In[733]:=
```
ABMInputs1600 = Import[
   "/Users/thorsilver/Downloads/ABM outputs1/LPtau1600runs_GEMSA_inputs.csv"];

ABMOutputs1600 = Import[
   "/Users/thorsilver/Downloads/ABM outputs1/LPtau1600runs GEMSA outputs.csv"];
```

In[735]:=
```
ABMOutputs1600 = Function[x, x / 1000] /@ ABMOutputs1600;
ABMAssoc1600 = AssociationThread[ABMInputs1600 → Flatten[ABMOutputs1600]];
ABMnewData1600 = Dataset[ABMAssoc1600];
ABMNormal1600 = Normal[ABMAssoc1600];
ABMNormalRandom = RandomSample[ABMNormal1600];
ABMtrain1600 = TakeDrop[ABMNormal1600, 1280];
ABMtest1600 = ABMtrain1600[[2]];
ABMtraining1600 = ABMtrain1600[[1]];
trainDevSplit1600 = TakeDrop[ABMtraining1600, 1024];
finalTrain1600 = trainDevSplit1600[[1]];
finalDev1600 = trainDevSplit1600[[2]];
finaltest1600 = ABMtest1600;
```

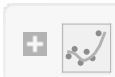
In[747]:=
```
Length[finalDev1600]
Length[finalTrain1600]
Length[finaltest1600]
```

Out[747]= 256

Out[748]= 1024

Out[749]= 320

In[750]:=
```
pRFv1600 = Predict[finalTrain1600, Method → "RandomForest"]
```

Out[750]= PredictorFunction[ Input type: Mixed (number: 10)
Method: RandomForest]

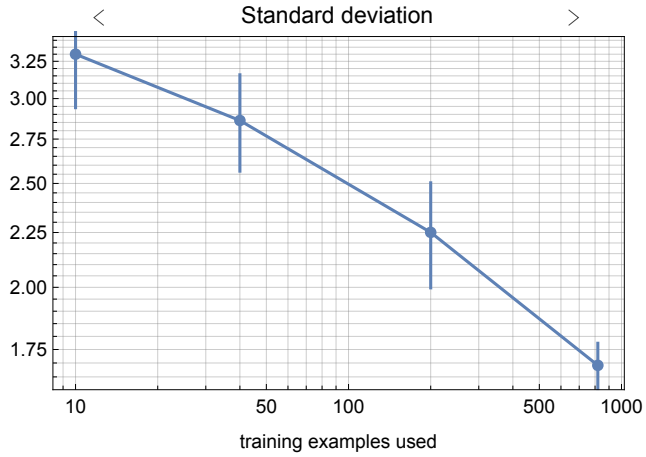
In[751]:=
```
pmRFv1600 = PredictorMeasurements[pRFv1600, finaltest1600]
PredictorInformation[pRFv1600]
pmRFv1600["ComparisonPlot"]
pmRFv1600["ResidualPlot"]
```

Out[751]= PredictorMeasurementsObject[ Predictor: RandomForest
Number of test examples: 320]

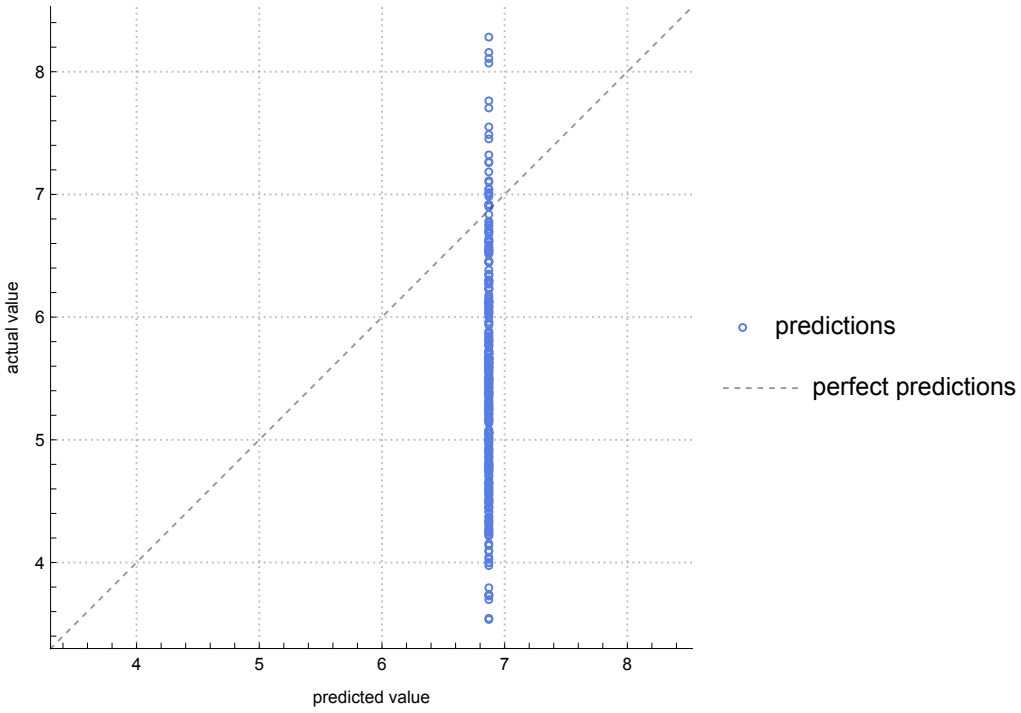
Predictor information

Data type	Mixed (number: 10)
Standard deviation	1.69 ± 0.082
Method	RandomForest
Single evaluation time	3.47 ms/example
Batch evaluation speed	44. examples/ms
Loss	1.88 ± 0.042
Model memory	293. kB
Training examples used	1024 examples
Training time	666. ms

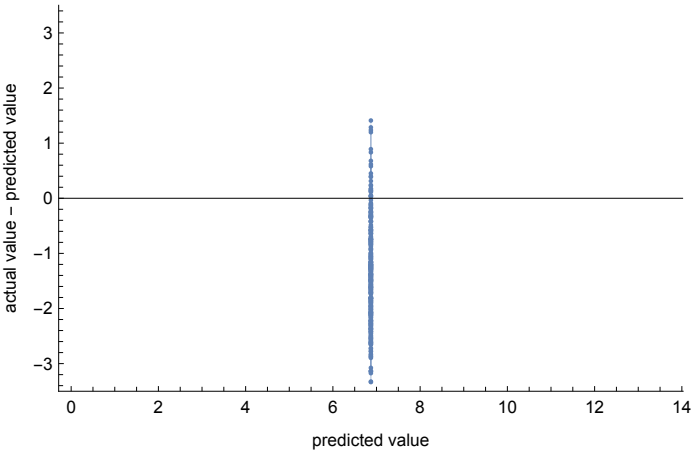
Out[752]=



Out[753]=



Out[754]=



In[755]:=
```
pXGTv1600 = Predict[finalTrain1600, Method -> "GradientBoostedTrees"]
pmXGTv1600 = PredictorMeasurements[pXGTv1600, finaltest1600]
PredictorInformation[pXGTv1600]
pmXGTv1600["ComparisonPlot"]
pmXGTv1600["ResidualPlot"]
```

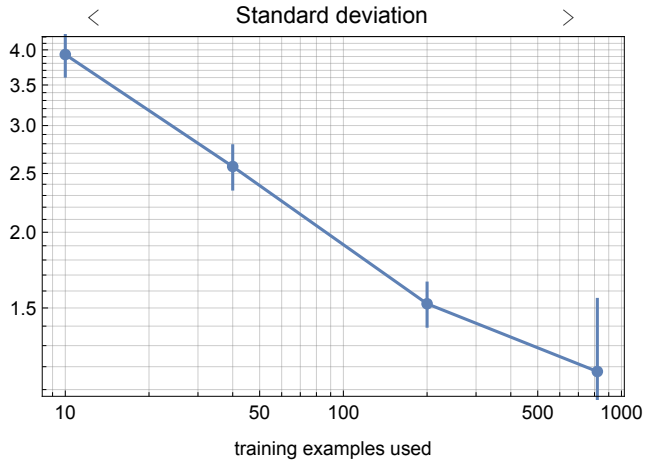
Out[755]= PredictorFunction[Input type: Mixed (number: 10) Method: GradientBoostedTrees]

Out[756]= PredictorMeasurementsObject[Predictor: GradientBoostedTrees Number of test examples: 320]

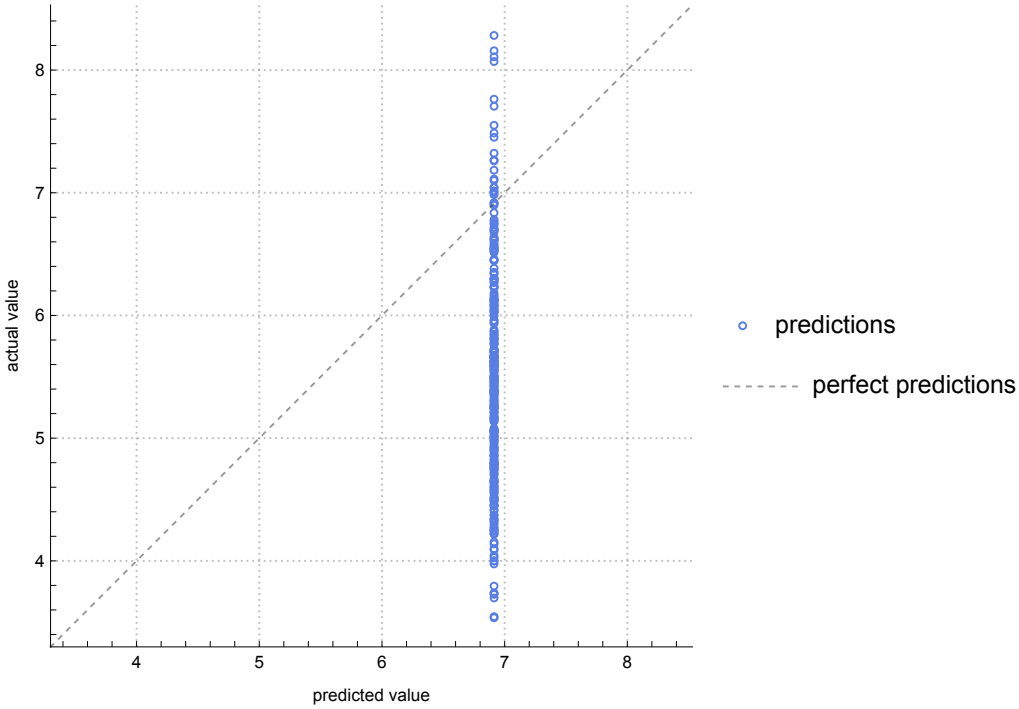
Predictor information

Data type	Mixed (number: 10)
Standard deviation	1.18 ± 0.37
Method	GradientBoostedTrees
Single evaluation time	3.62 ms/example
Batch evaluation speed	49. examples/ms
Loss	6.01 ± 1.4
Model memory	451. kB
Training examples used	1024 examples
Training time	2.57 s

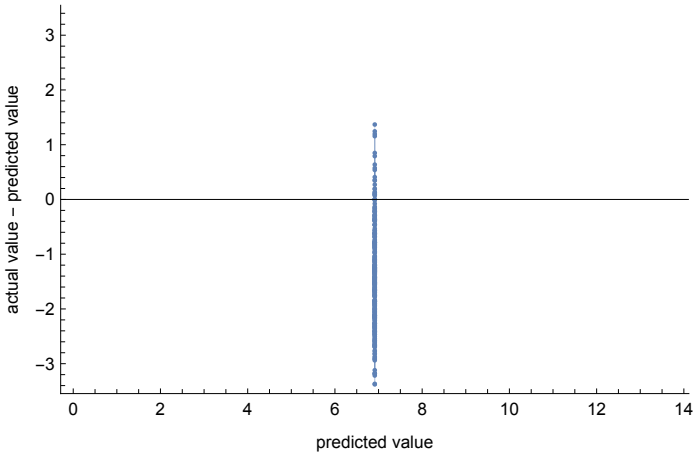
Out[757]=



Out[758]=



Out[759]=

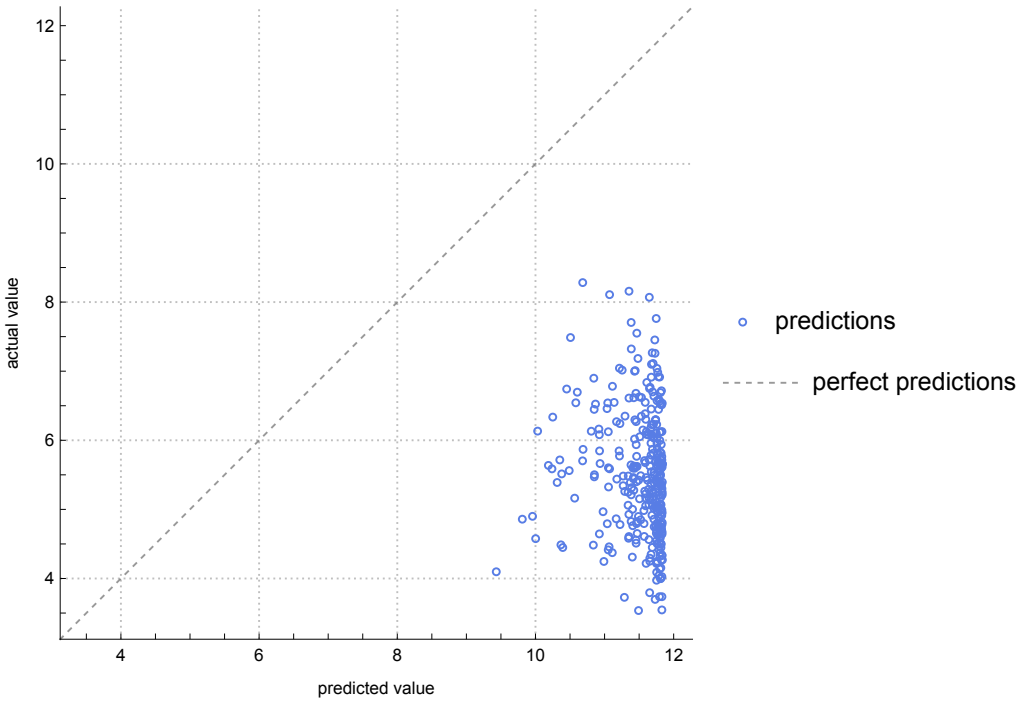

actual value − predicted value
predicted value
3
2
1
0
−1
−2
−3
0
2
4
6
8
10
12
14

In[760]:=
```
pGPEv1600 = Predict[finalTrain1600, Method → "GaussianProcess"]
pmGPEv1600 = PredictorMeasurements[pGPEv1600, finaltest1600]
pmGPEv1600["ComparisonPlot"]
pmGPEv1600["ResidualPlot"]
```

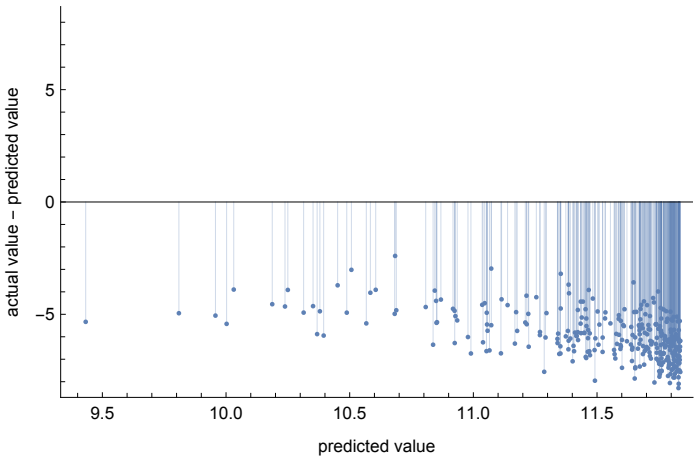
Out[760]= PredictorFunction[Input type: Mixed (number: 10) Method: GaussianProcess]

Out[761]= PredictorMeasurementsObject[Predictor: GaussianProcess Number of test examples: 320]

Out[762]=



Out[763]=



In[764]:=
```
pDTv1600 = Predict[finalTrain1600, Method → "DecisionTree"]
pmDTv1600 = PredictorMeasurements[pDTv1600, finaltest1600]
PredictorInformation[pDTv1600]
pmDTv1600["ComparisonPlot"]
pmDTv1600["ResidualPlot"]
```

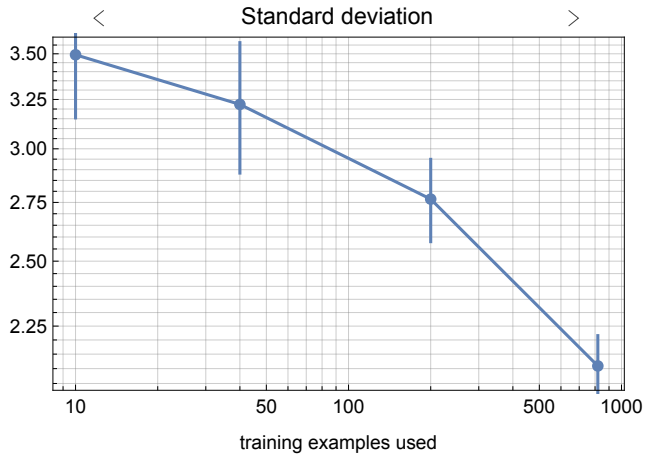
Out[764]= PredictorFunction[Input type: Mixed (number: 10) Method: DecisionTree]

Out[765]= PredictorMeasurementsObject[Predictor: DecisionTree Number of test examples: 320]

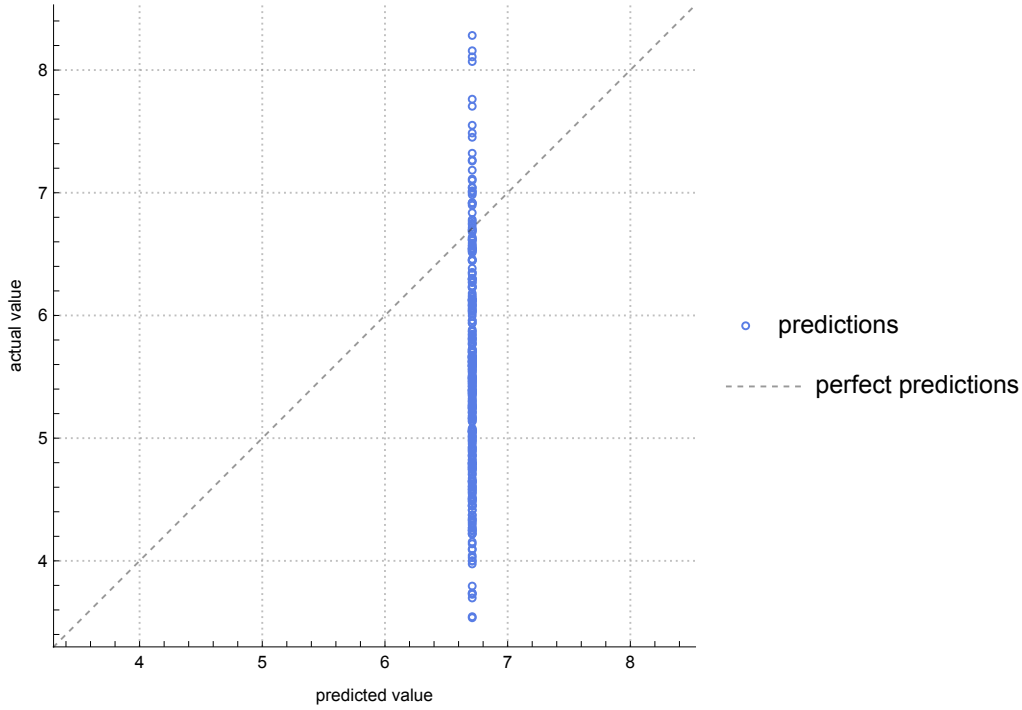
Out[766]=

Predictor information

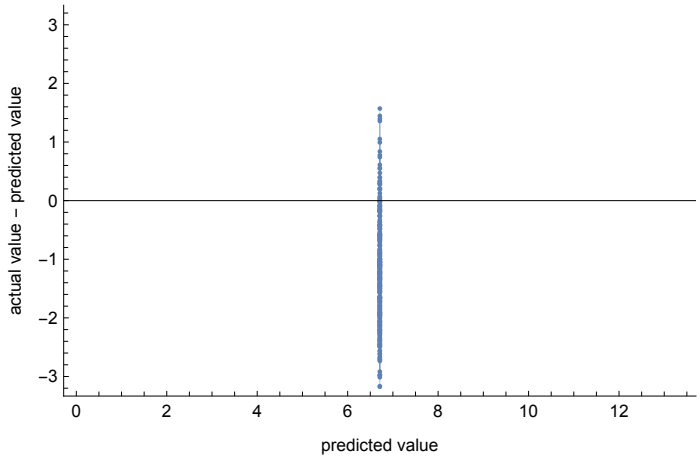
Data type	Mixed (number: 10)
Standard deviation	2.11 ± 0.11
Method	DecisionTree
Single evaluation time	1.4 ms/example
Batch evaluation speed	384. examples/ms
Loss	2.18 ± 0.072
Model memory	153. kB
Training examples used	1024 examples
Training time	487. ms



Out[767]=



Out[768]=



In[769]:=
```
pNearestv1600 = Predict[finalTrain1600, Method → "NearestNeighbors"]
pmNearestv1600 = PredictorMeasurements[pNearestv1600, finaltest1600]
PredictorInformation[pNearestv1600]
pmNearestv1600["ComparisonPlot"]
pmNearestv1600["ResidualPlot"]
```

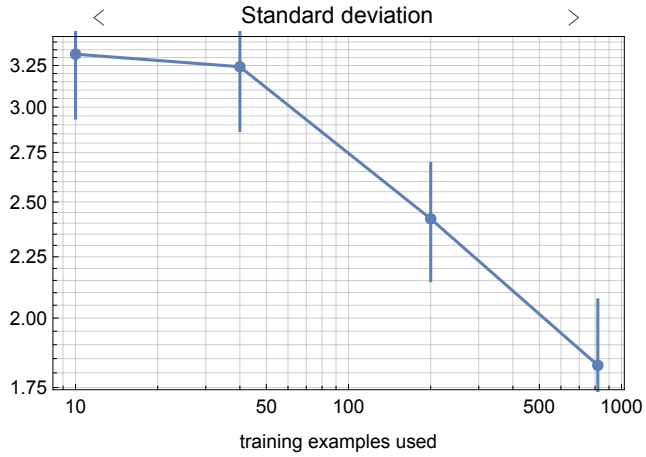
Out[769]= PredictorFunction[Input type: Mixed (number: 10) Method: NearestNeighbors]

Out[770]= PredictorMeasurementsObject[Predictor: NearestNeighbors Number of test examples: 320]

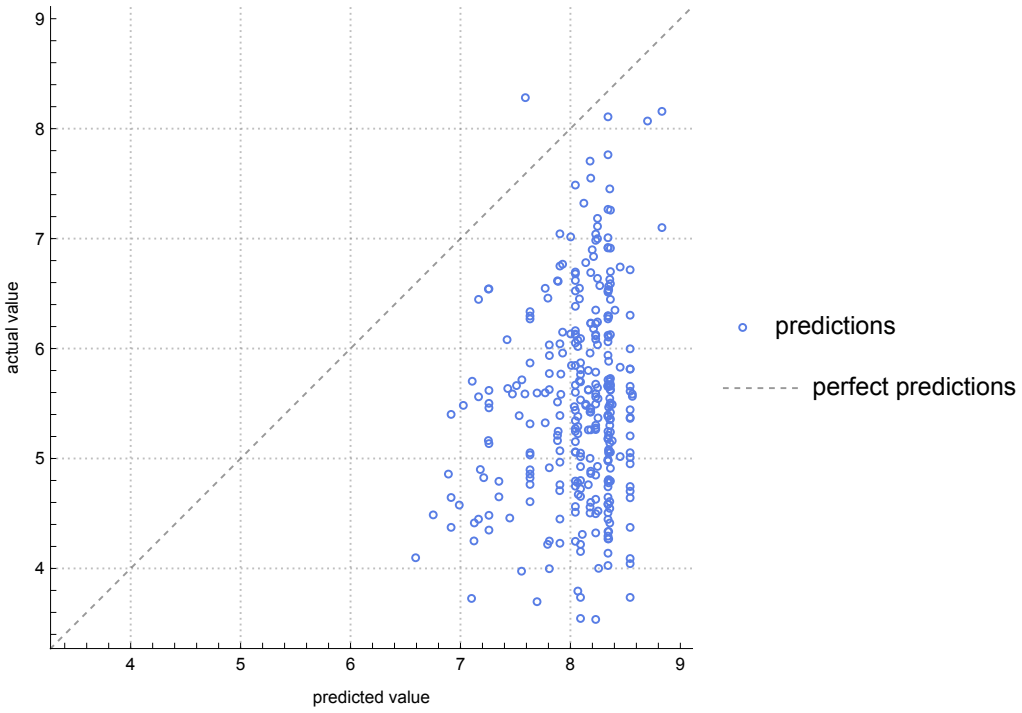
Predictor information

Data type	Mixed (number: 10)
Standard deviation	1.83 ± 0.25
Method	NearestNeighbors
Single evaluation time	1.5 ms/example
Batch evaluation speed	99. examples/ms
Loss	2.11 ± 0.12
Model memory	218. kB
Training examples used	1024 examples
Training time	500. ms

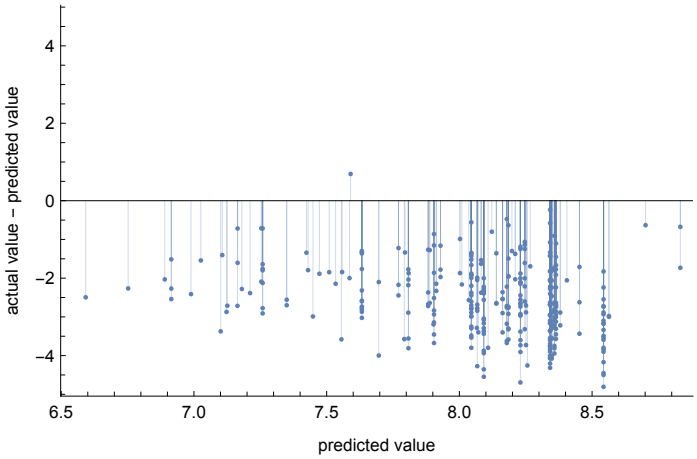
Out[771]=



Out[772]=



Out[773]=



In[774]:=
```
pLRv1600 = Predict[finalTrain1600, Method → "LinearRegression"]
pmLRv1600 = PredictorMeasurements[pLRv1600, finaltest1600]
PredictorInformation[pLRv1600]
pmLRv1600["ComparisonPlot"]
pmLRv1600["ResidualPlot"]
```

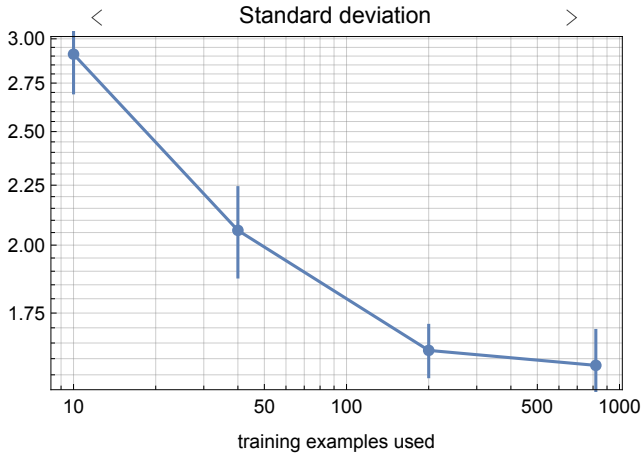
Out[774]= PredictorFunction[Input type: Mixed (number: 10) Method: LinearRegression]

Out[775]= PredictorMeasurementsObject[Predictor: LinearRegression Number of test examples: 320]

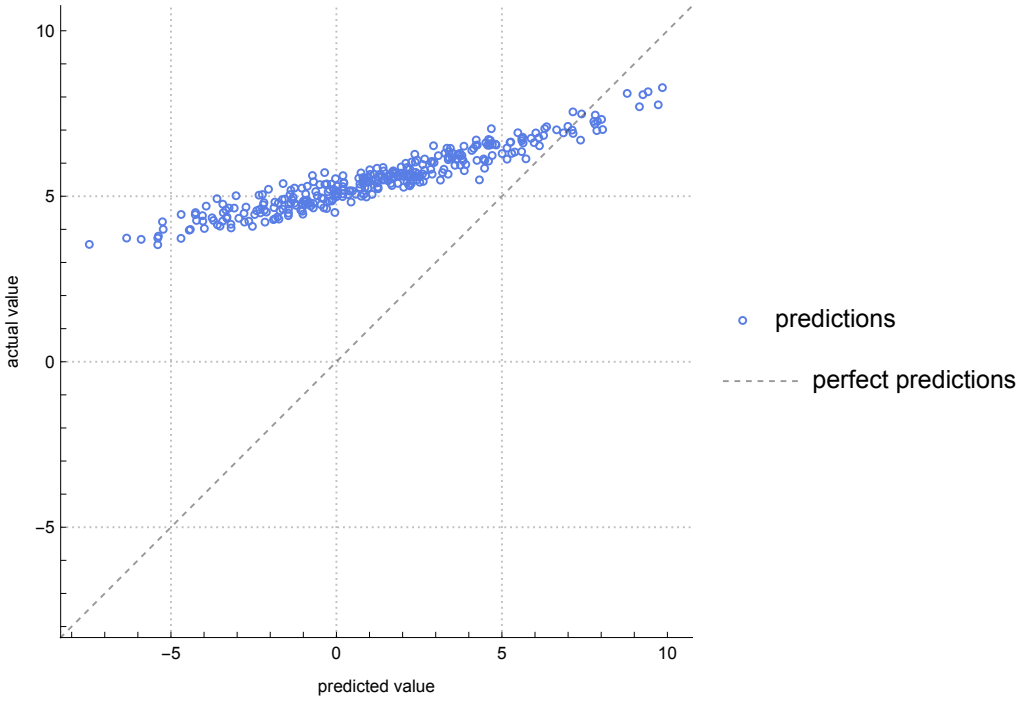
Out[776]=

Predictor information

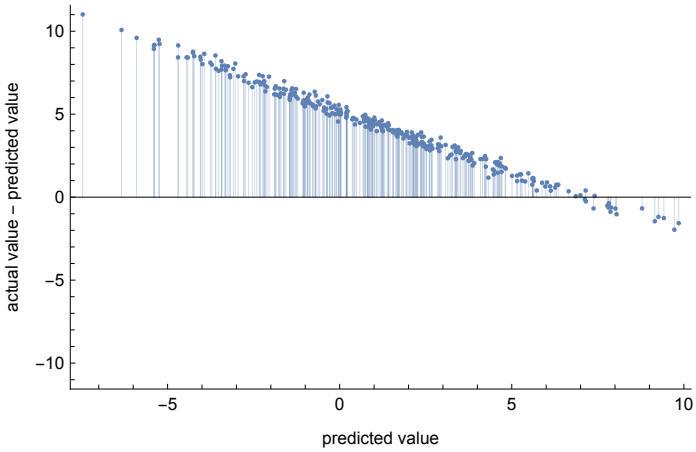
Data type	Mixed (number: 10)
Standard deviation	1.58 ± 0.11
Method	LinearRegression
Single evaluation time	1.61 ms/example
Batch evaluation speed	288. examples/ms
Loss	1.79 ± 0.055
Model memory	325. kB
Training examples used	1024 examples
Training time	2.02 s



Out[777]=



Out[778]=



In[779]:=
```
pNNv1600 = Predict[finalTrain1600, Method →
    {"NeuralNetwork", "NetworkDepth" → 3, "NetworkType" → "FullyConnected",
     "L2Regularization" → 0.05, MaxTrainingRounds → 1000}]
```

Out[779]= PredictorFunction[Input type: Mixed (number: 10) Method: NeuralNetwork]

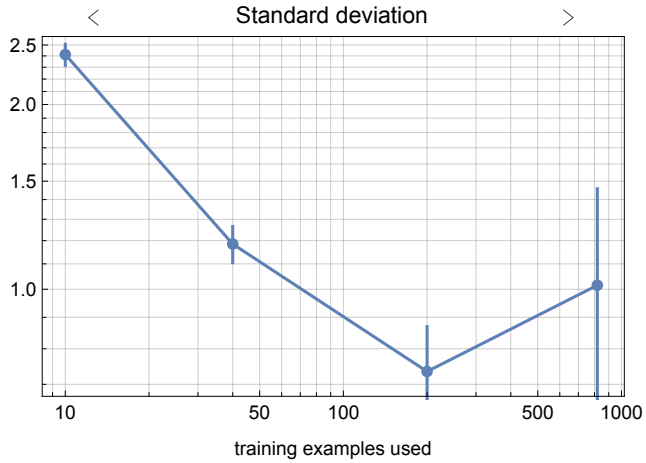
In[780]:=
```
pmNNv1600 = PredictorMeasurements[pNNv1600, finaltest1600]
PredictorInformation[pNNv1600]
pmNNv1600["ComparisonPlot"]
pmNNv1600["ResidualPlot"]
```

Out[780]= PredictorMeasurementsObject[Predictor: NeuralNetwork Number of test examples: 320]

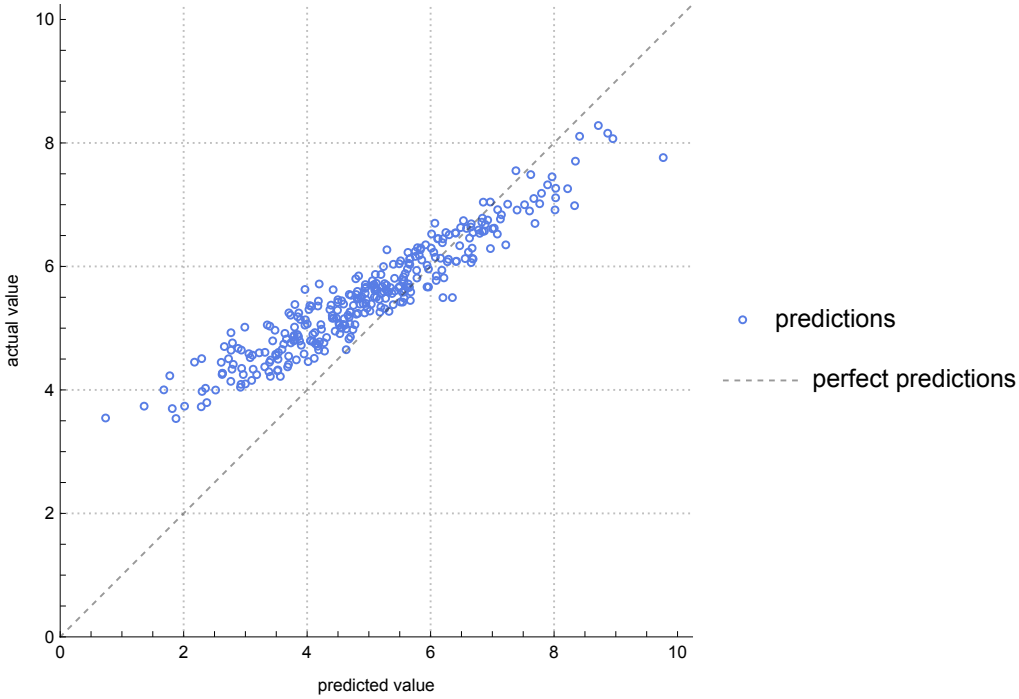
Out[781]=

Predictor information

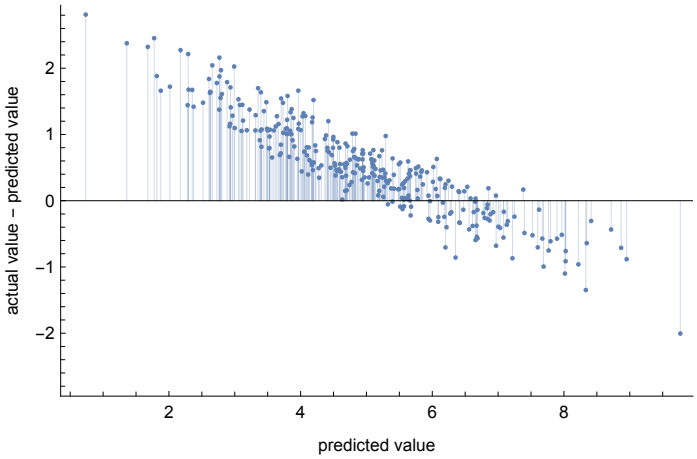
Data type	Mixed (number: 10)
Standard deviation	0.734 ± 0.13
Method	NeuralNetwork
Single evaluation time	2.13 ms/example
Batch evaluation speed	49.2 examples/ms
Loss	0.881 ± 0.082
Model memory	251. kB
Training examples used	1024 examples
Training time	16.3 s



Out[782]=



Out[783]=



In[794]:=

```
netSimple1 =
 NetChain[{BatchNormalizationLayer[], Tanh, 100, 100, 100, 100, 100, 100, 100,
   BatchNormalizationLayer[], Tanh, 1}, "Input" → 10, "Output" → "Scalar"]
trainedNetSimple1 = NetTrain[netSimple1, finalTrain1600, All,
  ValidationSet → finaltest1600, TargetDevice → "CPU", MaxTrainingRounds → 3000,
  Method → {"ADAM", "LearningRate" → 0.0001, "L2Regularization" → 0.03}]
```

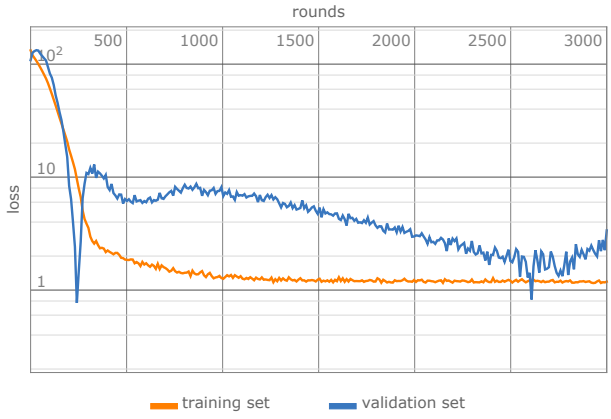
Out[794]=

NetChain[uninitialized | Input port: vector (size: 10) | Output port: scalar | Number of layers: 12]

Out[795]=

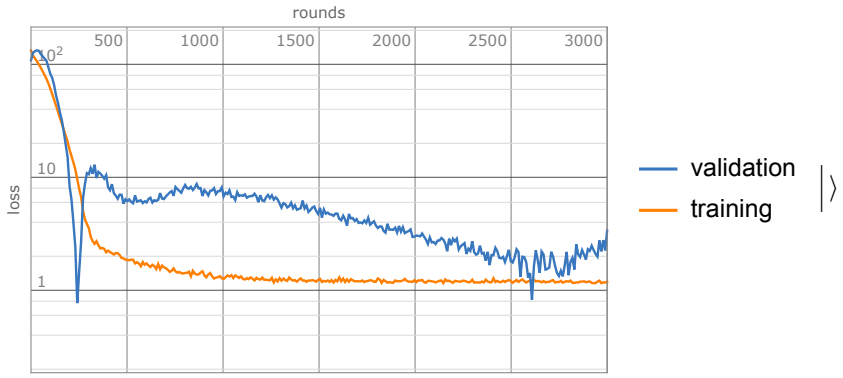
NetTrain Results

summary	batches: 48 000, rounds: 3000, time: 59s, examples/s: 52 142
data	training examples: 1024, validation examples: 320, processed examples: 3 072 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 1.22
validation	loss: 4.03



In[796]:=
```
trainedNetSimple1["FinalPlots"]
trainedNetSimple1["RoundMeasurements"]
best = trainedNetSimple1["BestValidationRound"]
trainedNetSimple1["ValidationLossList"][[best]]
NetInformation[trainedNetSimple1["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNetSimple1["TrainedNet"], "SummaryGraphic"]
```

Out[796]=

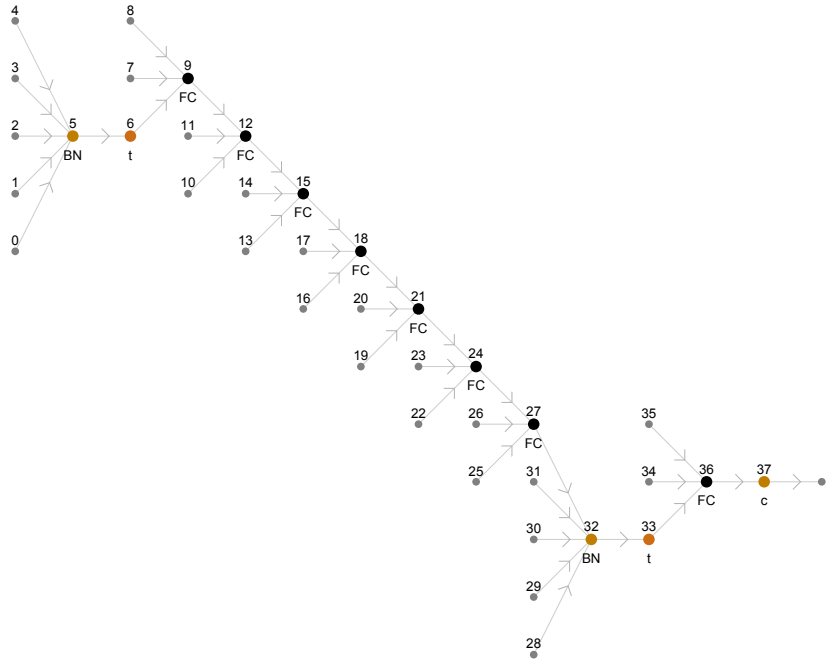


Out[797]= ⟨| Loss → 1.22379 |⟩

Out[798]= 2606

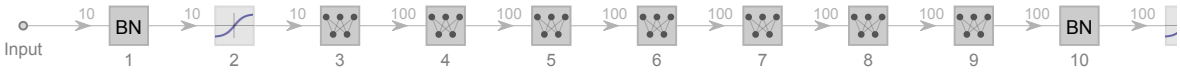
Out[799]= 0.304149

Out[800]=



Tensor BatchNorm tanh FullyConnected _copy

Out[801]=



In[810]:=
```
netSimple2 = NetChain[{BatchNormalizationLayer[], Tanh, 100, 100, 100,
    100, 100, 100, 100, 100, 100, BatchNormalizationLayer[], Tanh, 1}]
trainedNetSimple2 = NetTrain[netSimple2, finalTrain1600, All,
  ValidationSet → finaltest1600, TargetDevice → "CPU", MaxTrainingRounds → 6000,
  Method → {"ADAM", "LearningRate" → 0.0001, "L2Regularization" → 0.05}]
```

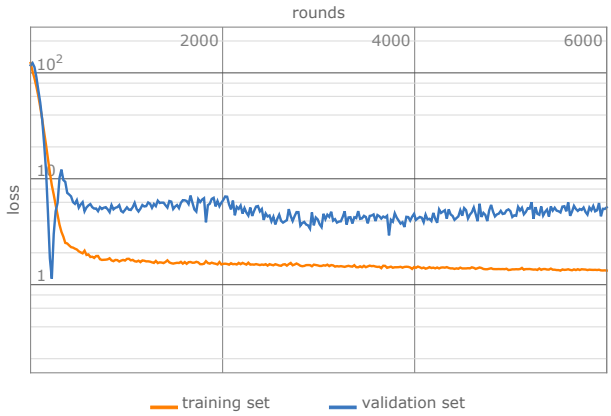
Out[810]=

NetChain[uninitialized | Input port: array of rank ≥ 1 | Output port: vector (size: 1) | Number of layers: 14]

Out[811]=

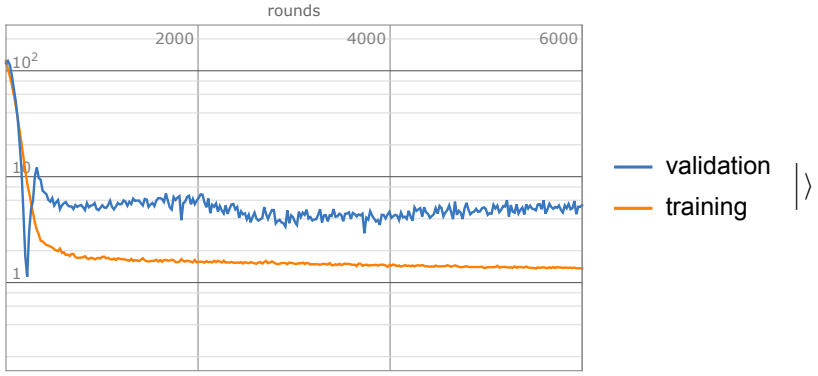
NetTrain Results

summary	batches: 96 000, rounds: 6000, time: 2.9min, examples/s: 35 130
data	training examples: 1024, validation examples: 320, processed examples: 6 144 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 1.33
validation	loss: 5.31



In[812]:=
```
trainedNetSimple2["FinalPlots"]
trainedNetSimple2["RoundMeasurements"]
best = trainedNetSimple2["BestValidationRound"]
trainedNetSimple2["ValidationLossList"][[best]]
NetInformation[trainedNetSimple2["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNetSimple2["TrainedNet"], "SummaryGraphic"]
```

Out[812]=



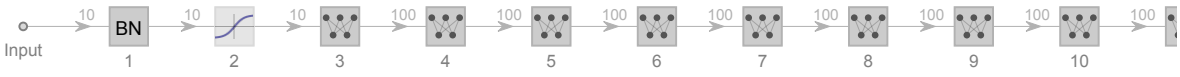
Out[813]= ⟨| Loss → 1.32644 |⟩

Out[814]= 3167

Out[815]= 0.234034

Out[816]=

Out[817]=



In[818]:=
```
netSimple3 = NetChain[{BatchNormalizationLayer[], Tanh, 50, 50,
    50, 50, 50, 50, 50, 50, 50, 50, BatchNormalizationLayer[], Tanh, 1}]
trainedNetSimple3 = NetTrain[netSimple3, finalTrain1600, All,
  ValidationSet → finaltest1600, TargetDevice → "CPU", MaxTrainingRounds → 6000,
  Method → {"ADAM", "LearningRate" → 0.0001, "L2Regularization" → 0.05}]
```

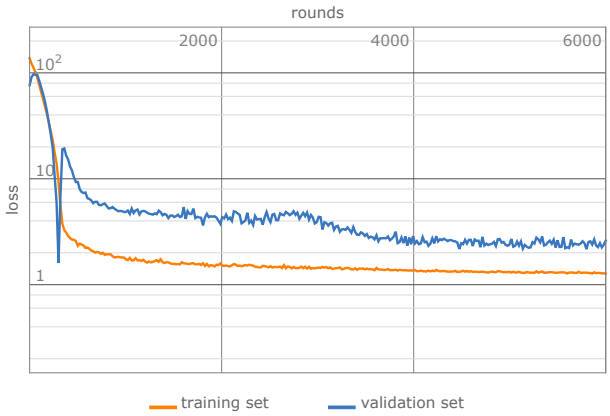
Out[818]=

NetChain[uninitialized | Input port: array of rank ≥ 1 | Output port: vector (size: 1) | Number of layers: 15]

Out[819]=

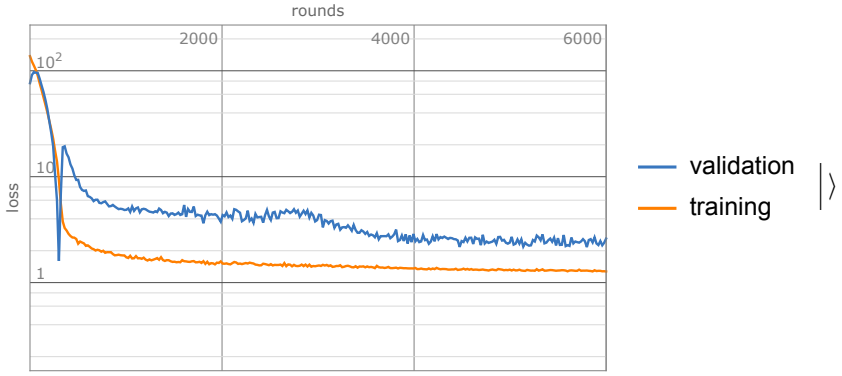
NetTrain Results

summary	batches: 96 000, rounds: 6000, time: 2.0min, examples/s: 52 104
data	training examples: 1024, validation examples: 320, processed examples: 6 144 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 1.31
validation	loss: 4.17



In[820]:=
```
trainedNetSimple3["FinalPlots"]
trainedNetSimple3["RoundMeasurements"]
best = trainedNetSimple3["BestValidationRound"]
trainedNetSimple3["ValidationLossList"][[best]]
NetInformation[trainedNetSimple3["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNetSimple3["TrainedNet"], "SummaryGraphic"]
```

Out[820]=

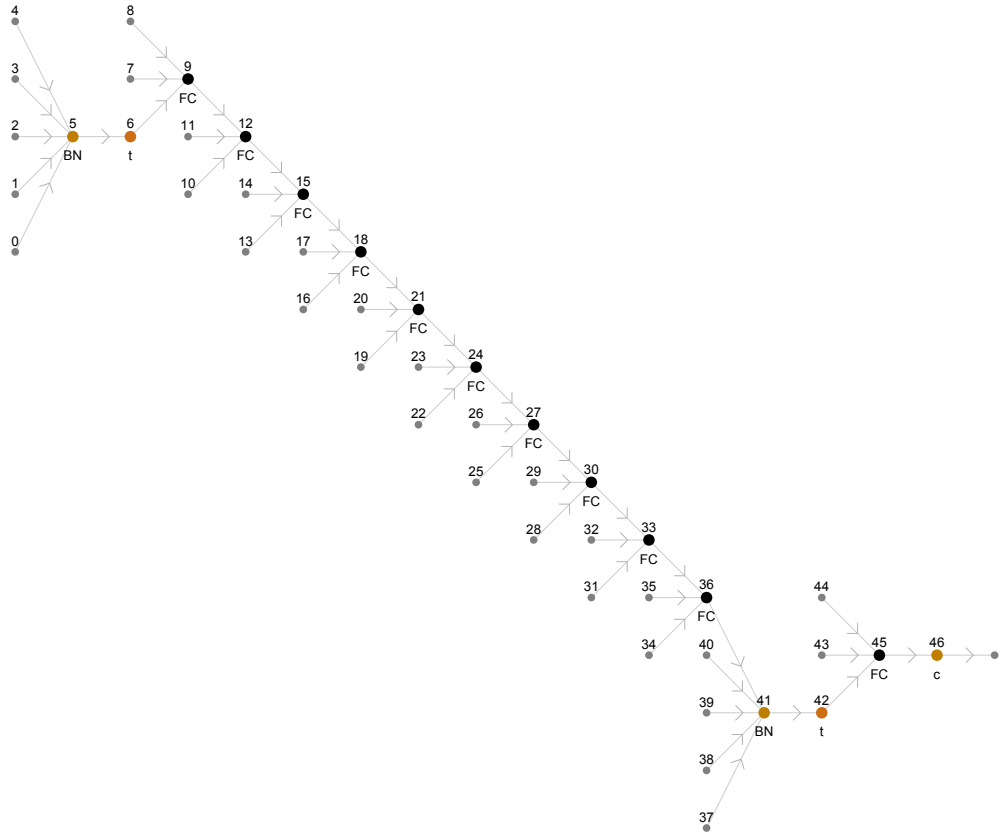


Out[821]= ⟨| Loss → 1.31077 |⟩

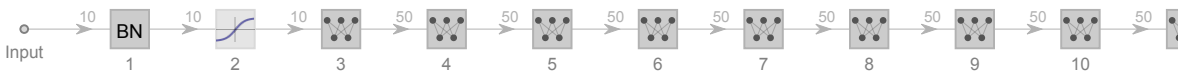
Out[822]= 4527

Out[823]= 0.223906

Out[824]=



Out[825]=



In[826]:=
```
netSimple4 = NetChain[{BatchNormalizationLayer[], Tanh, 50, 50, 50, 50,
    50, 50, 50, 50, 50, 50, 50, 50, BatchNormalizationLayer[], Tanh, 1}]
trainedNetSimple4 = NetTrain[netSimple4, finalTrain1600, All,
  ValidationSet → finaltest1600, TargetDevice → "CPU", MaxTrainingRounds → 6000,
  Method → {"ADAM", "LearningRate" → 0.0001, "L2Regularization" → 0.05}]
```

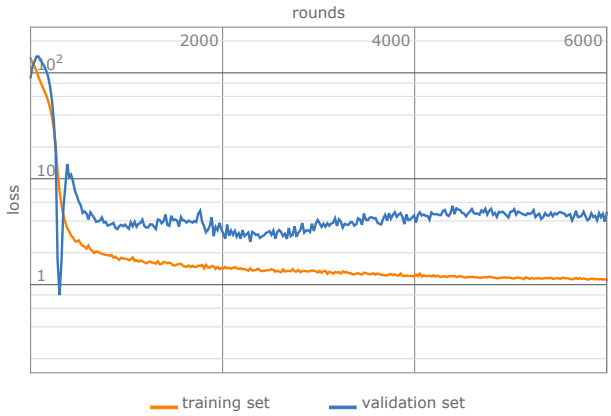
Out[826]=

NetChain[uninitialized | Input port: array of rank ≥ 1 | Output port: vector (size: 1) | Number of layers: 17]

Out[827]=

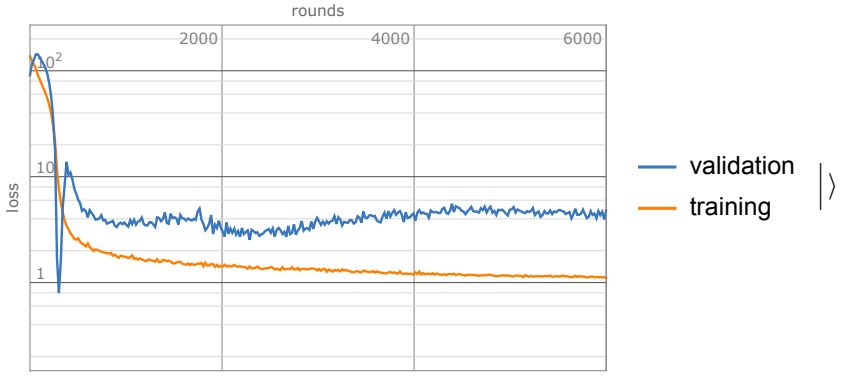
NetTrain Results

summary	batches: 96 000, rounds: 6000, time: 2.2min, examples/s: 46 989
data	training examples: 1024, validation examples: 320, processed examples: 6 144 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 1.1
validation	loss: 6.15



In[828]:=
```
trainedNetSimple4["FinalPlots"]
trainedNetSimple4["RoundMeasurements"]
best = trainedNetSimple4["BestValidationRound"]
trainedNetSimple4["ValidationLossList"][[best]]
NetInformation[trainedNetSimple4["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNetSimple4["TrainedNet"], "SummaryGraphic"]
```

Out[828]=



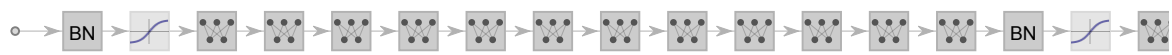
Out[829]= ⟨| Loss → 1.09542 |⟩

Out[830]= 4094

Out[831]= 0.228744

Out[832]=

Out[833]=



In[834]:=
```
netSimple5 = NetChain[{BatchNormalizationLayer[], Tanh, 100, 100, 100, 100,
    100, 100, 100, 100, 100, 100, BatchNormalizationLayer[], Tanh, 1}]
trainedNetSimple5 = NetTrain[netSimple5, finalTrain1600, All,
  ValidationSet → finaltest1600, TargetDevice → "CPU", MaxTrainingRounds → 5000,
  Method → {"ADAM", "LearningRate" → 0.0001, "L2Regularization" → 0.05}]
```

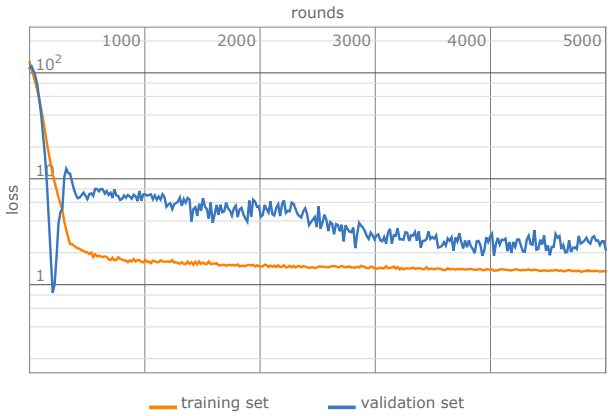
Out[834]=

NetChain[uninitialized | Input port: array of rank ≥ 1 | Output port: vector (size: 1) | Number of layers: 15]

Out[835]=

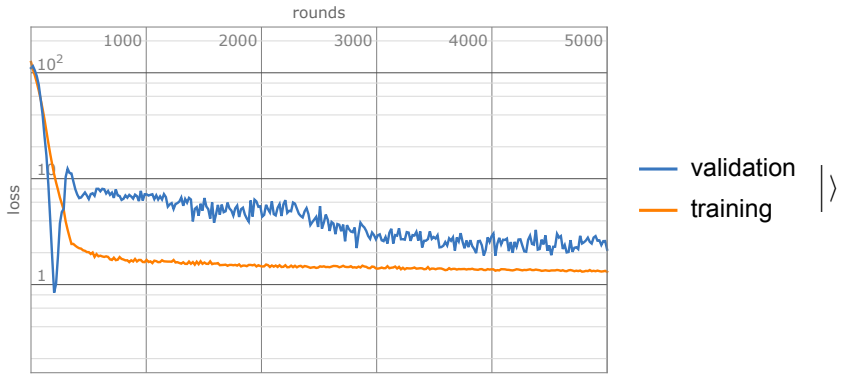
NetTrain Results

summary	batches: 80 000, rounds: 5000, time: 2.5min, examples/s: 33 834
data	training examples: 1024, validation examples: 320, processed examples: 5 120 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 1.31
validation	loss: 2.42



In[836]:=
```
trainedNetSimple5["FinalPlots"]
trainedNetSimple5["RoundMeasurements"]
best = trainedNetSimple5["BestValidationRound"]
trainedNetSimple5["ValidationLossList"][[best]]
NetInformation[trainedNetSimple5["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNetSimple5["TrainedNet"], "SummaryGraphic"]
```

Out[836]=

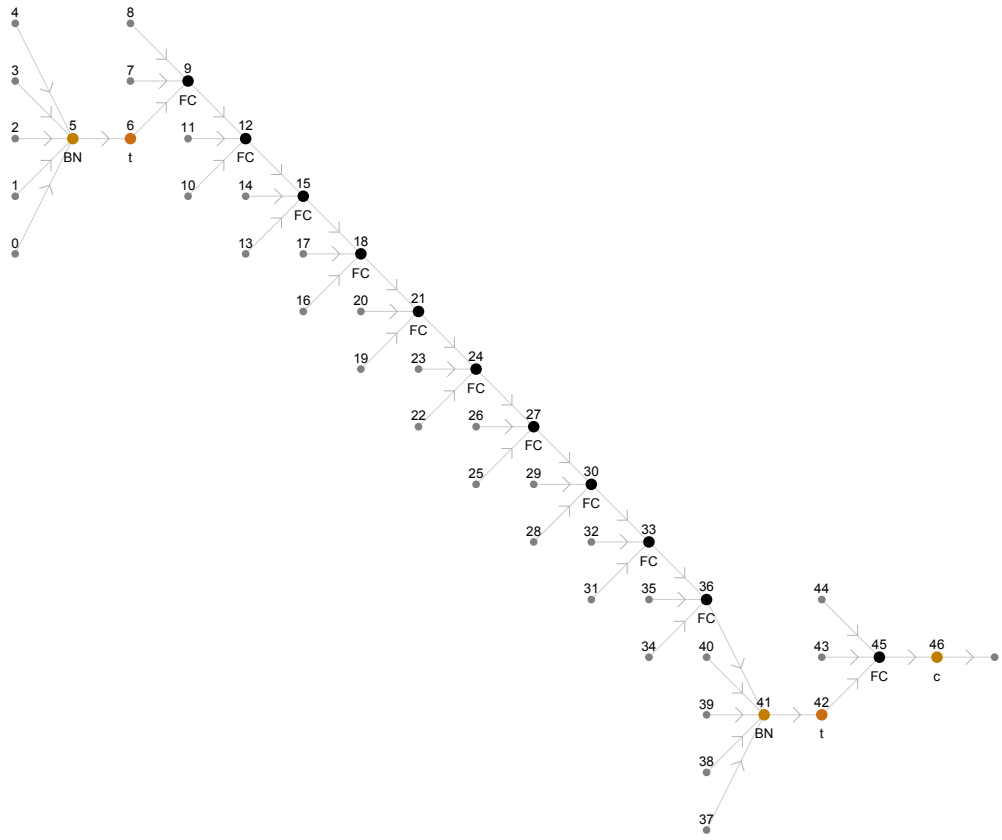


Out[837]= ⟨| Loss → 1.30938 |⟩

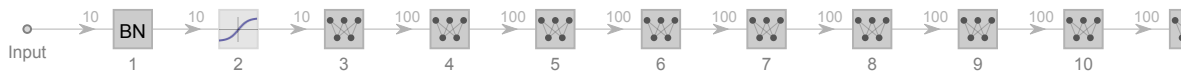
Out[838]= 3872

Out[839]= 0.214454

Out[840]=



Out[841]=



In[888]:=
```
netSimple6 = NetChain[{BatchNormalizationLayer[], Tanh, 100, 100,
    100, 100, 100, 100, 100, 100, BatchNormalizationLayer[], Tanh, 1}]
trainedNetSimple6 = NetTrain[netSimple6, finalTrain1600, All,
  ValidationSet → finaltest1600, TargetDevice → "CPU", MaxTrainingRounds → 5000,
  Method → {"ADAM", "LearningRate" → 0.0002, "L2Regularization" → 0.05}]
```

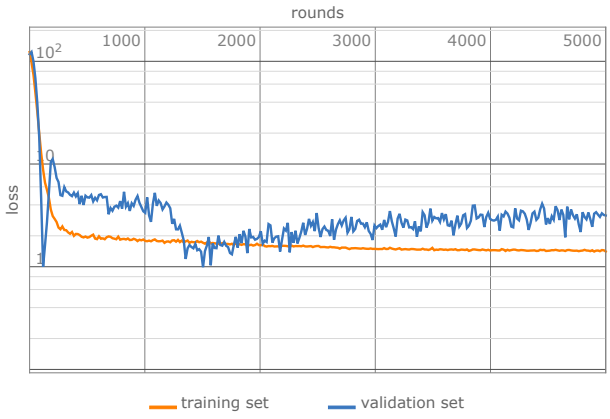
Out[888]=

NetChain[uninitialized | Input port: array of rank ≥ 1 | Output port: vector (size: 1) | Number of layers: 13]

Out[889]=

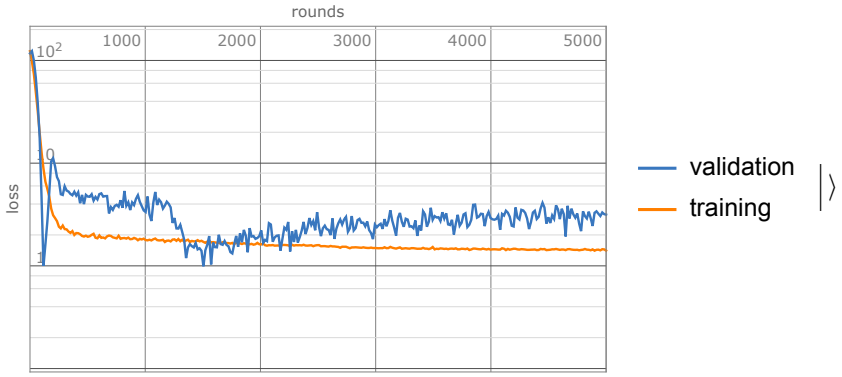
NetTrain Results

summary	batches: 80 000, rounds: 5000, time: 2.1min, examples/s: 40 876
data	training examples: 1024, validation examples: 320, processed examples: 5 120 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 1.41
validation	loss: 6.



In[896]:=
```
trainedNetSimple6["FinalPlots"]
trainedNetSimple6["TotalTrainingTime"]
trainedNetSimple6["RoundMeasurements"]
best = trainedNetSimple6["BestValidationRound"]
trainedNetSimple6["ValidationLossList"][[best]]
NetInformation[trainedNetSimple6["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNetSimple6["TrainedNet"], "SummaryGraphic"]
```

Out[896]=



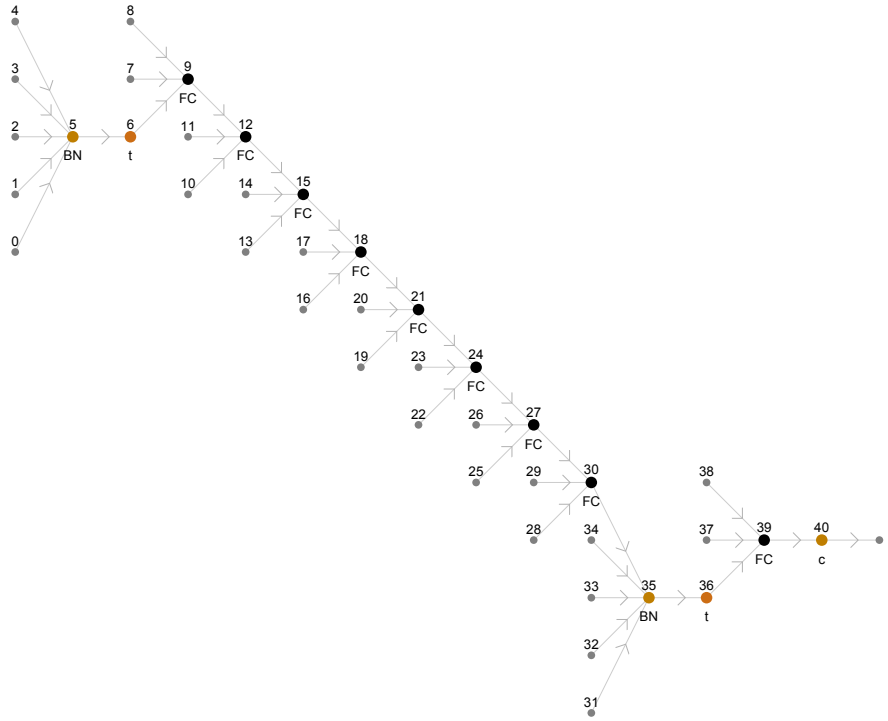
Out[897]= 125.258

Out[898]= ⟨| Loss → 1.40556 |⟩

Out[899]= 2098

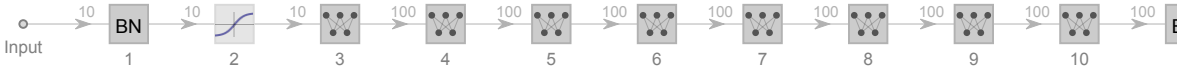
Out[900]= 0.163367

Out[901]=



Tensor BatchNorm tanh FullyConnected _copy

Out[902]=



In[905]:=
```
netSimple7 =
 NetChain[{BatchNormalizationLayer[], Tanh, 100, 100, 100, 100, 100, 100, 100, 100,
   100, BatchNormalizationLayer[], Tanh, 1}, "Input" → 10, "Output" → "Scalar"]
trainedNetSimple7 = NetTrain[netSimple7, finalTrain1600, All,
  ValidationSet → finaltest1600, TargetDevice → "CPU", MaxTrainingRounds → 10 000,
  Method → {"ADAM", "LearningRate" → 0.0002, "L2Regularization" → 0.05}]
```

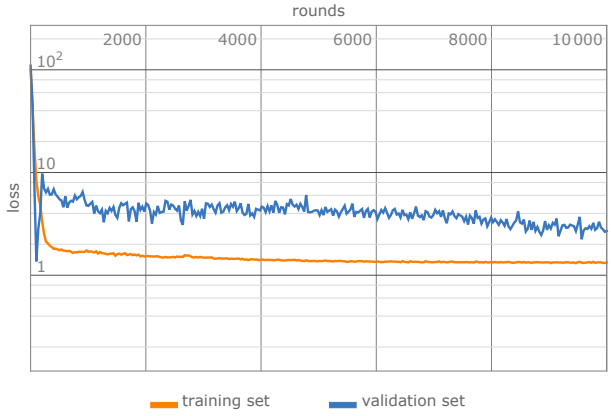
Out[905]=

NetChain[uninitialized | Input port: vector (size: 10) | Output port: scalar | Number of layers: 14]

Out[906]=

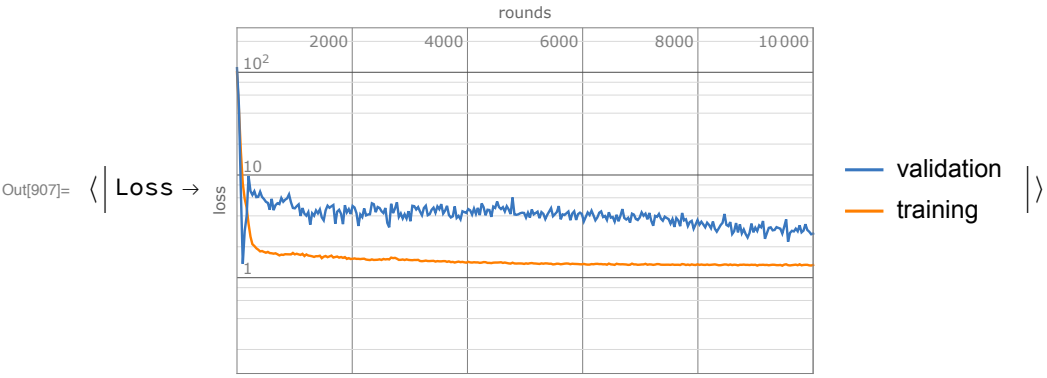
NetTrain Results

summary	batches: 160 000, rounds: 10 000, time: 5.0min, examples/s: 33 806
data	training examples: 1024, validation examples: 320, processed examples: 10 240 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 1.33
validation	loss: 1.13



In[907]:=
```
trainedNetSimple7["FinalPlots"]
trainedNetSimple7["TotalTrainingTime"]
trainedNetSimple7["RoundMeasurements"]
best = trainedNetSimple7["BestValidationRound"]
trainedNetSimple7["ValidationLossList"][[best]]
NetInformation[trainedNetSimple7["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNetSimple7["TrainedNet"], "SummaryGraphic"]
```

Out[907]=



Out[908]= 302.904

Out[909]= ⟨| Loss → 1.32892 |⟩

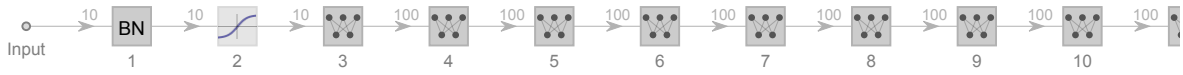
Out[910]= 2006

Out[911]= 0.186545

Out[912]=

Tensor BatchNorm tanh FullyConnected _copy

Out[913]=



In[924]:=
```
net16Simple8 = NetChain[{BatchNormalizationLayer[], Tanh, 100, 100, 100,
    100, 100, 100, 100, 100, 100, 100, BatchNormalizationLayer[], Tanh, 1}]
trainedNet16Simple8 = NetTrain[net16Simple8, finalTrain1600, All,
  ValidationSet → finaltest1600, TargetDevice → "CPU", MaxTrainingRounds → 5000,
  Method → {"ADAM", "LearningRate" → 0.0002, "L2Regularization" → 0.03}]
```

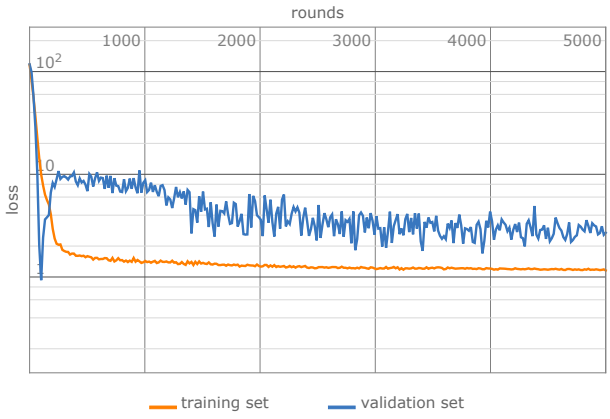
Out[924]=

NetChain[uninitialized | Input port: array of rank ≥ 1 | Output port: vector (size: 1) | Number of layers: 15]

Out[925]=

NetTrain Results

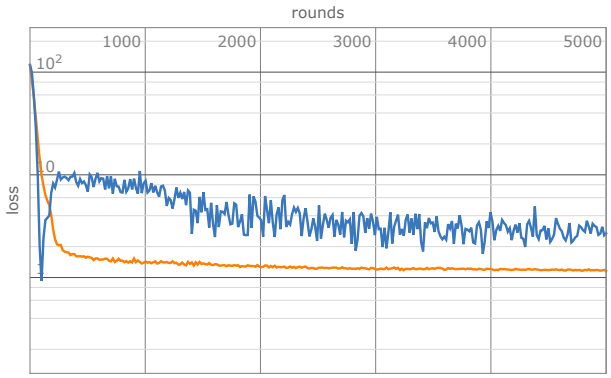
summary	batches: 80 000, rounds: 5000, time: 2.4min, examples/s: 34 955
data	training examples: 1024, validation examples: 320, processed examples: 5 120 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 1.15
validation	loss: 7.01



In[926]:=
```
trainedNet16Simple8["FinalPlots"]
trainedNet16Simple8["TotalTrainingTime"]
trainedNet16Simple8["RoundMeasurements"]
best = trainedNet16Simple8["BestValidationRound"]
trainedNet16Simple8["ValidationLossList"][[best]]
NetInformation[trainedNet16Simple8["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNet16Simple8["TrainedNet"], "SummaryGraphic"]
```

Out[926]=

⟨ Loss →



⟩

Out[927]= 146.474

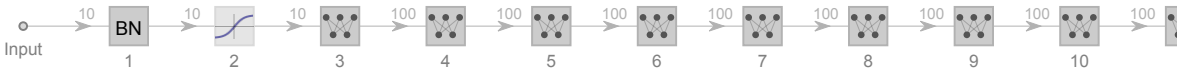
Out[928]= ⟨| Loss → 1.15024 |⟩

Out[929]= 2363

Out[930]= 0.189905

Out[931]=

Out[932]=



In[942]:=
```
net16Simple9 = NetChain[{BatchNormalizationLayer[], Tanh, 100, 100, 100, 100,
    100, 100, 100, 100, 100, 100, 100, BatchNormalizationLayer[], Tanh, 1}]
trainedNet16Simple9 = NetTrain[net16Simple9, finalTrain1600, All,
  ValidationSet → finaltest1600, TargetDevice → "CPU", MaxTrainingRounds → 5000,
  Method → {"ADAM", "LearningRate" → 0.0002, "L2Regularization" → 0.01}]
```

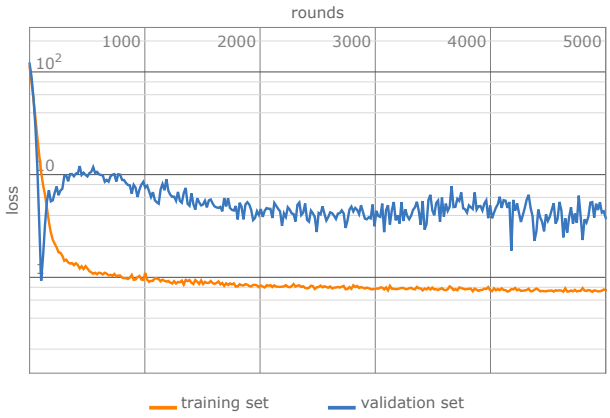
Out[942]=

NetChain[uninitialized | Input port: array of rank ≥ 1 | Output port: vector (size: 1) | Number of layers: 16]

Out[943]=

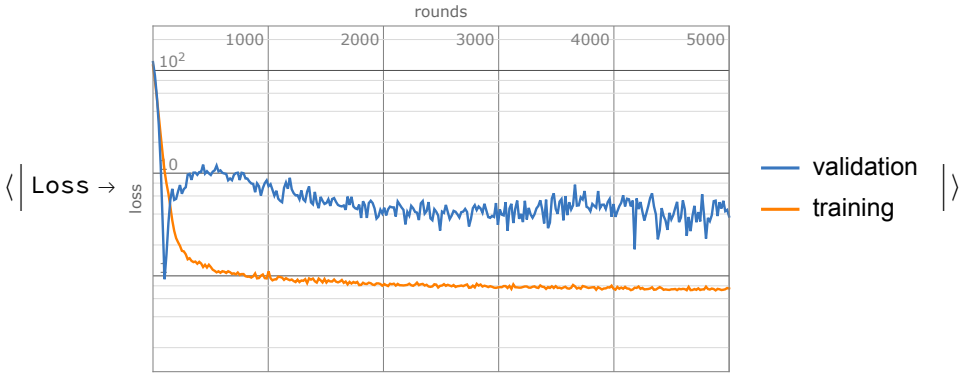
NetTrain Results

summary	batches: 80 000, rounds: 5000, time: 2.7min, examples/s: 31 984
data	training examples: 1024, validation examples: 320, processed examples: 5 120 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 7.08×10^{-1}
validation	loss: 2.07



In[944]:=
```
trainedNet16Simple9["FinalPlots"]
trainedNet16Simple9["TotalTrainingTime"]
trainedNet16Simple9["RoundMeasurements"]
best = trainedNet16Simple9["BestValidationRound"]
trainedNet16Simple9["ValidationLossList"][[best]]
NetInformation[trainedNet16Simple9["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNet16Simple9["TrainedNet"], "SummaryGraphic"]
```

Out[944]=



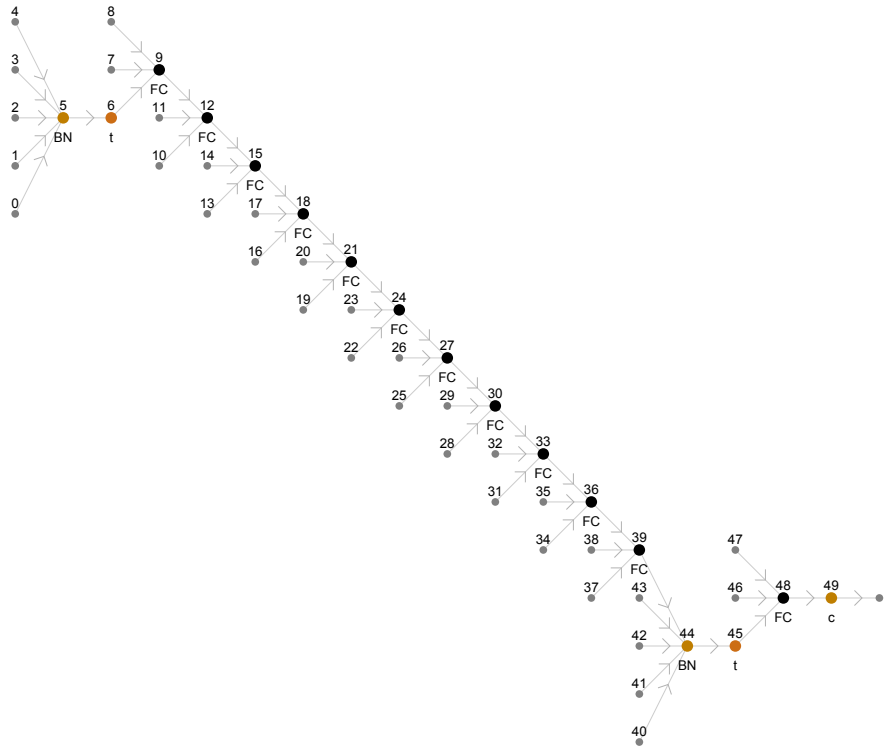
Out[945]= 160.082

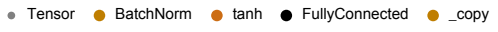
Out[946]= ⟨| Loss → 0.707874 |⟩

Out[947]= 4992

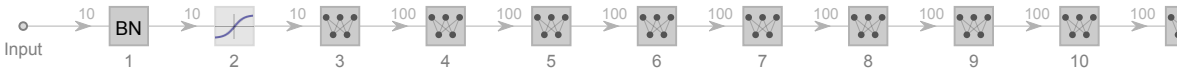
Out[948]= 0.171547

Out[949]=





Out[950]=



In[979]:=
```
net16Simple10 = NetChain[{BatchNormalizationLayer[], Tanh, 100, 100, 100, 100,
    100, 100, 100, 100, 100, 100, 100, BatchNormalizationLayer[], Tanh, 1}]
trainedNet16Simple10 = NetTrain[net16Simple10, finalTrain1600, All,
  ValidationSet → finaltest1600, TargetDevice → "CPU", MaxTrainingRounds → 15 000,
  Method → {"ADAM", "LearningRate" → 0.00008, "L2Regularization" → 0.02}]
```

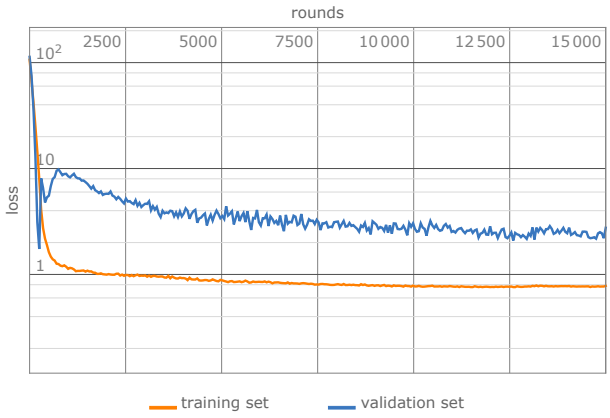
Out[979]=

NetChain[uninitialized | Input port: array of rank ≥ 1 | Output port: vector (size: 1) | Number of layers: 16]

Out[980]=

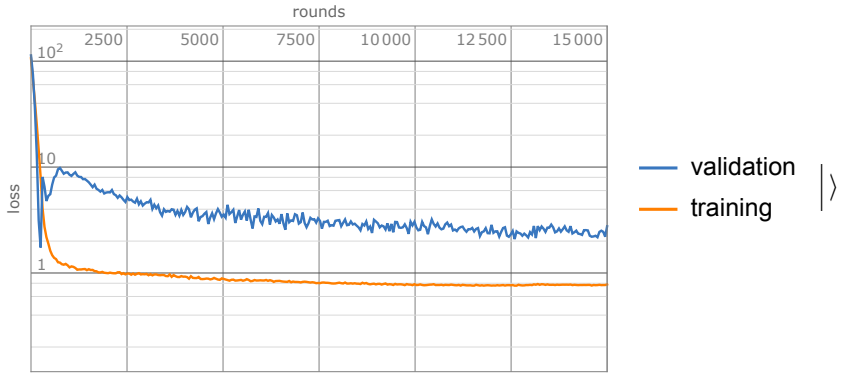
NetTrain Results

summary	batches: 240 000, rounds: 15 000, time: 12min, examples/s: 22 091
data	training examples: 1024, validation examples: 320, processed examples: 15 360 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 7.55×10^{-1}
validation	loss: 1.48



In[988]:=
```
trainedNet16Simple10["FinalPlots"]
trainedNet16Simple10["TotalTrainingTime"]
trainedNet16Simple10["RoundMeasurements"]
best = trainedNet16Simple10["BestValidationRound"]
trainedNet16Simple10["ValidationLossList"][[best]]
NetInformation[trainedNet16Simple10["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNet16Simple10["TrainedNet"], "SummaryGraphic"]
```

Out[988]=



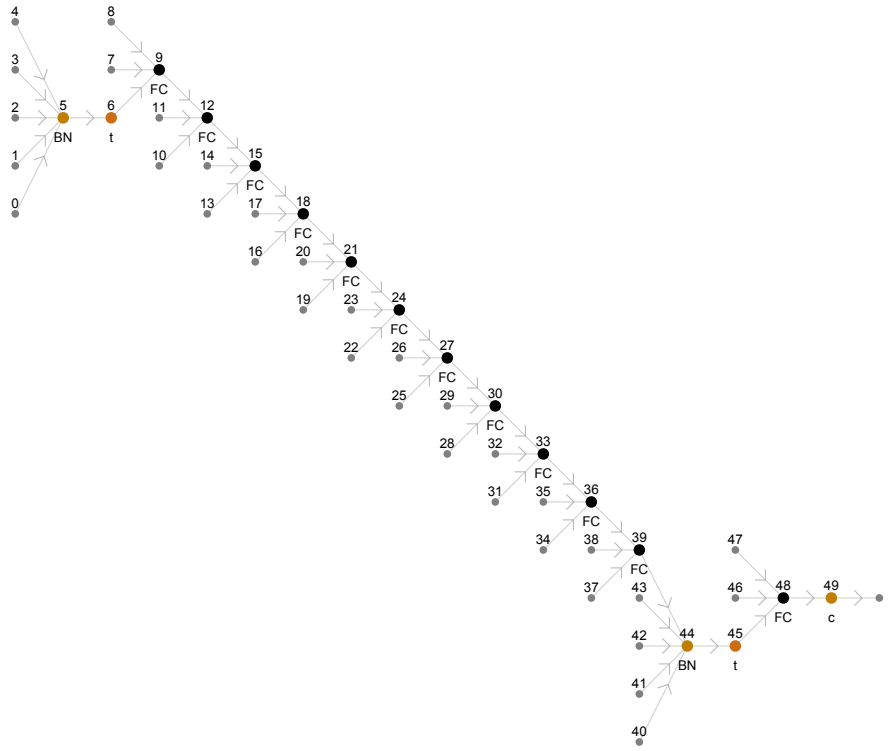
Out[989]= 695.307

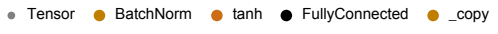
Out[990]= ⟨| Loss → 0.755217 |⟩

Out[991]= 12 670

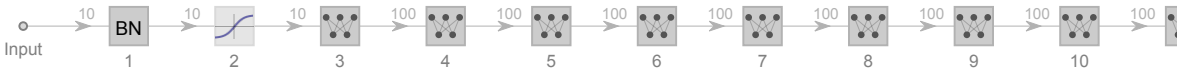
Out[992]= 0.188035

Out[993]=





Out[994]=



In[997]:=
```
net16Simple11 = NetChain[{BatchNormalizationLayer[],
    Tanh, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
    BatchNormalizationLayer[], Tanh, 1}, "Input" → 10, "Output" → "Scalar"]
trainedNet16Simple11 = NetTrain[net16Simple11, finalTrain1600, All,
  ValidationSet → finaltest1600, TargetDevice → "CPU", MaxTrainingRounds → 15 000,
  Method → {"ADAM", "LearningRate" → 0.0001, "L2Regularization" → 0.02}]
```

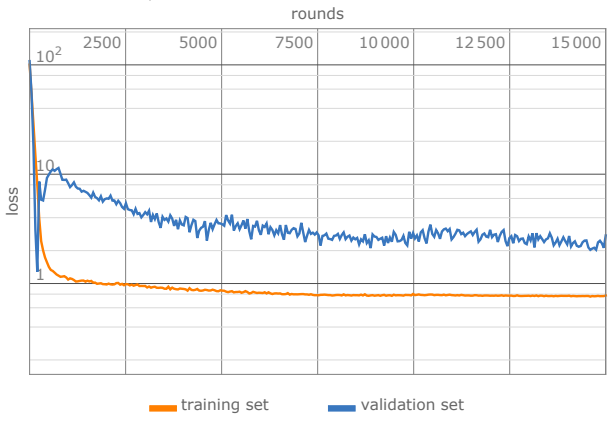
Out[997]=

NetChain[uninitialized | Input port: vector (size: 10) | Output port: scalar | Number of layers: 16]

NetTrain Results

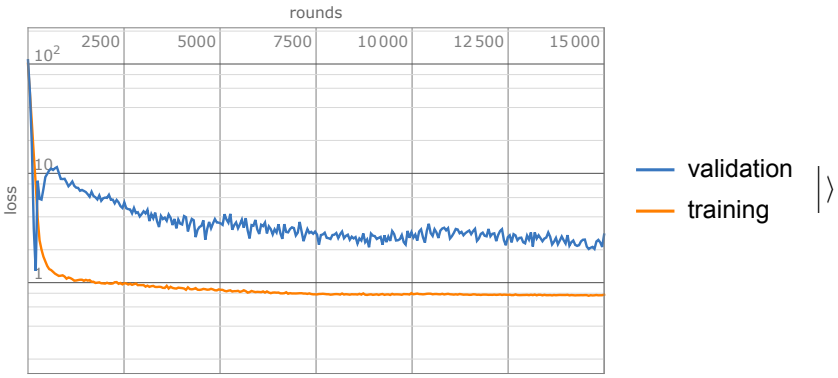
summary	batches: 240 000, rounds: 15 000, time: 8.8min, examples/s: 29 132
data	training examples: 1024, validation examples: 320, processed examples: 15 360 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 7.53×10^{-1}
validation	loss: 1.15

Out[998]=



In[999]:=
```
trainedNet16Simple11["FinalPlots"]
trainedNet16Simple11["TotalTrainingTime"]
trainedNet16Simple11["RoundMeasurements"]
best = trainedNet16Simple11["BestValidationRound"]
trainedNet16Simple11["ValidationLossList"][[best]]
NetInformation[trainedNet16Simple11["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNet16Simple11["TrainedNet"], "SummaryGraphic"]
```

Out[999]=



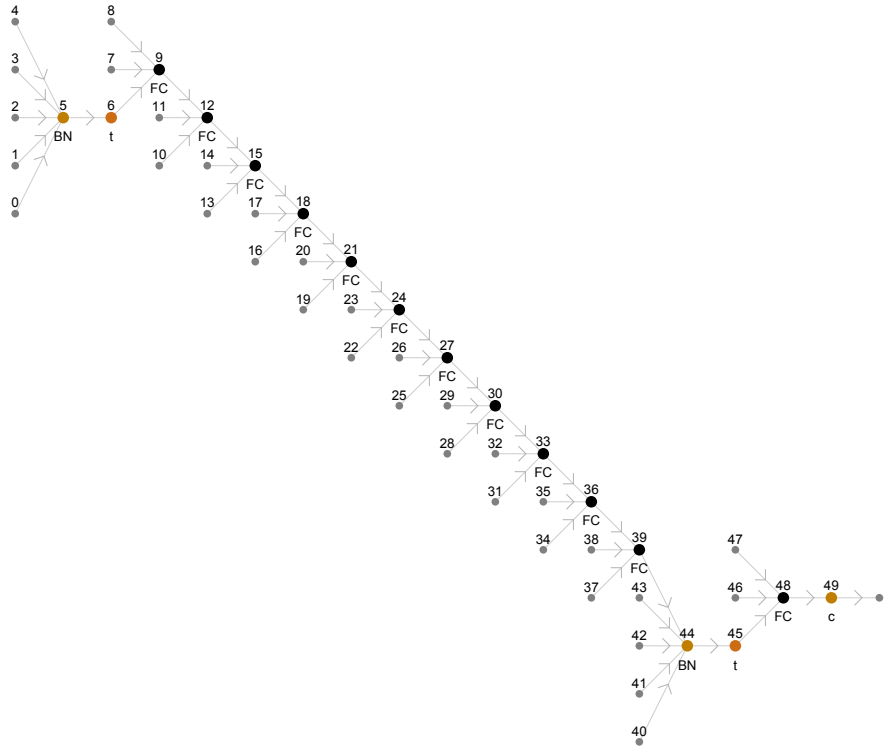
Out[1000]= 527.261

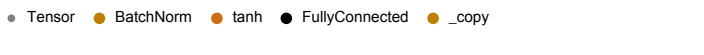
Out[1001]= ⟨|Loss → 0.75316|⟩

Out[1002]= 5110

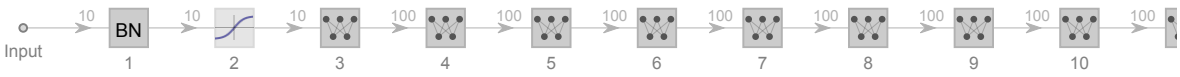
Out[1003]= 0.209511

Out[1004]=





Out[1005]=



MeanSquare

In[1112]:=
```
pmNNv1600["MeanSquare"]
pmDTv1600["MeanSquare"]
pmLRv1600["MeanSquare"]
pmRFv1600["MeanSquare"]
pmXGTv1600["MeanSquare"]
pmGPEv1600["MeanSquare"]
pmLRv1600["MeanSquare"]
pmNearestv1600["MeanSquare"]
```

Out[1112]= 0.806247

Out[1113]= 2.24735

Out[1114]= 24.1802

Out[1115]= 2.65628

Out[1116]= 2.77351

Out[1117]= 37.0196

Out[1118]= 24.1802

Out[1119]= 7.40404