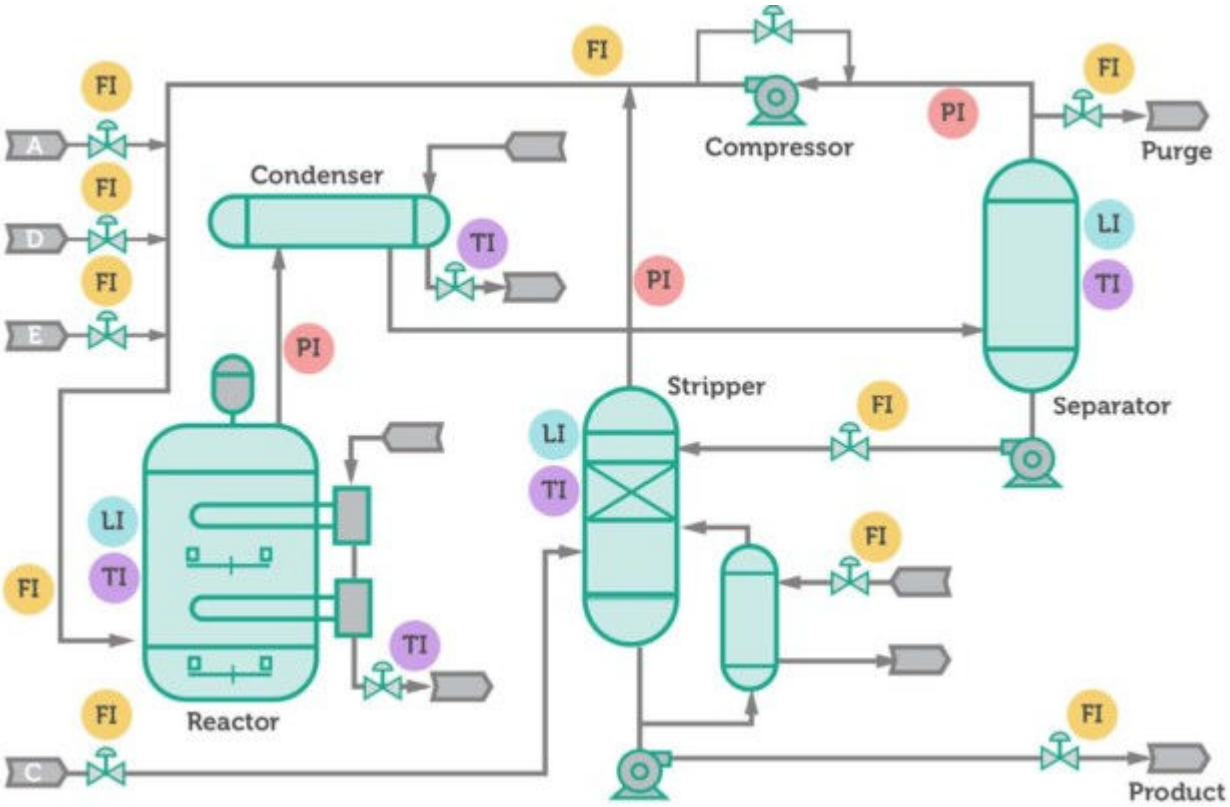
Chemical Process Fault Detection [using LSTM]

- Anomaly detection in chemical process engineering
- Data consisting **fault free** (normal operational conditions) sensor's data and **faulty data** (20 different process faults) in simulated real-world industrial chemical plant



Key Variables in the TEP Dataset and Their Importance :

1. Measured Variables (xmeas_1 to xmeas_41)

These variables represent real-time sensor measurements from the plant, including:

- **Reactor temperature & pressure** (xmeas_1, xmeas_2)
- **Separator level & pressure** (xmeas_9, xmeas_10)
- **Flow rates of reactants and products** (xmeas_15 to xmeas_20)
- **Composition of process streams** (mole fractions of A, B, C, etc.)

2. Manipulated Variables (xmv_1 to xmv_11)

These are control parameters that operators can adjust, such as:

- **Cooling water flow rate** (xmv_2)
- **Agitator speed** (xmv_7)
- **Valve positions for feed streams** (xmv_5, xmv_6)

3. Fault Labels

The dataset includes **20 fault types**, each representing a specific anomaly:

- **Step changes in feed composition (Fault 1-3)**
- **Random variations in reactor cooling water (Fault 7)**
- **Valve sticking (Fault 10)**
- **Unknown disturbances (Fault 15-20)**

source: The Tennessee Eastman Process: An Open-Source Benchmark for Anomaly Detection in Process Engineering

MATLAB tutorial: Chemical Process Fault Detection Using Deep Learning - MATLAB & Simulink

Helper functions for preprocessing and normalizing data later on:

```
function processed = helperPreprocess(mydata,limit)
    H = size(mydata,1);
    processed = {};
    for ind = 1:limit:H
        x = mydata(ind:(ind+(limit-1)),4:end);
        processed = [processed; x]; %#ok<AGROW>
    end
end
```

```
function data = helperNormalize(data,m,s)
    for ind = 1:size(data,1)
        data{ind} = (data{ind} - m)./s;
    end
end
```

Get data

original source: Additional Tennessee Eastman Process Simulation Data for Anomaly Detection Evaluation - Harvard Dataverse

```
url = 'https://www.mathworks.com/supportfiles/predmaint/chemical-process-fault-
detection-data/faultytesting.mat';
websave('data\faultytesting.mat',url);
url = 'https://www.mathworks.com/supportfiles/predmaint/chemical-process-fault-
detection-data/faultytraining.mat';
websave('data\faultytraining.mat',url);
url = 'https://www.mathworks.com/supportfiles/predmaint/chemical-process-fault-
detection-data/faultfreetesting.mat';
websave('data\faultfreetesting.mat',url);
url = 'https://www.mathworks.com/supportfiles/predmaint/chemical-process-fault-
detection-data/faultfreetraining.mat';
websave('data\faultfreetraining.mat',url);
```

```
load('data\faultfreetesting.mat');
```

```
load('data\faultfreetraining.mat');
load('data\faultytesting.mat');
load('data\faultytraining.mat');
```

```
head(faultytraining, 4)
```

faultNumber	simulationRun	sample	xmeas_1	xmeas_2	xmeas_3	xmeas_4	xmeas_5	xmeas_6	xmea
1	1	1	0.25038	3674	4529	9.232	26.889	42.402	2704
1	1	2	0.25109	3659.4	4556.6	9.4264	26.721	42.576	27
1	1	3	0.25038	3660.3	4477.8	9.4426	26.875	42.07	2706
1	1	4	0.24977	3661.3	4512.1	9.4776	26.758	42.063	2707

```
tail(faultytraining, 4)
```

faultNumber	simulationRun	sample	xmeas_1	xmeas_2	xmeas_3	xmeas_4	xmeas_5	xmeas_6	xmea
20	500	497	0.26704	3647.4	4540.2	9.3546	27.034	42.671	2704
20	500	498	0.26543	3630.3	4571.6	9.4089	27.129	42.47	2705
20	500	499	0.27671	3655.7	4498.9	9.3781	27.353	42.281	2705
20	500	500	0.27421	3640.4	4474.4	9.3866	27.145	41.985	27

- *faultNumber* - fault type, **0** is fault free, and nums **1-20** are different fault types
- *simulationRun* - number of times simulation ran to obtain data (nums **1-500**)
- *sample* - number of times TEP variables were recorded per simulation, 1 to 500 for the training data sets and from 1 to 960 for the testing data sets
- *cols 4-44* = **measured** variables of TEP
- *45-55* = **manipulated** variables of TEP

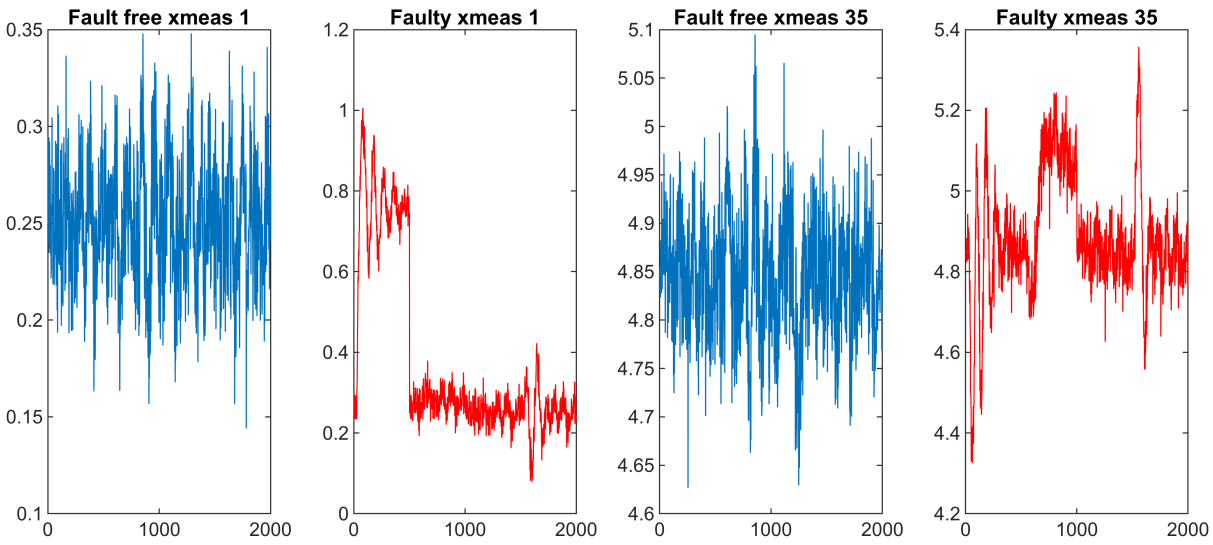
```
% remove vars with fault nums 3,9,15 - theyre not valid
faultytesting(faultytesting.faultNumber == 3,:) = [];
faultytesting(faultytesting.faultNumber == 9,:) = [];
faultytesting(faultytesting.faultNumber == 15,:) = [];

faultytraining(faultytraining.faultNumber == 3,:) = [];
faultytraining(faultytraining.faultNumber == 9,:) = [];
faultytraining(faultytraining.faultNumber == 15,:) = [];
```

Exploration

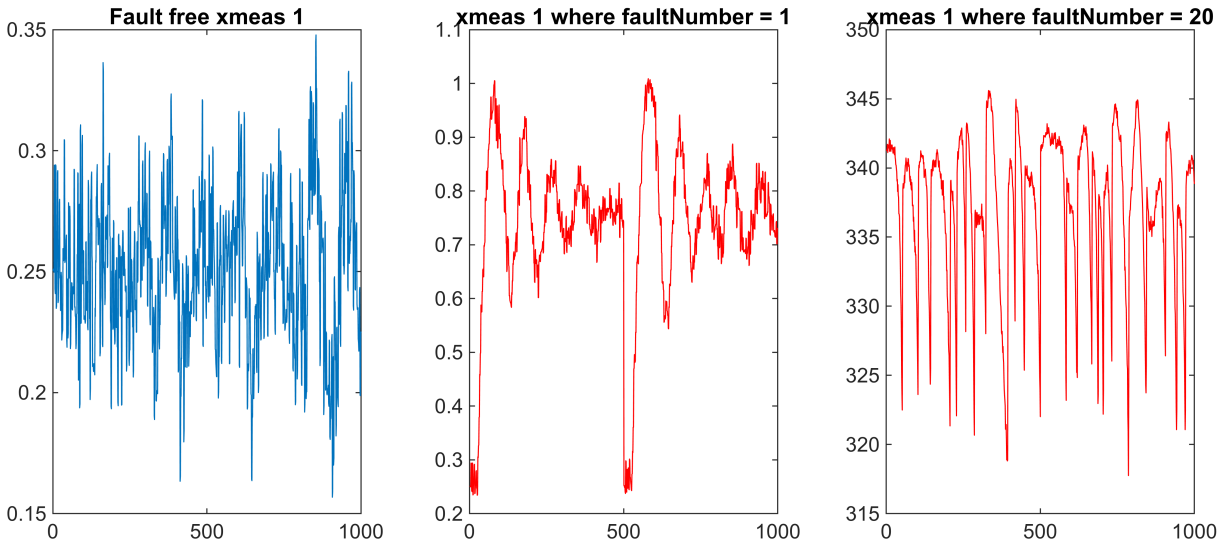
```
f = figure;
f.Position = [100 100 1000 400];
subplot(1,4,1);
plot(faultfreetraining.xmeas_1(1:2000));
title('Fault free xmeas 1');
subplot(1,4,2);
plot(faultytraining.xmeas_1(1:2000), 'r');
title('Faulty xmeas 1');
subplot(1,4,3);
```

```
plot(faultfreetraining.xmeas_35(1:2000));
title('Fault free xmeas 35');
subplot(1,4,4);
plot(faultytraining.xmeas_35(1:2000), 'r');
title('Faulty xmeas 35');
```




```
xmeas1_fault_1 = faultytraining(faultytraining.faultNumber == 1, :).xmeas_1;
xmeas1_fault_20 = faultytraining(faultytraining.faultNumber == 20, :).xmeas_20;
```

```
f = figure;
f.Position = [100 100 1000 400];
subplot(1,3,1);
plot(faultfreetraining.xmeas_1(1:1000));
title('Fault free xmeas 1');
subplot(1,3,2);
plot(xmeas1_fault_1(1:1000), 'r');
title('xmeas 1 where faultNumber = 1');
subplot(1,3,3);
plot(xmeas1_fault_20(1:1000), 'r');
title('xmeas 1 where faultNumber = 20');
```



Split into train/test/val sets (80/10/10)

```
H1 = height(faultfreetraining);
H2 = height(faultytraining);
msTrain = max(faultfreetraining.simulationRun)
```

```
msTrain =
500
```

```
msTest = max(faultytesting.simulationRun)
```

```
msTest =
500
```

```
rTrain = 0.80;
msVal = ceil(msTrain*(1 - rTrain))
```

```
msVal =
100
```

```
msTrain = msTrain*rTrain
```

```
msTrain =
400
```

```
sampleTrain = max(faultfreetraining.sample)
```

```
sampleTrain =
500
```

```
sampleTest = max(faultfreetesting.sample)
```

```
sampleTest =
960
```

```
rowLim1 = ceil(rTrain*H1);
rowLim2 = ceil(rTrain*H2);

trainingData = [faultfreetraining{1:rowLim1,:}; faultytraining{1:rowLim2,:}];
validationData = [faultfreetraining{rowLim1 + 1:end,:}; faultytraining{rowLim2 +
1:end,:}];
testingData = [faultfreetesting{:,:}; faultytesting{:,:}]
```

```
testingData = 8640000×55
10^3 ×
         0    0.0010    0.0010    0.0003    3.6724    4.4663    0.0095    0.0271 ···
         0    0.0010    0.0020    0.0003    3.6422    4.5687    0.0094    0.0270
         0    0.0010    0.0030    0.0002    3.6431    4.5075    0.0093    0.0269
         0    0.0010    0.0040    0.0003    3.6283    4.5193    0.0093    0.0270
         0    0.0010    0.0050    0.0002    3.6558    4.5710    0.0093    0.0269
         0    0.0010    0.0060    0.0002    3.6468    4.4782    0.0094    0.0269
         0    0.0010    0.0070    0.0002    3.7096    4.5144    0.0092    0.0273
         0    0.0010    0.0080    0.0002    3.6907    4.5317    0.0094    0.0266
         0    0.0010    0.0090    0.0002    3.6758    4.4840    0.0092    0.0268
         0    0.0010    0.0100    0.0002    3.6680    4.5273    0.0094    0.0269
     ⋮
```

Preprocess and normalize data

```
Xtrain = helperPreprocess(trainingData,sampleTrain);
size(Xtrain)
```

```
ans = 1×2
        7200           1
```

```
Ytrain = categorical([zeros(msTrain,1);repmat([1,2,4:8,10:14,16:20],1,msTrain)']);
XVal = helperPreprocess(validationData,sampleTrain);
size(XVal)
```

```
ans = 1×2
        1800           1
```

```
YVal = categorical([zeros(msVal,1);repmat([1,2,4:8,10:14,16:20],1,msVal)']);

Xtest = helperPreprocess(testingData,sampleTest);
size(Xtest)
```

```
ans = 1×2
        9000           1
```

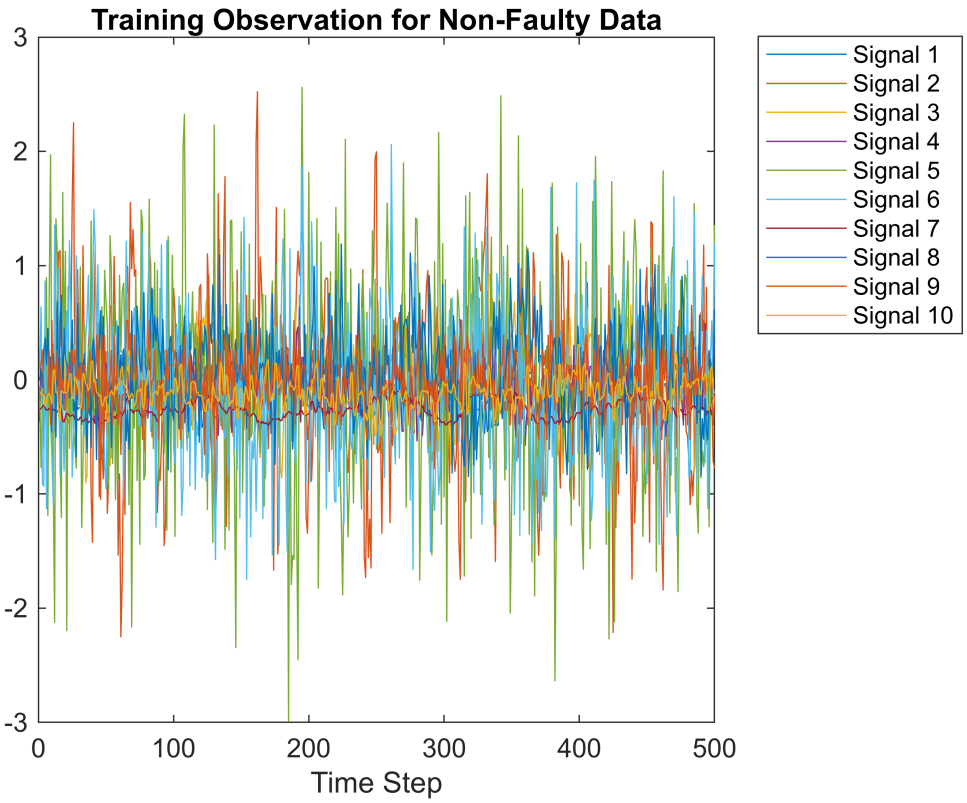
```
Ytest = categorical([zeros(msTest,1);repmat([1,2,4:8,10:14,16:20],1,msTest)']);
```

```
tMean = mean(trainingData(:,4:end));
tSigma = std(trainingData(:,4:end));
```

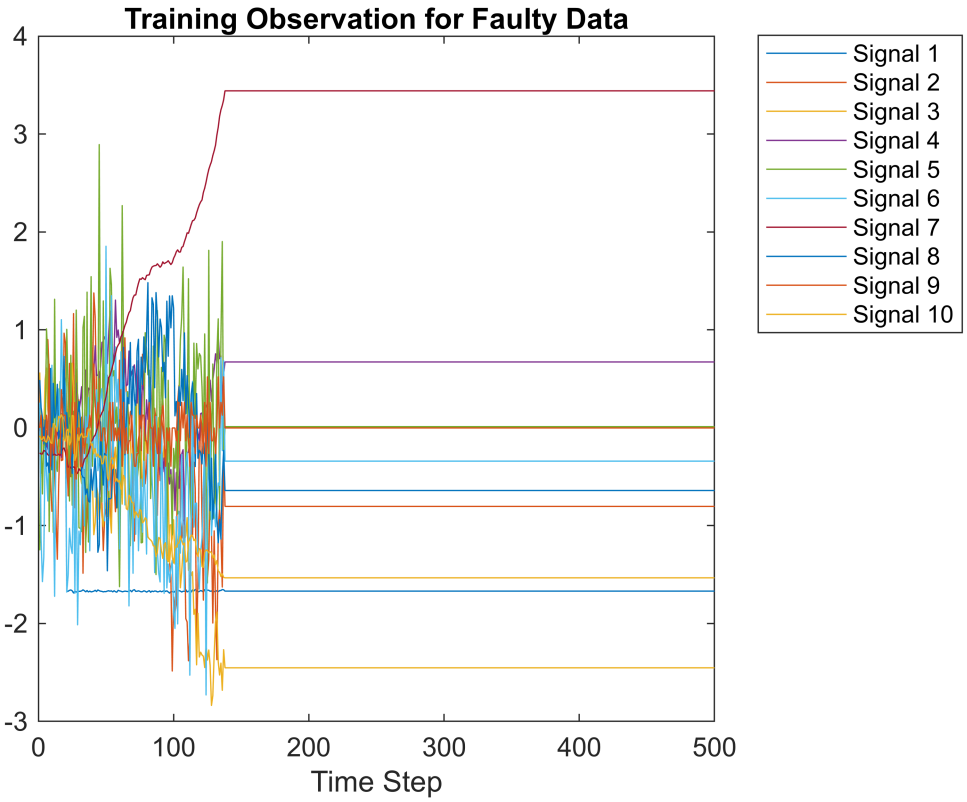
```
Xtrain = helperNormalize(Xtrain, tMean, tSigma);
XVal = helperNormalize(XVal, tMean, tSigma);
```

```
Xtest = helperNormalize(Xtest, tMean, tSigma);
```

```
figure;
splot = 10;
plot(Xtrain{1}(:,1:10));
xlabel("Time Step");
title("Training Observation for Non-Faulty Data");
legend("Signal " + string(1:splot),'Location','northeastoutside');
```



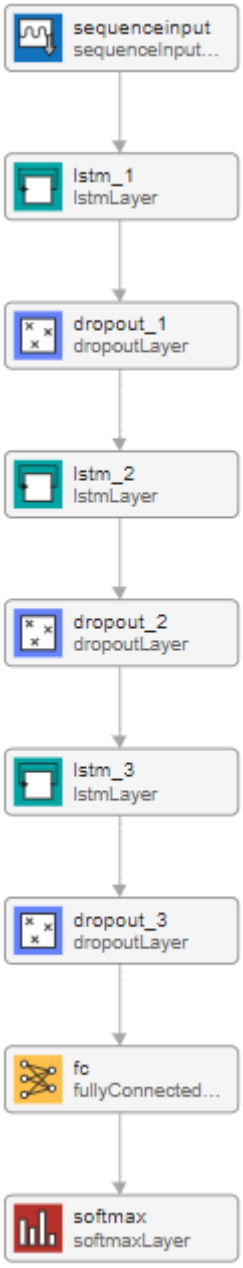

```
figure;
plot(Xtrain{1000}(:,1:10));
xlabel("Time Step");
title("Training Observation for Faulty Data");
legend("Signal " + string(1:splot),'Location','northeastoutside');
```



Define model's structure and training options

```
numSignals = 52;
numHiddenUnits2 = 52;
numHiddenUnits3 = 40;
numHiddenUnits4 = 25;
numClasses = 18; % fault free class and all different possible process faults

layers = [ ...
    sequenceInputLayer(numSignals)
    lstmLayer(numHiddenUnits2,'OutputMode','sequence')
    dropoutLayer(0.2)
    lstmLayer(numHiddenUnits3,'OutputMode','sequence')
    dropoutLayer(0.2)
    lstmLayer(numHiddenUnits4,'OutputMode','last')
    dropoutLayer(0.2)
    fullyConnectedLayer(numClasses)
    softmaxLayer];
```



LAYER INFORMATION

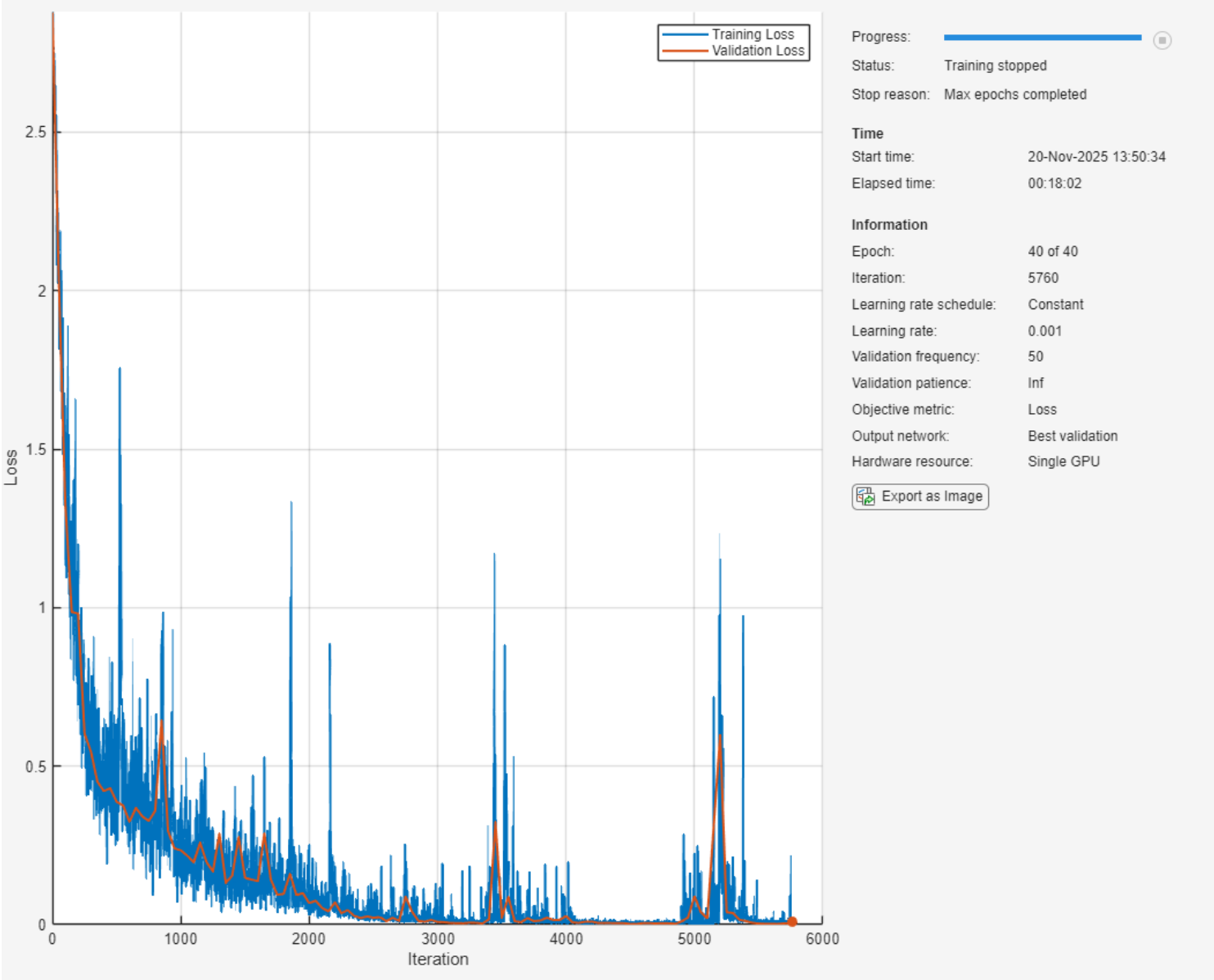
	Name	Type	Activations	Learnable Sizes	State Sizes
1	sequenceinput Sequence input with 52 dimensions	Sequence Input	52(C) × 1(B) × 1(T)	-	-
2	lstm_1 LSTM with 52 hidden units	LSTM	52(C) × 1(B) × 1(T)	InputWeigh… 208 … RecurrentW… 208 … Bias 208 …	HiddenState 52 × 1 CellState 52 × 1
3	dropout_1 20% dropout	Dropout	52(C) × 1(B) × 1(T)	-	-
4	lstm_2 LSTM with 40 hidden units	LSTM	40(C) × 1(B) × 1(T)	InputWeigh… 160 … RecurrentW… 160 … Bias 160 …	HiddenState 40 × 1 CellState 40 × 1
5	dropout_2 20% dropout	Dropout	40(C) × 1(B) × 1(T)	-	-
6	lstm_3 LSTM with 25 hidden units	LSTM	25(C) × 1(B)	InputWeigh… 100 … RecurrentW… 100 … Bias 100 …	HiddenState 25 × 1 CellState 25 × 1
7	dropout_3 20% dropout	Dropout	25(C) × 1(B)	-	-
8	fc 18 fully connected layer	Fully Connected	18(C) × 1(B)	Weights 18 × 25 Bias 18 × 1	-
9	softmax softmax	Softmax	18(C) × 1(B)	-	-

```
maxEpochs = 40;
miniBatchSize = 50;

options = trainingOptions('adam', ...
    'ExecutionEnvironment','auto', ...
    'GradientThreshold',1, ...
    'MaxEpochs',maxEpochs, ...
    'MiniBatchSize', miniBatchSize,...
    'Shuffle','every-epoch', ...
    'Verbose',0, ...
    'Plots','training-progress',...
    'ValidationData',{XVal,YVal});
```

Train and evaluate network

```
net = trainnet(Xtrain,Ytrain,layers,"crossentropy",options);
```

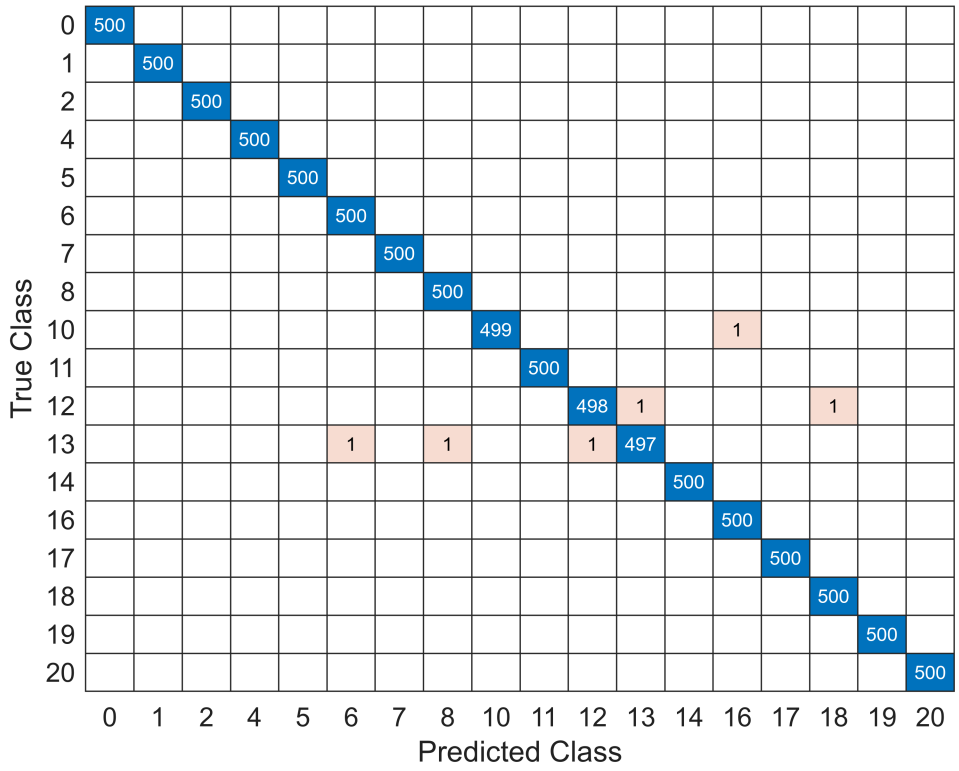



```
scores = minibatchpredict(net,Xtest);
Ypred = scores2label(scores,unique(Ytrain));
```

```
acc = sum(Ypred == Ytest)./numel(Ypred)
```

```
acc =
0.9993
```

```
confusionchart(Ytest,Ypred)
```


True Class
Predicted Class
0 1 2 4 5 6 7 8 10 11 12 13 14 16 17 18 19 20
500
500
500
500
500
500
500
500
499
1
500
498
1
1
1
1
497
500
500
500
500
500