% | 0.0007 | 0.0786 | 0.01 |
| 6 | 480 | 00:00:57 | 100.00% | 97.30% | 0.0004 | 0.0685 | 0.01 |
| 7 | 500 | 00:00:58 | 100.00% | | 0.0009 | | 0.01 |
| 7 | 510 | 00:01:00 | 100.00% | 97.07% | 0.0003 | 0.0708 | 0.01 |
| 7 | 540 | 00:01:03 | 100.00% | 97.07% | 0.0013 | 0.0694 | 0.01 |
| 7 | 550 | 00:01:03 | 100.00% | | 0.0005 | | 0.01 |
| 7 | 570 | 00:01:05 | 100.00% | 97.52% | 0.0005 | 0.0643 | 0.01 |
| 8 | 600 | 00:01:08 | 100.00% | 97.07% | 0.0006 | 0.0652 | 0.01 |
| 8 | 630 | 00:01:11 | 100.00% | 97.30% | 0.0021 | 0.0665 | 0.01 |
| 8 | 650 | 00:01:13 | 100.00% | | 0.0002 | | 0.01 |
| 8 | 660 | 00:01:14 | 100.00% | 97.07% | 2.5962e-05 | 0.0697 | 0.01 |
| 9 | 690 | 00:01:17 | 100.00% | 97.75% | 0.0012 | 0.0622 | 0.01 |
| 9 | 700 | 00:01:18 | 100.00% | | 0.0013 | | 0.01 |
| 9 | 720 | 00:01:20 | 100.00% | 96.85% | 6.8530e-05 | 0.0650 | 0.01 |
| 10 | 750 | 00:01:23 | 100.00% | 97.52% | 0.0001 | 0.0656 | 0.01 |
| 10 | 780 | 00:01:25 | 100.00% | 97.30% | 6.3996e-05 | 0.0660 | 0.01 |
| 10 | 800 | 00:01:27 | 100.00% | | 8.4134e-05 | | 0.01 |
| 10 | 810 | 00:01:28 | 100.00% | 97.30% | 0.0002 | 0.0666 | 0.01 |
| 11 | 840 | 00:01:31 | 100.00% | 97.30% | 0.0001 | 0.0674 | 0.01 |
| 11 | 850 | 00:01:31 | 100.00% | | 0.0012 | | 0.01 |
| 11 | 870 | 00:01:34 | 100.00% | 97.30% | 0.0002 | 0.0681 | 0.01 |
| 11 | 900 | 00:01:36 | 100.00% | 97.30% | 0.0012 | 0.0638 | 0.01 |
| 12 | 930 | 00:01:39 | 100.00% | 97.30% | 0.0002 | 0.0639 | 0.01 |
| 12 | 950 | 00:01:40 | 100.00% | | 0.0001 | | 0.01 |
| 12 | 960 | 00:01:42 | 100.00% | 97.30% | 9.9904e-05 | 0.0677 | 0.01 |
| 12 | 990 | 00:01:44 | 100.00% | 97.07% | 5.9957e-05 | 0.0649 | 0.01 |
| 13 | 1000 | 00:01:45 | 100.00% | | 6.7338e-05 | | 0.01 |
| 13 | 1020 | 00:01:47 | 100.00% | 97.30% | 0.0004 | 0.0652 | 0.01 |
| 13 | 1050 | 00:01:50 | 100.00% | 97.52% | 5.5128e-05 | 0.0644 | 0.01 |
| 14 | 1080 | 00:01:52 | 100.00% | 97.52% | 0.0003 | 0.0631 | 0.01 |
| 14 | 1100 | 00:01:54 | 100.00% | | 0.0001 | | 0.01 |
| 14 | 1110 | 00:01:55 | 100.00% | 97.52% | 9.3230e-05 | 0.0666 | 0.01 |
| 14 | 1140 | 00:01:58 | 100.00% | 97.30% | 0.0003 | 0.0632 | 0.01 |
| 14 | 1150 | 00:01:59 | 100.00% | | 0.0001 | | 0.01 |
| 15 | 1170 | 00:02:01 | 100.00% | 97.52% | 2.5916e-05 | 0.0640 | 0.01 |
| 15 | 1200 | 00:02:03 | 100.00% | 97.52% | 0.0002 | 0.0644 | 0.01 |
| 15 | 1230 | 00:02:06 | 100.00% | 97.07% | 0.0005 | 0.0647 | 0.01 |
| 15 | 1245 | 00:02:08 | 100.00% | 97.52% | 2.0451e-05 | 0.0654 | 0.01 |

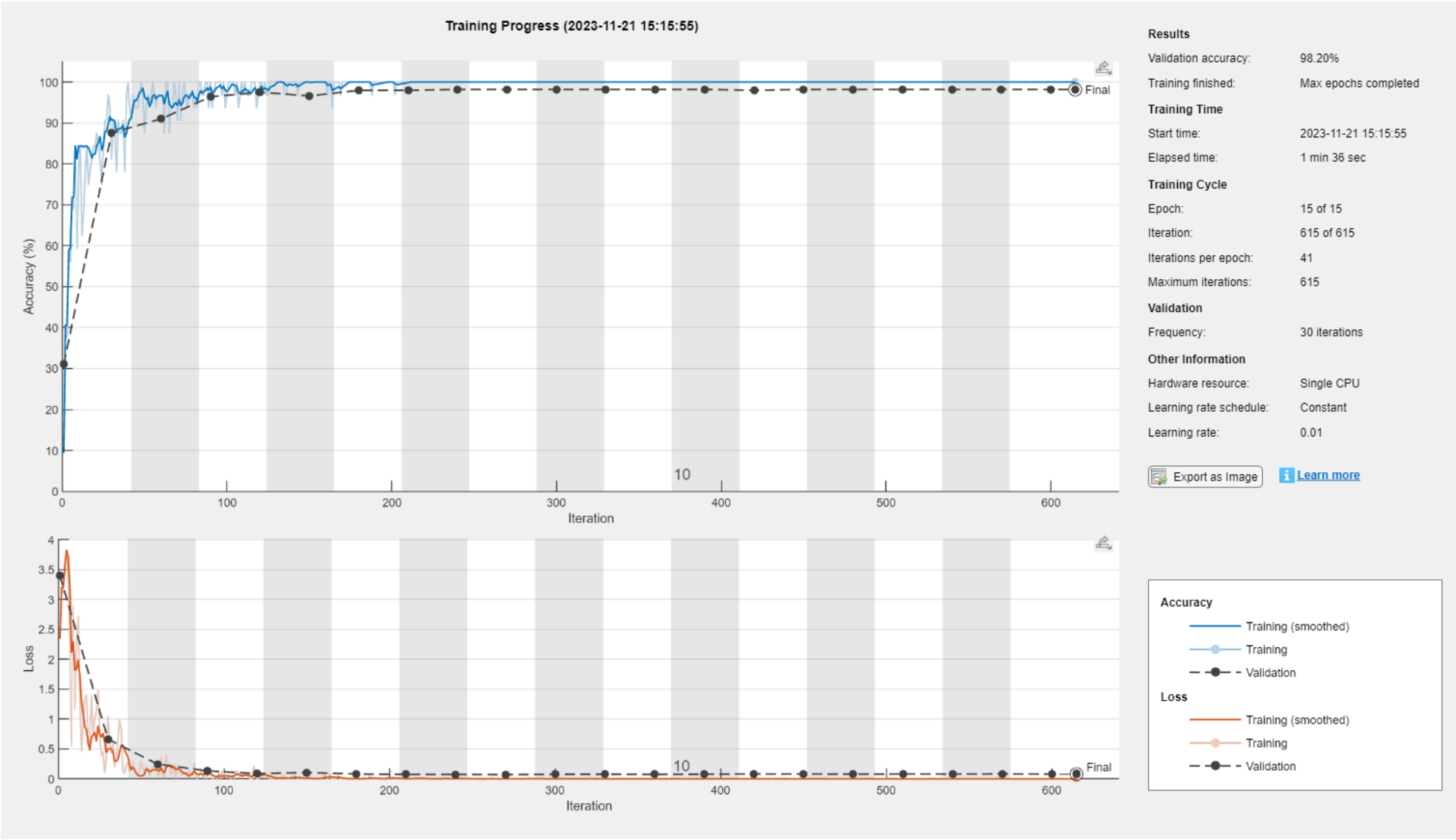
Training finished: Max epochs completed.




Training on single CPU.
Initializing input data normalization.

| Epoch | Iteration | Time Elapsed (hh:mm:ss) | Mini-batch Accuracy | Validation Accuracy | Mini-batch Loss | Validation Loss | Base Learni Rate |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 00:00:02 | 9.38% | 31.08% | 2.3488 | 3.3944 | 0.01 |
| 1 | 30 | 00:00:07 | 81.25% | 87.61% | 1.0391 | 0.6638 | 0.01 |
| 2 | 50 | 00:00:10 | 87.50% | | 0.2941 | | 0.01 |
| 2 | 60 | 00:00:11 | 96.88% | 90.99% | 0.0456 | 0.2467 | 0.01 |
| 3 | 90 | 00:00:16 | 100.00% | 96.40% | 0.0116 | 0.1339 | 0.01 |
| 3 | 100 | 00:00:17 | 96.88% | | 0.0818 | | 0.01 |
| 3 | 120 | 00:00:21 | 100.00% | 97.52% | 0.0011 | 0.0885 | 0.01 |
| 4 | 150 | 00:00:25 | 100.00% | 96.62% | 0.0120 | 0.1052 | 0.01 |
| 5 | 180 | 00:00:30 | 100.00% | 97.97% | 0.0074 | 0.0830 | 0.01 |
| 5 | 200 | 00:00:32 | 100.00% | | 0.0011 | | 0.01 |
| 6 | 210 | 00:00:34 | 100.00% | 97.97% | 0.0062 | 0.0758 | 0.01 |
| 6 | 240 | 00:00:38 | 100.00% | 98.20% | 0.0081 | 0.0752 | 0.01 |
| 7 | 250 | 00:00:40 | 100.00% | | 0.0012 | | 0.01 |
| 7 | 270 | 00:00:43 | 100.00% | 98.20% | 0.0040 | 0.0706 | 0.01 |
| 8 | 300 | 00:00:47 | 100.00% | 98.20% | 0.0006 | 0.0785 | 0.01 |
| 9 | 330 | 00:00:51 | 100.00% | 98.20% | 8.6674e-05 | 0.0796 | 0.01 |
| 9 | 350 | 00:00:54 | 100.00% | | 0.0010 | | 0.01 |
| 9 | 360 | 00:00:56 | 100.00% | 98.20% | 0.0002 | 0.0763 | 0.01 |
| 10 | 390 | 00:01:00 | 100.00% | 98.20% | 0.0011 | 0.0799 | 0.01 |
| 10 | 400 | 00:01:01 | 100.00% | | 0.0009 | | 0.01 |
| 11 | 420 | 00:01:04 | 100.00% | 97.97% | 0.0007 | 0.0843 | 0.01 |
| 11 | 450 | 00:01:08 | 100.00% | 98.20% | 0.0006 | 0.0832 | 0.01 |
| 12 | 480 | 00:01:13 | 100.00% | 98.20% | 0.0001 | 0.0815 | 0.01 |
| 13 | 500 | 00:01:15 | 100.00% | | 0.0013 | | 0.01 |
| 13 | 510 | 00:01:17 | 100.00% | 98.20% | 0.0005 | 0.0827 | 0.01 |
| 14 | 540 | 00:01:22 | 100.00% | 98.20% | 0.0008 | 0.0823 | 0.01 |
| 14 | 550 | 00:01:23 | 100.00% | | 0.0004 | | 0.01 |
| 14 | 570 | 00:01:27 | 100.00% | 98.20% | 0.0009 | 0.0847 | 0.01 |

```
|       15 |          600 |       00:01:32 |       100.00% |        98.20% |        0.0004 |        0.0815 |       0.0
|       15 |          615 |       00:01:35 |       100.00% |        98.20% |        0.0003 |        0.0869 |       0.0
|======================================================================================================================
Training finished: Max epochs completed.
```




```
% Plot the accuracy performance
batch_size = [8,16,32];
figure;
bar(batch_size, batchaccuracyMatrix);
xlabel('Batch Size');
ylabel('Accuracy');
ylim([0.9, 1]);
title('Accuracy Performance vs Batch Size');
grid on;

% Save the plot as an EPS file
eps_filename = 'results/batchsize.eps';
saveas(gcf, eps_filename, 'epsc');
fprintf('Saved %s\n', eps_filename);
```

```
% Plot the training time
figure;
bar(batch_size, batchTrainingTime);

xlabel('Batch Size');
ylabel('Training Time');
```

```
ylim([1, 500]);

title('Training Time vs Batch Size');
grid on;

% Save the plot as an EPS file
eps_filename = 'results/batch_size_trainingtime.eps'
saveas(gcf, eps_filename, 'epsc');
fprintf('Saved %s\n', eps_filename);
```

```
% Constants
numConvLayersList = [2, 3, 4]; % Vary the number of convolution layers
numConfigs = numel(numConvLayersList);
LayersaccuracyResults = zeros(numConfigs, 1);
LayersTrainingTime = zeros(numConfigs, 1);

% Loop through different configurations
for i = 1:numConfigs
    % Define the neural network architecture with varying convolution layers
    layers = [
        imageInputLayer([imageSize 1])
    ];

    for j = 1:numConvLayersList(i)
        layers = [layers
            convolution2dLayer(3, 16, 'Padding', 'same')
            batchNormalizationLayer
            reluLayer
            maxPooling2dLayer(2, 'Stride', 2)
        ];
    end

    layers = [layers
        fullyConnectedLayer(numel(categories))
        softmaxLayer
        classificationLayer
    ];

    % Set training options
    options = trainingOptions( ...
        'sgdm', ... % Stochastic Gradient Descent with Momentum
        'InitialLearnRate', 0.01, ...
        'MaxEpochs', 10, ...
        'Shuffle', 'every-epoch', ...
        'ValidationData', {testImages, testLabels}, ...
        'ValidationFrequency', 30, ...
        'Verbose', true, ...
        'Plots', 'training-progress' ...
        );
```

```
    trainingStartTime = tic;

    % Train the CNN
    net = trainNetwork(trainImages, trainLabels, layers, options);

    trainingTime = toc(trainingStartTime);

    % Classify Test Images
    predictedLabels = classify(net, testImages);

    % Calculate the Accuracy
    accuracy = sum(predictedLabels == testLabels) / numel(testLabels);

    % Store the accuracy for this configuration
    LayersaccuracyResults(i) = accuracy;
    LayersTrainingTime(i) = trainingTime;
end
```

```
% Plot the results
figure;
bar(numConvLayersList, LayersaccuracyResults)
xlabel('Number of Convolution Layers')
ylabel('Accuracy')
ylim([0.9, 1]);
title('Accuracy Performance vs Number of Convolution Layers')
grid on;

% Save the plot as an EPS file
eps_filename = 'results/layers.eps'
saveas(gcf, eps_filename, 'epsc');
fprintf('Saved %s\n', eps_filename);
```

```
% Plot the training time
figure;
bar(numConvLayersList, LayersTrainingTime);
xticks(learningRateValues);
xticklabels(learningRates);

xlabel('Number of Convolution Layers');
ylabel('Training Time');
ylim([1, 500]);

title('Training Time vs Number of Convolution Layers');
grid on;

% Save the plot as an EPS file
eps_filename = 'results/layers_trainingtime.eps'
```

```
saveas(gcf, eps_filename, 'epsc');
fprintf('Saved %s\n', eps_filename);
```