New NN test, Train/Dev/Test Split

In[1252]:=
```
ABMInputsV2 = Import[
   "/Users/thorsilver/Downloads/SocialCareOptimisation/ABM-for-social-care/
     lptau10round2_GEMSA_inputs.csv"];

ABMOutputsV2 = Import[
  "/Users/thorsilver/Downloads/SocialCareOptimisation/ABM-for-social-care/
    lptau10round2 GEMSA outputs.csv"];
ABMOutputsV2 = Function[x, x / 1000] /@ ABMOutputsV2;
ABMAssocV2 = AssociationThread[ABMInputsV2 → Flatten[ABMOutputsV2]];
ABMnewDataV2 = Dataset[ABMAssocV2];
ABMNormalV2 = Normal[ABMAssocV2];
ABMNormalRandom = RandomSample[ABMNormalV2];
ABMtrainv2 = TakeDrop[ABMNormalV2, 320];
ABMtestv2 = ABMtrainv2[[2]];
ABMtraining = ABMtrainv2[[1]];
trainDevSplit = TakeDrop[ABMtraining, 256];
finalTrain = trainDevSplit[[1]];
finalDev = trainDevSplit[[2]];
finaltest = ABMtestv2;
```

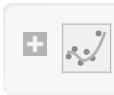
In[1265]:=
```
Length[finalTrain]
Length[finalDev]
Length[finaltest]
```

Out[1265]= 256

Out[1266]= 64

Out[1267]= 80

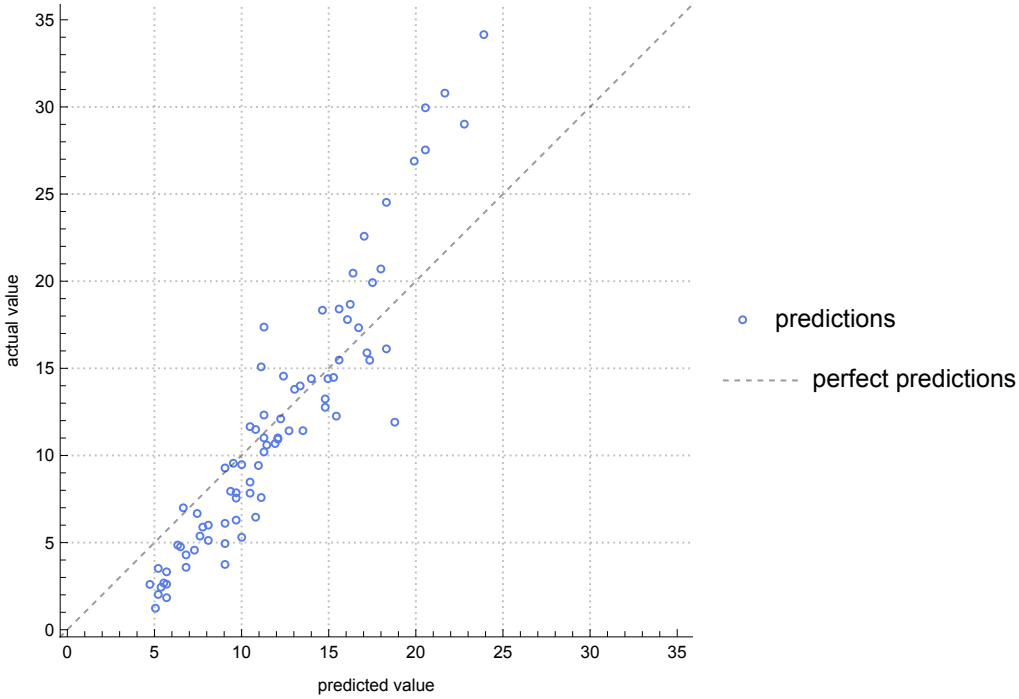
In[1268]:=
```
pRFv2 = Predict[finalTrain, Method → "RandomForest"]
```

Out[1268]= PredictorFunction[ Input type: Mixed (number: 10)
Method: RandomForest]

In[1289]:=
```
pmRFv2 = PredictorMeasurements[pRFv2, finaltest]
pmRFv2["ComparisonPlot"]
pmRFv2["MeanSquare"]
pmRFv2["ResidualPlot"]
Information[pRFv2]
```

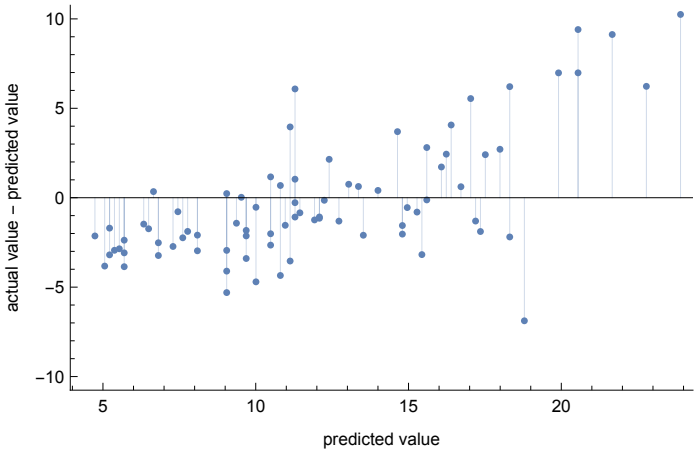
Out[1289]= PredictorMeasurementsObject[ Predictor: RandomForest
Number of test examples: 80]

Out[1290]=



Out[1291]= 12.0599

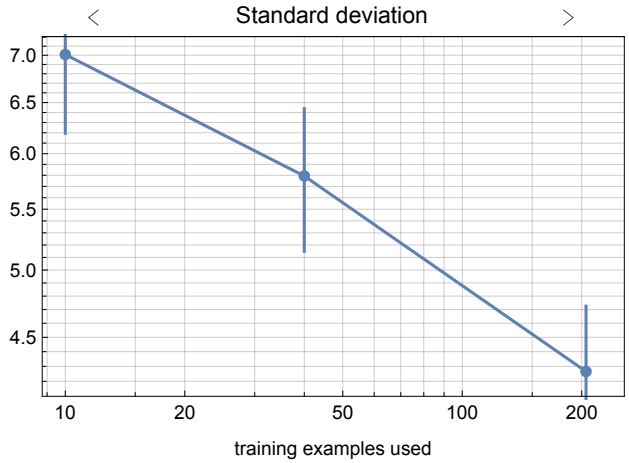
Out[1292]=



Out[1293]=

Predictor information

Data type	Mixed (number: 10)
Standard deviation	4.27 ± 0.46
Method	RandomForest
Single evaluation time	4.33 ms/example
Batch evaluation speed	52. examples/ms
Loss	2.88 ± 0.092
Model memory	238. kB
Training examples used	256 examples
Training time	397. ms



In[1273]:=

```
pXGTv2 = Predict[finalTrain, Method → "GradientBoostedTrees"]
```

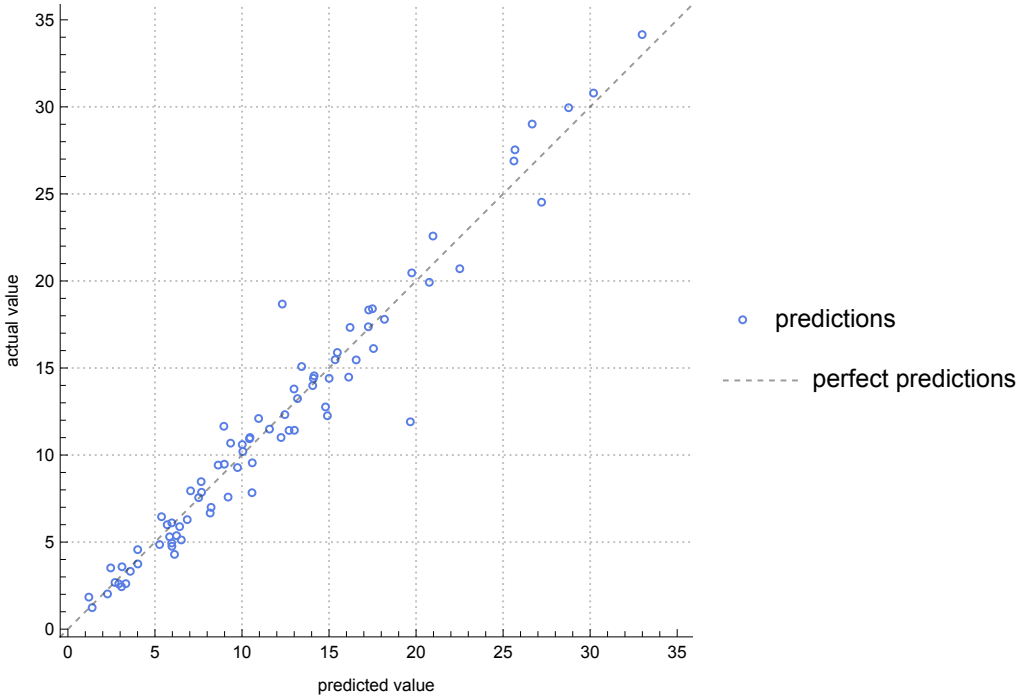
Out[1273]= PredictorFunction[Input type: Mixed (number: 10) Method: GradientBoostedTrees]

In[1294]:=

```
pmXGTv2 = PredictorMeasurements[pXGTv2, finaltest]
pmXGTv2["ComparisonPlot"]
pmXGTv2["MeanSquare"]
pmXGTv2["ResidualPlot"]
Information[pXGTv2]
```

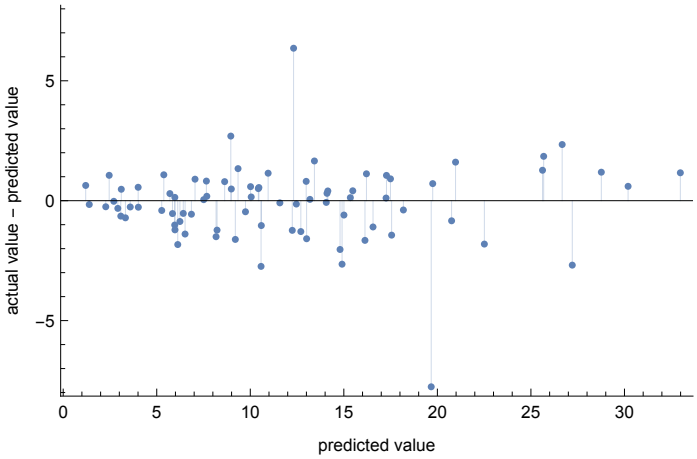
Out[1294]= PredictorMeasurementsObject[Predictor: GradientBoostedTrees Number of test examples: 80]

Out[1295]=



Out[1296]= 2.52565

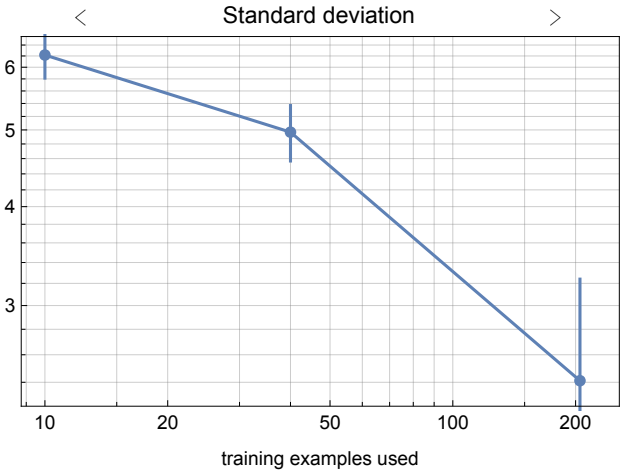
Out[1297]=



Predictor information

Data type	Mixed (number: 10)
Standard deviation	2.41 ± 0.83
Method	GradientBoostedTrees
Single evaluation time	3.8 ms/example
Batch evaluation speed	43.8 examples/ms
Loss	3.77 ± 1.0
Model memory	391. kB
Training examples used	256 examples
Training time	1.47 s

Out[1298]=



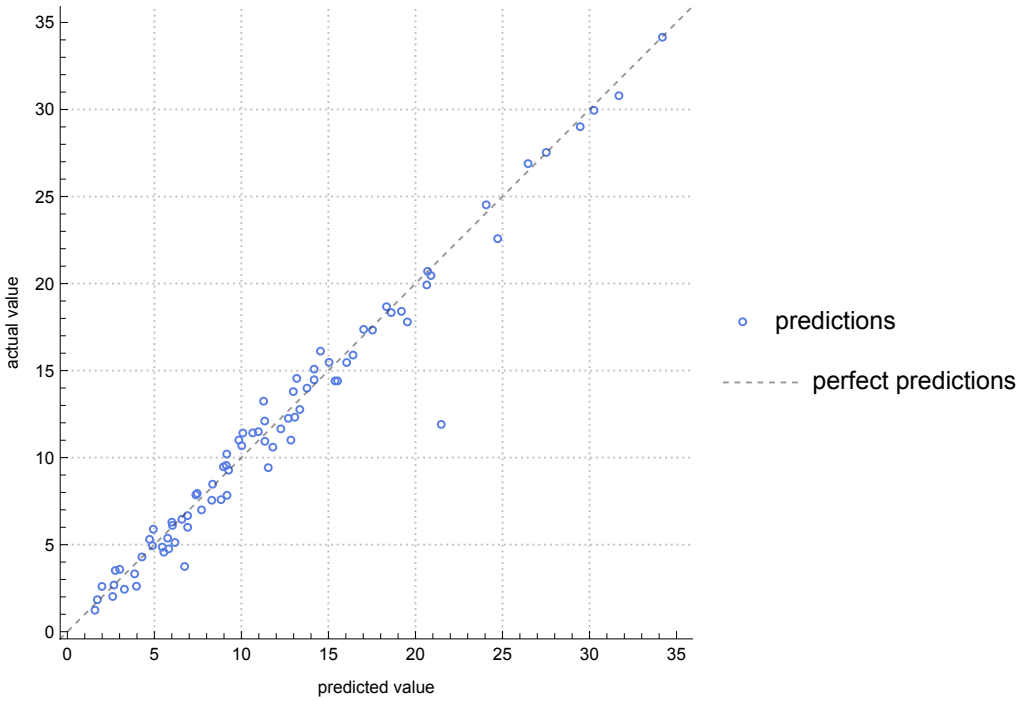
In[1278]:=
```
pNNv2 = Predict[finalTrain, Method →
    {"NeuralNetwork", "NetworkDepth" → 3, "NetworkType" → "FullyConnected",
     "L2Regularization" → 0.05, MaxTrainingRounds → 16 000}]
```

Out[1278]= PredictorFunction[Input type: Mixed (number: 10) Method: NeuralNetwork]

In[1279]:=
```
pmNNv2 = PredictorMeasurements[pNNv2, finaltest]
pmNNv2["ComparisonPlot"]
pmNNv2["MeanSquare"]
pmNNv2["ResidualPlot"]
Information[pNNv2]
```

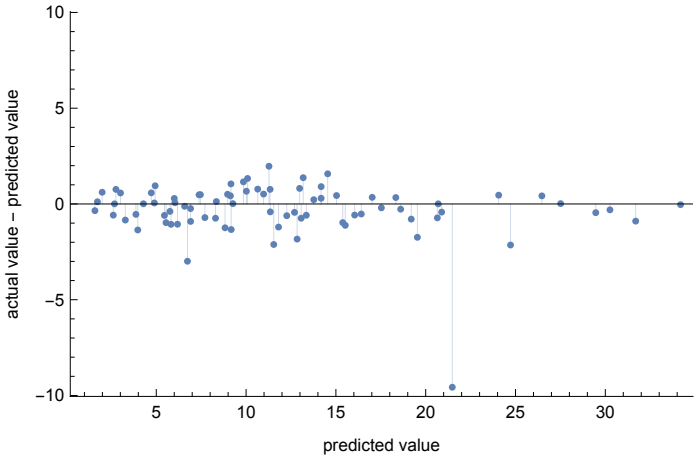
Out[1279]= PredictorMeasurementsObject[Predictor: NeuralNetwork Number of test examples: 80]

Out[1280]=



Out[1281]= 1.96365

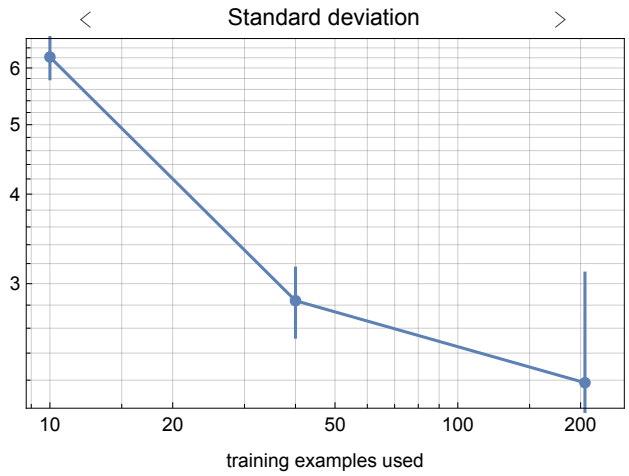
Out[1282]=



Predictor information

Data type	Mixed (number: 10)
Standard deviation	2.18 ± 0.92
Method	NeuralNetwork
Single evaluation time	2.16 ms/example
Batch evaluation speed	46.7 examples/ms
Loss	1.87 ± 0.22
Model memory	246. kB
Training examples used	256 examples
Training time	1 min 23 s

Out[1283]=

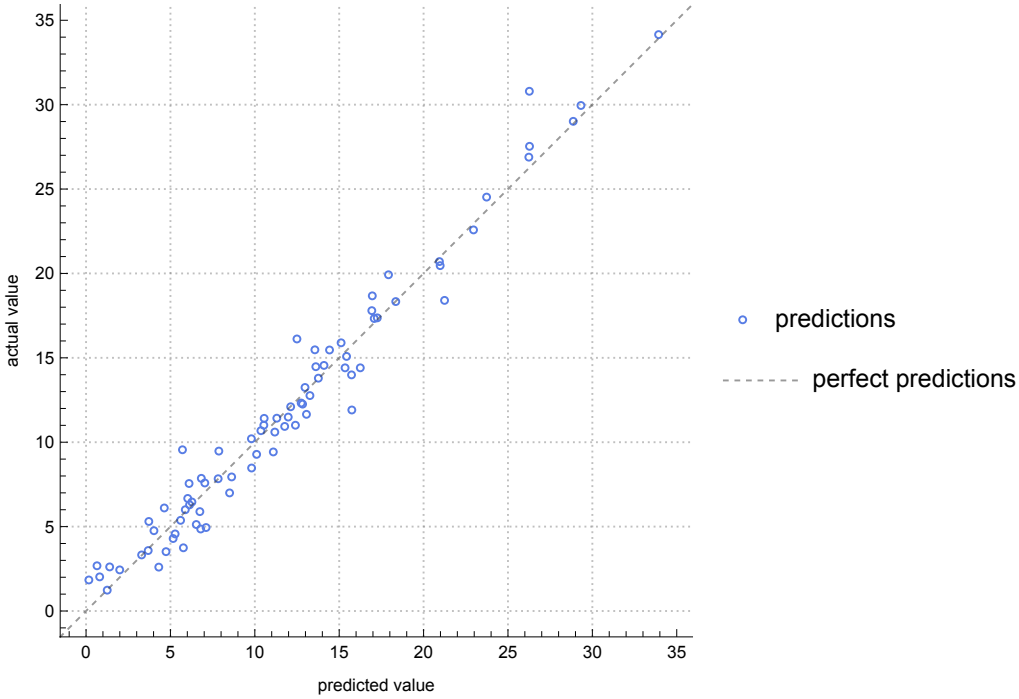


In[1299]:=
```
pGPEv2 = Predict[finalTrain, Method → "GaussianProcess"]
pmGPEv2 = PredictorMeasurements[pGPEv2, finaltest]
pmGPEv2["ComparisonPlot"]
pmGPEv2["MeanSquare"]
pmGPEv2["ResidualPlot"]
Information[pGPEv2]
```

Out[1299]= PredictorFunction[Input type: Mixed (number: 10) Method: GaussianProcess]

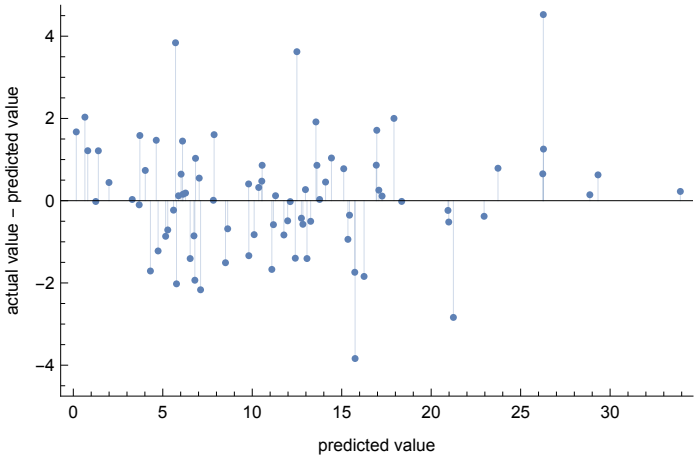
Out[1300]= PredictorMeasurementsObject[Predictor: GaussianProcess Number of test examples: 80]

Out[1301]=



Out[1302]= 1.92845

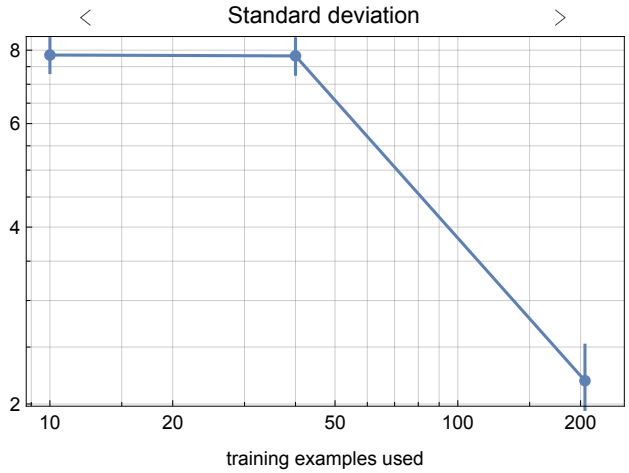
Out[1303]=



Out[1304]=

Predictor information

Data type	Mixed (number: 10)
Standard deviation	2.19 ± 0.33
Method	GaussianProcess
Single evaluation time	3.95 ms/example
Batch evaluation speed	25.7 examples/ms
Loss	1.97 ± 0.12
Model memory	548. kB
Training examples used	256 examples
Training time	2.97 s



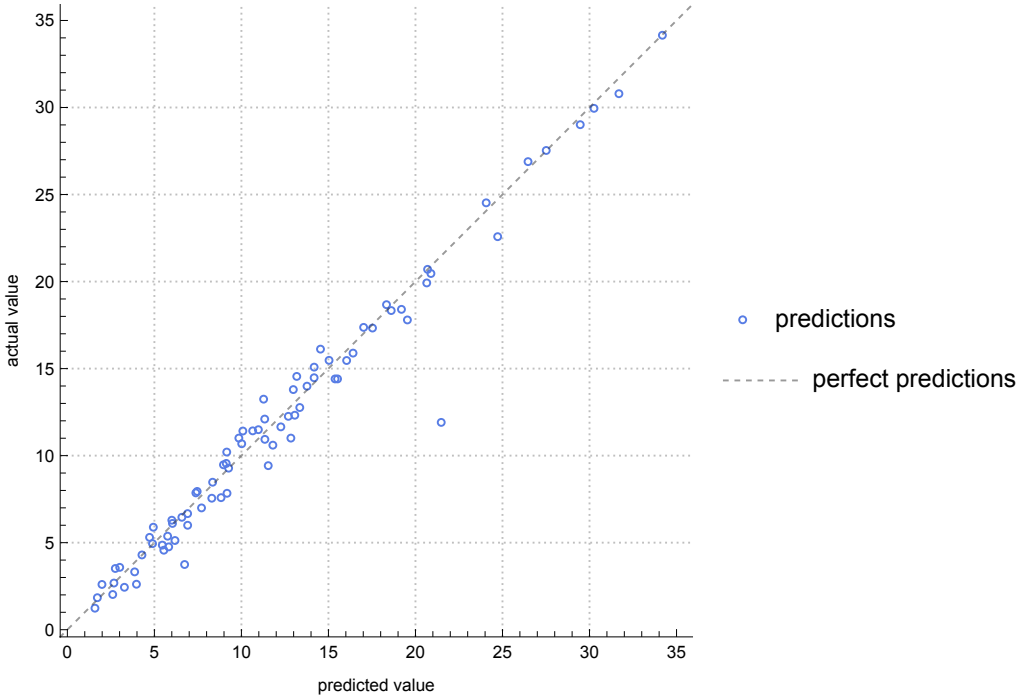
In[1305]:=

```
pNNv3 = Predict[finalTrain, Method →
    {"NeuralNetwork", "NetworkDepth" → 3, "NetworkType" → "FullyConnected",
     "L2Regularization" → 0.05, MaxTrainingRounds → 24 000}]
pmNNv3 = PredictorMeasurements[pNNv3, finaltest]
pmNNv3["ComparisonPlot"]
pmNNv3["MeanSquare"]
pmNNv3["ResidualPlot"]
Information[pNNv3]
```

Out[1305]= PredictorFunction[Input type: Mixed (number: 10) Method: NeuralNetwork]

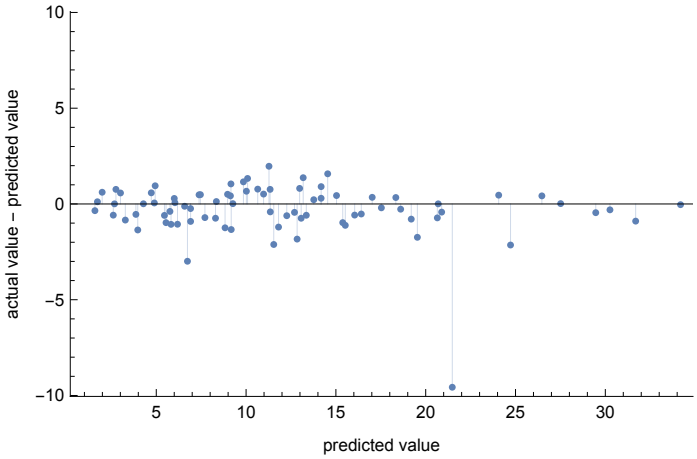
Out[1306]= PredictorMeasurementsObject[Predictor: NeuralNetwork Number of test examples: 80]

Out[1307]=



Out[1308]= 1.96365

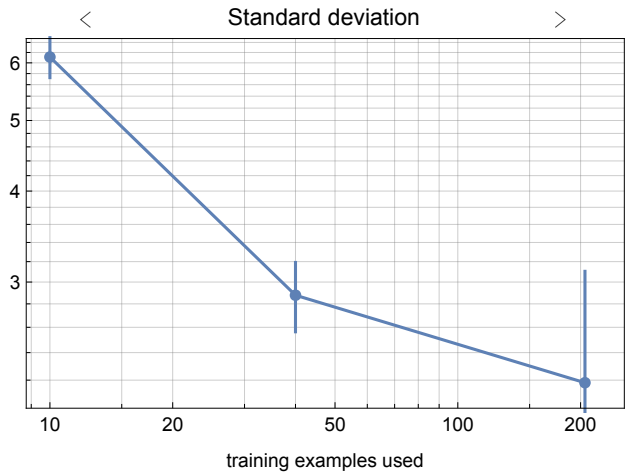
Out[1309]=



Out[1310]=

Predictor information

Data type	Mixed (number: 10)
Standard deviation	2.18 ± 0.92
Method	NeuralNetwork
Single evaluation time	2.31 ms/example
Batch evaluation speed	45.4 examples/ms
Loss	1.87 ± 0.22
Model memory	246. kB
Training examples used	256 examples
Training time	2 min 8 s

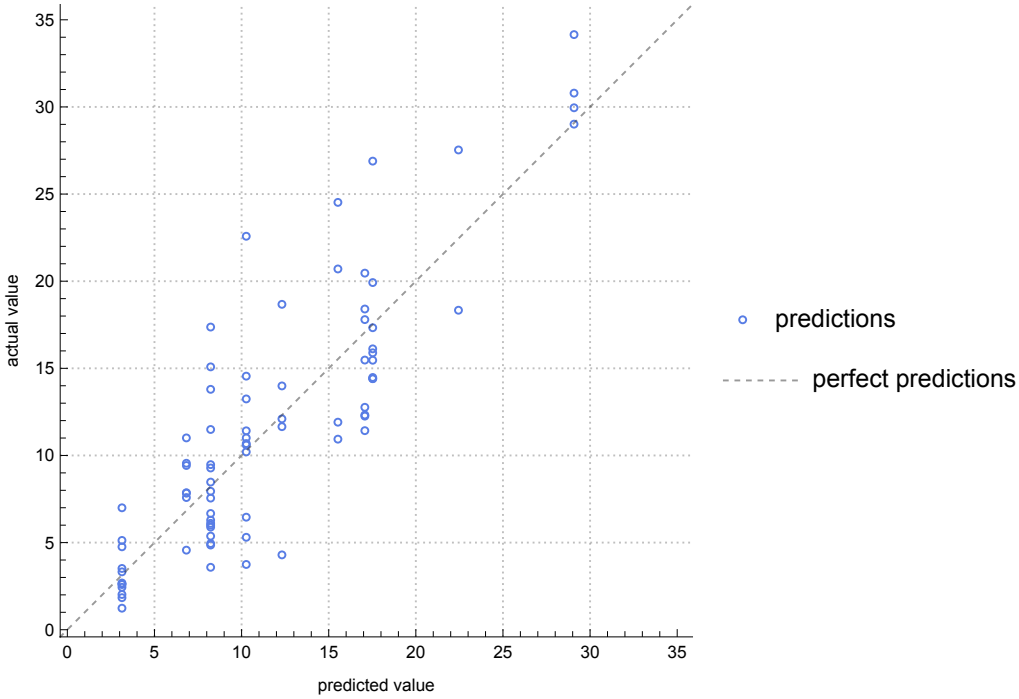


In[1311]:=
```
pDTv2 = Predict[finalTrain, Method → "DecisionTree"]
pmDTv2 = PredictorMeasurements[pDTv2, finaltest]
pmDTv2["ComparisonPlot"]
pmDTv2["MeanSquare"]
pmDTv2["ResidualPlot"]
Information[pDTv2]
```

Out[1311]= PredictorFunction[Input type: Mixed (number: 10) Method: DecisionTree]

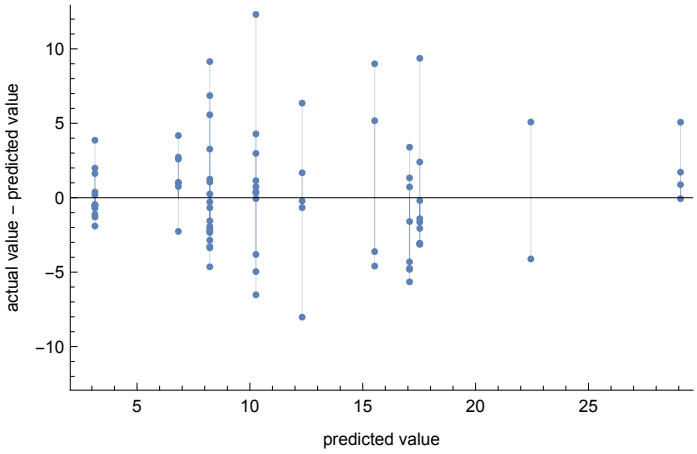
Out[1312]= PredictorMeasurementsObject[Predictor: DecisionTree Number of test examples: 80]

Out[1313]=



Out[1314]= 14.3214

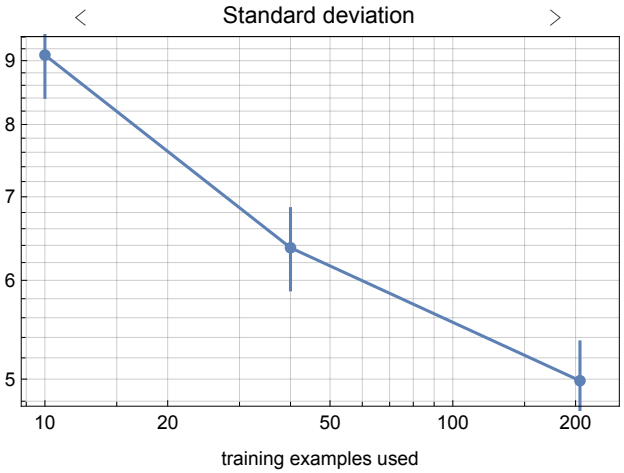
Out[1315]=



Predictor information

Data type	Mixed (number: 10)
Standard deviation	4.99 ± 0.37
Method	DecisionTree
Single evaluation time	1.32 ms/example
Batch evaluation speed	305. examples/ms
Loss	3.01 ± 0.13
Model memory	147. kB
Training examples used	256 examples
Training time	355. ms

Out[1316]=

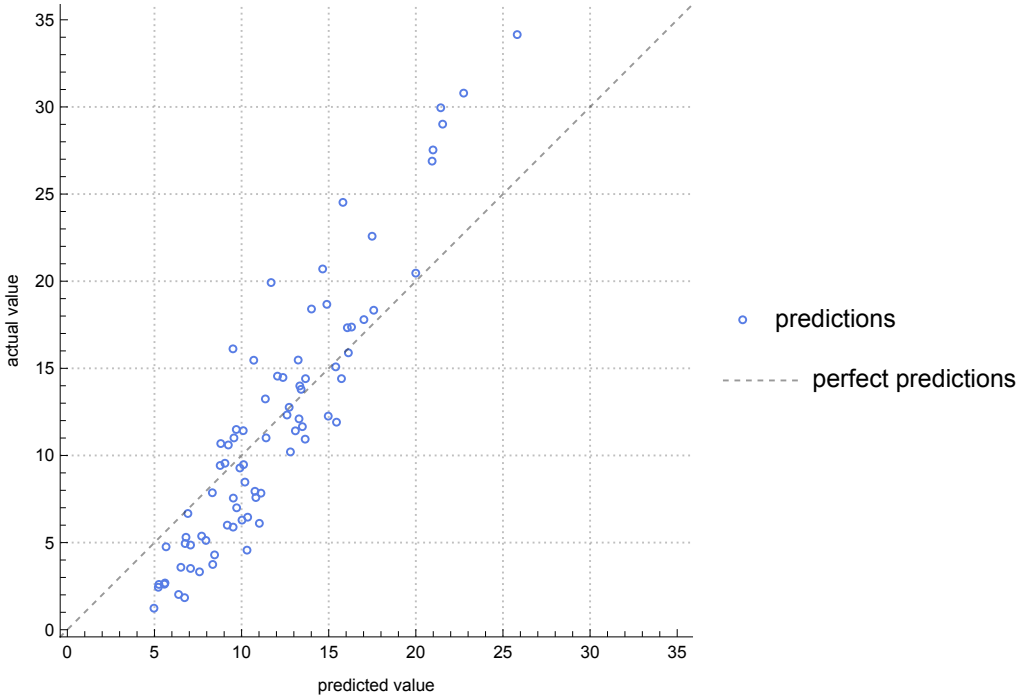


In[1317]:=
```
pNearestv2 = Predict[finalTrain, Method → "NearestNeighbors"]
pmNearestv2 = PredictorMeasurements[pNearestv2, finaltest]
pmNearestv2["ComparisonPlot"]
pmNearestv2["MeanSquare"]
pmNearestv2["ResidualPlot"]
Information[pNearestv2]
```

Out[1317]= PredictorFunction[Input type: Mixed (number: 10) Method: NearestNeighbors]

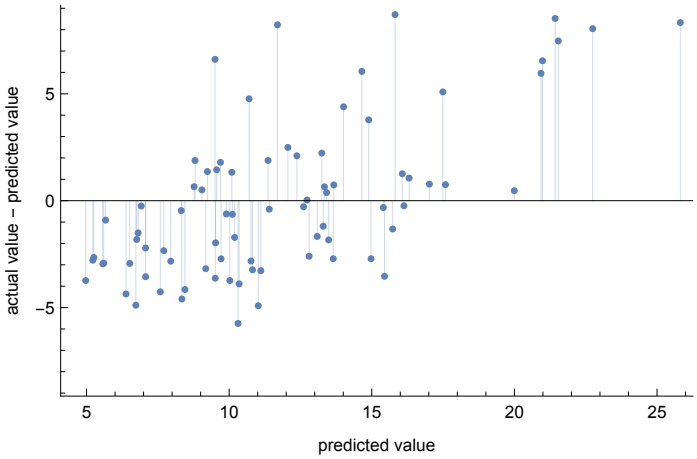
Out[1318]= PredictorMeasurementsObject[Predictor: NearestNeighbors Number of test examples: 80]

Out[1319]=



Out[1320]= 13.4224

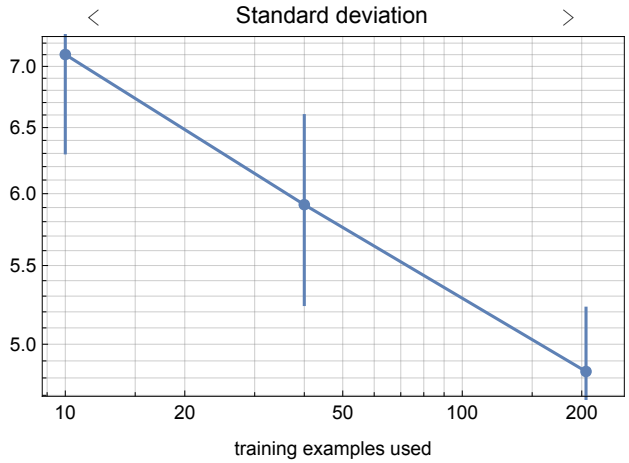
Out[1321]=



Predictor information

Data type	Mixed (number: 10)
Standard deviation	4.84 ± 0.39
Method	NearestNeighbors
Single evaluation time	1.28 ms/example
Batch evaluation speed	86.2 examples/ms
Loss	3.02 ± 0.045
Model memory	171. kB
Training examples used	256 examples
Training time	350. ms

Out[1322]=

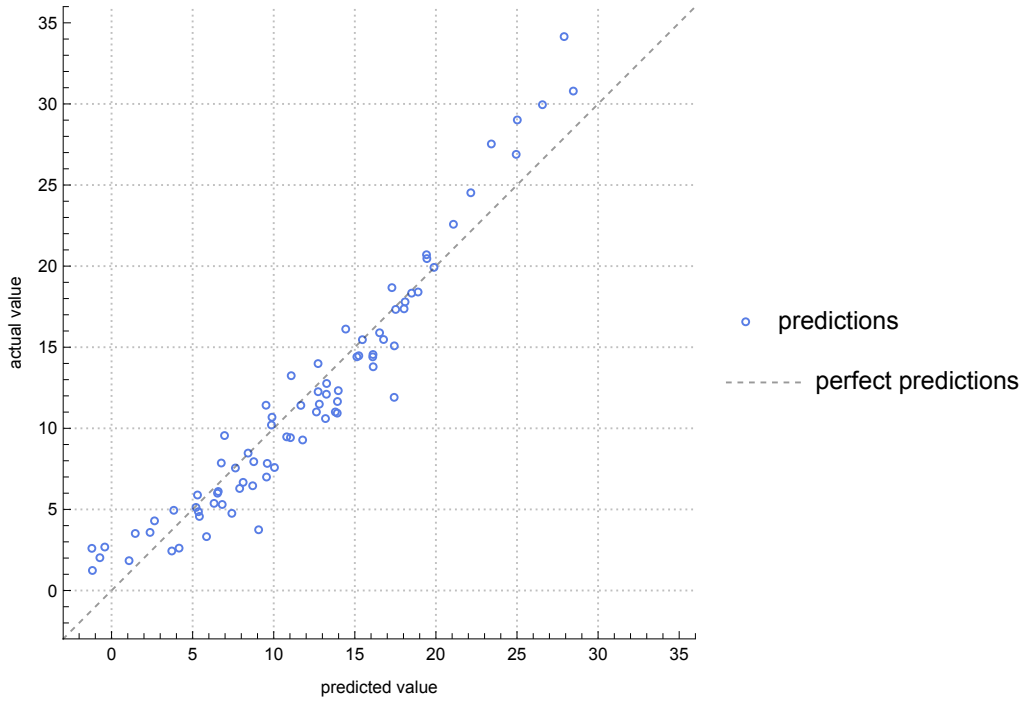


In[1323]:=
```
pLRv2 = Predict[finalTrain, Method → "LinearRegression"]
pmLRv2 = PredictorMeasurements[pLRv2, finaltest]
pmLRv2["ComparisonPlot"]
pmLRv2["MeanSquare"]
pmLRv2["ResidualPlot"]
Information[pLRv2]
```

Out[1323]= PredictorFunction[Input type: Mixed (number: 10) Method: LinearRegression]

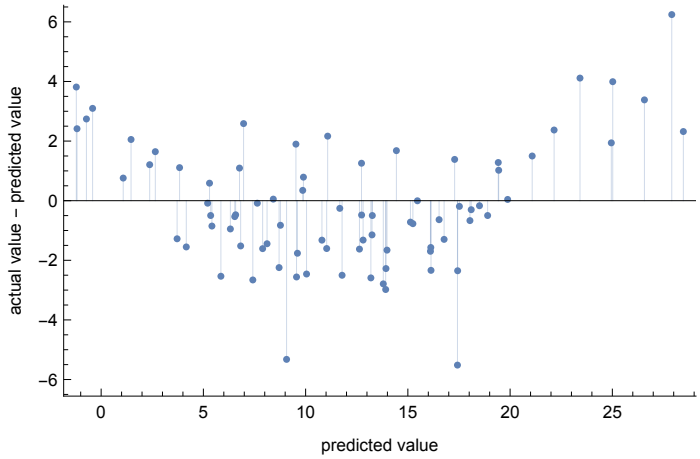
Out[1324]= PredictorMeasurementsObject[Predictor: LinearRegression Number of test examples: 80]

Out[1325]=



Out[1326]= 4.374

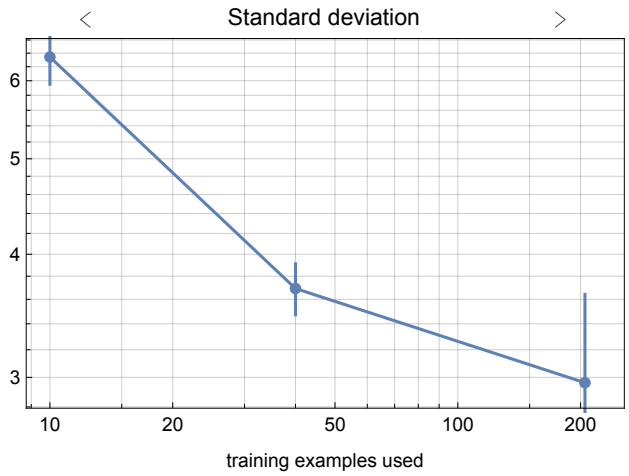
Out[1327]=



Predictor information

Data type	Mixed (number: 10)
Standard deviation	2.96 ± 0.68
Method	LinearRegression
Single evaluation time	1.54 ms/example
Batch evaluation speed	234. examples/ms
Loss	2.42 ± 0.12
Model memory	281. kB
Training examples used	256 examples
Training time	1.43 s

Out[1328]=

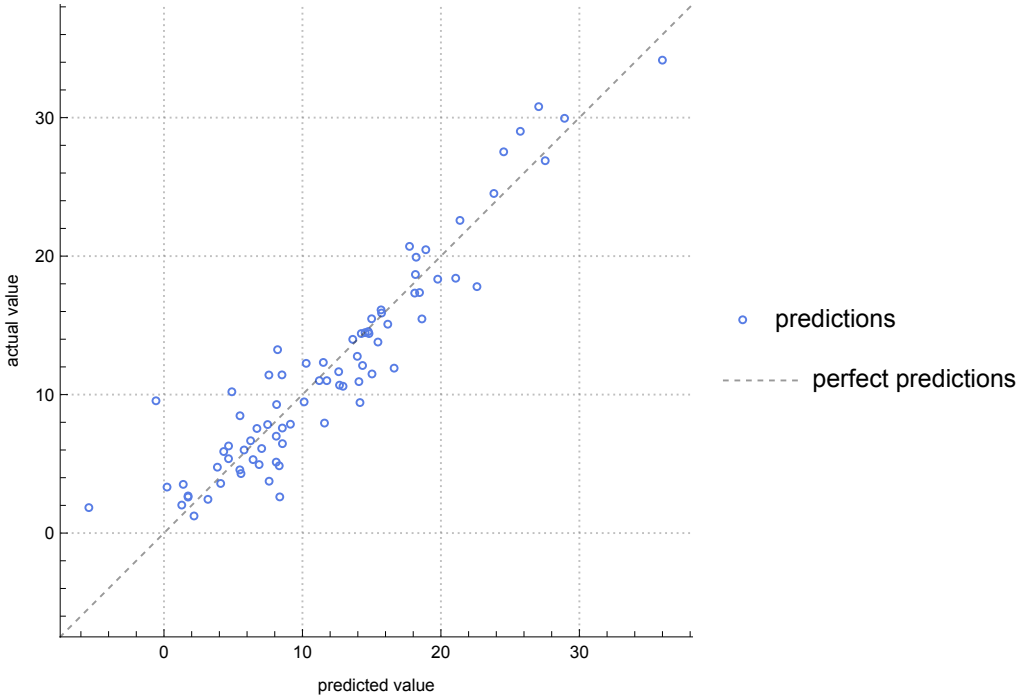


In[1335]:=
```
pNNnoReg = Predict[finalTrain, Method → {"NeuralNetwork", "NetworkDepth" → 3,
    "NetworkType" → "FullyConnected", MaxTrainingRounds → 24000}]
pmNNnoReg = PredictorMeasurements[pNNnoReg, finaltest]
pmNNnoReg["ComparisonPlot"]
pmNNnoReg["MeanSquare"]
pmNNnoReg["ResidualPlot"]
Information[pNNnoReg]
```

Out[1335]= PredictorFunction[Input type: Mixed (number: 10) Method: NeuralNetwork]

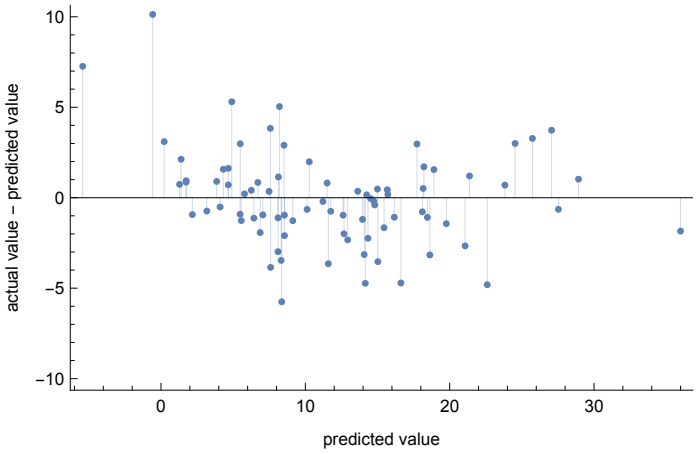
Out[1336]= PredictorMeasurementsObject[Predictor: NeuralNetwork Number of test examples: 80]

Out[1337]=



Out[1338]= 6.99146

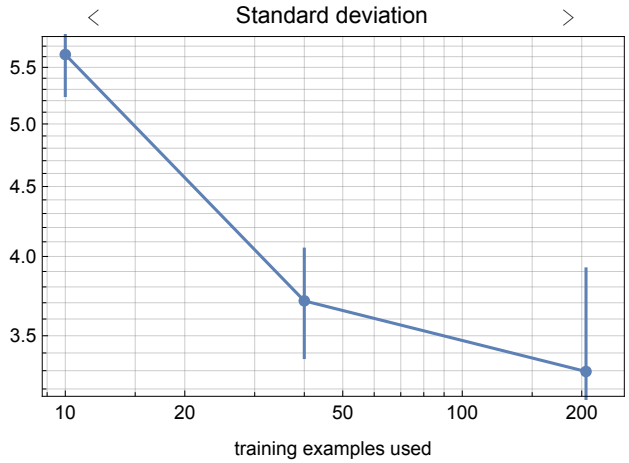
Out[1339]=



Predictor information

Data type	Mixed (number: 10)
Standard deviation	3.30 ± 0.62
Method	NeuralNetwork
Single evaluation time	2.26 ms/example
Batch evaluation speed	46.9 examples/ms
Loss	2.57 ± 0.14
Model memory	246. kB
Training examples used	256 examples
Training time	2 min 4 s

Out[1340]=

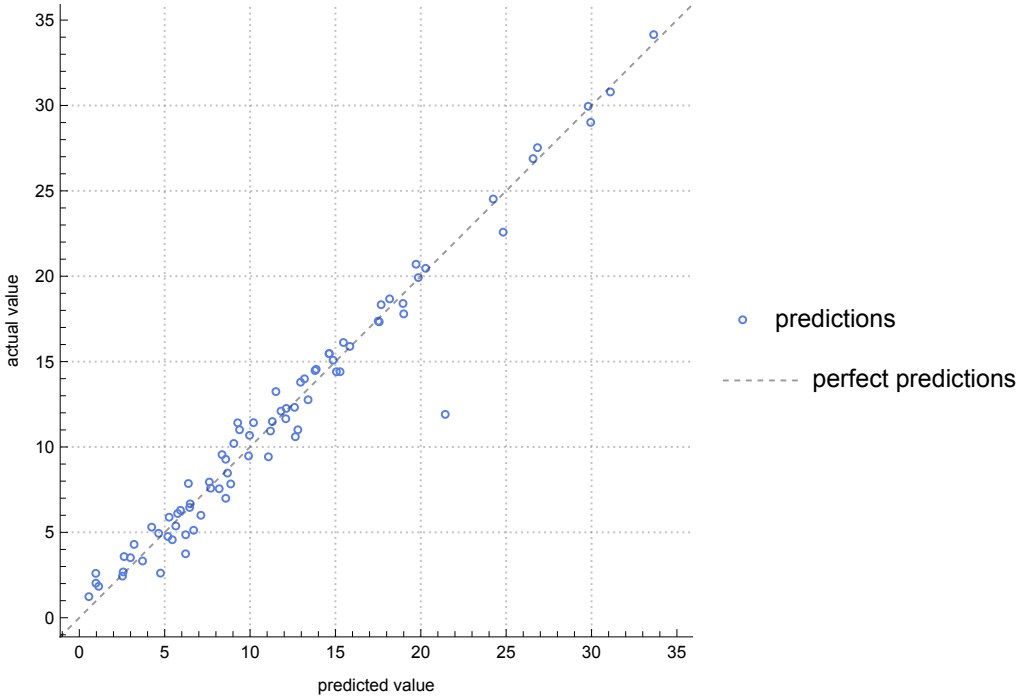


In[1341]:=
```
pNN2layer = Predict[finalTrain, Method →
    {"NeuralNetwork", "NetworkDepth" → 2, "NetworkType" → "FullyConnected",
     "L2Regularization" → 0.05, MaxTrainingRounds → 24 000}]
pmNN2layer = PredictorMeasurements[pNN2layer, finaltest]
pmNN2layer["ComparisonPlot"]
pmNN2layer["MeanSquare"]
pmNN2layer["ResidualPlot"]
Information[pNN2layer]
```

Out[1341]= PredictorFunction[Input type: Mixed (number: 10) Method: NeuralNetwork]

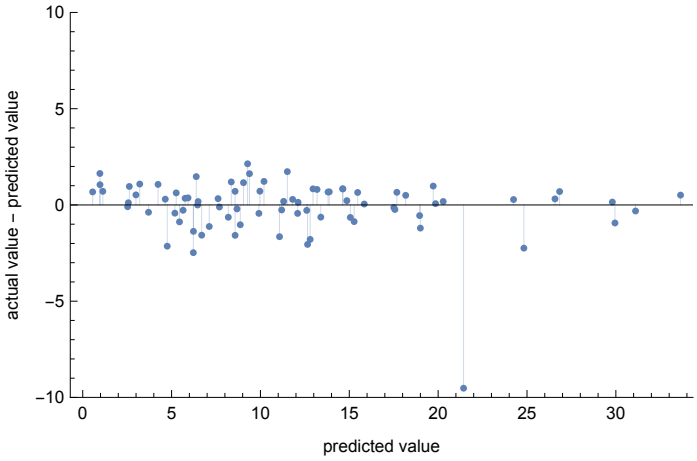
Out[1342]= PredictorMeasurementsObject[Predictor: NeuralNetwork Number of test examples: 80]

Out[1343]=



Out[1344]= 2.07432

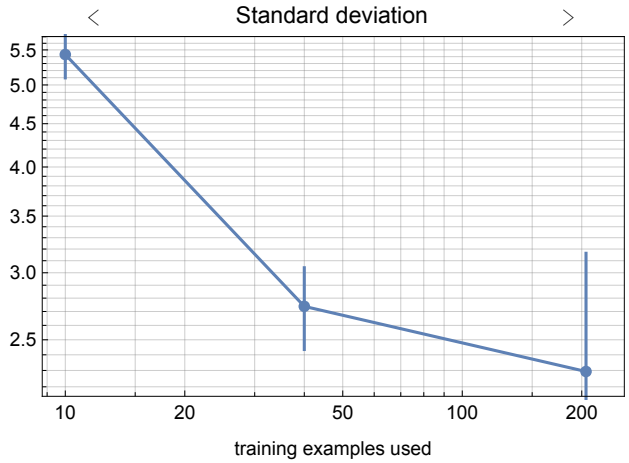
Out[1345]=



Out[1346]=

Predictor information

Data type	Mixed (number: 10)
Standard deviation	2.29 ± 0.87
Method	NeuralNetwork
Single evaluation time	2.13 ms/example
Batch evaluation speed	52.9 examples/ms
Loss	2.08 ± 0.28
Model memory	226. kB
Training examples used	256 examples
Training time	1 min 55 s



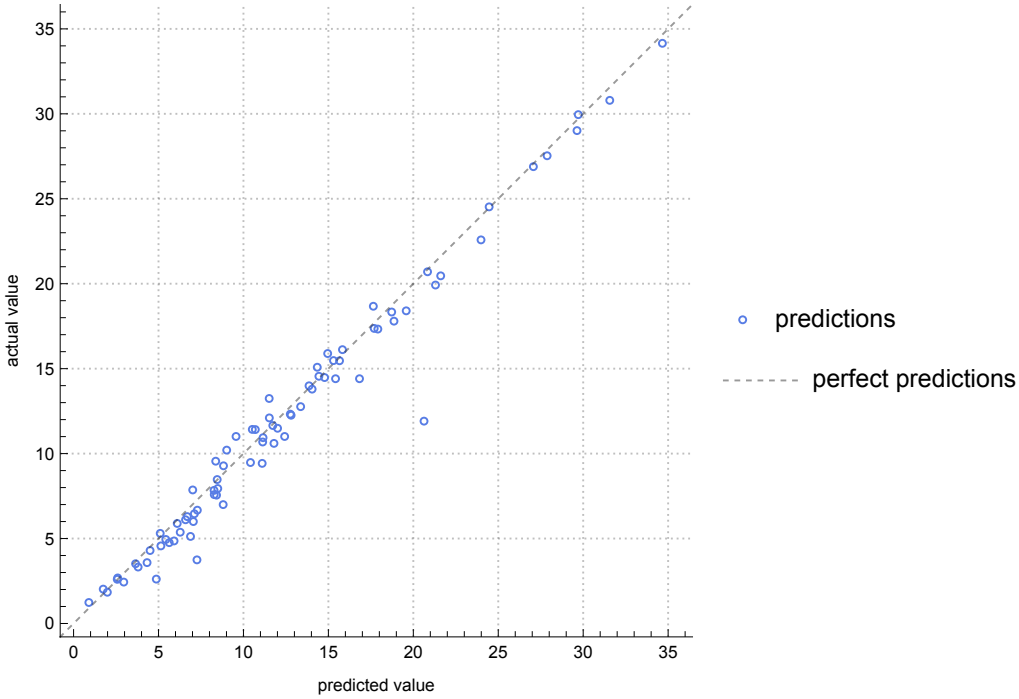
In[1347]:=

```
pNN4layer = Predict[finalTrain, Method →
    {"NeuralNetwork", "NetworkDepth" → 4, "NetworkType" → "FullyConnected",
     "L2Regularization" → 0.05, MaxTrainingRounds → 24 000}]
pmNN4layer = PredictorMeasurements[pNN4layer, finaltest]
pmNN4layer["ComparisonPlot"]
pmNN4layer["MeanSquare"]
pmNN4layer["ResidualPlot"]
Information[pNN4layer]
```

Out[1347]= PredictorFunction[Input type: Mixed (number: 10) Method: NeuralNetwork]

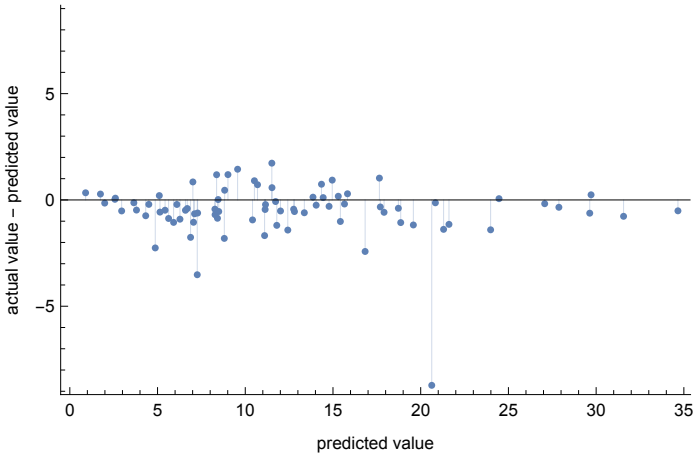
Out[1348]= PredictorMeasurementsObject[Predictor: NeuralNetwork Number of test examples: 80]

Out[1349]=



Out[1350]= 1.83864

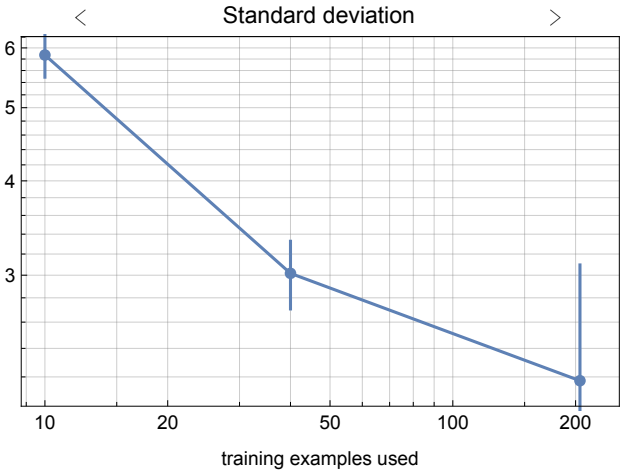
Out[1351]=



Out[1352]=

Predictor information

Data type	Mixed (number: 10)
Standard deviation	2.17 ± 0.92
Method	NeuralNetwork
Single evaluation time	2.17 ms/example
Batch evaluation speed	46.9 examples/ms
Loss	1.88 ± 0.20
Model memory	260. kB
Training examples used	256 examples
Training time	2 min 9 s



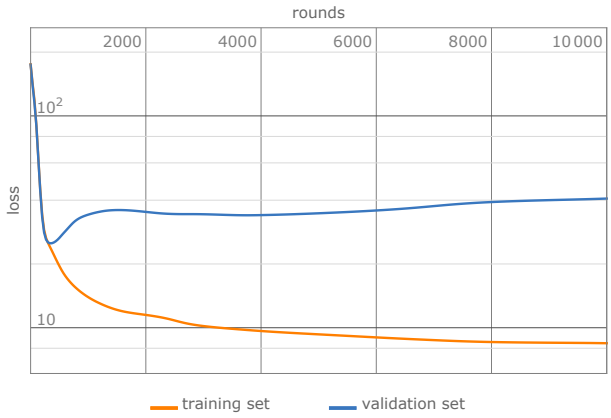
In[1353]:=
```
netSimple400v2 = NetChain[{10, BatchNormalizationLayer[],
    Tanh, 30, 30, 30, 30, BatchNormalizationLayer[], Tanh, 1}]
trainedNetSimple400v2 = NetTrain[netSimple400v2, finalTrain,
  All, ValidationSet → finaltest, MaxTrainingRounds → 10 000,
  Method → {"ADAM", "LearningRate" → 0.0005, "L2Regularization" → 0.3}]
```

Out[1353]=
```
NetChain[ uninitialized | Input port: array
                          Output port: vector (size: 1)
                          Number of layers: 10 ]
```

Out[1354]=

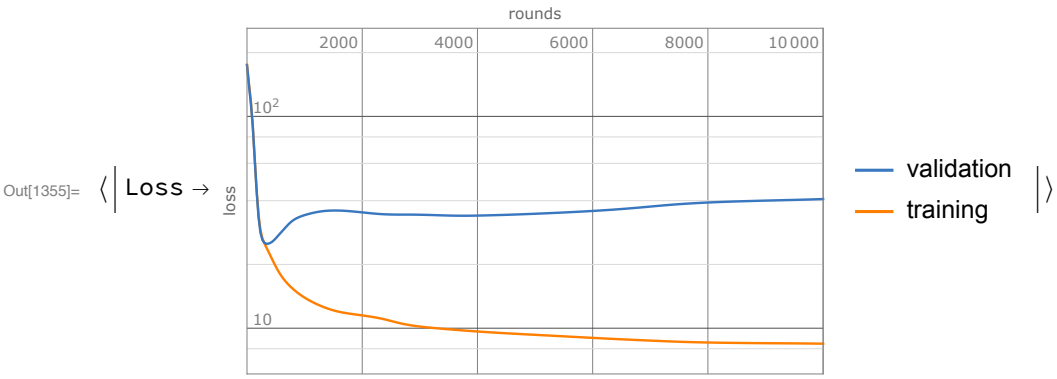
NetTrain Results

summary	batches: 40 000, rounds: 10 000, time: 45s, examples/s: 57 164
data	training examples: 256, validation examples: 80, processed examples: 2 560 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 8.45
validation	loss: 4.07×10^{1}



In[1355]:=
```
trainedNetSimple400v2["FinalPlots"]
trainedNetSimple400v2["TotalTrainingTime"]
trainedNetSimple400v2["RoundMeasurements"]
best = trainedNetSimple400v2["BestValidationRound"]
trainedNetSimple400v2["ValidationLossList"][[best]]
NetInformation[trainedNetSimple400v2["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNetSimple400v2["TrainedNet"], "SummaryGraphic"]
```

Out[1355]=



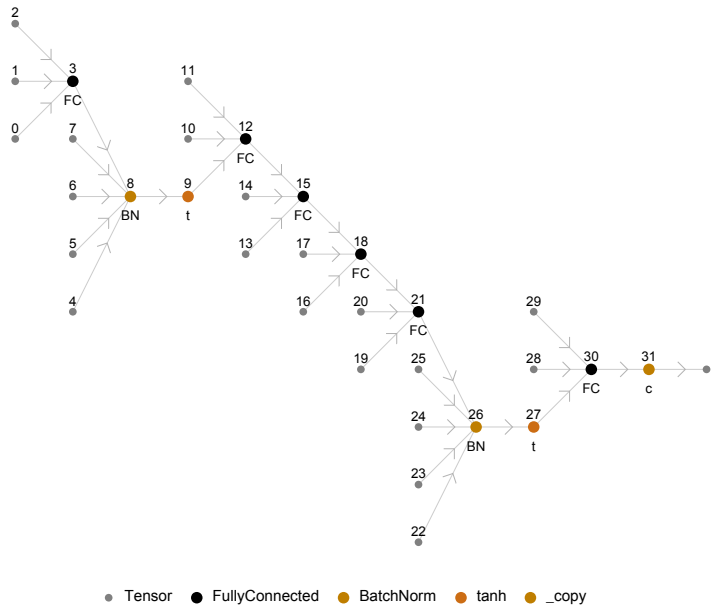
Out[1356]= 44.7837

Out[1357]= ⟨| Loss → 8.44841 |⟩

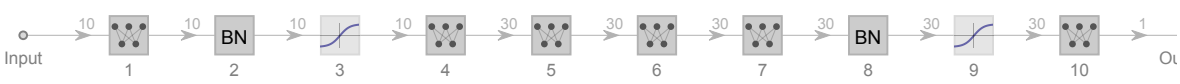
Out[1358]= 380

Out[1359]= 25.0069

Out[1360]=



Out[1361]=



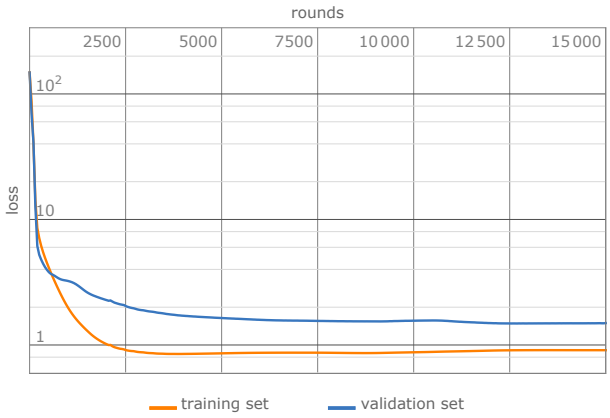
In[1362]:=
```
netSimple400v3 = NetChain[{BatchNormalizationLayer[],
    Tanh, 50, 50, 50, 50, 50, 50, BatchNormalizationLayer[], Tanh, 1}]
trainedNetSimple400v3 = NetTrain[netSimple400v3, finalTrain,
  All, ValidationSet → finaltest, MaxTrainingRounds → 15 000,
  Method → {"ADAM", "LearningRate" → 0.0005, "L2Regularization" → 0.05}]
```

Out[1362]=
```
NetChain[ uninitialized
  Input port:              array of rank ≥ 1
  Output port:             vector (size: 1)
  Number of layers:        11 ]
```

Out[1363]=

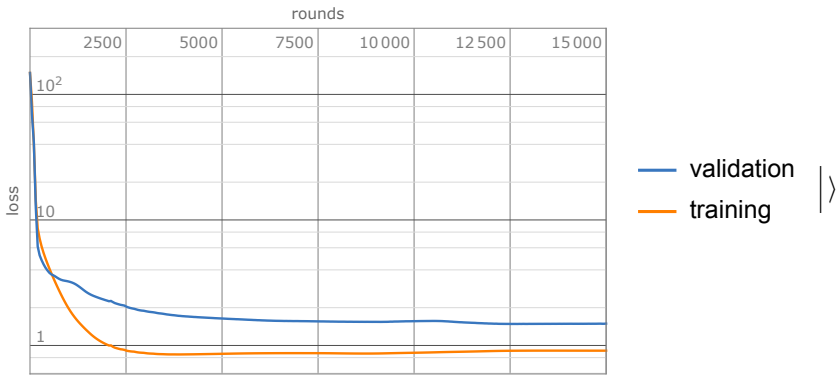
NetTrain Results

summary	batches: 60 000, rounds: 15 000, time: 1.2min, examples/s: 53 811
data	training examples: 256, validation examples: 80, processed examples: 3 840 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 9.09×10^{-1}
validation	loss: 1.48



In[1364]:=
```
trainedNetSimple400v3["FinalPlots"]
trainedNetSimple400v3["TotalTrainingTime"]
trainedNetSimple400v3["RoundMeasurements"]
best = trainedNetSimple400v3["BestValidationRound"]
trainedNetSimple400v3["ValidationLossList"][[best]]
NetInformation[trainedNetSimple400v3["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNetSimple400v3["TrainedNet"], "SummaryGraphic"]
```

Out[1364]=



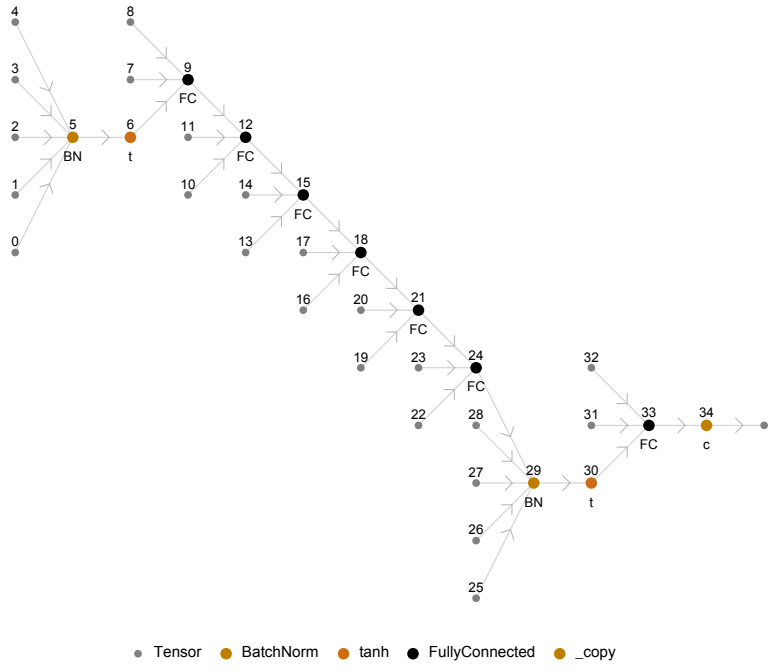
Out[1365]= 71.3604

Out[1366]= ⟨| Loss → 0.908735 |⟩

Out[1367]= 12 853

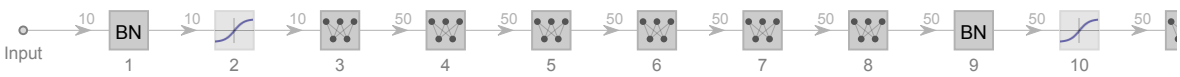
Out[1368]= 1.46082

Out[1369]=



Tensor BatchNorm tanh FullyConnected _copy

Out[1370]=



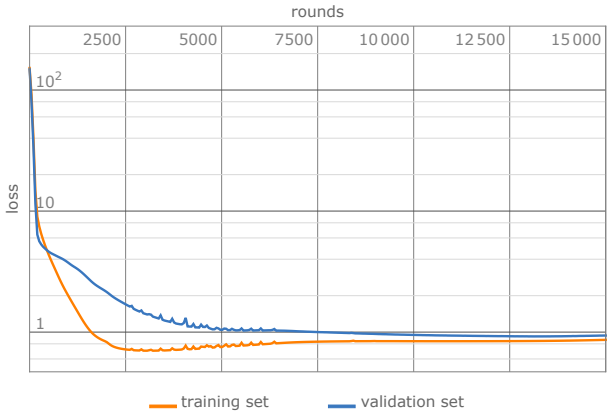
In[1371]:=
```
netSimple400v4 = NetChain[{BatchNormalizationLayer[],
    Tanh, 50, 50, 50, 50, 50, BatchNormalizationLayer[], Tanh, 1}]
trainedNetSimple400v4 = NetTrain[netSimple400v4, finalTrain,
  All, ValidationSet → finaltest, MaxTrainingRounds → 15 000,
  Method → {"ADAM", "LearningRate" → 0.0005, "L2Regularization" → 0.05}]
```

Out[1371]=
NetChain[uninitialized
Input port: array of rank ≥ 1
Output port: vector (size: 1)
Number of layers: 10]

Out[1372]=

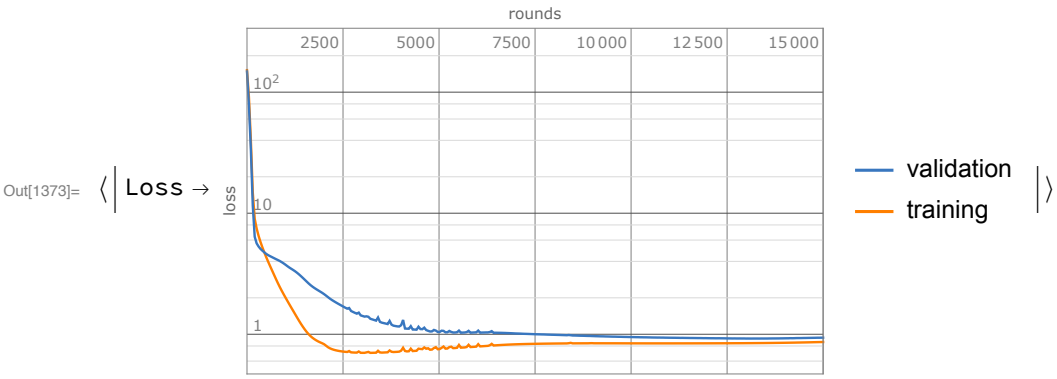
NetTrain Results

summary	batches: 60 000, rounds: 15 000, time: 1.2min, examples/s: 55 549
data	training examples: 256, validation examples: 80, processed examples: 3 840 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 8.63×10^{-1}
validation	loss: 9.37×10^{-1}



In[1373]:=
```
trainedNetSimple400v4["FinalPlots"]
trainedNetSimple400v4["TotalTrainingTime"]
trainedNetSimple400v4["RoundMeasurements"]
best = trainedNetSimple400v4["BestValidationRound"]
trainedNetSimple400v4["ValidationLossList"][[best]]
NetInformation[trainedNetSimple400v4["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNetSimple400v4["TrainedNet"], "SummaryGraphic"]
```

Out[1373]=



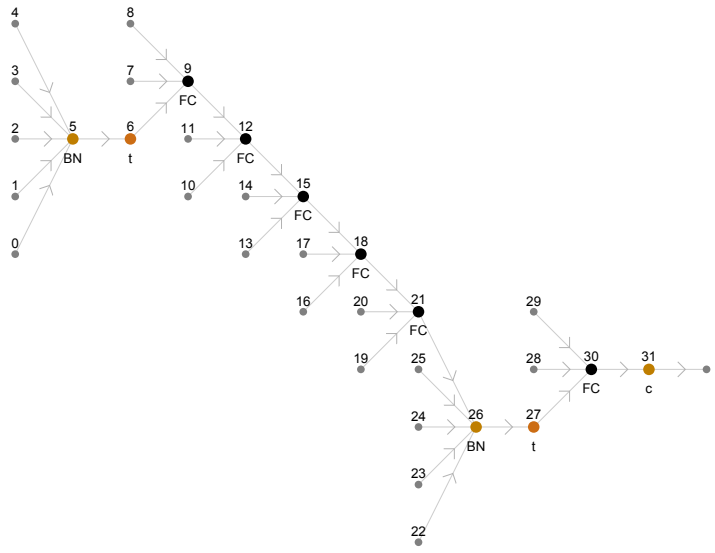
Out[1374]= 69.128

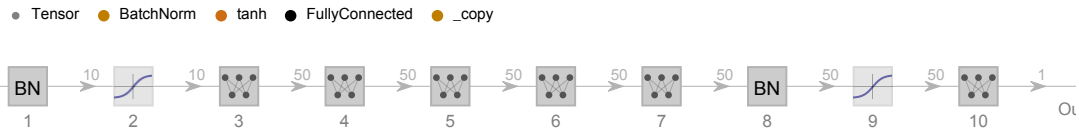
Out[1375]= ⟨| Loss → 0.863038 |⟩

Out[1376]= 13 218

Out[1377]= 0.919938

Out[1378]=





Out[1379]=

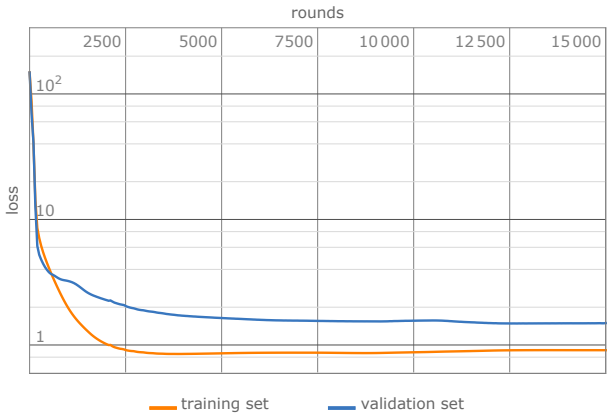
In[1380]:=
```
netSimple400v5 = NetChain[{BatchNormalizationLayer[],
    Tanh, 50, 50, 50, 50, 50, 50, BatchNormalizationLayer[], Tanh, 1}]
trainedNetSimple400v5 = NetTrain[netSimple400v5, finalTrain,
  All, ValidationSet → finaltest, MaxTrainingRounds → 15 000,
  Method → {"ADAM", "LearningRate" → 0.0005, "L2Regularization" → 0.05}]
```

Out[1380]=
```
NetChain[ uninitialized
  Input port:              array of rank ≥ 1
  Output port:             vector (size: 1)
  Number of layers:        11 ]
```

Out[1381]=

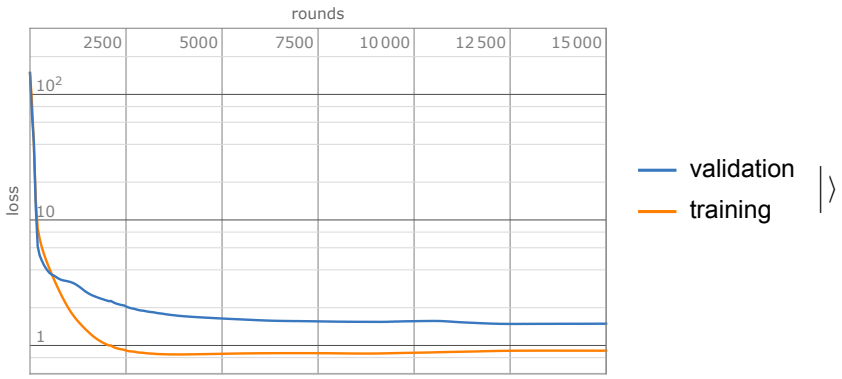
NetTrain Results

summary	batches: 60 000, rounds: 15 000, time: 1.2min, examples/s: 54 887
data	training examples: 256, validation examples: 80, processed examples: 3 840 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 9.09×10^{-1}
validation	loss: 1.48



In[1382]:=
```
trainedNetSimple400v5["FinalPlots"]
trainedNetSimple400v5["TotalTrainingTime"]
trainedNetSimple400v5["RoundMeasurements"]
best = trainedNetSimple400v5["BestValidationRound"]
trainedNetSimple400v5["ValidationLossList"][[best]]
NetInformation[trainedNetSimple400v5["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNetSimple400v5["TrainedNet"], "SummaryGraphic"]
```

Out[1382]=



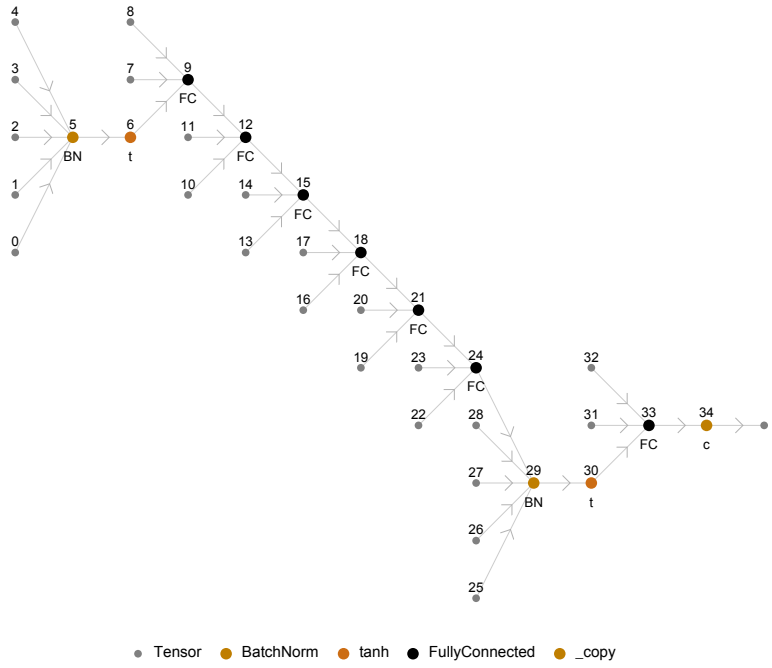
Out[1383]= 69.9616

Out[1384]= ⟨| Loss → 0.908735 |⟩

Out[1385]= 12 853

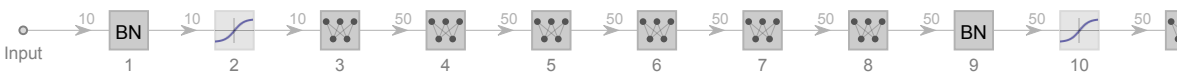
Out[1386]= 1.46082

Out[1387]=



Tensor BatchNorm tanh FullyConnected _copy

Out[1388]=



In[1389]:=
```
netSimple400v6 = NetChain[{BatchNormalizationLayer[], Tanh,
    100, 100, 100, 100, 100, BatchNormalizationLayer[], Tanh, 1}]
trainedNetSimple400v6 = NetTrain[netSimple400v6, finalTrain,
  All, ValidationSet → finaltest, MaxTrainingRounds → 15 000,
  Method → {"ADAM", "LearningRate" → 0.0005, "L2Regularization" → 0.05}]
```

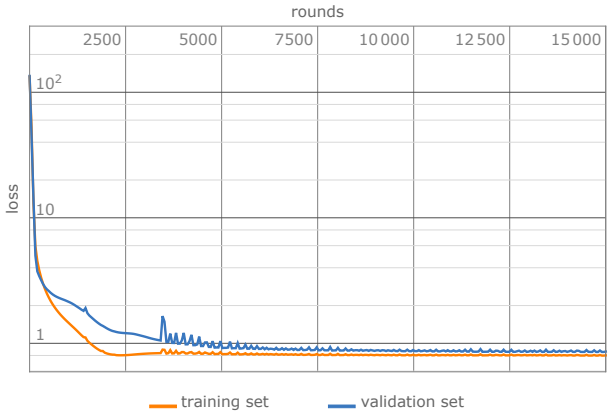
Out[1389]=

NetChain[uninitialized | Input port: array of rank ≥ 1 | Output port: vector (size: 1) | Number of layers: 10]

Out[1390]=

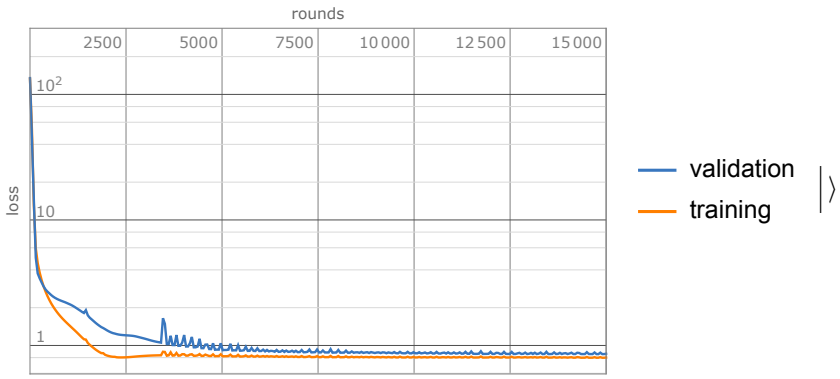
NetTrain Results

summary	batches: 60 000, rounds: 15 000, time: 1.3min, examples/s: 49 423
data	training examples: 256, validation examples: 80, processed examples: 3 840 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 8.13×10^{-1}
validation	loss: 9.07×10^{-1}



In[1391]:=
```
trainedNetSimple400v6["FinalPlots"]
trainedNetSimple400v6["TotalTrainingTime"]
trainedNetSimple400v6["RoundMeasurements"]
best = trainedNetSimple400v6["BestValidationRound"]
trainedNetSimple400v6["ValidationLossList"][[best]]
NetInformation[trainedNetSimple400v6["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNetSimple400v6["TrainedNet"], "SummaryGraphic"]
```

Out[1391]=



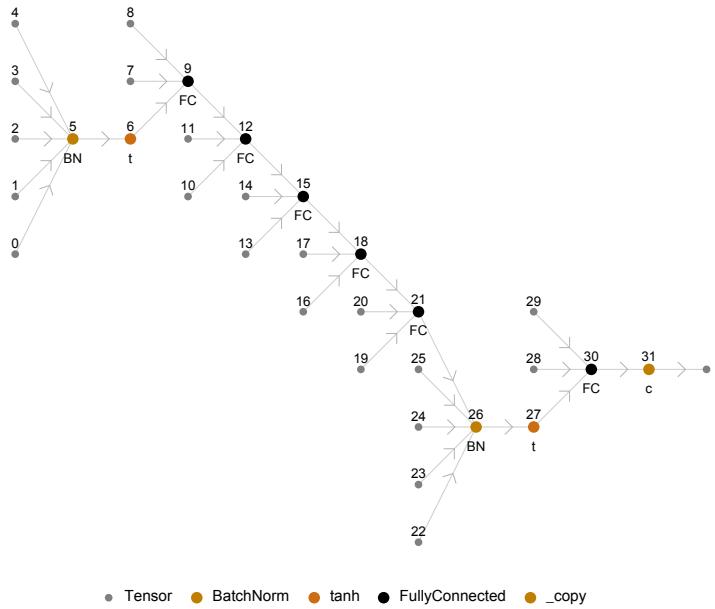
Out[1392]= 77.6971

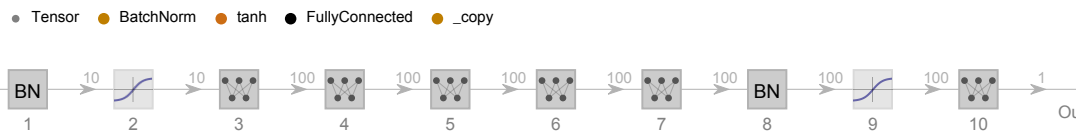
Out[1393]= ⟨| Loss → 0.812835 |⟩

Out[1394]= 14 281

Out[1395]= 0.827167

Out[1396]=





In[1398]:=
```
netSimple400v7 = NetChain[{BatchNormalizationLayer[], Tanh,
    100, 100, 100, 100, 100, 100, BatchNormalizationLayer[], Tanh, 1}]
trainedNetSimple400v7 = NetTrain[netSimple400v7, finalTrain,
  All, ValidationSet → finaltest, MaxTrainingRounds → 15 000,
  Method → {"ADAM", "LearningRate" → 0.0005, "L2Regularization" → 0.05}]
```

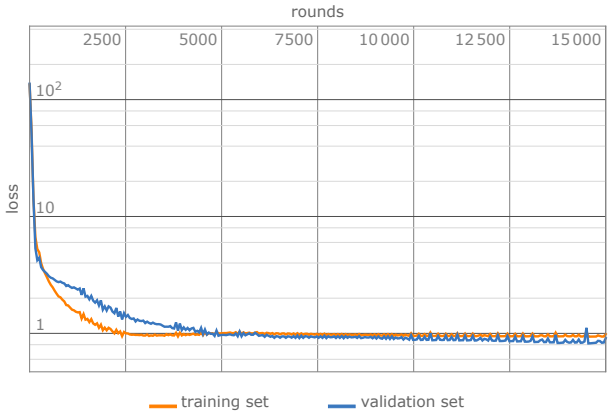
Out[1398]=

NetChain[uninitialized | Input port: array of rank ≥ 1 | Output port: vector (size: 1) | Number of layers: 11]

Out[1399]=

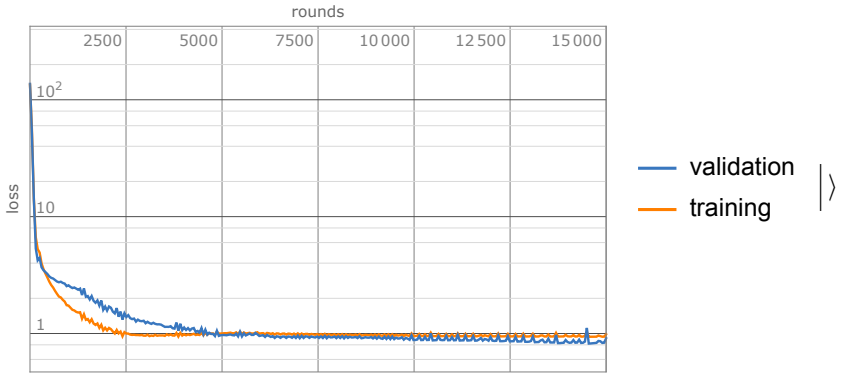
NetTrain Results

summary	batches: 60 000, rounds: 15 000, time: 1.5min, examples/s: 41 845
data	training examples: 256, validation examples: 80, processed examples: 3 840 000, skipped examples: 0
method	ADAM optimizer, batch size 64, CPU
round	loss: 9.38×10^{-1}
validation	loss: 8.41×10^{-1}



In[1400]:=
```
trainedNetSimple400v7["FinalPlots"]
trainedNetSimple400v7["TotalTrainingTime"]
trainedNetSimple400v7["RoundMeasurements"]
best = trainedNetSimple400v7["BestValidationRound"]
trainedNetSimple400v7["ValidationLossList"][[best]]
NetInformation[trainedNetSimple400v7["TrainedNet"], "MXNetNodeGraphPlot"]
NetInformation[trainedNetSimple400v7["TrainedNet"], "SummaryGraphic"]
```

Out[1400]=



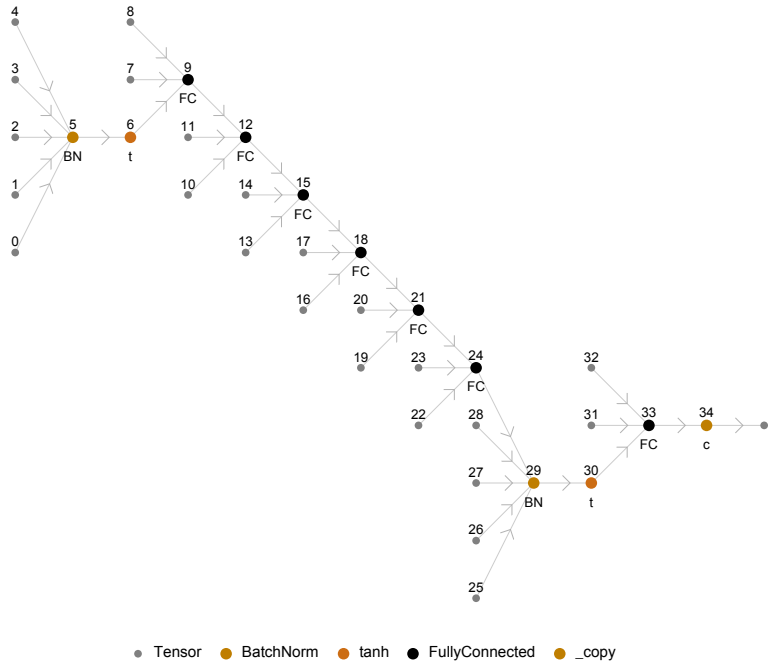
Out[1401]= 91.7673

Out[1402]= ⟨| Loss → 0.937657 |⟩

Out[1403]= 13 707

Out[1404]= 0.760799

Out[1405]=



Tensor BatchNorm tanh FullyConnected _copy

Out[1406]=