In[1007]:=
```
ABMInputs200 = Import[
    "/Users/thorsilver/Documents/ABM-for-social-care/LPtau200runs_GEMSA_inputs.
      csv"];

ABMOutputs200 =
  Import["/Users/thorsilver/Documents/ABM-for-social-care/LPtau200runs
     GEMSA outputs only.csv"];
```

In[1009]:=
```
ABMOutputs200 = Function[x, x / 1000] /@ ABMOutputs200;
ABMAssoc200 = AssociationThread[ABMInputs200 → Flatten[ABMOutputs200]];
ABMnewData200 = Dataset[ABMAssoc200];
ABMNormal200 = Normal[ABMAssoc200];
ABMNormalRandom = RandomSample[ABMNormal200];
ABMtrain200 = TakeDrop[ABMNormal200, 160];
ABMtest200 = ABMtrain200[[2]];
ABMtraining200 = ABMtrain200[[1]];
trainDevSplit200 = TakeDrop[ABMtraining200, 128];
finalTrain200 = trainDevSplit200[[1]];
finalDev200 = trainDevSplit200[[2]];
finaltest200 = ABMtest200;
```

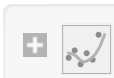
In[1021]:=
```
Length[finalDev200]
Length[finalTrain200]
Length[finaltest200]
```

Out[1021]= 32

Out[1022]= 128

Out[1023]= 40

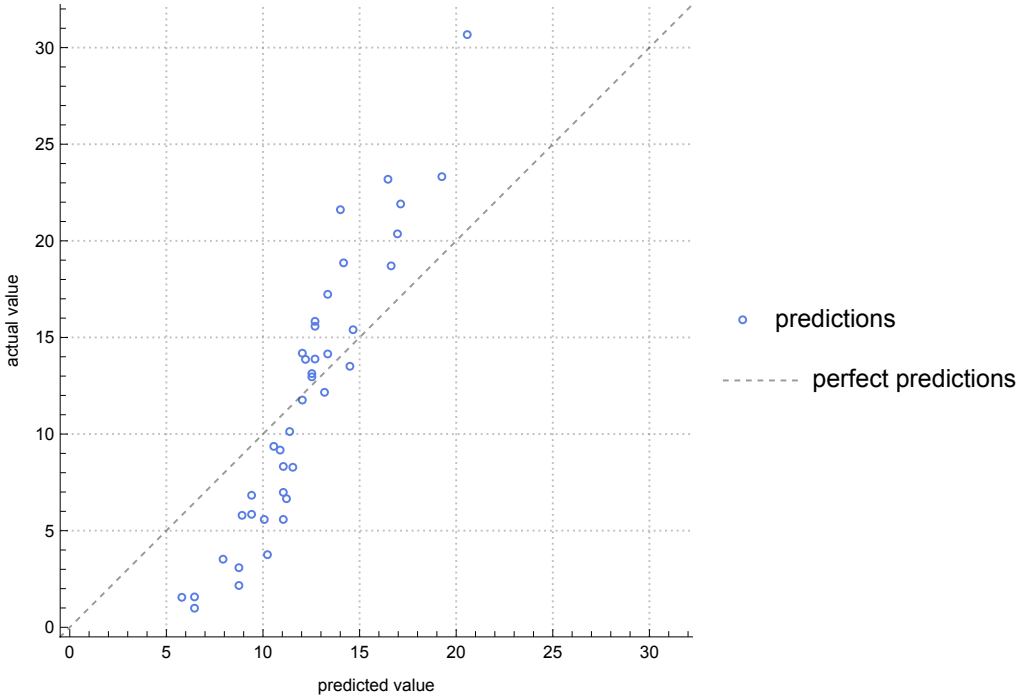
In[1024]:=
```
pRFv200 = Predict[finalTrain200, Method → "RandomForest"]
```

Out[1024]= PredictorFunction[ Input type: Mixed (number: 10)
Method: RandomForest]

In[1050]:=
```
pmRFv200 = PredictorMeasurements[pRFv200, finaltest200]
pmRFv200["ComparisonPlot"]
pmRFv200["MeanSquare"]
Information[pRFv200]
```

Out[1050]= PredictorMeasurementsObject[ Predictor: RandomForest
Number of test examples: 40]

Out[1051]=

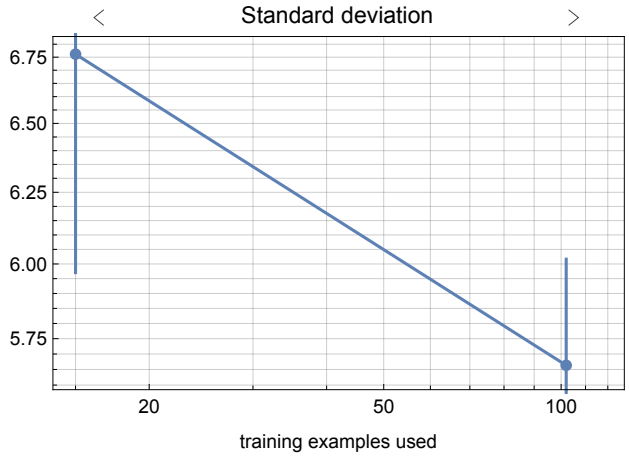


Out[1052]= 16.9357

Predictor information

Data type	Mixed (number: 10)
Standard deviation	5.66 ± 0.35
Method	RandomForest
Single evaluation time	3.55 ms/example
Batch evaluation speed	44.2 examples/ms
Loss	3.19 ± 0.061
Model memory	235. kB
Training examples used	128 examples
Training time	2.14 s

Out[1053]=

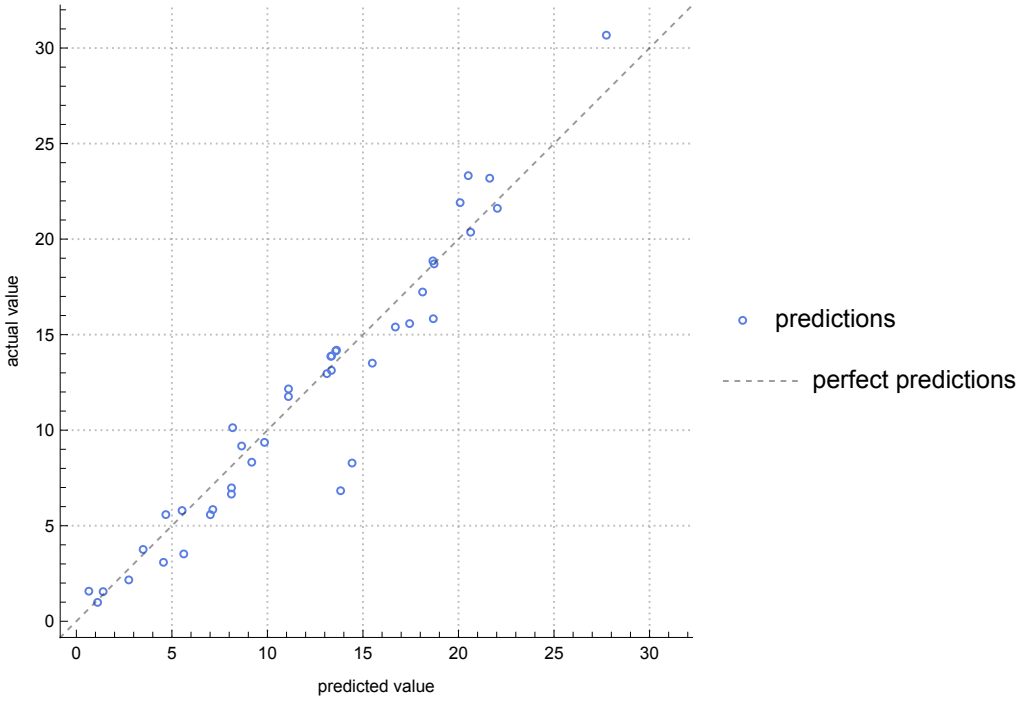


In[1045]:=
```
pXGTv200 = Predict[finalTrain200, Method -> "GradientBoostedTrees"]
pmXGTv200 = PredictorMeasurements[pXGTv200, finaltest200]
pmXGTv200["ComparisonPlot"]
pmXGTv200["MeanSquare"]
Information[pXGTv200]
```

Out[1045]= PredictorFunction[Input type: Mixed (number: 10) Method: GradientBoostedTrees]

Out[1046]= PredictorMeasurementsObject[Predictor: GradientBoostedTrees Number of test examples: 40]

Out[1047]=

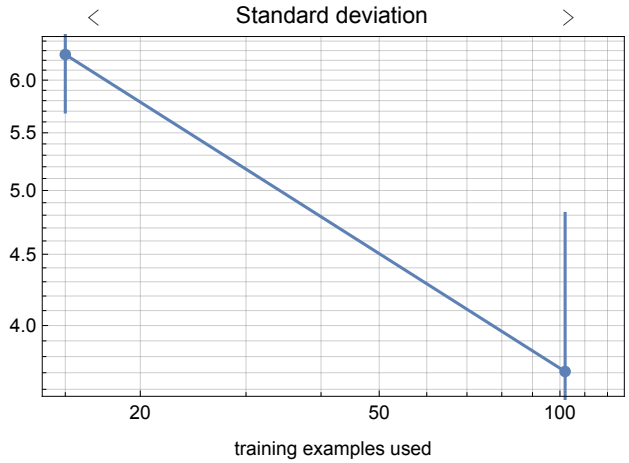


Out[1048]= 3.77753

Predictor information

Data type	Mixed (number: 10)
Standard deviation	3.71 ± 1.1
Method	GradientBoostedTrees
Single evaluation time	3.78 ms/example
Batch evaluation speed	43.4 examples/ms
Loss	64.2 ± 29.
Model memory	389. kB
Training examples used	128 examples
Training time	1.4 s

Out[1049]=



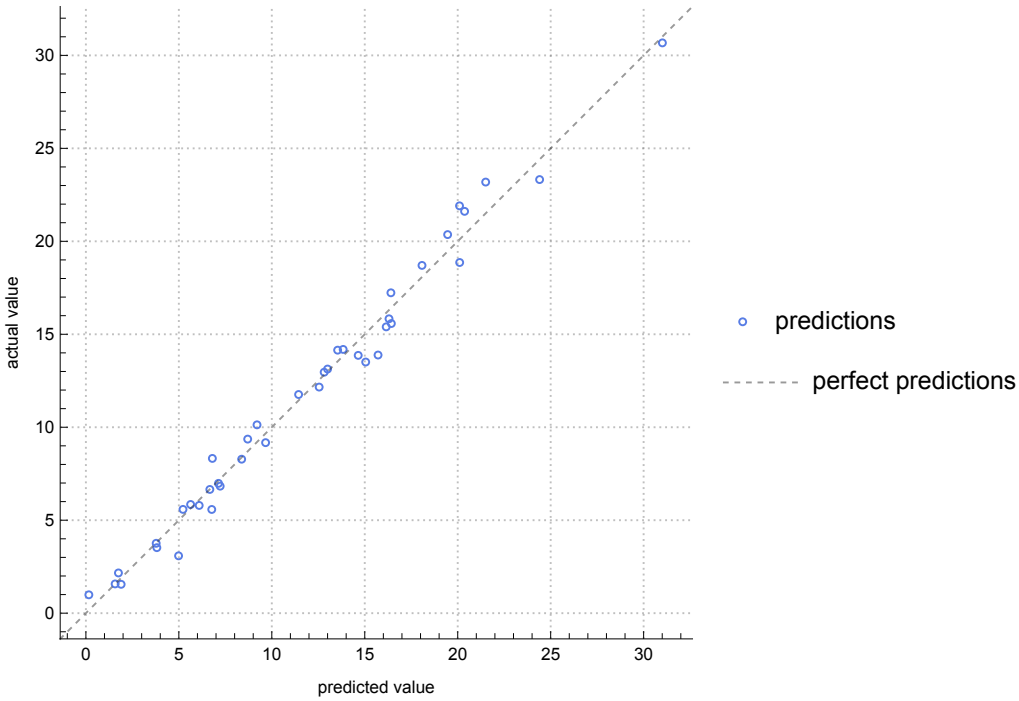
In[1034]:=
```
pNNv200 = Predict[finalTrain200, Method →
    {"NeuralNetwork", "NetworkDepth" → 3, "NetworkType" → "FullyConnected",
     "L2Regularization" → 0.05, MaxTrainingRounds → 16 000}]
```

Out[1034]= PredictorFunction[Input type: Mixed (number: 10) Method: NeuralNetwork]

In[1041]:=
```
pmNNv200 = PredictorMeasurements[pNNv200, finaltest200]
pmNNv200["ComparisonPlot"]
pmNNv200["MeanSquare"]
Information[pNNv200]
```

Out[1041]= PredictorMeasurementsObject[Predictor: NeuralNetwork Number of test examples: 40]

Out[1042]=

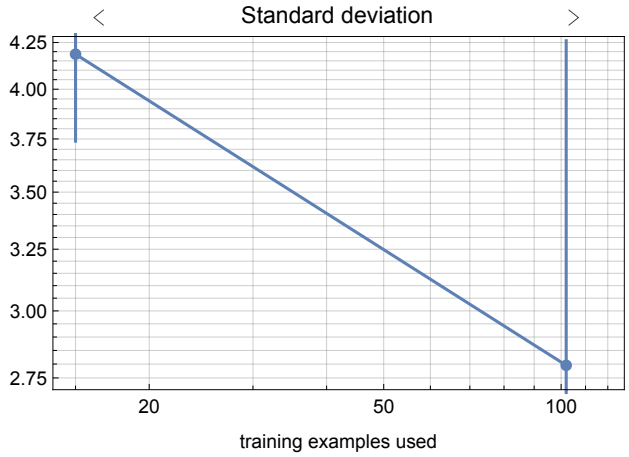


Out[1043]= 0.785093

Predictor information

Data type	Mixed (number: 10)
Standard deviation	2.80 ± 1.5
Method	NeuralNetwork
Single evaluation time	2.34 ms/example
Batch evaluation speed	49.7 examples/ms
Loss	2.30 ± 0.52
Model memory	253. kB
Training examples used	128 examples
Training time	1 min 33 s

Out[1044]=

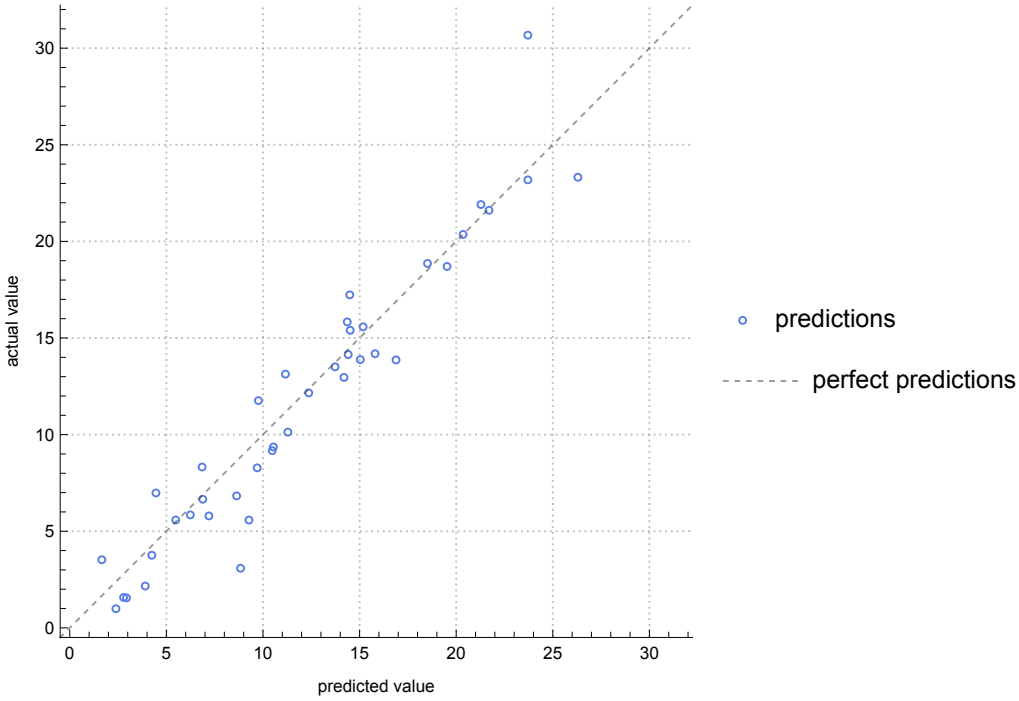


In[1054]:=
```
pGPEv200 = Predict[finalTrain200, Method → "GaussianProcess"]
pmGPEv200 = PredictorMeasurements[pGPEv200, finaltest200]
pmGPEv200["ComparisonPlot"]
pmGPEv200["MeanSquare"]
Information[pGPEv200]
```

Out[1054]= PredictorFunction[Input type: Mixed (number: 10) Method: GaussianProcess]

Out[1055]= PredictorMeasurementsObject[Predictor: GaussianProcess Number of test examples: 40]

Out[1056]=

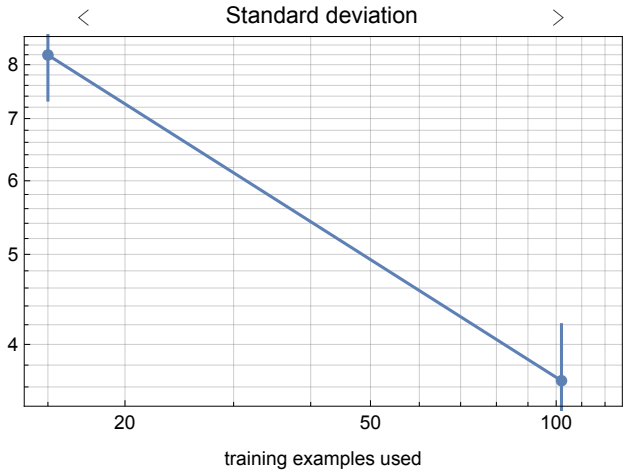


Out[1057]= 4.28377

Out[1058]=

Predictor information

Data type	Mixed (number: 10)
Standard deviation	3.66 ± 0.54
Method	GaussianProcess
Single evaluation time	3.59 ms/example
Batch evaluation speed	30.9 examples/ms
Loss	2.63 ± 0.083
Model memory	265. kB
Training examples used	128 examples
Training time	2.02 s

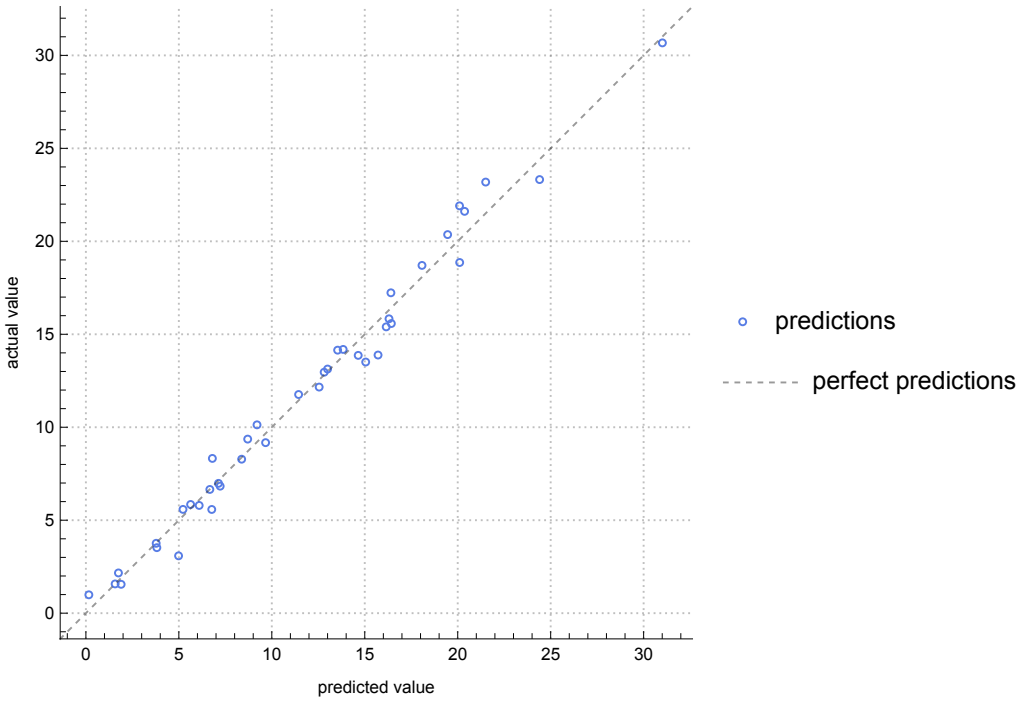


In[1059]:=
```
pNNv200l3 = Predict[finalTrain200, Method →
    {"NeuralNetwork", "NetworkDepth" → 3, "NetworkType" → "FullyConnected",
     "L2Regularization" → 0.05, MaxTrainingRounds → 24 000}]
pmNNv200l3 = PredictorMeasurements[pNNv200l3, finaltest200]
pmNNv200l3["ComparisonPlot"]
pmNNv200l3["MeanSquare"]
Information[pNNv200l3]
```

Out[1059]= PredictorFunction[Input type: Mixed (number: 10) Method: NeuralNetwork]

Out[1060]= PredictorMeasurementsObject[Predictor: NeuralNetwork Number of test examples: 40]

Out[1061]=

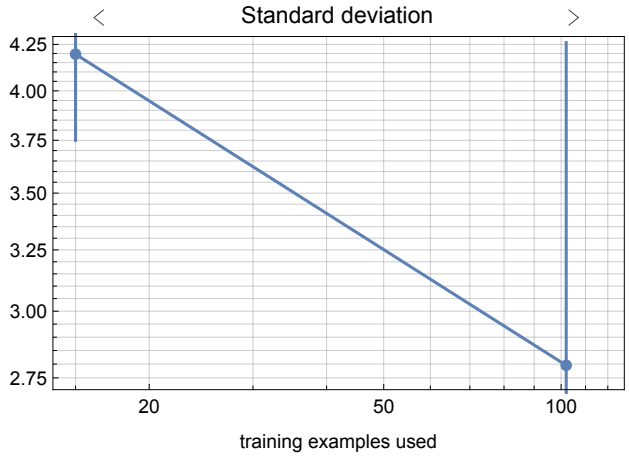


Out[1062]= 0.785093

Predictor information

Data type	Mixed (number: 10)
Standard deviation	2.80 ± 1.5
Method	NeuralNetwork
Single evaluation time	2.19 ms/example
Batch evaluation speed	44.8 examples/ms
Loss	2.30 ± 0.52
Model memory	253. kB
Training examples used	128 examples
Training time	2 min 13 s

Out[1063]=

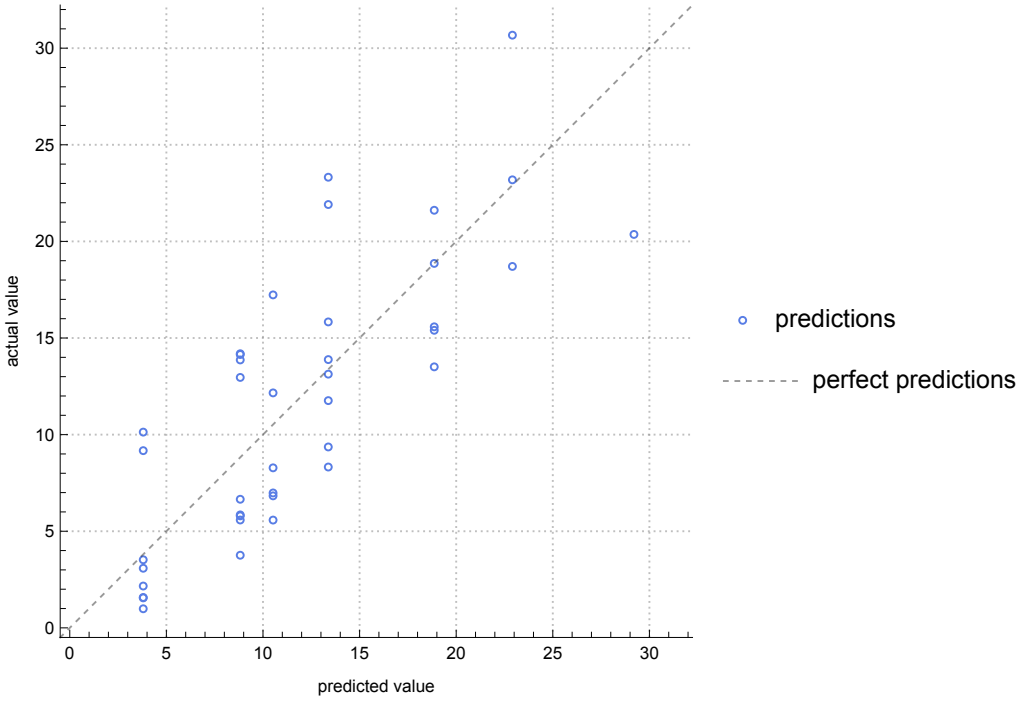


In[1064]:=
```
pDTv200 = Predict[finalTrain200, Method -> "DecisionTree"]
pmDTv200 = PredictorMeasurements[pDTv200, finaltest200]
pmDTv200["ComparisonPlot"]
pmDTv200["MeanSquare"]
Information[pDTv200]
```

Out[1064]= PredictorFunction[Input type: Mixed (number: 10) Method: DecisionTree]

Out[1065]= PredictorMeasurementsObject[Predictor: DecisionTree Number of test examples: 40]

Out[1066]=

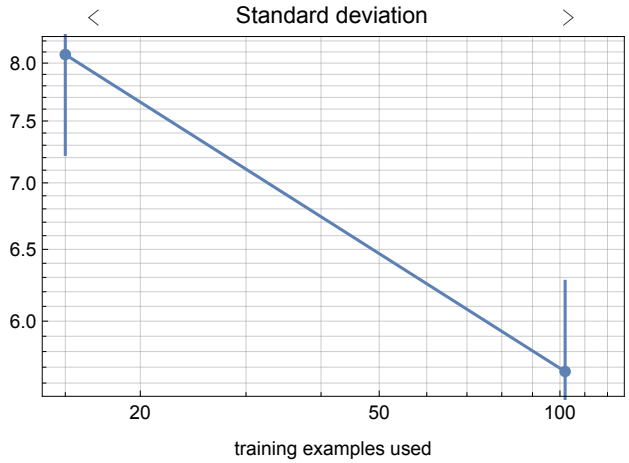


Out[1067]= 19.7339

Out[1068]=

Predictor information

Data type	Mixed (number: 10)
Standard deviation	5.67 ± 0.60
Method	DecisionTree
Single evaluation time	1.31 ms/example
Batch evaluation speed	378. examples/ms
Loss	3.29 ± 0.19
Model memory	143. kB
Training examples used	128 examples
Training time	332. ms

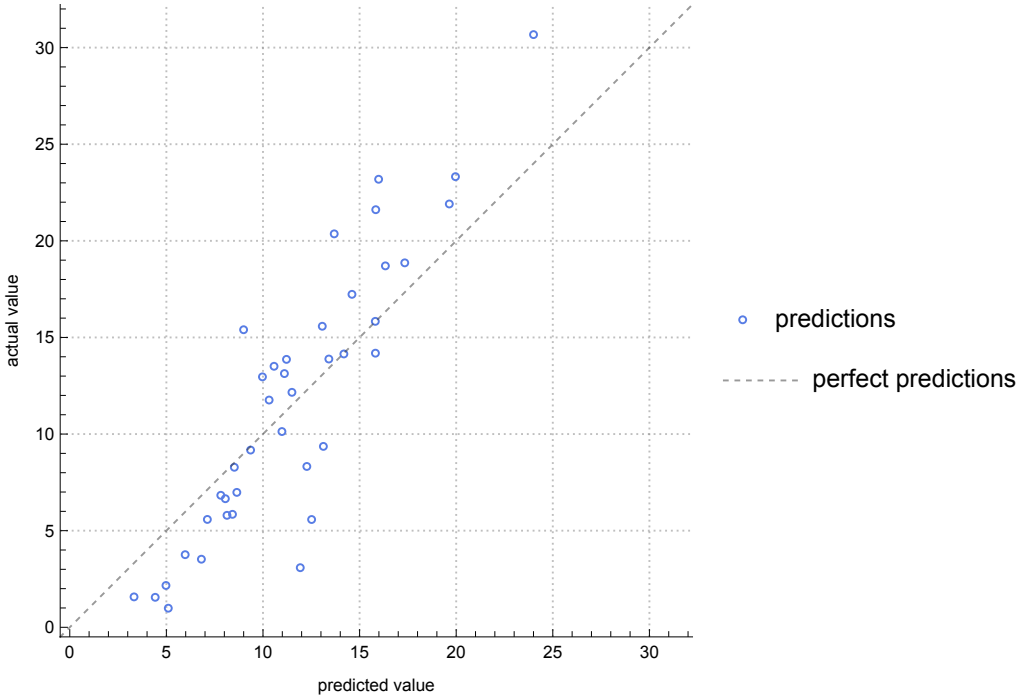


In[1069]:=
```
pNearestv200 = Predict[finalTrain200, Method -> "NearestNeighbors"]
pmNearestv200 = PredictorMeasurements[pNearestv200, finaltest200]
pmNearestv200["ComparisonPlot"]
pmNearestv200["MeanSquare"]
Information[pNearestv200]
```

Out[1069]= PredictorFunction[Input type: Mixed (number: 10) Method: NearestNeighbors]

Out[1070]= PredictorMeasurementsObject[Predictor: NearestNeighbors Number of test examples: 40]

Out[1071]=

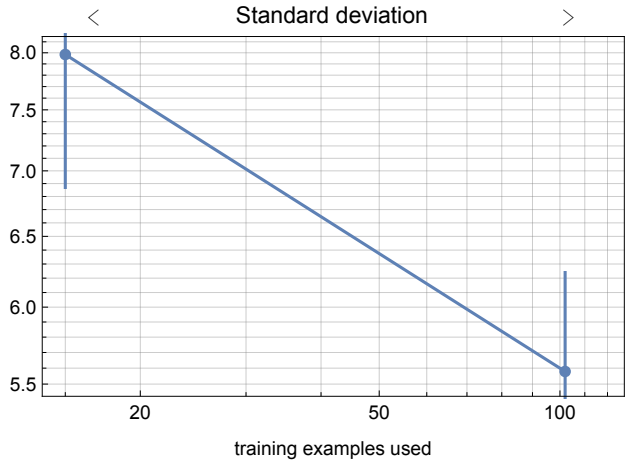


Out[1072]= 12.8954

Predictor information

Data type	Mixed (number: 10)
Standard deviation	5.58 ± 0.66
Method	NearestNeighbors
Single evaluation time	1.37 ms/example
Batch evaluation speed	89.2 examples/ms
Loss	3.21 ± 0.13
Model memory	155. kB
Training examples used	128 examples
Training time	333. ms

Out[1073]=

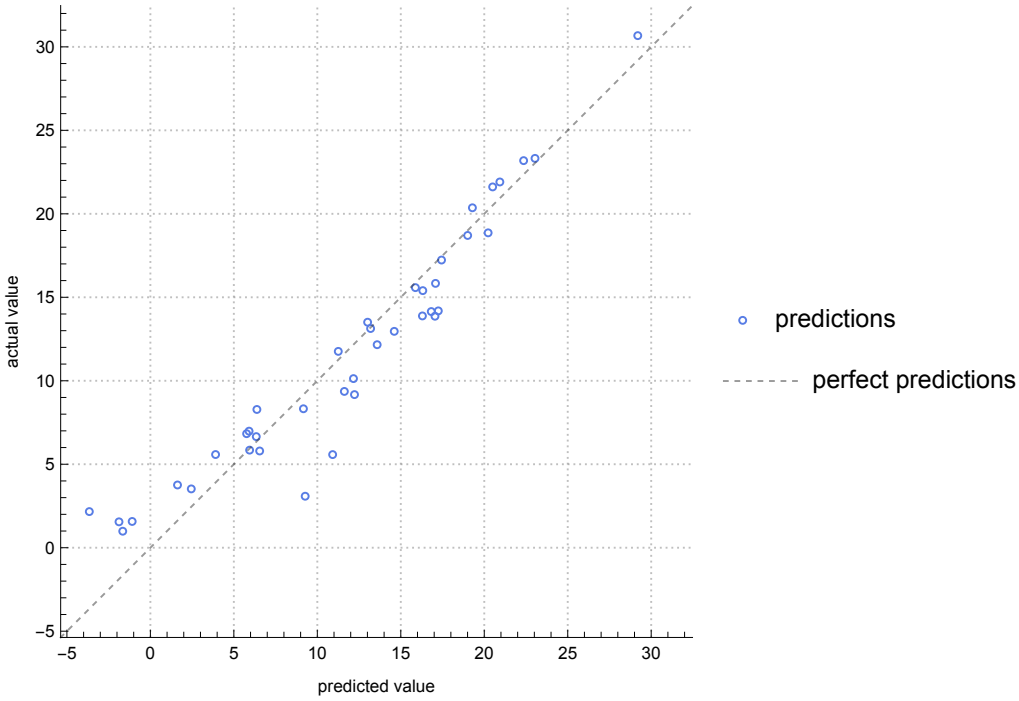


In[1074]:=
```
pLRv200 = Predict[finalTrain200, Method -> "LinearRegression"]
pmLRv200 = PredictorMeasurements[pLRv200, finaltest200]
pmLRv200["ComparisonPlot"]
pmLRv200["MeanSquare"]
Information[pLRv200]
```

Out[1074]= PredictorFunction[Input type: Mixed (number: 10) Method: LinearRegression]

Out[1075]= PredictorMeasurementsObject[Predictor: LinearRegression Number of test examples: 40]

Out[1076]=

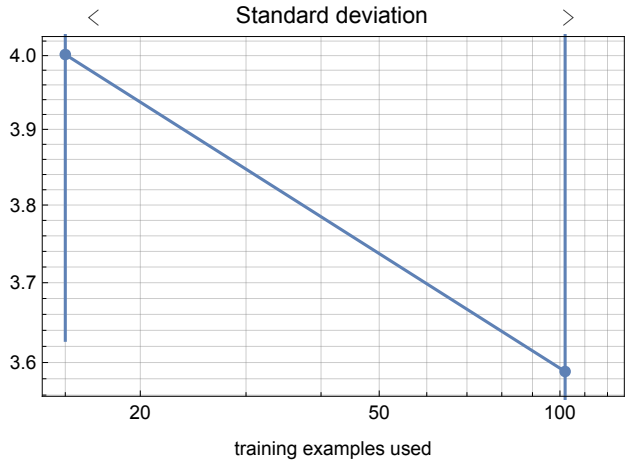


Out[1077]= 5.23679

Out[1078]=

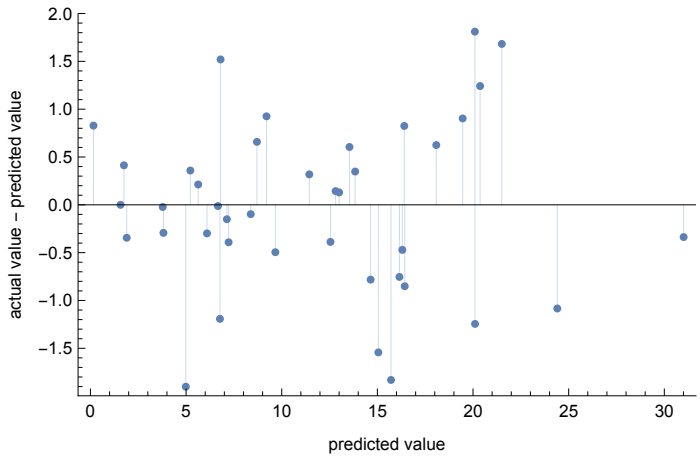
Predictor information

Data type	Mixed (number: 10)
Standard deviation	3.59 ± 0.46
Method	LinearRegression
Single evaluation time	1.53 ms/example
Batch evaluation speed	289. examples/ms
Loss	2.58 ± 0.086
Model memory	257. kB
Training examples used	128 examples
Training time	1.47 s



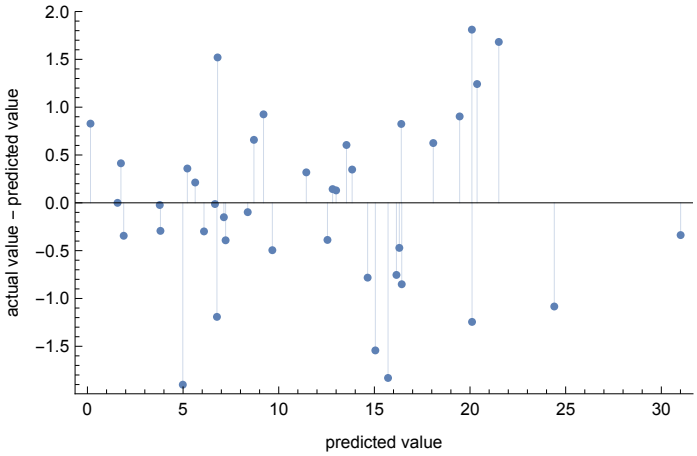
In[1079]:=
```
pmNNv200["ResidualPlot"]
```

Out[1079]=



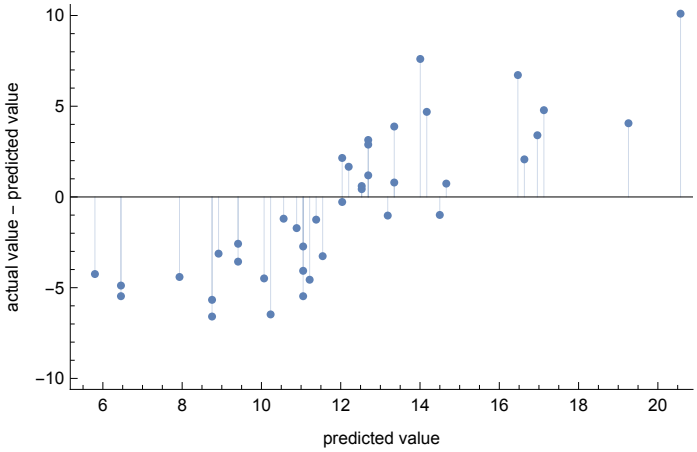
In[1080]:=
```
pmNNv200l3["ResidualPlot"]
```

Out[1080]=



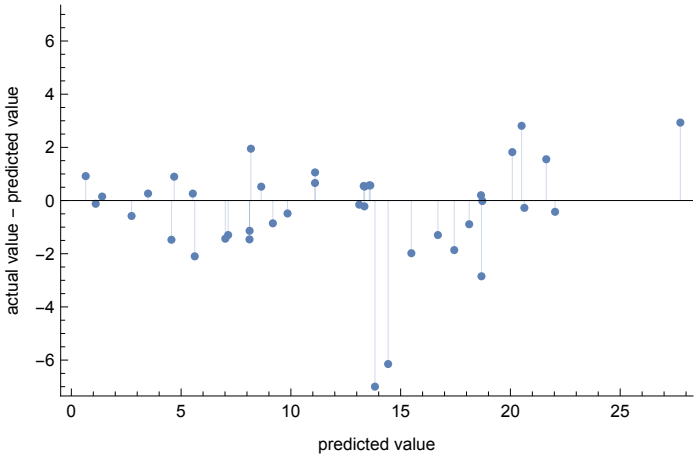
In[1090]:=
```
pmRFv200["ResidualPlot"]
```

Out[1090]=



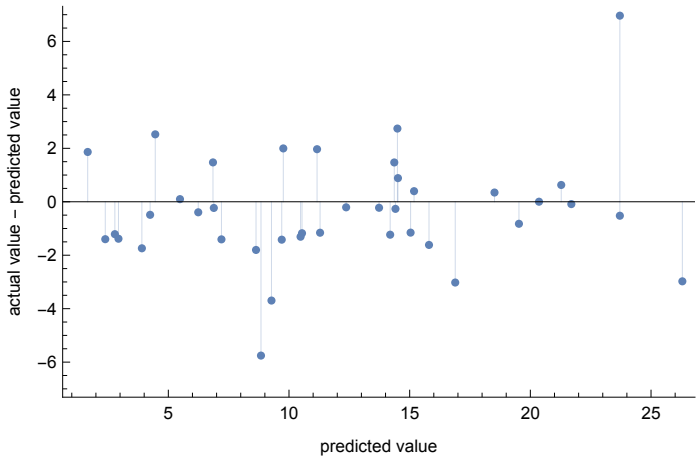
In[1081]:=
```
pmXGTv200["ResidualPlot"]
```

Out[1081]=



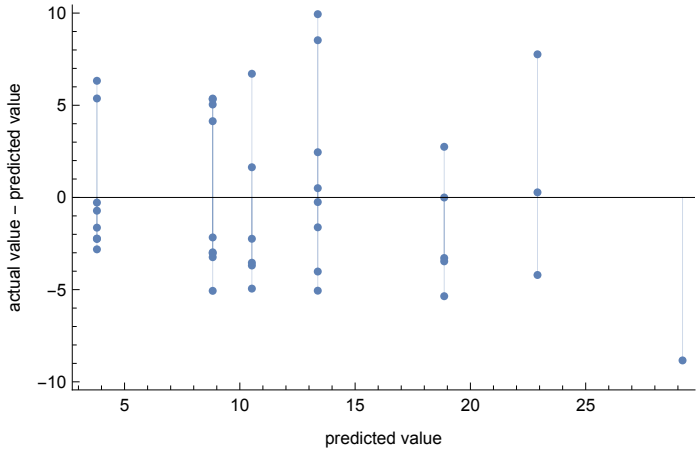
In[1082]:= **pmGPEv200["ResidualPlot"]**

Out[1082]=



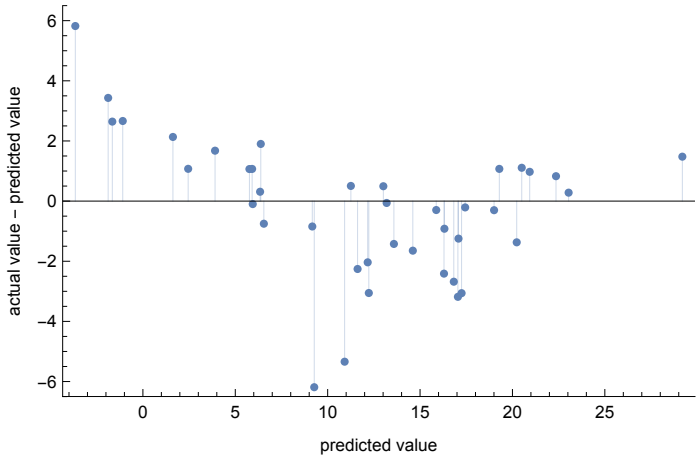
In[1083]:= **pmDTv200["ResidualPlot"]**

Out[1083]=



In[1084]:= **pmLRv200["ResidualPlot"]**

Out[1084]=



MeanSquare

In[1128]:=
```
pmNNv200["MeanSquare"]
pmDTv200["MeanSquare"]
pmLRv200["MeanSquare"]
pmRFv200["MeanSquare"]
pmXGTv200["MeanSquare"]
pmGPEv200["MeanSquare"]
pmLRv200["MeanSquare"]
pmNearestv200["MeanSquare"]
```

Out[1128]= 0.785093

Out[1129]= 19.7339

Out[1130]= 5.23679

Out[1131]= 16.9357

Out[1132]= 3.77753

Out[1133]= 4.28377

Out[1134]= 5.23679

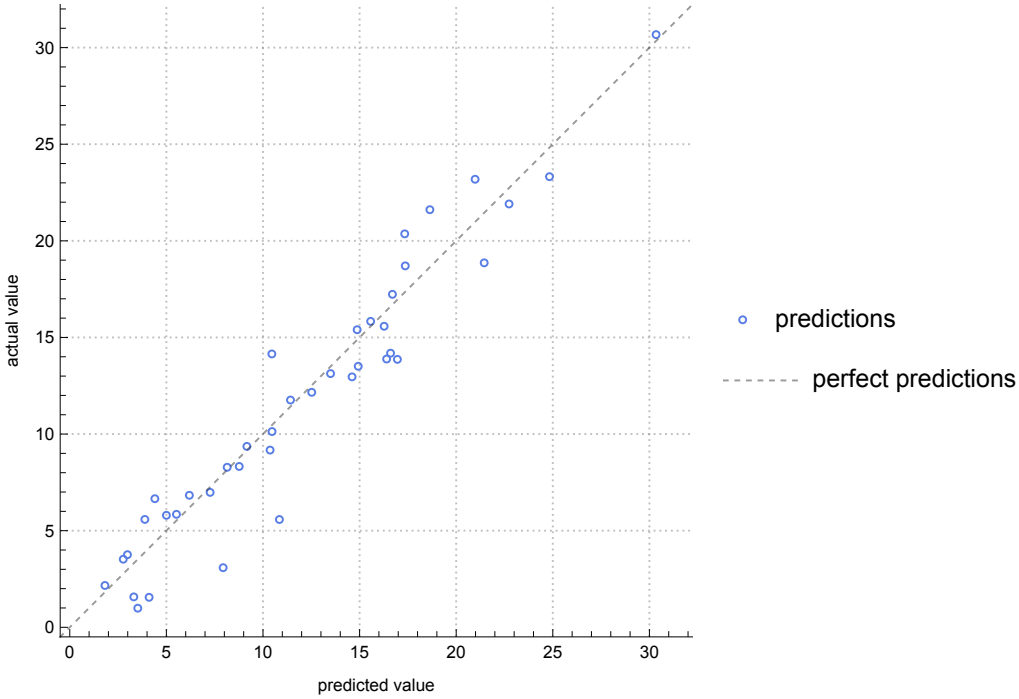
Out[1135]= 12.8954

In[1085]:=
```
pNNnoReg200 =
 Predict[finalTrain200, Method → {"NeuralNetwork", "NetworkDepth" → 3,
    "NetworkType" → "FullyConnected", MaxTrainingRounds → 24000}]
pmNNnoReg200 = PredictorMeasurements[pNNnoReg200, finaltest200]
pmNNnoReg200["ComparisonPlot"]
pmNNnoReg200["MeanSquare"]
Information[pNNnoReg200]
```

Out[1085]= PredictorFunction[ Input type: Mixed (number: 10)
Method: NeuralNetwork]

Out[1086]= PredictorMeasurementsObject[ Predictor: NeuralNetwork
Number of test examples: 40]

Out[1087]=

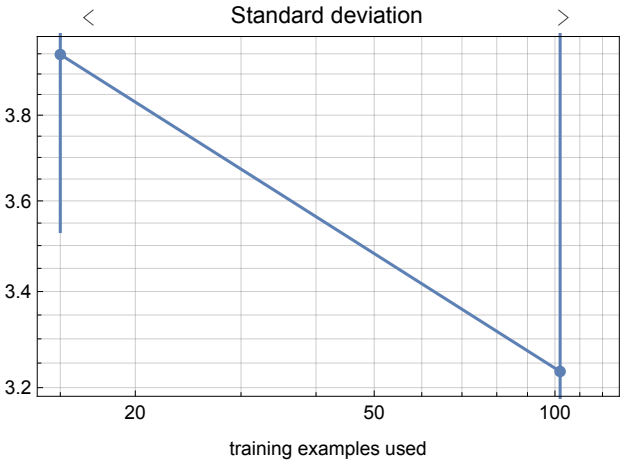


Out[1088]= 3.8887

Predictor information

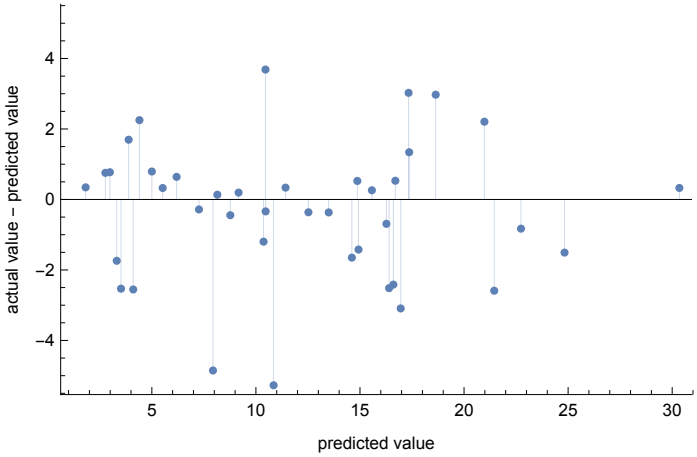
Data type	Mixed (number: 10)
Standard deviation	3.23 ± 1.2
Method	NeuralNetwork
Single evaluation time	2.15 ms/example
Batch evaluation speed	46.2 examples/ms
Loss	2.55 ± 0.38
Model memory	249. kB
Training examples used	128 examples
Training time	1 min 26 s

Out[1089]=



In[1091]:=
```
pmNNnoReg200["ResidualPlot"]
```

Out[1091]=

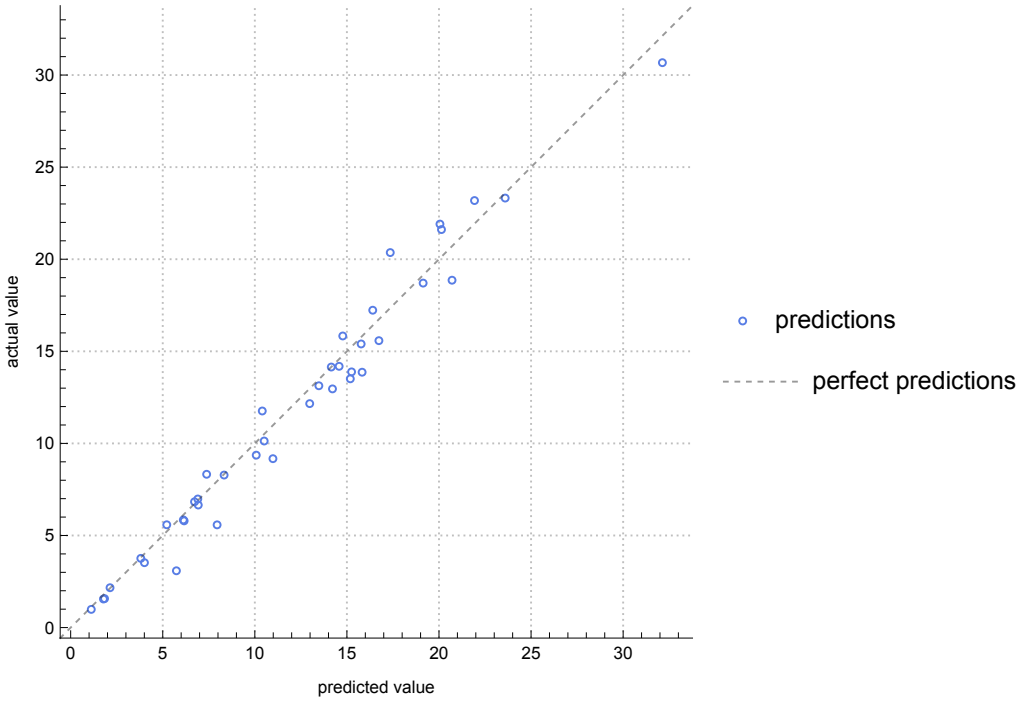


In[1092]:=
```
pNN2layer200 = Predict[finalTrain200, Method →
    {"NeuralNetwork", "NetworkDepth" → 2, "NetworkType" → "FullyConnected",
     "L2Regularization" → 0.05, MaxTrainingRounds → 24 000}]
pmNN2layer200 = PredictorMeasurements[pNN2layer200, finaltest200]
pmNN2layer200["ComparisonPlot"]
pmNN2layer200["MeanSquare"]
pmNN2layer200["ResidualPlot"]
Information[pNN2layer200]
```

Out[1092]= PredictorFunction[Input type: Mixed (number: 10) Method: NeuralNetwork]

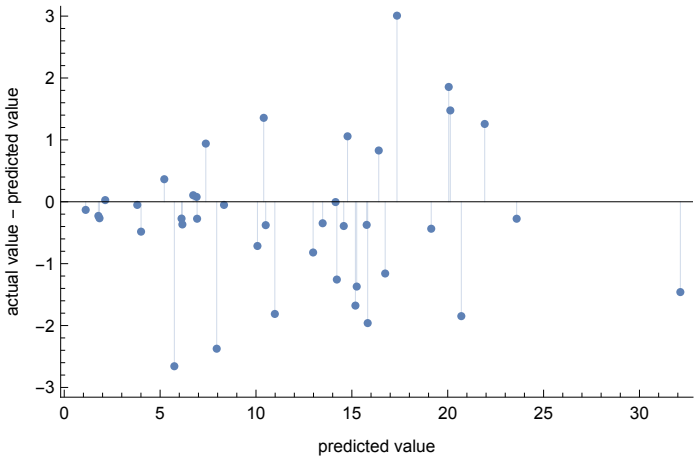
Out[1093]= PredictorMeasurementsObject[Predictor: NeuralNetwork Number of test examples: 40]

Out[1094]=



Out[1095]= 1.41434

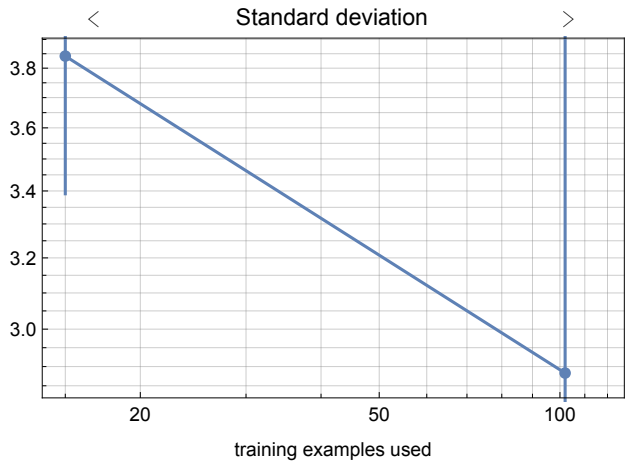
Out[1096]=



Predictor information

Data type	Mixed (number: 10)
Standard deviation	2.88 ± 1.4
Method	NeuralNetwork
Single evaluation time	2.19 ms/example
Batch evaluation speed	53.1 examples/ms
Loss	2.49 ± 0.69
Model memory	234. kB
Training examples used	128 examples
Training time	2 min 6 s

Out[1097]=

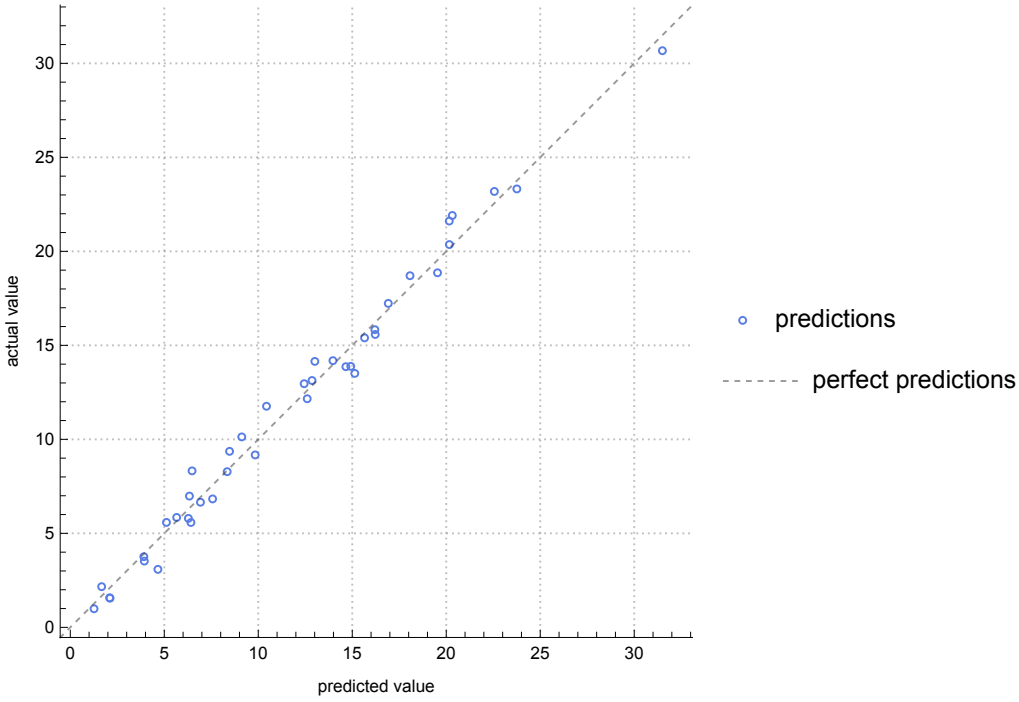


In[1136]:=
```
pNN4layer200 = Predict[finalTrain200, Method →
    {"NeuralNetwork", "NetworkDepth" → 4, "NetworkType" → "FullyConnected",
     "L2Regularization" → 0.05, MaxTrainingRounds → 24 000}]
pmNN4layer200 = PredictorMeasurements[pNN4layer200, finaltest200]
pmNN4layer200["ComparisonPlot"]
pmNN4layer200["MeanSquare"]
pmNN4layer200["ResidualPlot"]
PredictorInformation[pNN4layer200]
```

Out[1136]= PredictorFunction[Input type: Mixed (number: 10) Method: NeuralNetwork]

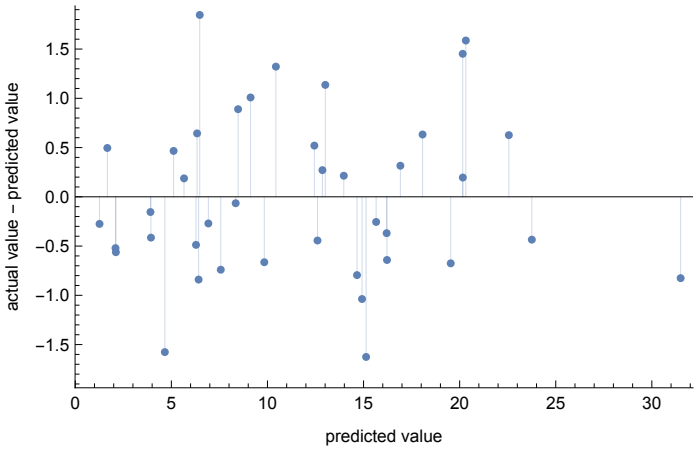
Out[1137]= PredictorMeasurementsObject[Predictor: NeuralNetwork Number of test examples: 40]

Out[1138]=



Out[1139]= 0.673797

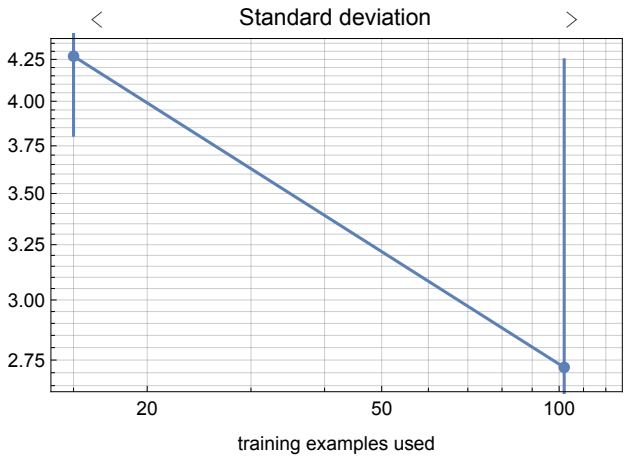
Out[1140]=



Out[1141]=

Predictor information

Data type	Mixed (number: 10)
Standard deviation	2.72 ± 1.5
Method	NeuralNetwork
Single evaluation time	3.3 ms/example
Batch evaluation speed	38.7 examples/ms
Loss	1.94 ± 0.42
Model memory	269. kB
Training examples used	128 examples
Training time	2 min 26 s



In[1142]:=
```
netSimple200 = NetChain[
  {BatchNormalizationLayer[], Tanh, 50, BatchNormalizationLayer[], Tanh, 1}]
trainedNetSimple200 = NetTrain[netSimple200, finalTrain200, All,
  ValidationSet → finaltest200, MaxTrainingRounds → 15 000,
  Method → {"ADAM", "LearningRate" → 0.0003, "L2Regularization" → 0.03}]
```

Out[1142]=

NetChain[uninitialized

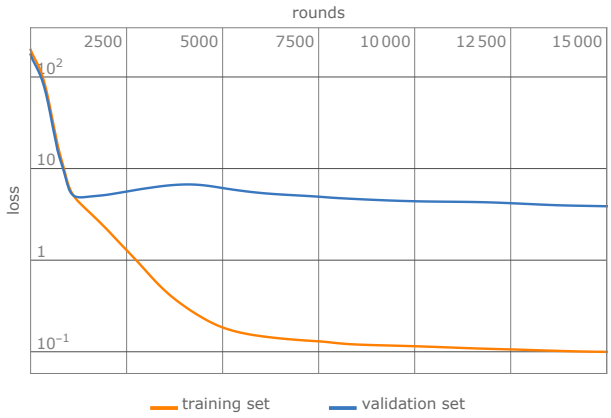
Input port:	array of rank ≥ 1
Output port:	vector (size: 1)
Number of layers:	6

]

Out[1143]=

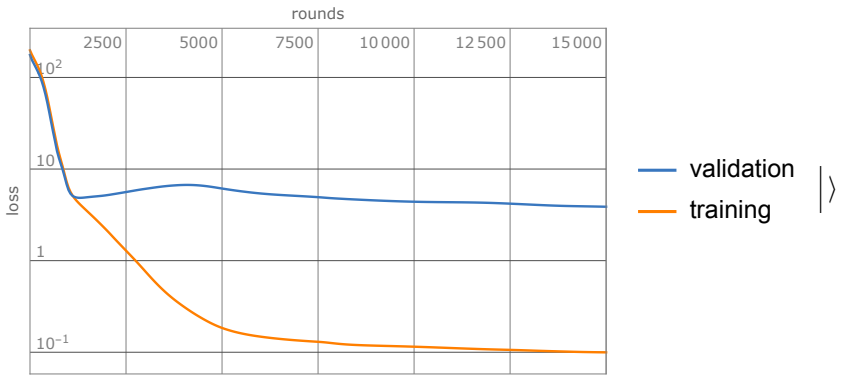
NetTrain Results

summary	batches: 30 000, rounds: 15 000, time: 32s, examples/s: 61 616
data	training examples: 128, validation examples: 40, processed examples: 1 920 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 9.97×10^{-2}
validation	loss: 3.88



In[1144]:=
```
trainedNetSimple200["FinalPlots"]
trainedNetSimple200["TotalTrainingTime"]
trainedNetSimple200["RoundMeasurements"]
best = trainedNetSimple200["BestValidationRound"]
trainedNetSimple200["ValidationLossList"][[best]]
NetInformation[trainedNetSimple200["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNetSimple200["TrainedNet"], "SummaryGraphic"]
```

Out[1144]=



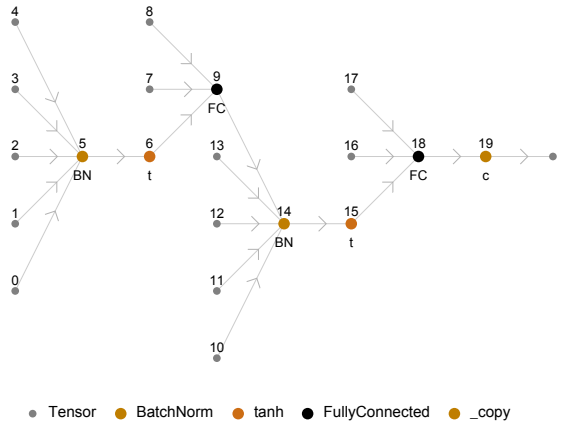
Out[1145]= 31.1608

Out[1146]= ⟨| Loss → 0.0997103 |⟩

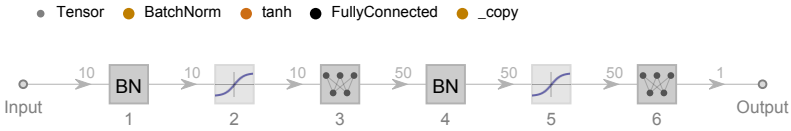
Out[1147]= 14 994

Out[1148]= 3.87713

Out[1149]=



Out[1150]=



In[1151]:=
```
netSimple200v2 = NetChain[{BatchNormalizationLayer[], Tanh, 25,
    BatchNormalizationLayer[], Tanh, 1}, "Input" → 10, "Output" → "Scalar"]
trainedNetSimple200v2 = NetTrain[netSimple200v2, finalTrain200,
  All, ValidationSet → finaltest200, MaxTrainingRounds → 15 000,
  Method → {"ADAM", "LearningRate" → 0.0002, "L2Regularization" → 0.05}]
```

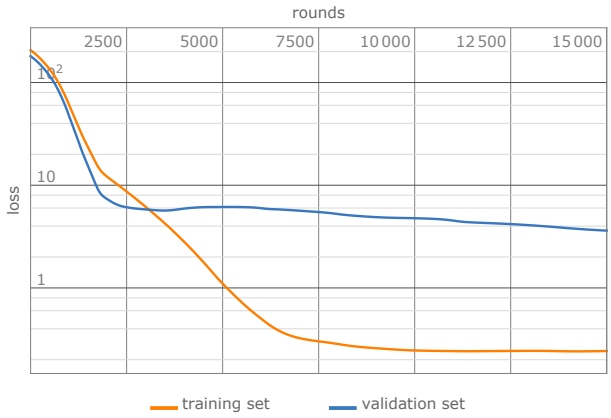
Out[1151]=

NetChain[uninitialized | Input port: vector (size: 10) | Output port: scalar | Number of layers: 6]

Out[1152]=

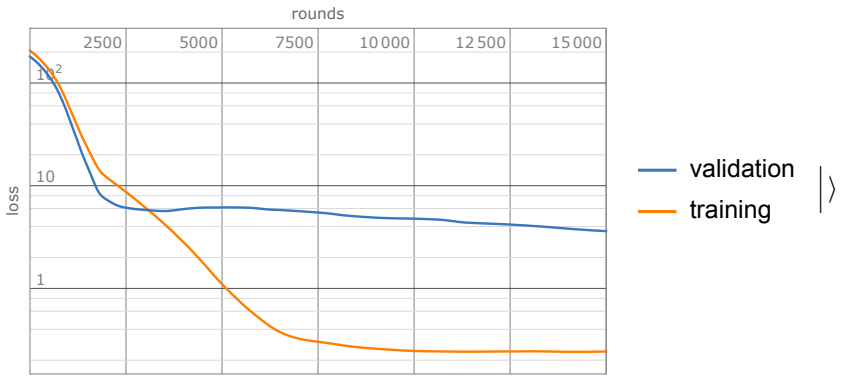
NetTrain Results

summary	batches: 30 000, rounds: 15 000, time: 32s, examples/s: 61 441
data	training examples: 128, validation examples: 40, processed examples: 1 920 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 2.43×10^{-1}
validation	loss: 3.61



In[1153]:=
```
trainedNetSimple200v2["FinalPlots"]
trainedNetSimple200v2["TotalTrainingTime"]
trainedNetSimple200v2["RoundMeasurements"]
best = trainedNetSimple200v2["BestValidationRound"]
trainedNetSimple200v2["ValidationLossList"][[best]]
NetInformation[trainedNetSimple200v2["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNetSimple200v2["TrainedNet"], "SummaryGraphic"]
```

Out[1153]=



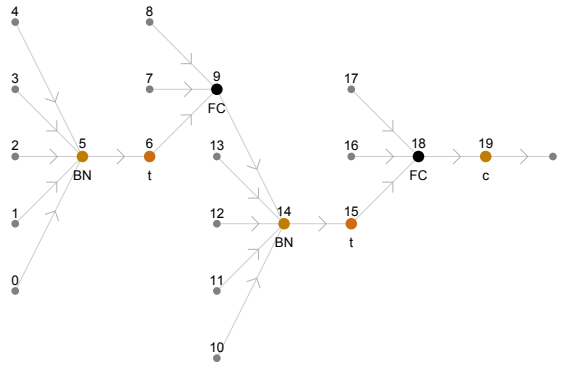
Out[1154]= 31.2493

Out[1155]= ⟨|Loss → 0.242993|⟩

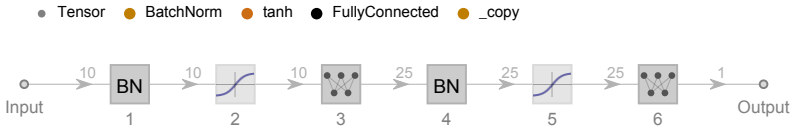
Out[1156]= 14 986

Out[1157]= 3.61068

Out[1158]=



Out[1159]=



In[1160]:=
```
netSimple200v3 = NetChain[{BatchNormalizationLayer[], Tanh, 25, 25, 25, 25, 25,
    BatchNormalizationLayer[], Tanh, 1}, "Input" → 10, "Output" → "Scalar"]
trainedNetSimple200v3 = NetTrain[netSimple200v3, finalTrain200,
  All, ValidationSet → finaltest200, MaxTrainingRounds → 15 000,
  Method → {"ADAM", "LearningRate" → 0.0002, "L2Regularization" → 0.05}]
```

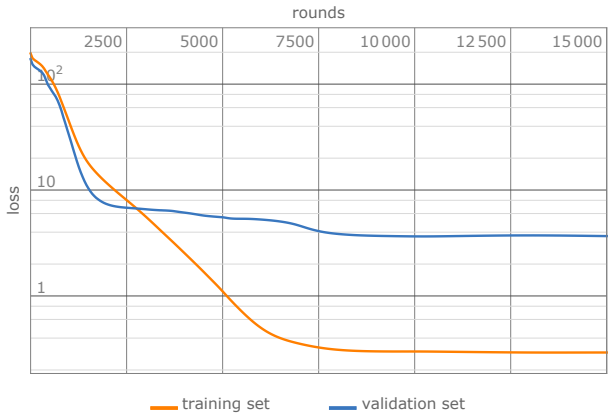
Out[1160]=

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

Out[1161]=

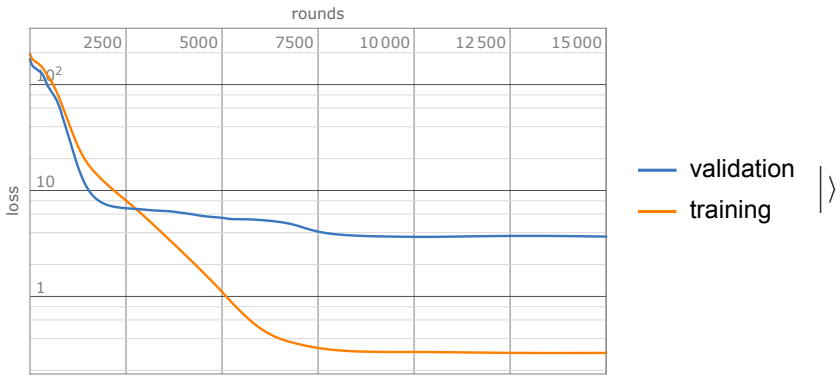
NetTrain Results

summary	batches: 30 000, rounds: 15 000, time: 40s, examples/s: 48 103
data	training examples: 128, validation examples: 40, processed examples: 1 920 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 2.93×10^{-1}
validation	loss: 3.68



In[1162]:=
```
trainedNetSimple200v3["FinalPlots"]
trainedNetSimple200v3["TotalTrainingTime"]
trainedNetSimple200v3["RoundMeasurements"]
best = trainedNetSimple200v3["BestValidationRound"]
trainedNetSimple200v3["ValidationLossList"][[best]]
NetInformation[trainedNetSimple200v3["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNetSimple200v3["TrainedNet"], "SummaryGraphic"]
```

Out[1162]=



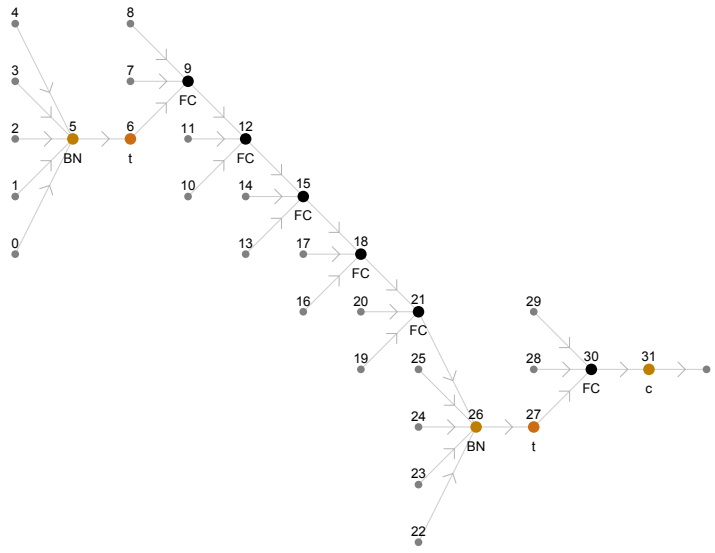
Out[1163]= 39.914

Out[1164]= ⟨| Loss → 0.292506 |⟩

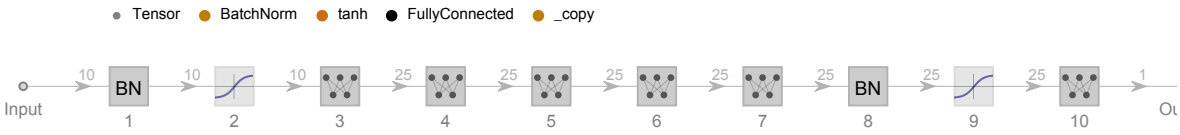
Out[1165]= 10 262

Out[1166]= 3.64528

Out[1167]=



Out[1168]=



In[1169]:=
```
netSimple200v4 = NetChain[{BatchNormalizationLayer[], Tanh, 50, 50, 50, 50, 50,
    BatchNormalizationLayer[], Tanh, 1}, "Input" → 10, "Output" → "Scalar"]
trainedNetSimple200v4 = NetTrain[netSimple200v4, finalTrain200,
  All, ValidationSet → finaltest200, MaxTrainingRounds → 15 000,
  Method → {"ADAM", "LearningRate" → 0.0002, "L2Regularization" → 0.05}]
```

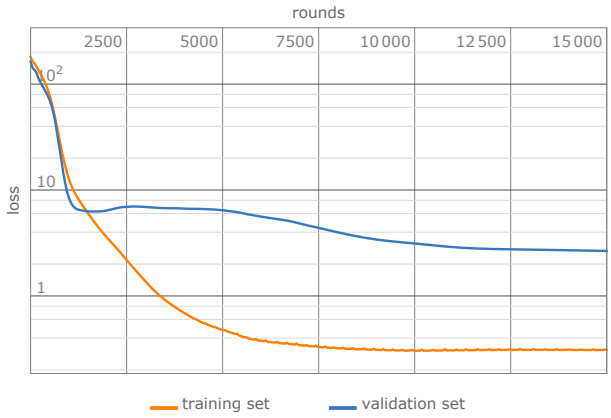
Out[1169]=

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

Out[1170]=

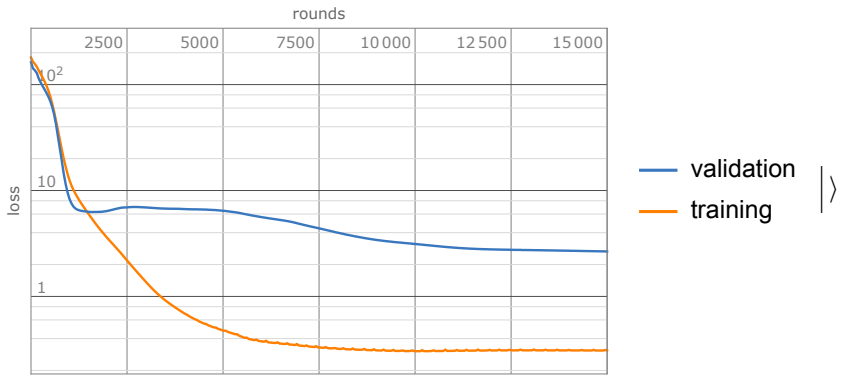
NetTrain Results

summary	batches: 30 000, rounds: 15 000, time: 36s, examples/s: 53 878
data	training examples: 128, validation examples: 40, processed examples: 1 920 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 3.08×10^{-1}
validation	loss: 2.65



In[1171]:=
```
trainedNetSimple200v4["FinalPlots"]
trainedNetSimple200v4["TotalTrainingTime"]
trainedNetSimple200v4["RoundMeasurements"]
best = trainedNetSimple200v4["BestValidationRound"]
trainedNetSimple200v4["ValidationLossList"][[best]]
NetInformation[trainedNetSimple200v4["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNetSimple200v4["TrainedNet"], "SummaryGraphic"]
```

Out[1171]=



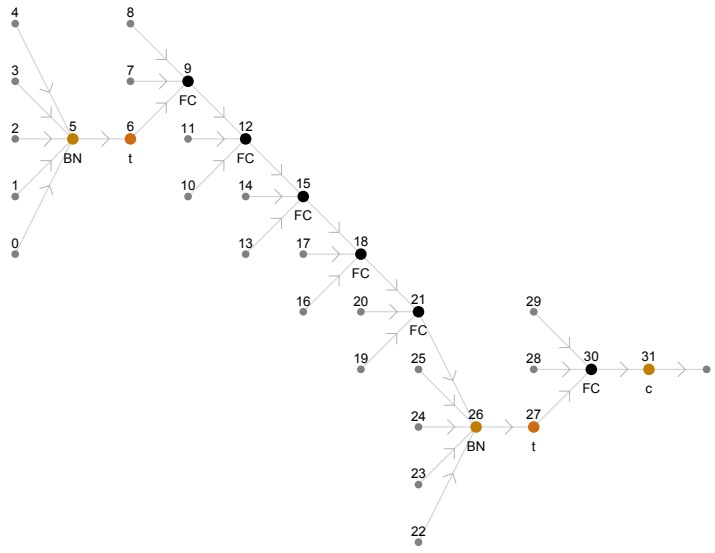
Out[1172]= 35.6362

Out[1173]= ⟨| Loss → 0.308341 |⟩

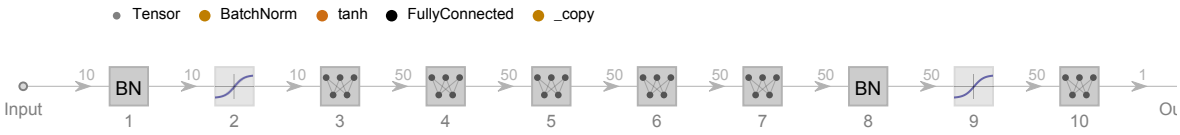
Out[1174]= 14 964

Out[1175]= 2.61222

Out[1176]=



Out[1177]=



In[1178]:=
```
netSimple200v5 = NetChain[{BatchNormalizationLayer[], Tanh, 50, 50, 50, 50, 50, 50,
    50, BatchNormalizationLayer[], Tanh, 1}, "Input" → 10, "Output" → "Scalar"]
trainedNetSimple200v5 = NetTrain[netSimple200v5, finalTrain200,
  All, ValidationSet → finaltest200, MaxTrainingRounds → 15 000,
  Method → {"ADAM", "LearningRate" → 0.0003, "L2Regularization" → 0.05}]
```

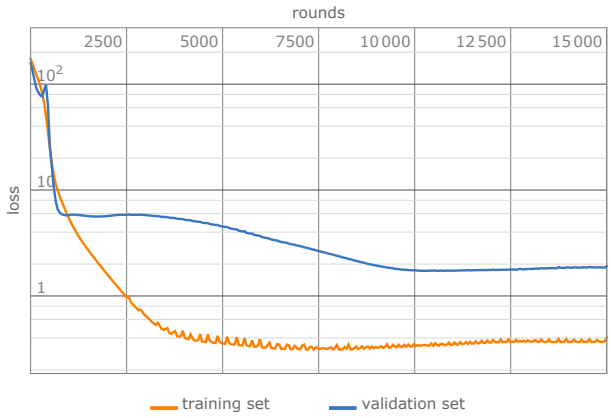
Out[1178]=

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

Out[1179]=

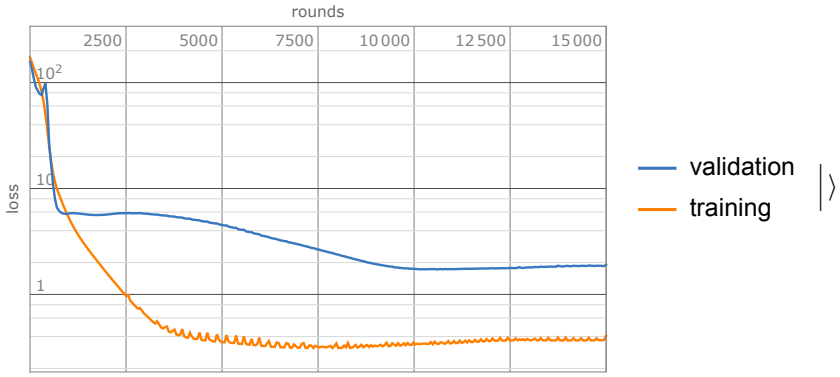
NetTrain Results

summary	batches: 30 000, rounds: 15 000, time: 39s, examples/s: 49 944
data	training examples: 128, validation examples: 40, processed examples: 1 920 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 3.72×10^{-1}
validation	loss: 1.87



In[1180]:=
```
trainedNetSimple200v5["FinalPlots"]
trainedNetSimple200v5["TotalTrainingTime"]
trainedNetSimple200v5["RoundMeasurements"]
best = trainedNetSimple200v5["BestValidationRound"]
trainedNetSimple200v5["ValidationLossList"][[best]]
NetInformation[trainedNetSimple200v5["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNetSimple200v5["TrainedNet"], "SummaryGraphic"]
```

Out[1180]=



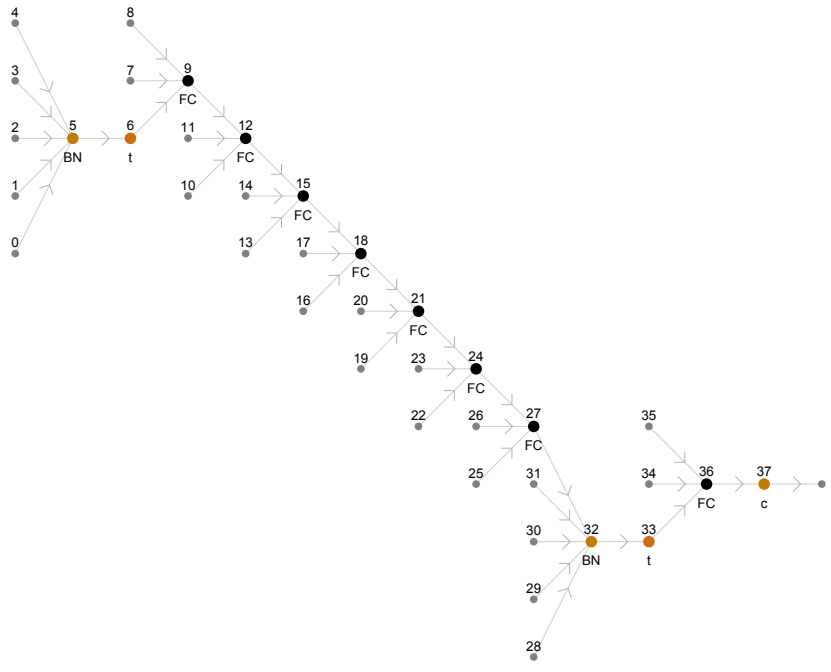
Out[1181]= 38.443

Out[1182]= ⟨| Loss → 0.371582 |⟩

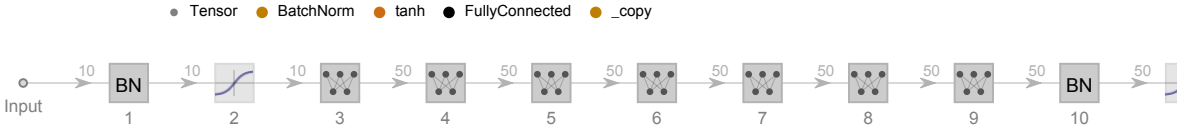
Out[1183]= 10 879

Out[1184]= 1.58206

Out[1185]=



Out[1186]=



In[1187]:=
```
netSimple200v6 =
 NetChain[{BatchNormalizationLayer[], Tanh, 100, 100, 100, 100, 100, 100, 100,
   BatchNormalizationLayer[], Tanh, 1}, "Input" → 10, "Output" → "Scalar"]
trainedNetSimple200v6 = NetTrain[netSimple200v6, finalTrain200,
  All, ValidationSet → finaltest200, MaxTrainingRounds → 15 000,
  Method → {"ADAM", "LearningRate" → 0.0005, "L2Regularization" → 0.05}]
```

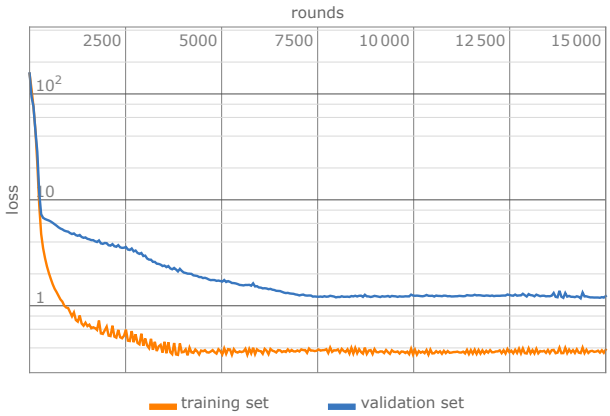
Out[1187]=

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

Out[1188]=

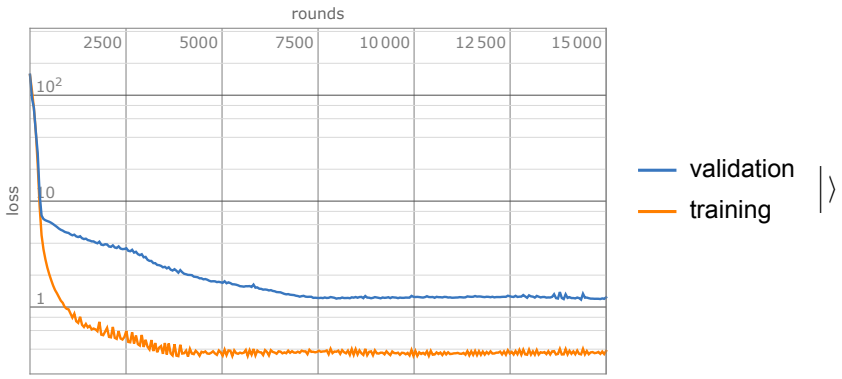
NetTrain Results

summary	batches: 30 000, rounds: 15 000, time: 49s, examples/s: 39 868
data	training examples: 128, validation examples: 40, processed examples: 1 920 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 4.05×10^{-1}
validation	loss: 1.6



In[1189]:=
```
trainedNetSimple200v6["FinalPlots"]
trainedNetSimple200v6["TotalTrainingTime"]
trainedNetSimple200v6["RoundMeasurements"]
best = trainedNetSimple200v6["BestValidationRound"]
trainedNetSimple200v6["ValidationLossList"][[best]]
NetInformation[trainedNetSimple200v6["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNetSimple200v6["TrainedNet"], "SummaryGraphic"]
```

Out[1189]=



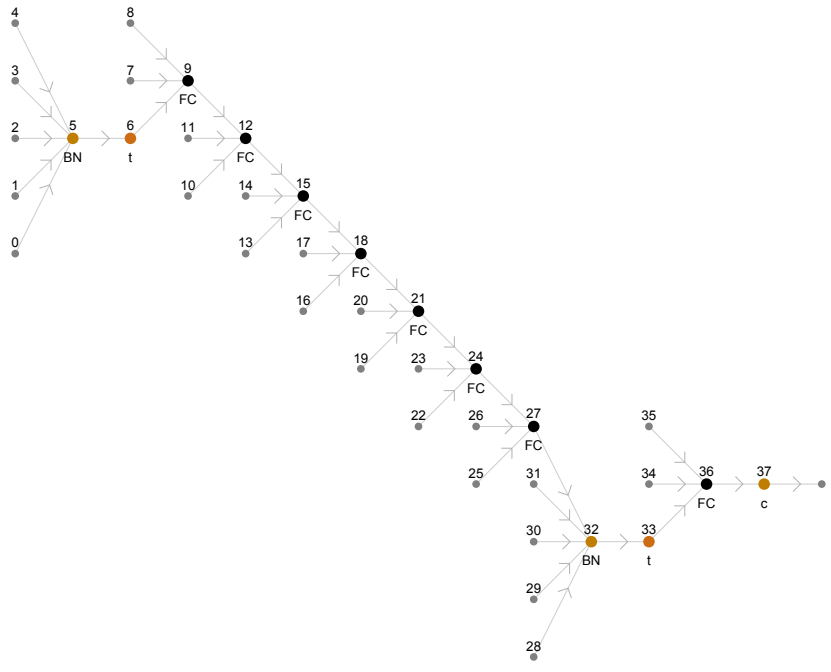
Out[1190]= 48.1588

Out[1191]= ⟨| Loss → 0.40492 |⟩

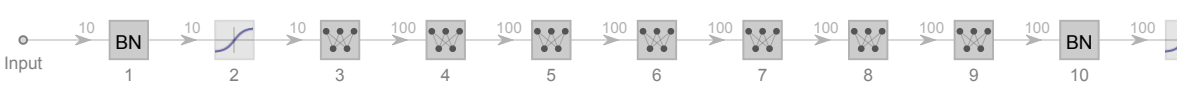
Out[1192]= 14 395

Out[1193]= 0.964567

Out[1194]=



Out[1195]=



In[1196]:=
```
netSimple200v7 = NetChain[{BatchNormalizationLayer[], Tanh, 250, 250, 250,
    BatchNormalizationLayer[], Tanh, 1}, "Input" → 10, "Output" → "Scalar"]
trainedNetSimple200v7 = NetTrain[netSimple200v7, finalTrain200,
  All, ValidationSet → finaltest200, MaxTrainingRounds → 15 000,
  Method → {"ADAM", "LearningRate" → 0.0005, "L2Regularization" → 0.05}]
```

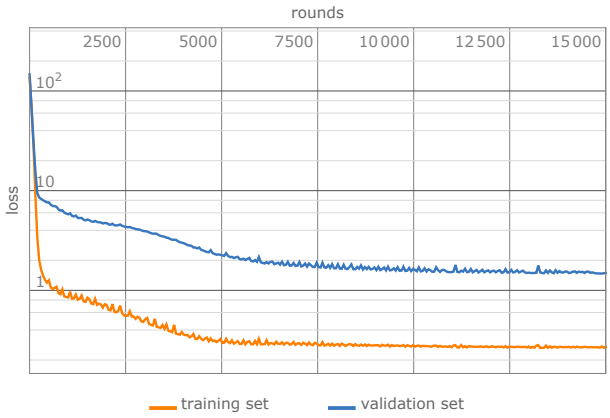
Out[1196]=

NetChain[uninitialized | Input port: vector (size: 10) | Output port: scalar | Number of layers: 8]

Out[1197]=

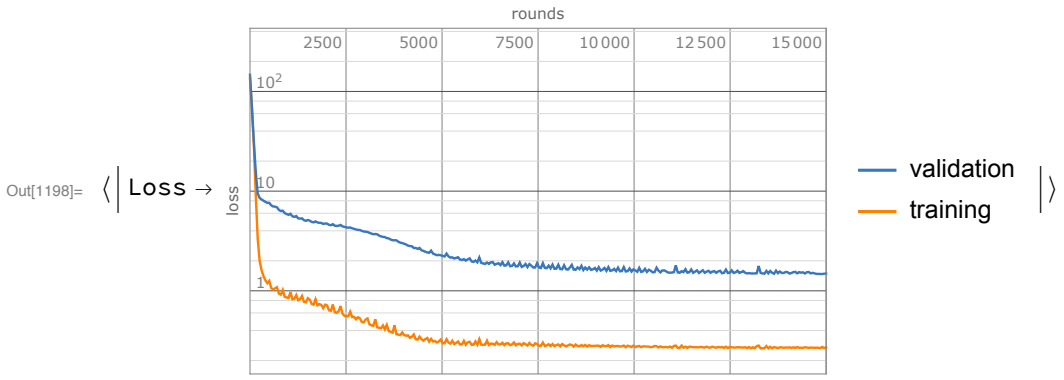
NetTrain Results

summary	batches: 30 000, rounds: 15 000, time: 1.1min, examples/s: 27 922
data	training examples: 128, validation examples: 40, processed examples: 1 920 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 2.77×10^{-1}
validation	loss: 1.46



In[1198]:=
```
trainedNetSimple200v7["FinalPlots"]
trainedNetSimple200v7["TotalTrainingTime"]
trainedNetSimple200v7["RoundMeasurements"]
best = trainedNetSimple200v7["BestValidationRound"]
trainedNetSimple200v7["ValidationLossList"][[best]]
NetInformation[trainedNetSimple200v7["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNetSimple200v7["TrainedNet"], "SummaryGraphic"]
```

Out[1198]=



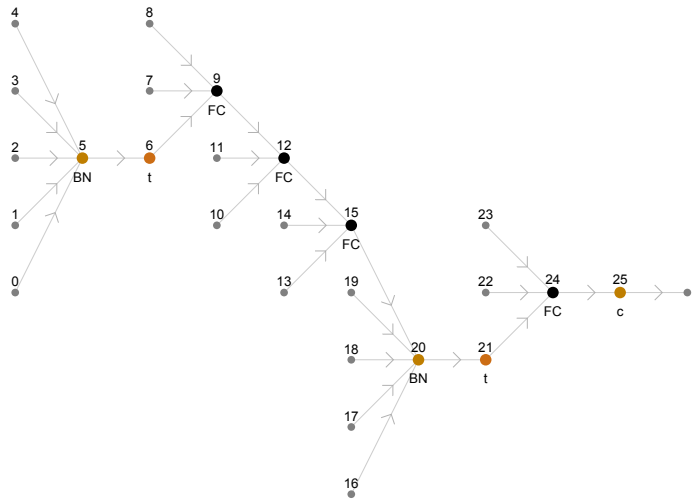
Out[1199]= 68.7619

Out[1200]= ⟨|Loss → 0.276535|⟩

Out[1201]= 13 193

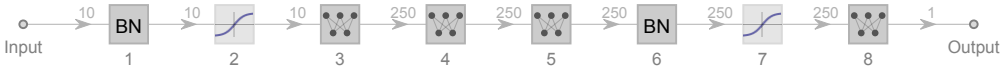
Out[1202]= 1.33129

Out[1203]=



Tensor BatchNorm tanh FullyConnected _copy

Out[1204]=



In[1205]:=
```
netSimple200v8 = NetChain[{BatchNormalizationLayer[], Tanh, 250, 250, 250, 250,
    250, BatchNormalizationLayer[], Tanh, 1}, "Input" → 10, "Output" → "Scalar"]
trainedNetSimple200v8 = NetTrain[netSimple200v8, finalTrain200,
  All, ValidationSet → finaltest200, MaxTrainingRounds → 15 000,
  Method → {"ADAM", "LearningRate" → 0.0005, "L2Regularization" → 0.05}]
```

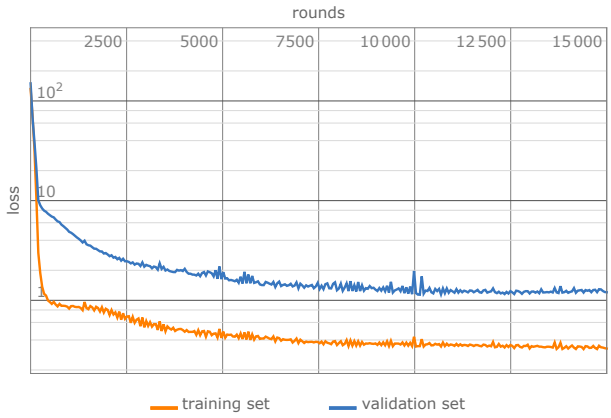
Out[1205]=

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

Out[1206]=

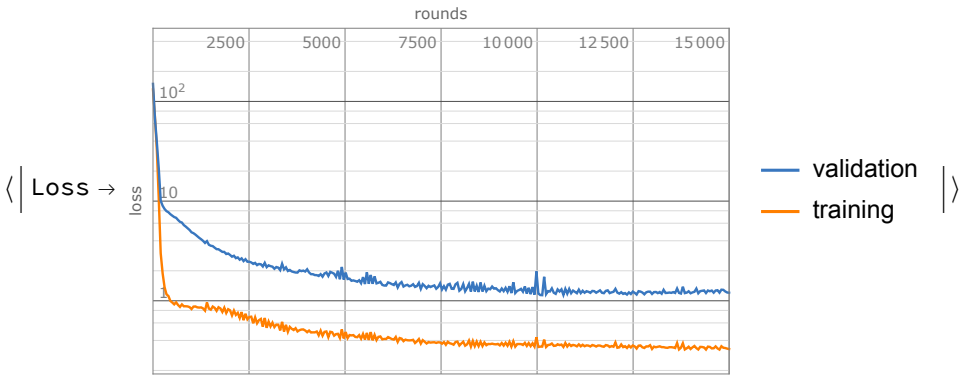
NetTrain Results

summary	batches: 30 000, rounds: 15 000, time: 1.8min, examples/s: 17 692
data	training examples: 128, validation examples: 40, processed examples: 1 920 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 3.26×10^{-1}
validation	loss: 1.2



In[1207]:=
```
trainedNetSimple200v8["FinalPlots"]
trainedNetSimple200v8["TotalTrainingTime"]
trainedNetSimple200v8["RoundMeasurements"]
best = trainedNetSimple200v8["BestValidationRound"]
trainedNetSimple200v8["ValidationLossList"][[best]]
NetInformation[trainedNetSimple200v8["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNetSimple200v8["TrainedNet"], "SummaryGraphic"]
```

Out[1207]=



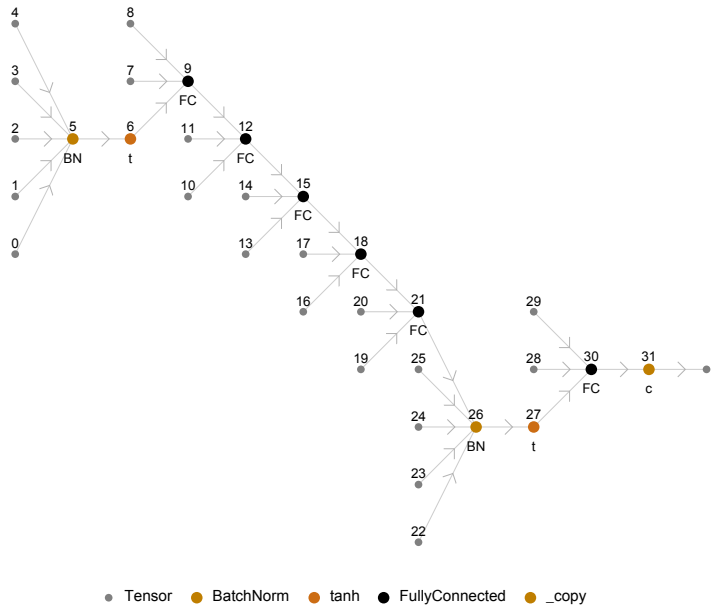
Out[1208]= 108.521

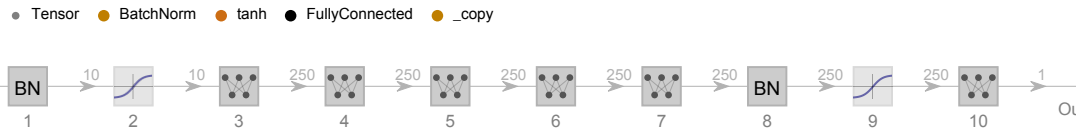
Out[1209]= ⟨| Loss → 0.326359 |⟩

Out[1210]= 9900

Out[1211]= 1.02354

Out[1212]=





In[1243]:=
```
netSimple200v9 =
 NetChain[{BatchNormalizationLayer[], Tanh, 100, 100, 100, 100, 100, 100, 100,
   BatchNormalizationLayer[], Tanh, 1}, "Input" → 10, "Output" → "Scalar"]
trainedNetSimple200v9 = NetTrain[netSimple200v9, finalTrain200,
  All, ValidationSet → finaltest200, MaxTrainingRounds → 15 000,
  Method → {"ADAM", "LearningRate" → 0.0005, "L2Regularization" → 0.05}]
```

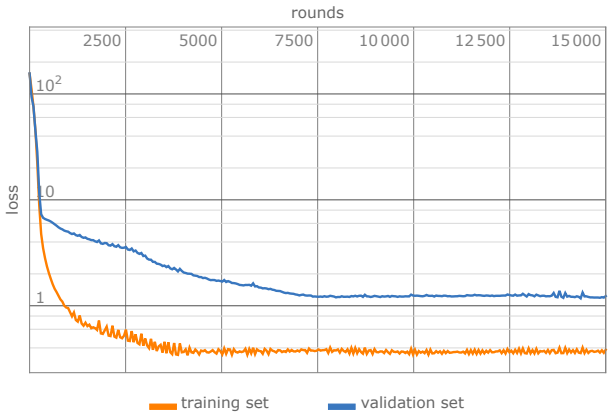
Out[1243]=

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

Out[1244]=

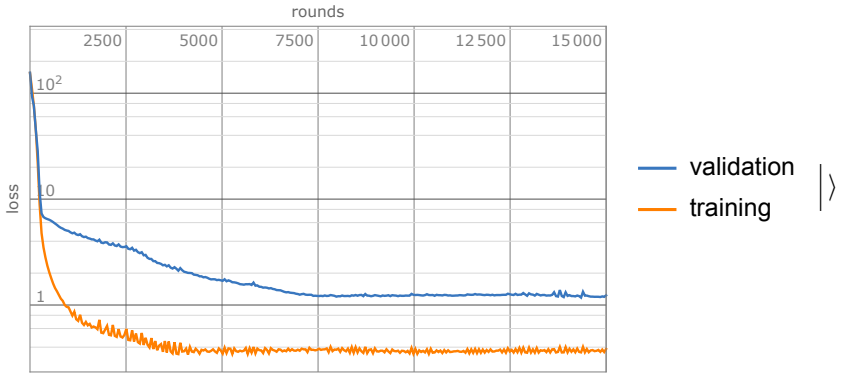
NetTrain Results

summary	batches: 30 000, rounds: 15 000, time: 48s, examples/s: 40 418
data	training examples: 128, validation examples: 40, processed examples: 1 920 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 4.05×10^{-1}
validation	loss: 1.6



In[1245]:=
```
trainedNetSimple200v9["FinalPlots"]
trainedNetSimple200v9["TotalTrainingTime"]
trainedNetSimple200v9["RoundMeasurements"]
best = trainedNetSimple200v9["BestValidationRound"]
trainedNetSimple200v9["ValidationLossList"][[best]]
NetInformation[trainedNetSimple200v9["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNetSimple200v9["TrainedNet"], "SummaryGraphic"]
```

Out[1245]=



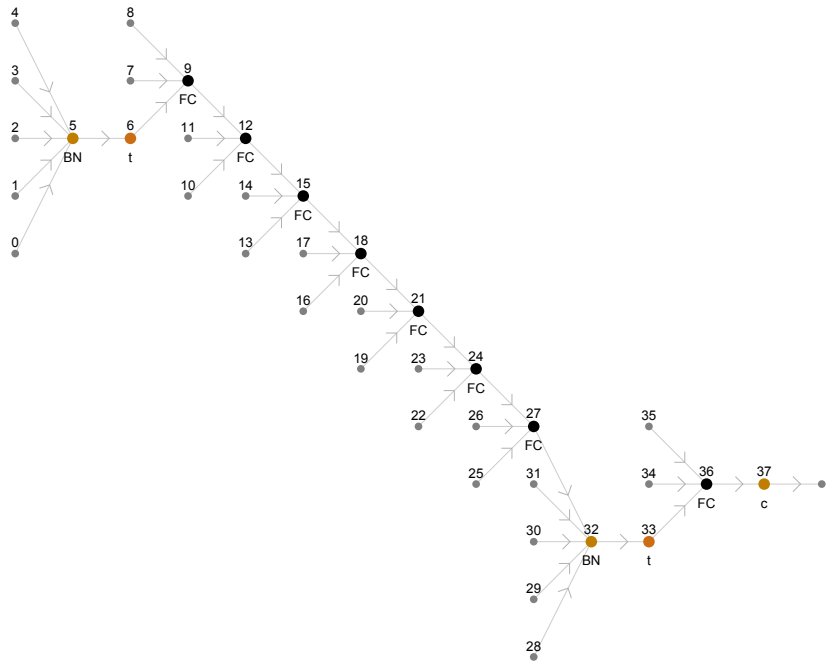
Out[1246]= 47.5036

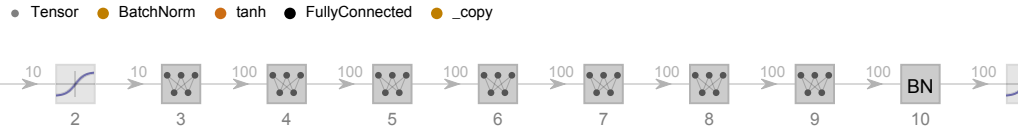
Out[1247]= ⟨| Loss → 0.40492 |⟩

Out[1248]= 14 395

Out[1249]= 0.964567

Out[1250]=





Out[1251]=