```
% Features: n × m → m × n for deep learning input
XTrain = X_train_raw';
XVal = X_val';
XTest = X_test';

% Labels: Convert to categorical
YTrain = categorical(Y_train_raw);
YVal = categorical(Y_val);
YTest = categorical(Y_test);
```

```
numFeatures = size(XTrain, 1);      % number of input features
numClasses = numel(unique(YTrain)); % number of heartbeat classes

layers = [
    featureInputLayer(numFeatures)
    fullyConnectedLayer(64)
    reluLayer
    fullyConnectedLayer(32)
    reluLayer
    fullyConnectedLayer(numClasses)
    softmaxLayer
    classificationLayer];

options = trainingOptions('adam', ...
    'MaxEpochs', 30, ...
    'MiniBatchSize', 64, ...
    'Shuffle', 'every-epoch', ...
    'ValidationData', {XVal', YVal}, ...
    'ValidationFrequency', 30, ...
    'ValidationPatience',5,...
    'Verbose', false, ...
    'Plots', 'training-progress');
```

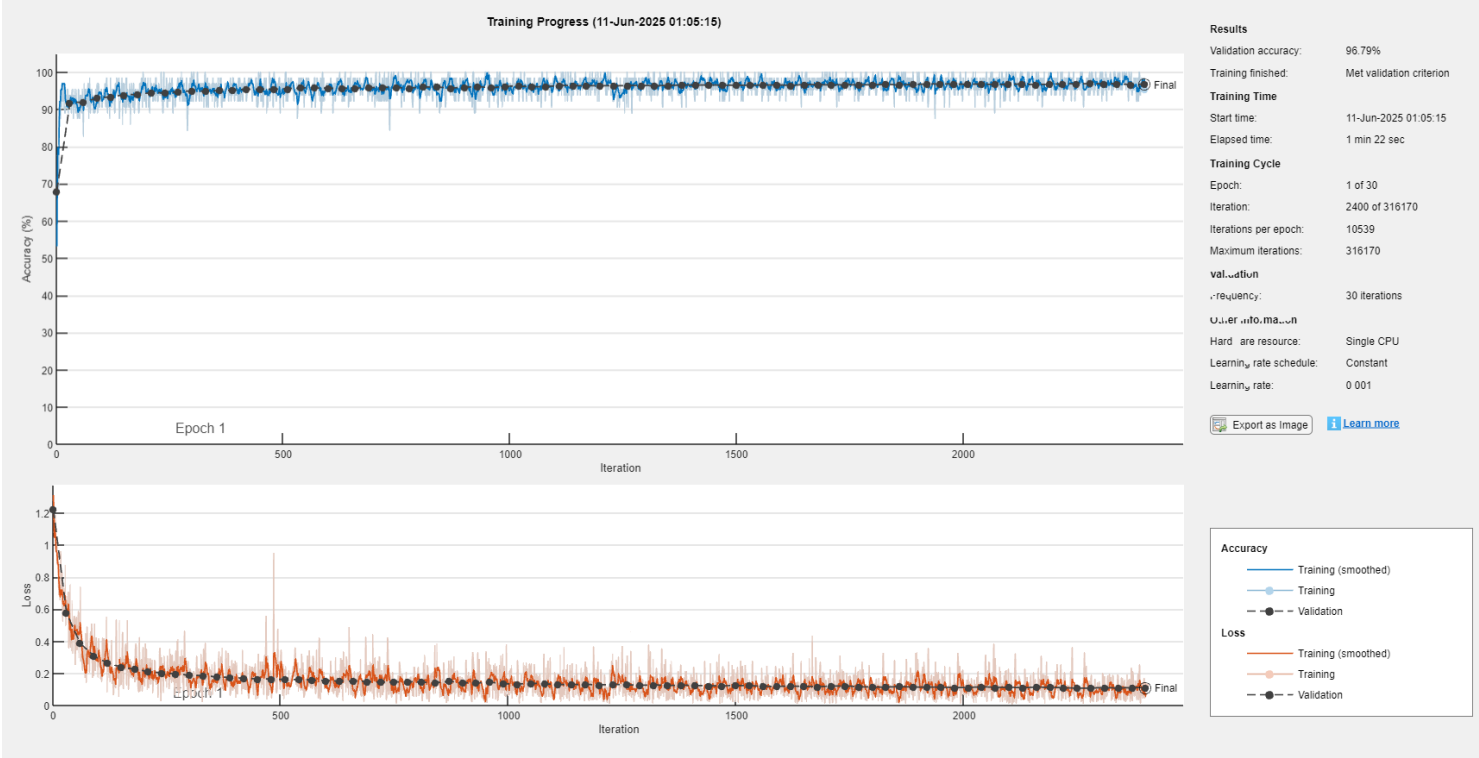
```
disp ("Train Labels:"); disp(unique(YTrain));
```

```
Train Labels:
     F
     N
     Q
     SVEB
     VEB
```

```
disp (["XTrain Size:"], num2str(size(XTrain)));
```

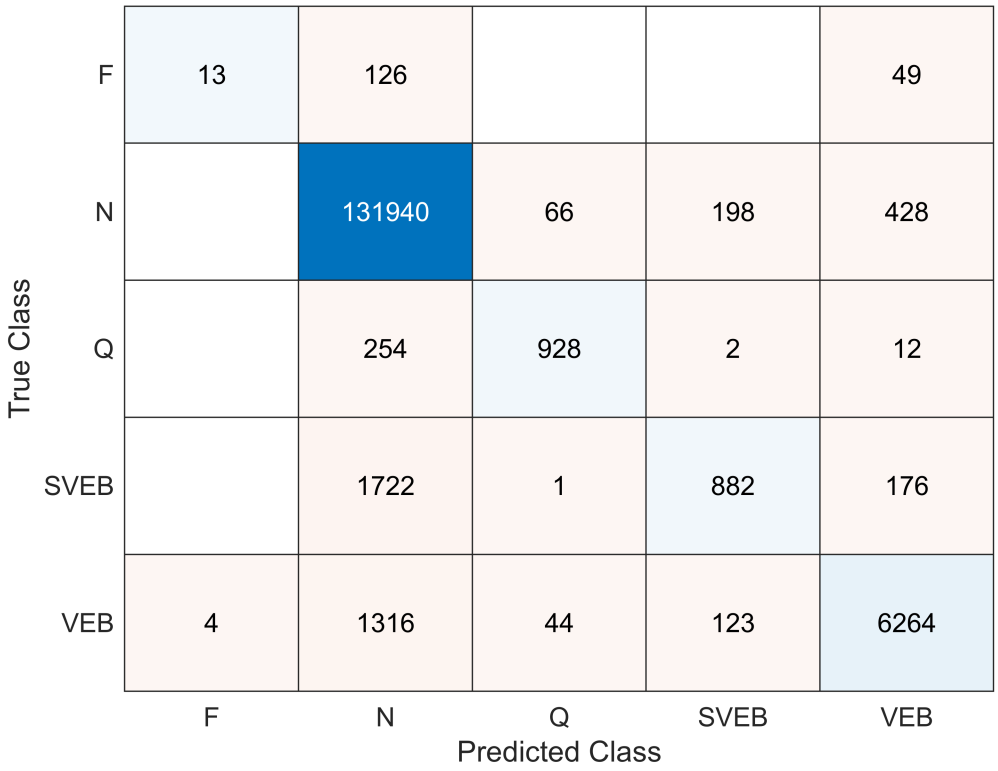
```
XTrain Size:
```

```
net = trainNetwork(XTrain', YTrain, layers, options);
```




```
% Predict labels
YPred = classify(net, XTest');

% Confusion matrix and accuracy
confusionchart(YTest, YPred);
```




```
accuracy = sum(YPred == YTest) / numel(YTest);
fprintf('Test Accuracy: %.2f%%\n', accuracy * 100);
```

```
Test Accuracy: 96.87%
```

```
% Assume YTest and YPred are categorical arrays
classes = categories(YTest);
numClasses = numel(classes);

% Get confusion matrix
confMat = confusionmat(YTest, YPred);

% Preallocate
precision = zeros(numClasses, 1);
recall = zeros(numClasses, 1);
f1_score = zeros(numClasses, 1);
support = sum(confMat, 2);  % True instances per class

% Compute per-class metrics
for i = 1:numClasses
    TP = confMat(i, i);  % True Positives
    FP = sum(confMat(:, i)) - TP;  % False Positives
    FN = sum(confMat(i, :)) - TP;  % False Negatives

    precision(i) = TP / (TP + FP + eps);
```

```
    recall(i) = TP / (TP + FN + eps);
    f1_score(i) = 2 * (precision(i) * recall(i)) / (precision(i) + recall(i) + eps);
end

% Display as table
metrics_table = table(classes, precision, recall, f1_score, support, ...
    'VariableNames', {'Class', 'Precision', 'Recall', 'F1_Score', 'Support'});

disp(metrics_table)
```

Class	Precision	Recall	F1_Score	Support
{'F'}	0.76471	0.069149	0.12683	188
{'N'}	0.97475	0.99478	0.98466	1.3263e+05
{'Q'}	0.89317	0.77592	0.83043	1196
{'SVEB'}	0.73195	0.31715	0.44255	2781
{'VEB'}	0.90403	0.80815	0.85341	7751