```
% Load USPS dataset
load("usps_all.mat");

% Select the classes corresponding to the last three digits of your UMD ID
class_labels = [9, 3, 8];  % Corresponding to the last three digits of your UMD ID

% Initialize variables to store training and testing data
train_data = [];
test_data = [];
train_labels = [];
test_labels = [];

% Reshape each image to 16x16
data = reshape(data, 16, 16, 1100, 10);

% Iterate over selected classes
for i = 1:numel(class_labels)
    % Select instances for the current class
    class_idx = class_labels(i) + 1; % Adjust for 0-based indexing
    class_data = data(:, :, :, class_idx);

    % Split the instances into training and testing data
    train_data = cat(3, train_data, class_data(:, :, 1:1000));
    train_labels = [train_labels; repmat(class_labels(i), 1000, 1)];

    test_data = cat(3, test_data, class_data(:, :, 1001:1100));
    test_labels = [test_labels; repmat(class_labels(i), 100, 1)];
end

% Reshape data for SVM training
train_data = reshape(train_data, size(train_data, 1) * size(train_data, 2), size(train_data, 3))';
test_data = reshape(test_data, size(test_data, 1) * size(test_data, 2), size(test_data, 3))';

% Normalize pixel values
train_data = double(train_data) ./ 255;
test_data = double(test_data) ./ 255;

% Train binary classifiers for each class using one-vs-all approach
svmModels = cell(1, numel(class_labels));
for i = 1:numel(class_labels)
    % Set labels for the current class as 1 and others as -1
    binary_train_labels = ones(size(train_labels));
    binary_train_labels(train_labels ~= class_labels(i)) = -1;

    % Train binary SVM classifier
    svmModels{i} = fitcsvm(train_data, binary_train_labels, 'KernelFunction', 'linear');
end

% Predict labels for test data
predictedLabels_remaining = zeros(size(test_labels));
for i = 1:numel(class_labels)
    % Predict using the binary SVM classifier for the current class
    predictedLabels_remaining(predict(svmModels{i}, test_data) == 1) = class_labels(i);
end

% Calculate global accuracy
global_accuracy = sum(predictedLabels_remaining == test_labels) / numel(test_labels);

% Calculate local accuracy (accuracy by label)
local_accuracy = zeros(1, numel(class_labels));
for i = 1:numel(class_labels)
    % Extract true labels for the current class
    true_labels_class = test_labels(test_labels == class_labels(i));

    % Calculate accuracy for the current class
    local_accuracy(i) = sum(predictedLabels_remaining(test_labels == class_labels(i)) == true_labels_class) / numel(true_labels_class);
end

% Display global accuracy
disp(['Global Accuracy: ', num2str(global_accuracy)]);

% Display local accuracy
disp('Local Accuracy (Accuracy by Label):');
for i = 1:numel(class_labels)
```

```
    disp(['Class ', num2str(class_labels(i)), ' Accuracy: ', num2str(local_accuracy(i))]);
end
```

```
Global Accuracy: 0.95667
Local Accuracy (Accuracy by Label):
Class 9 Accuracy: 0.97
Class 3 Accuracy: 0.93
Class 8 Accuracy: 0.97
```

Published with MATLAB® R2023b