```
% MaTLab
% The "Chest X-Ray Images (Pneumonia)" Dataset
% By Meetra Nouri
```

```
% The "Chest X-Ray Images (Pneumonia)" dataset is a collection of chest X-
ray images that have been labeled as
% normal or containing pneumonia, along with a file containing the image
labels.
% The dataset is labeled by experienced radiologists from the NIH.
```

```
% Define the path to the image directory
dataDir = '/MATLAB Drive/Chest X-Ray Images (Pneumonia)';

% Create an imageDatastore for the entire dataset
imds = imageDatastore(dataDir, "IncludeSubfolders", true, 'LabelSource',
'foldernames');

% Define the number of classes
num_classes = 2;

% Define the size of each set
num_train = 320;   % number of samples in the training set
num_val = 32;      % number of samples in the validation set
num_test = 32;     % number of samples in the test set

% Shuffle the data
imds = shuffle(imds);

% Split the data into the train, validation, and test sets
[imdsTrain, imdsVal, imdsTest] = splitEachLabel(imds, num_train, num_val,
num_test, 'randomized');

% Get the unique labels from the image datastore
labels = unique(imds.Labels);

% Set the image augmentation options
augmenter = imageDataAugmenter( ...
    'RandXReflection', true, ...
    'RandRotation', [-20 20], ...
    'RandXTranslation', [-10 10], ...
    'RandYTranslation', [-10 10]);

% Create augmented image datastores for the train, val, and test sets
augimdsTrain = augmentedImageDatastore([224, 224], imdsTrain, ...
    'DataAugmentation', augmenter, ...
    'ColorPreprocessing', 'gray2rgb');
augimdsVal = augmentedImageDatastore([224, 224], imdsVal, ...
    'DataAugmentation', augmenter, ...
    'ColorPreprocessing', 'gray2rgb');
augimdsTest = augmentedImageDatastore([224, 224], imdsTest, ...
    'DataAugmentation', augmenter, ...
    'ColorPreprocessing', 'gray2rgb');

% Load the pre-trained VGG-16 model
net = vgg16;
layers = net.Layers;

% Modify the network for transfer learning
layers(end-2) = fullyConnectedLayer(num_classes);
layers(end) = classificationLayer;

% Set the training options
options = trainingOptions('sgdm', ...
    'MiniBatchSize', 32, ...
    'MaxEpochs', 10, ...
    'InitialLearnRate', 0.001, ...
    'Verbose', true, ...
    'Plots', 'training-progress', ...
    'ValidationData', augimdsVal);
```

```
% Train the network
net = trainNetwork(augimdsTrain, layers, options);
```

```
Training on single CPU.
Initializing input data normalization.
|====================================================================================================================
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Ba
|         |             |   (hh:mm:ss)   |   Accuracy   |   Accuracy   |     Loss     |     Loss     |
|====================================================================================================================
|       1 |           1 |       00:00:17 |       43.75% |       50.00% |       2.2578 |       3.7333 |
|       3 |          50 |       00:07:15 |       68.75% |       93.75% |       0.6759 |       0.1337 |
|       5 |         100 |       00:14:40 |      100.00% |       96.88% |       0.0386 |       0.0502 |
|       8 |         150 |       00:21:41 |       96.88% |       95.31% |       0.1982 |       0.1040 |
|      10 |         200 |       00:28:55 |       96.88% |       98.44% |       0.0528 |       0.0720 |
|====================================================================================================================
Training finished: Max epochs completed.
```

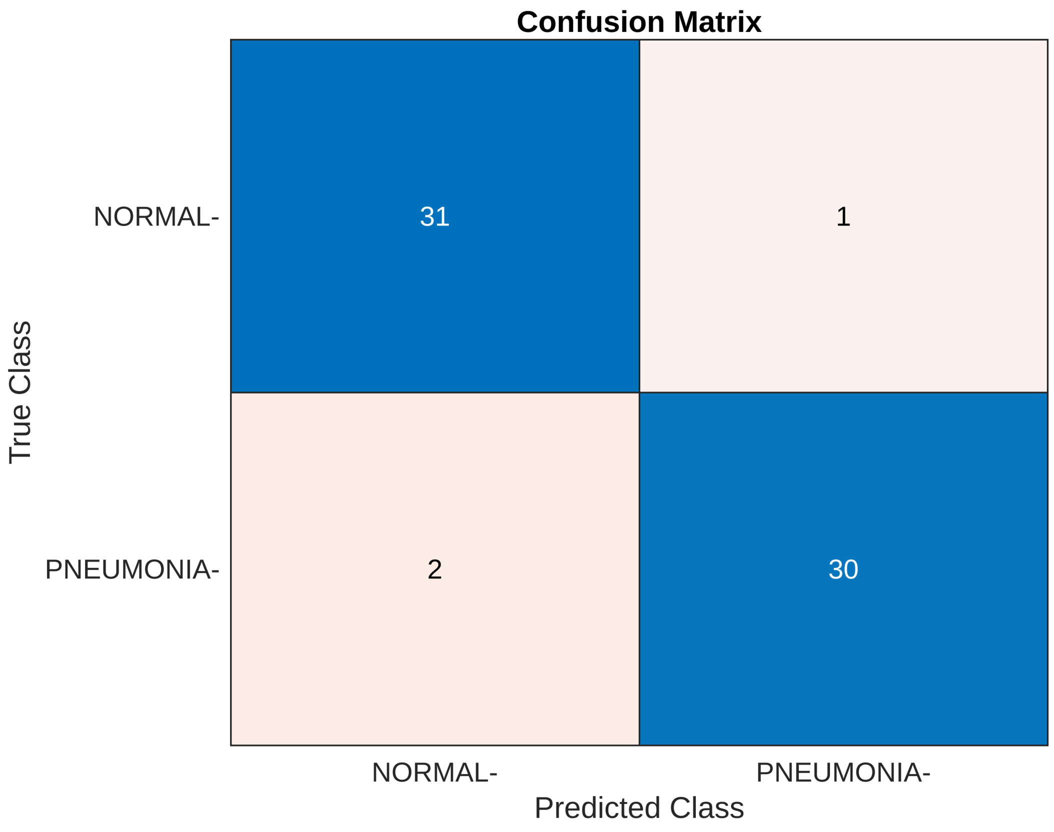
```
% Evaluate the model on the test set
predicted_labels = classify(net, augimdsTest);
test_labels = imdsTest.Labels;
```

```
% Get the unique labels from the test set
test_labels_unique = unique(test_labels);

% Add the missing label(s) to the labels variable
for i = 1:numel(test_labels_unique)
    if ~ismember(test_labels_unique(i), labels)
        labels = [labels; test_labels_unique(i)];
    end
end

% Calculate the confusion matrix
cm = confusionmat(test_labels, predicted_labels, 'Order', labels);

% Visualize the confusion matrix
figure
confusionchart(cm, labels)
title('Confusion Matrix')
```



```
% Calculate the classification accuracy
accuracy = sum(predicted_labels == test_labels) / numel(test_labels);

% Print the classification accuracy
fprintf('Test Accuracy: %.2f%%\n', accuracy*100);
```

```
Test Accuracy: 98.44%
```