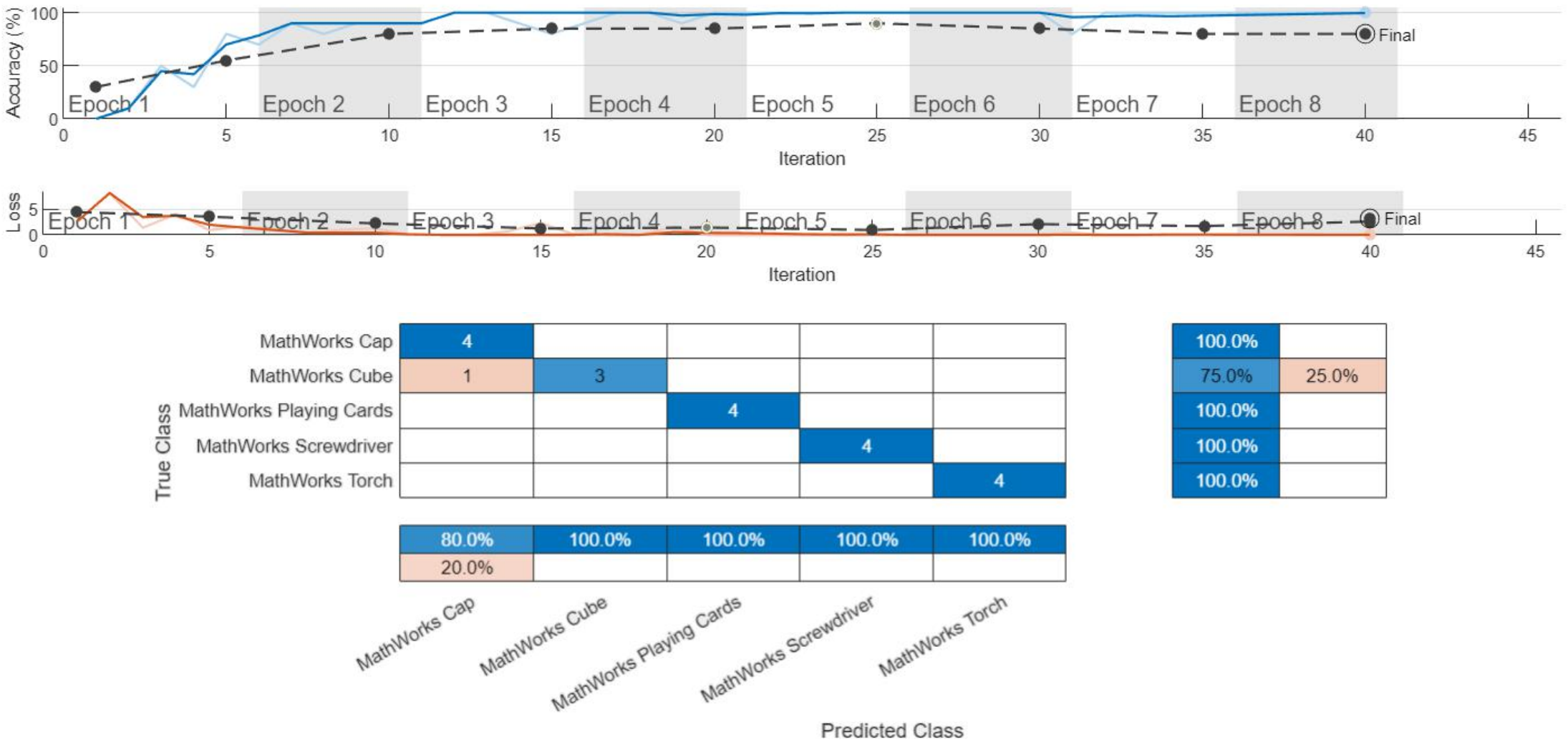
DL Experiments

with Experiment Manager App

(DL) Experiments play important roles in papers



Doing DL Experiments

We need …

- Keep track of training parameters
- Reuse training data across multiple networks
- Analyze and compare results

Experiment Manager App

The Experiment Manager app enables you to create deep learning experiments to train networks under various initial conditions and compare the results. For example, you can use deep learning experiments to:

- Sweep through a range of hyperparameter values or use Bayesian optimization to find optimal training options. Bayesian optimization requires Statistics and Machine Learning Toolbox™.
- Use the built-in function trainNetwork or define your own custom training function.
- Compare the results of using different data sets or test different deep network architectures.

Experiment Manager

EXPERIMENT MANAGER

New | Open | Save | Duplicate — FILE
Layout — ENVIRONMENT
Use Parallel — PARALLEL
Restart All Canceled | Stop — RUN
Training Plot | Confusion Matrix — REVIEW RESULTS
Filter — FILTER
Annotations — ANNOTATIONS
Export — EXPORT

EXPERIMENT BROWSER

- DeepLearningProject
  - Experiment1
    - Result4 (Running)
    - Result3
    - Result2
    - Result1
  - Experiment2
    - Result2
    - Result1

Experiment1 | Experiment1 | Result4

Exhaustive Sweep Result

Experiment1
(View Experiment Source)
Image Classification by Sweeping Hyperparameters

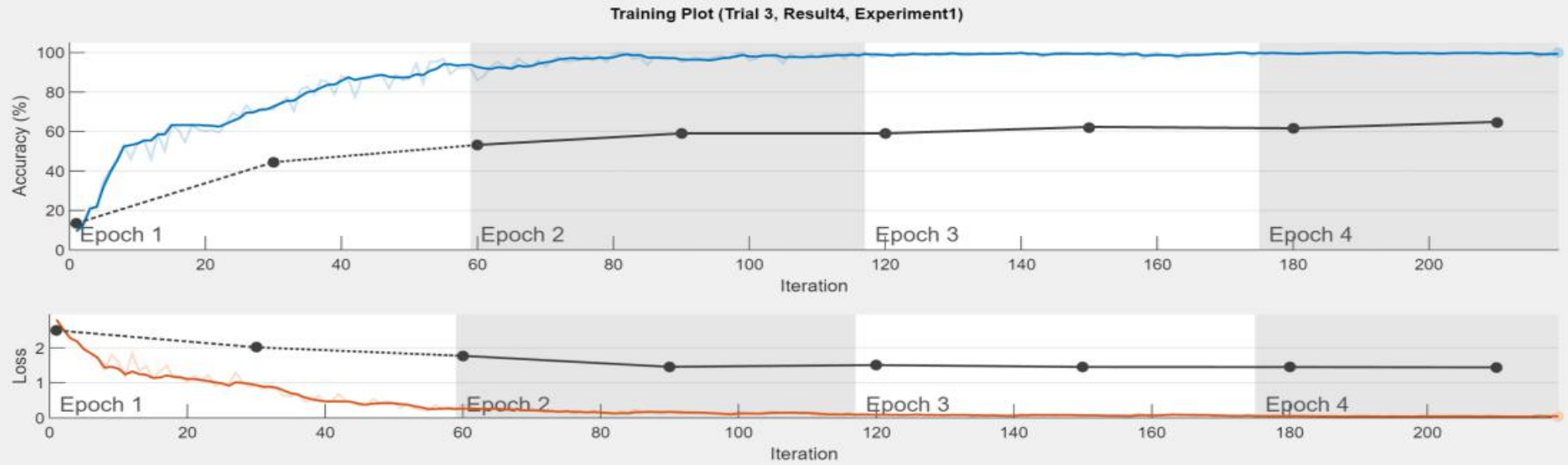
Start: 11/10/2020, 10:12:07 AM

2/6 Trials

Complete	2	Stopped	0	Error	0
Running	2	Queued	2	Canceled	0

Trial	Status	Progress	Elapsed Time	myInitialLearnRate	Training Accuracy (%)	Training Loss	Validation Accuracy (%)	Validation Loss
1	Complete	100.0%	0 hr 1 min 21 sec	0.0025	98.4375	0.1282	56.7200	1.3626
2	Complete	100.0%	0 hr 1 min 27 sec	0.0050	99.2188	0.0664	60.1600	1.4052
3	Running	73.4%	0 hr 0 min 45 sec	0.0075				
4	Running	65.2%	0 hr 0 min 37 sec	0.0100				
5	Queued	0.0%		0.0125				
6	Queued	0.0%		0.0150				

VISUALIZATIONS



Experiment Manager App

Experiment Manager organizes your experiments and results in a project.

- You can store several experiments in the same project.
- Each experiment contains a set of *results* for each time that you run the experiment.
- Each set of results consists of one or more *trials* corresponding to a different combination of hyperparameters.

By default, Experiment Manager runs one trial at a time. If you have Parallel Computing Toolbox™, you can configure your experiment to run multiple trials simultaneously. Running an experiment in parallel allows you to use MATLAB® while the training is in progress.

Open the Experiment Manager App

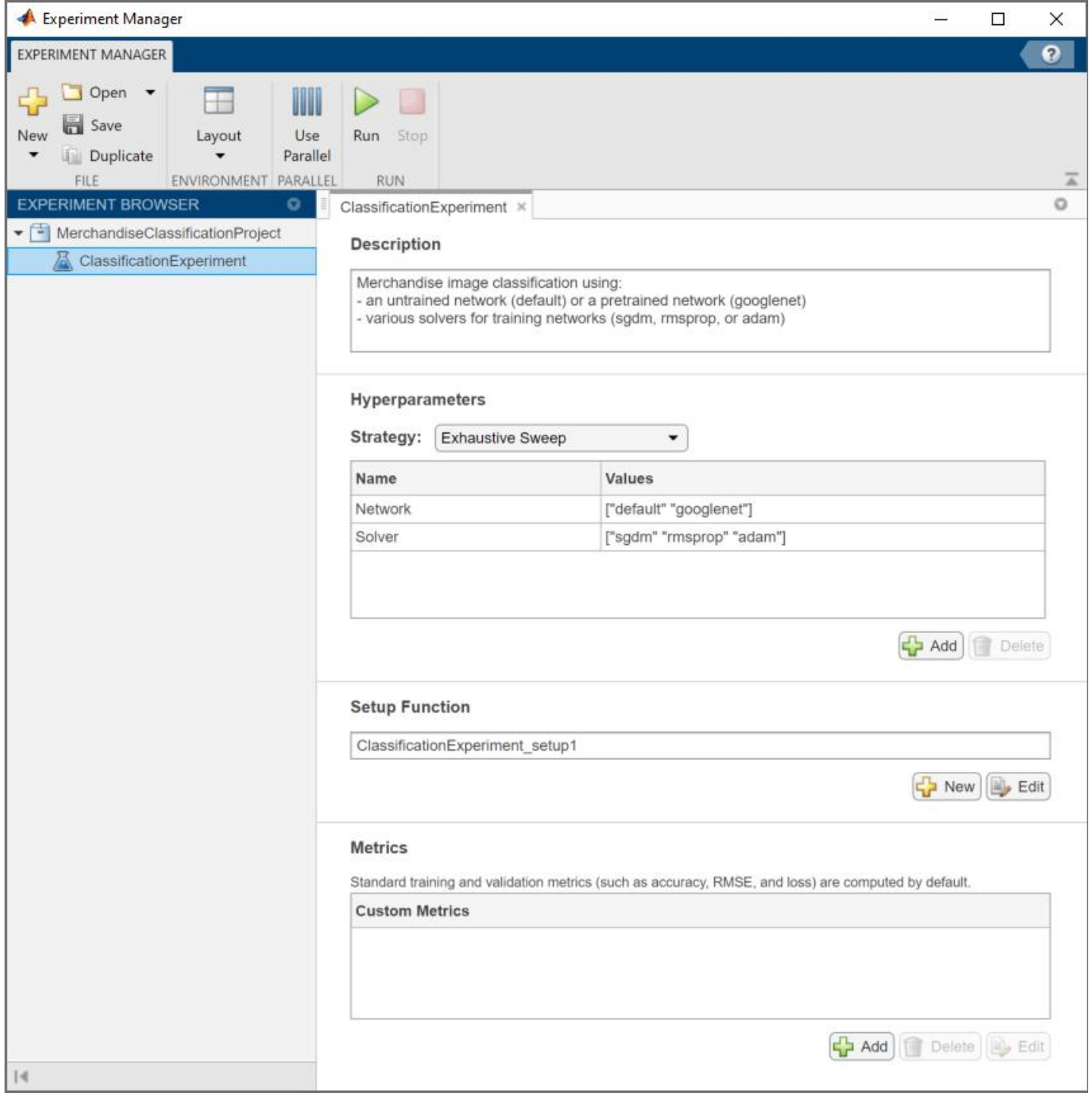
- MATLAB Toolstrip: On the Apps tab, under Machine Learning and Deep Learning, click the app icon.
- MATLAB command prompt: Enter experimentManager.

Experiment Manager Use cases

- Create a Deep Learning Experiment for Classification
- Create a Deep Learning Experiment for Regression
- Tune Experiment Hyperparameters by Using Bayesian Optimization

Create a Deep Learning Experiment for Classification

```
cd(setupExample('nnet/ExperimentManagerClassificationExample'));
setupExpMgr('MerchandiseClassificationProject');
```


Experiment Manager
EXPERIMENT MANAGER
New
Open
Save
Duplicate
FILE
Layout
ENVIRONMENT
Use Parallel
PARALLEL
Run
Stop
RUN
EXPERIMENT BROWSER
MerchandiseClassificationProject
ClassificationExperiment
ClassificationExperiment
Description
Merchandise image classification using:
- an untrained network (default) or a pretrained network (googlenet)
- various solvers for training networks (sgdm, rmsprop, or adam)
Hyperparameters
Strategy:
Exhaustive Sweep
Name
Values
Network
["default" "googlenet"]
Solver
["sgdm" "rmsprop" "adam"]
Add
Delete
Setup Function
ClassificationExperiment_setup1
New
Edit
Metrics
Standard training and validation metrics (such as accuracy, RMSE, and loss) are computed by default.
Custom Metrics
Add
Delete
Edit

Create a Deep Learning Experiment for Classification

The Description field contains a textual description of the experiment. For this example, the description is:

> Merchandise image classification using:
> - an untrained network (default) or a pretrained network (googlenet)
> - various solvers for training networks (sgdm, rmsprop, or adam)

The Hyperparameters section specifies the strategy (Exhaustive Sweep) and hyperparameter values to use for the experiment. When you run the experiment, Experiment Manager trains the network using every combination of hyperparameter values specified in the hyperparameter table. This example uses two hyperparameters:

Network specifies the network to train. The options include "default" (the default network provided by the experiment template for image classification) and "googlenet" (a pretrained GoogLeNet network with modified layers for transfer learning).

Solver indicates the algorithm used to train the network. The options include "sgdm" (stochastic gradient descent with momentum), "rmsprop" (root mean square propagation), and "adam" (adaptive moment estimation).

Create a Deep Learning Experiment for Classification

The Setup Function configures the training data, network architecture, and training options for the experiment. To inspect the setup function, under Setup Function, click Edit. The setup function opens in MATLAB Editor.

The input to the setup function is a structure with fields from the **hyperparameter table**. The setup function returns three outputs that you use to train a network for image classification problems. The setup function has three sections.

Load Training Data defines image datastores containing the training and validation data. This example loads images from the file MerchData.zip. This small data set contains 75 images of MathWorks merchandise, belonging to five different classes. The images are of size 227-by-227-by-3. For more information on this data set, see Image Data Sets.

Define Network Architecture defines the architecture for a convolutional neural network for deep learning classification. In this example, the choice of network to train depends on the value of the hyperparameter Network.

Specify Training Options defines a trainingOptions object for the experiment. The example trains the network for 8 epochs using the algorithm specified by the Solver entry in the hyperparameter table.

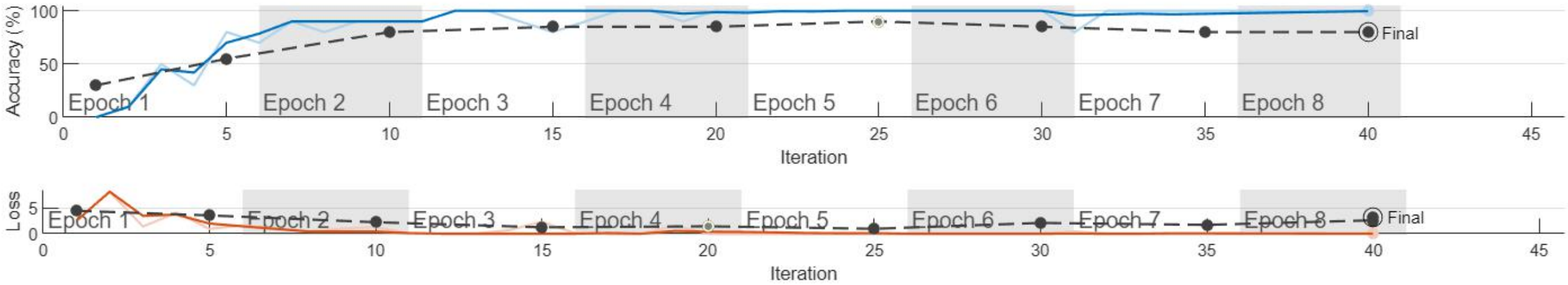
Create a Deep Learning Experiment for Classification

When you run the experiment, Experiment Manager trains the network defined by the setup function six times. Each trial uses a different combination of hyperparameter values. By default, Experiment Manager runs one trial at a time.

Trial	Status	Progress	Elapsed Time	Network	Solver	Training Accuracy (%)	Training Loss	Validation Accuracy (%)	Validation Loss
1	Complete	100.0%	0 hr 0 min 34 sec	default	sgdm	100.0000	1.3113e-7	80.0000	3.1913
2	Complete	100.0%	0 hr 0 min 29 sec	googlenet	sgdm	100.0000	0.0021	95.0000	0.0954
3	Complete	100.0%	0 hr 0 min 9 sec	default	rmsprop	100.0000	4.3908e-5	80.0000	2.0756
4	Complete	100.0%	0 hr 0 min 28 sec	googlenet	rmsprop	100.0000	0.0014	95.0000	0.1546
5	Complete	100.0%	0 hr 0 min 10 sec	default	adam	100.0000	0.0000	85.0000	2.3943
6	Complete	100.0%	0 hr 0 min 28 sec	googlenet	adam	100.0000	0.0002	95.0000	0.1333

Create a Deep Learning Experiment for Classification

While the experiment is running, click Training Plot to display the training plot and track the progress of each trial.



Create a Deep Learning Experiment for Classification

To find the best result for your experiment, sort the table of results by validation accuracy.

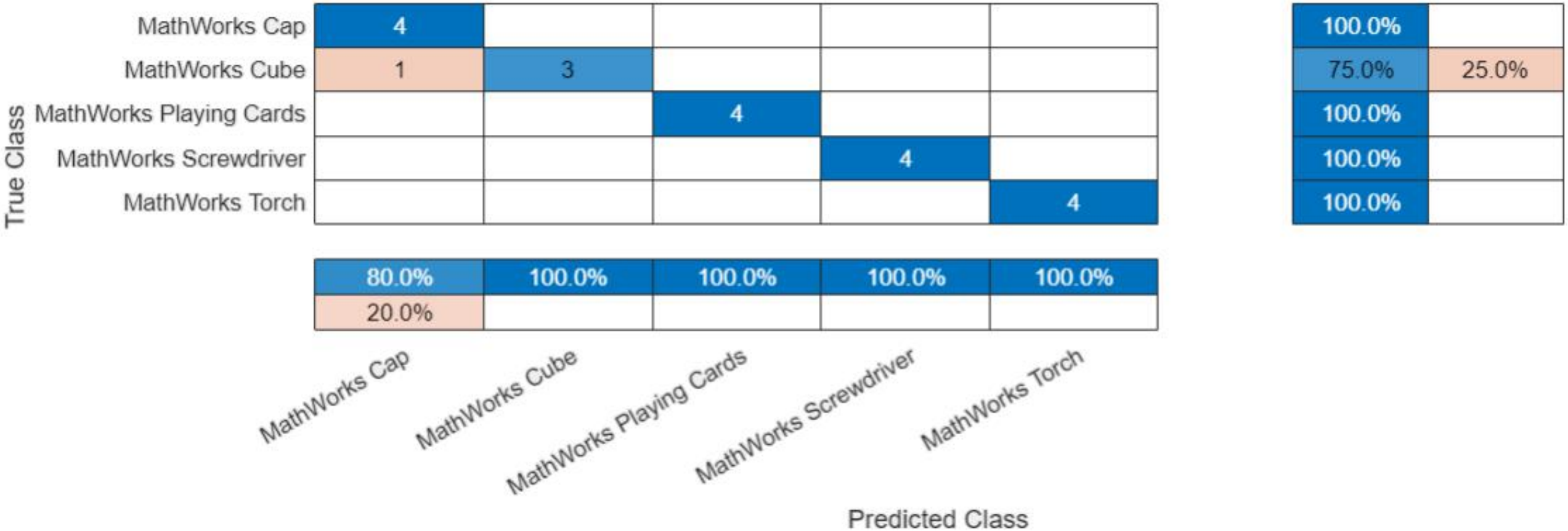
- Point to the Validation Accuracy column.
- Click the triangle icon.
- Select Sort in Descending Order.

The trial with the highest validation accuracy appears at the top of the results table.

Trial	Status	Progress	Elapsed Time	Network	Solver	Training Accuracy (%)	Training Loss	Validation Accuracy (%)	Validation Loss
2	Complete	100.0%	0 hr 0 min 29 sec	googlenet	sgdm	100.0000	0.0021	95.0000	0.0954
4	Complete	100.0%	0 hr 0 min 28 sec	googlenet	rmsprop	100.0000	0.0014	95.0000	0.1546
6	Complete	100.0%	0 hr 0 min 28 sec	googlenet	adam	100.0000	0.0002	95.0000	0.1333
5	Complete	100.0%	0 hr 0 min 10 sec	default	adam	100.0000	0.0000	85.0000	2.3943
1	Complete	100.0%	0 hr 0 min 34 sec	default	sgdm	100.0000	1.3113e-7	80.0000	3.1913
3	Complete	100.0%	0 hr 0 min 9 sec	default	rmsprop	100.0000	4.3908e-5	80.0000	2.0756

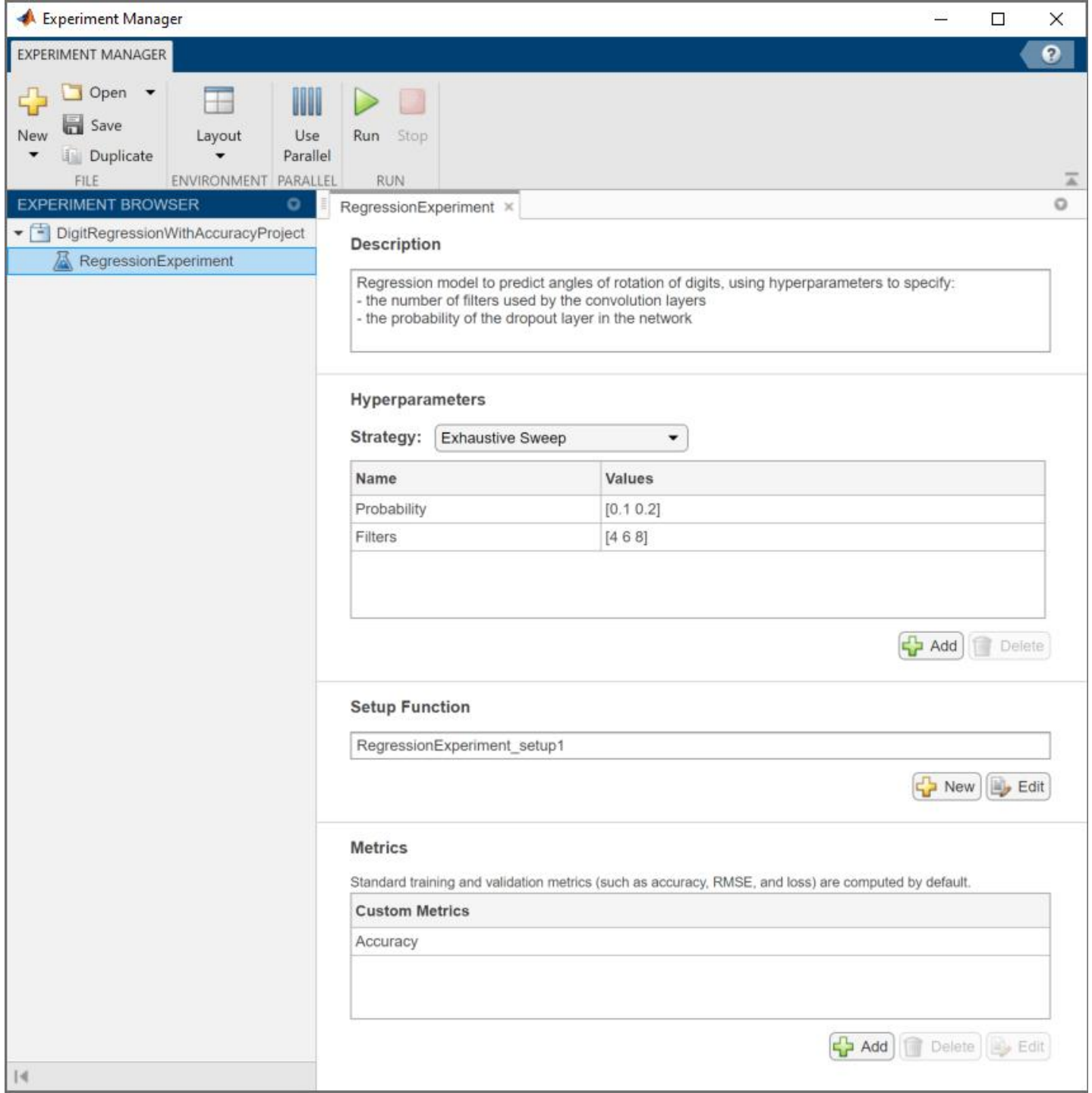
Create a Deep Learning Experiment for Classification

To display the confusion matrix for this trial, select the top row in the results table and click Confusion Matrix.



Create a Deep Learning Experiment for Regression

```
cd(setupExample('nnet/ExperimentManagerClassificationExample'));
setupExpMgr('MerchandiseClassificationProject');
```


Experiment Manager
EXPERIMENT MANAGER
New
Open
Save
Duplicate
FILE
Layout
ENVIRONMENT
Use Parallel
PARALLEL
Run
Stop
RUN
EXPERIMENT BROWSER
DigitRegressionWithAccuracyProject
RegressionExperiment
RegressionExperiment
Description
Regression model to predict angles of rotation of digits, using hyperparameters to specify:
- the number of filters used by the convolution layers
- the probability of the dropout layer in the network
Hyperparameters
Strategy:
Exhaustive Sweep
Name
Values
Probability
[0.1 0.2]
Filters
[4 6 8]
Add
Delete
Setup Function
RegressionExperiment_setup1
New
Edit
Metrics
Standard training and validation metrics (such as accuracy, RMSE, and loss) are computed by default.
Custom Metrics
Accuracy
Add
Delete
Edit

Create a Deep Learning Experiment for Regression

The Hyperparameters section specifies the strategy (Exhaustive Sweep) and hyperparameter values to use for the experiment. When you run the experiment, Experiment Manager trains the network using every combination of hyperparameter values specified in the hyperparameter table. This example uses two hyperparameters:

Probability sets the probability of the dropout layer in the neural network. By default, the values for this hyperparameter are specified as [0.1 0.2].

Filters indicates the number of filters used by the first convolution layer in the neural network. In the subsequent convolution layers, the number of filters is a multiple of this value. By default, the values of this hyperparameter are specified as [4 6 8].

Create a Deep Learning Experiment for Regression

The Setup Function configures the training data, network architecture, and training options for the experiment. To inspect the setup function, under Setup Function, click Edit. The setup function opens in MATLAB Editor.

The input to the setup function is a structure with fields from the hyperparameter table. The setup function returns four outputs that you use to train a network for image regression problems. The setup function has three sections.

Load Training Data defines the training and validation data for the experiment as 4-D arrays. The training and validation data sets each contain 5000 images of digits from 0 to 9. The regression values correspond to the angles of rotation of the digits.

Define Network Architecture defines the architecture for a convolutional neural network for regression.

Specify Training Options defines a trainingOptions object for the experiment. The example trains the network for 30 epochs. The learning rate is initially 0.001 and drops by a factor of 0.1 after 20 epochs. The software trains the network on the training data and calculates the root mean squared error (RMSE) and loss on the validation data at regular intervals during training. The validation data is not used to update the network weights.

Create a Deep Learning Experiment for Regression

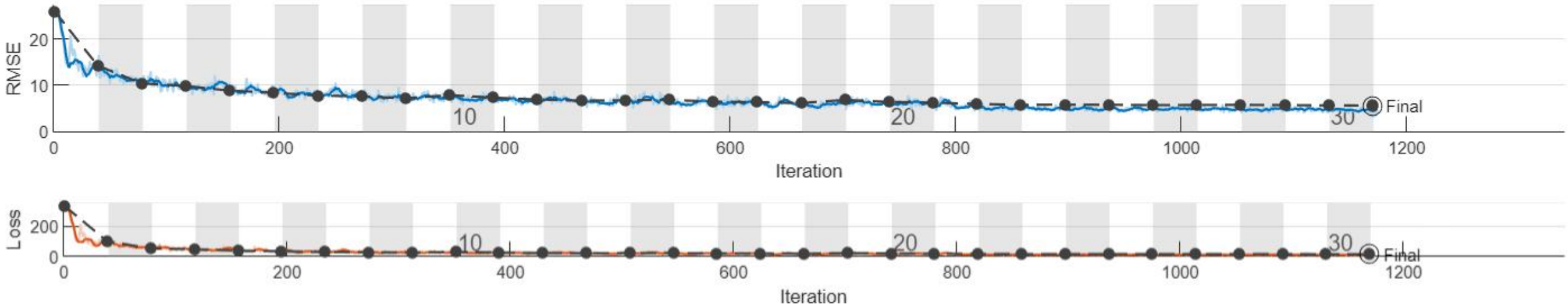
The Metrics section specifies optional functions that evaluate the results of the experiment. Experiment Manager evaluates these functions each time it finishes training the network. To inspect a metric function, select the name of the metric function and click Edit. The metric function opens in MATLAB Editor.

This example includes a metric function Accuracy that determines the percentage of angle predictions within an acceptable error margin from the true angles. By default, the function uses a threshold of 10 degrees.

Create a Deep Learning Experiment for Regression

When you run the experiment, Experiment Manager trains the network defined by the setup function six times. Each trial uses a different combination of hyperparameter values. By default, Experiment Manager runs one trial at a time.

Trial	Status	Progress	Elapsed Time	Probability	Filters	Training RMSE	Training Loss	Validation RMSE	Validation Loss	Accuracy
1	Complete	100.0%	0 hr 1 min 7 sec	0.1000	4.0000	4.4200	9.7682	5.5695	15.5097	93.5200
2	Complete	100.0%	0 hr 1 min 36 sec	0.2000	4.0000	5.5942	15.6475	5.9898	17.9387	90.8600
3	Complete	100.0%	0 hr 1 min 41 sec	0.1000	6.0000	3.9594	7.8382	4.7431	11.2487	96.5400
4	Complete	100.0%	0 hr 1 min 38 sec	0.2000	6.0000	4.2466	9.0170	4.8345	11.6863	95.4600
5	Complete	100.0%	0 hr 1 min 32 sec	0.1000	8.0000	3.2151	5.1684	4.4256	9.7931	97.3200
6	Complete	100.0%	0 hr 1 min 41 sec	0.2000	8.0000	3.6607	6.7003	4.6408	10.7685	96.3600



Create a Deep Learning Experiment for Regression

To test the performance of an individual trial, export the trained network and display a box plot of the residuals for each digit class.

Select the trial with the highest accuracy.

On the Experiment Manager toolstrip, click Export.

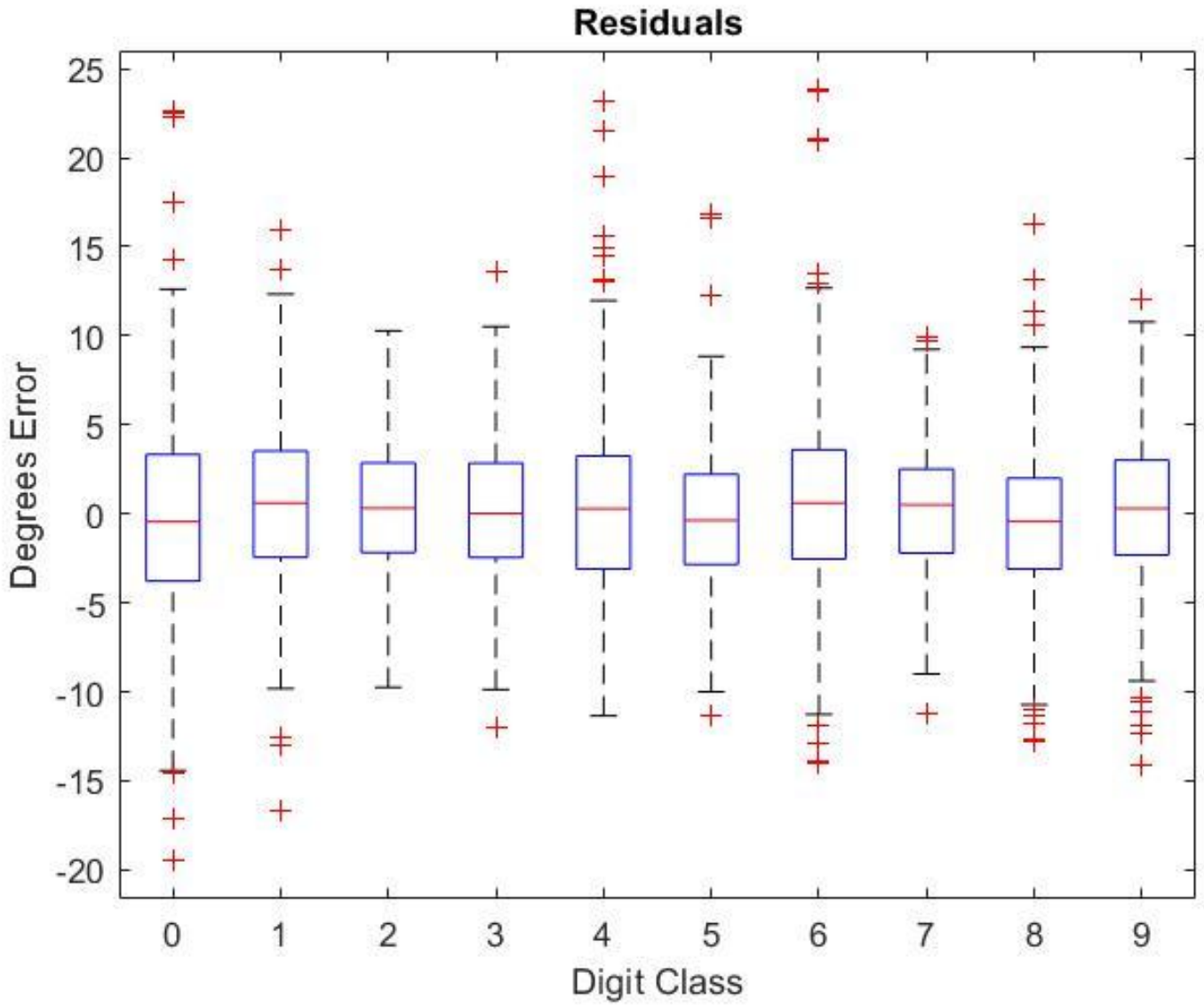
In the dialog window, enter the name of a workspace variable for the exported network. The default name is trainedNetwork.

Use the exported network as the input to the function plotResiduals. For instance, in the MATLAB Command Window, enter:

plotResiduals(trainedNetwork)

The function creates a residual box plot for each digit. The digit classes with highest accuracy have a mean close to zero and little variance.

Create a Deep Learning Experiment for Regression



Tune Experiment Hyperparameters by Using Bayesian Optimization

```
cd(setupExample('nnet/ExpMgrBayesOptExample'));
setupExpMgr('CIFARBayesianOptimizationProject');
```

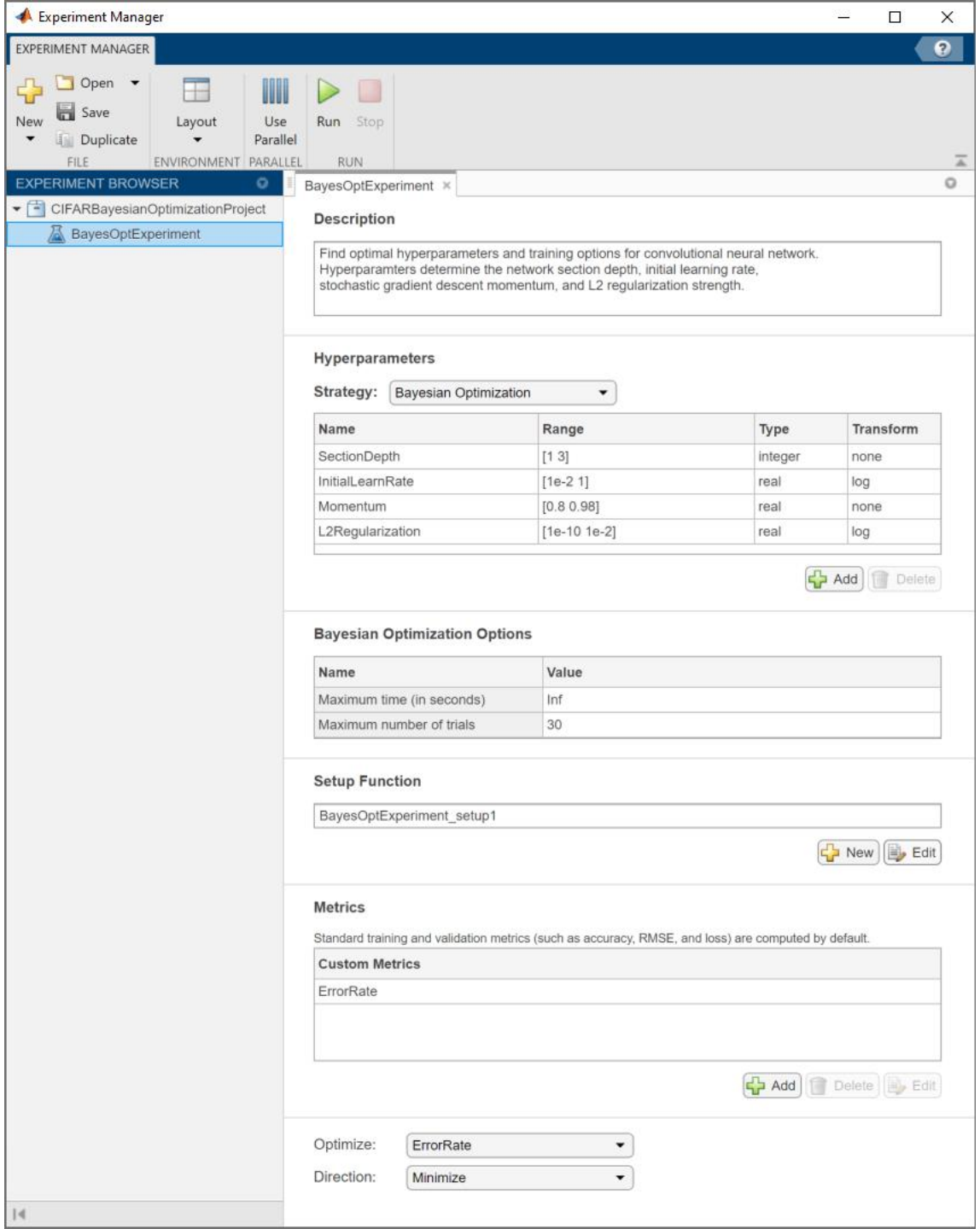
Tune Experiment Hyperparameters by Using Bayesian Optimization

This example shows how to use Bayesian optimization in Experiment Manager to find optimal network hyperparameters and training options for convolutional neural networks.

Bayesian optimization provides an alternative strategy to sweeping hyperparameters in an experiment. You specify a range of values for each hyperparameter and select a metric to optimize, and Experiment Manager searches for a combination of hyperparameters that optimizes your selected metric. Bayesian optimization requires Statistics and Machine Learning Toolbox.

Tune Experiment Hyperparameters by Using Bayesian Optimization

In this example, you train a network to classify images from the CIFAR-10 data set. The experiment uses Bayesian optimization to find the combination of hyperparameters that minimizes a custom metric function. The hyperparameters include options of the training algorithm, as well as parameters of the network architecture itself. The custom metric function determines the classification error on a randomly chosen test set.


Experiment Manager
EXPERIMENT MANAGER
New
Open
Save
Duplicate
FILE
Layout
ENVIRONMENT
Use Parallel
PARALLEL
Run
Stop
RUN
EXPERIMENT BROWSER
CIFARBayesianOptimizationProject
BayesOptExperiment
BayesOptExperiment
Description
Find optimal hyperparameters and training options for convolutional neural network.
Hyperparamters determine the network section depth, initial learning rate,
stochastic gradient descent momentum, and L2 regularization strength.
Hyperparameters
Strategy: Bayesian Optimization
Name | Range | Type | Transform
SectionDepth | [1 3] | integer | none
InitialLearnRate | [1e-2 1] | real | log
Momentum | [0.8 0.98] | real | none
L2Regularization | [1e-10 1e-2] | real | log
Add
Delete
Bayesian Optimization Options
Name | Value
Maximum time (in seconds) | Inf
Maximum number of trials | 30
Setup Function
BayesOptExperiment_setup1
New
Edit
Metrics
Standard training and validation metrics (such as accuracy, RMSE, and loss) are computed by default.
Custom Metrics
ErrorRate
Add
Delete
Edit
Optimize: ErrorRate
Direction: Minimize

Tune Experiment Hyperparameters by Using Bayesian Optimization

This example uses these hyperparameters:

SectionDepth — This parameter controls the depth of the network. The total number of layers in the network is 9*SectionDepth+7. In the experiment setup function, the number of convolutional filters in each layer is proportional to 1/sqrt(SectionDepth), so the number of parameters and the required amount of computation for each iteration are roughly the same for different section depths.

InitialLearnRate — The best learning rate can depend on your data as well as the network you are training.

Momentum — Stochastic gradient descent momentum adds inertia to the parameter updates by having the current update contain a contribution proportional to the update in the previous iteration. The inertial effect results in smoother parameter updates and a reduction of the noise inherent to stochastic gradient descent.

L2Regularization — Use L2 regularization to prevent overfitting. Search the space of regularization strength to find a good value. Data augmentation and batch normalization also help regularize the network.

Tune Experiment Hyperparameters by Using Bayesian Optimization

Under Bayesian Optimization Options, you can specify the duration of the experiment by entering the maximum time (in seconds) and the maximum number of trials to run. To best utilize the power of Bayesian optimization, you should perform at least 30 objective function evaluations.

The Setup Function configures the training data, network architecture, and training options for the experiment. To inspect the setup function, under Setup Function, click Edit. The setup function opens in MATLAB® Editor.

Tune Experiment Hyperparameters by Using Bayesian Optimization

In this example, the setup function has three sections.

- Load Training Data downloads and extracts images and labels from the CIFAR-10 data set. The data set is about 175 MB. Depending on your internet connection, the download process can take some time. For the training data, this example creates an augmentedImageDatastore by applying random translations and horizontal reflections. Data augmentation helps prevent the network from overfitting and memorizing the exact details of the training images. To enable network validation, the example uses 5000 images with no augmentation. For more information on this data set, see Image Data Sets.

Tune Experiment Hyperparameters by Using Bayesian Optimization

In this example, the setup function has three sections.

- Define Network Architecture defines the architecture for a convolutional neural network for deep learning classification. In this example, the network to train has three sections, each with SectionDepth identical convolutional layers. Each convolutional layer is followed by a batch normalization layer and a ReLU layer. The convolutional layers have added padding so that their spatial output size is always the same as the input size. Between sections, max pooling layers downsample the spatial dimensions by a factor of two.

Tune Experiment Hyperparameters by Using Bayesian Optimization

In this example, the setup function has three sections.

- Specify Training Options defines a trainingOptions object for the experiment using the values for the training options 'InitialLearnRate', 'Momentum', and 'L2Regularization' generated by the Bayesian optimization algorithm. The example trains the network for a fixed number of epochs, validating once per epoch and lowering the learning rate by a factor of 10 during the last epochs to reduce the noise of the parameter updates and allow the network parameters to settle down closer to a minimum of the loss function.

Tune Experiment Hyperparameters by Using Bayesian Optimization

The Metrics section specifies optional functions that evaluate the results of the experiment. Experiment Manager evaluates these functions each time it finishes training the network. To inspect a metric function, select the name of the metric function and click Edit. The metric function opens in MATLAB Editor.

This example includes the custom metric function ErrorRate. This function selects 5000 test images and labels at random, evaluates the trained network on these images, and calculates the proportion of images that the network misclassifies.

Tune Experiment Hyperparameters by Using Bayesian Optimization

```
function metricOutput = ErrorRate(trialInfo)
    datadir = tempdir;
    [~,~,XTest,YTest] = loadCIFARData(datadir);
    idx = randperm(numel(YTest),5000);
    XTest = XTest(:,:,:,idx);
    YTest = YTest(idx);
    YPredicted = classify(trialInfo.trainedNetwork,XTest);
    metricOutput = 1 - mean(YPredicted == YTest);
end
```

The Optimize and Direction fields indicate the metric that the Bayesian optimization algorithm uses as an objective function. For this experiment, Experiment Manager seeks to minimize the value of the ErrorRate metric.

Tune Experiment Hyperparameters by Using Bayesian Optimization

When you run the experiment, Experiment Manager searches for the best combination of hyperparameters with respect to the chosen metric. Each trial in the experiment uses a new combination of hyperparameter values based on the results of the previous trials. By default, Experiment Manager runs one trial at a time.

Tune Experiment Hyperparameters by Using Bayesian Optimization

Trial	Status	Progress	Elapsed Time	SectionDepth	InitialLearnRate	Momentum	L2Regularization	Training Accuracy (%)	Training Loss	Validation Accuracy (%)	Validation Loss	ValidationError
1	Complete	100.0%	0 hr 19 min 53 sec	2.0000	0.0127	0.8874	1.1585e-10	84.3750	0.5015	79.0000	0.6320	0.2075
2	Complete	100.0%	0 hr 19 min 57 sec	2.0000	0.0123	0.8289	0.0023	79.2969	0.5435	78.9000	0.6148	0.2175
3	Complete	100.0%	0 hr 24 min 29 sec	3.0000	0.3917	0.9169	0.0001	86.3281	0.4310	82.4600	0.5380	0.1786
4	Complete	100.0%	0 hr 19 min 23 sec	2.0000	0.4529	0.8289	0.0016	77.7344	0.6417	74.8400	0.7470	0.2550
5	Complete	100.0%	0 hr 23 min 57 sec	3.0000	0.6210	0.9141	1.4641e-5	89.0625	0.3588	81.7200	0.5696	0.1837
6	Complete	100.0%	0 hr 24 min 3 sec	3.0000	0.0380	0.9321	0.0002	86.7188	0.4229	82.3000	0.5338	0.1834
7	Complete	100.0%	0 hr 24 min 12 sec	3.0000	0.1274	0.8180	8.4335e-10	88.2813	0.3483	81.1200	0.5626	0.1856
8	Complete	100.0%	0 hr 15 min 25 sec	1.0000	0.2898	0.9680	4.4529e-10	82.4219	0.5038	77.5400	0.6936	0.2251
9	Complete	100.0%	0 hr 24 min 33 sec	3.0000	0.1608	0.9760	0.0092	34.7656	1.6631	33.4200	1.6412	0.6658
10	Complete	100.0%	0 hr 19 min 57 sec	2.0000	0.0103	0.9232	0.0001	83.5938	0.4843	78.5200	0.6288	0.2151
11	Complete	100.0%	0 hr 24 min 10 sec	3.0000	0.7808	0.9373	1.0351e-10	84.7656	0.3998	80.9000	0.5797	0.1891
12	Complete	100.0%	0 hr 24 min 11 sec	3.0000	0.0158	0.8481	2.9635e-10	84.7656	0.4036	79.6800	0.6093	0.2036
13	Complete	100.0%	0 hr 15 min 21 sec	1.0000	0.0102	0.8007	1.8735e-8	68.7500	0.8294	67.5400	0.9167	0.3229
14	Complete	100.0%	0 hr 15 min 36 sec	1.0000	0.9488	0.9401	2.3331e-10	75.7813	0.6522	76.0000	0.7395	0.2395
15	Complete	100.0%	0 hr 24 min 49 sec	3.0000	0.0108	0.8680	0.0037	84.3750	0.4456	80.3800	0.5607	0.1975
16	Complete	100.0%	0 hr 24 min 41 sec	3.0000	0.0101	0.8363	3.0689e-7	82.0313	0.5357	78.5600	0.6377	0.2202
17	Complete	100.0%	0 hr 24 min 26 sec	3.0000	0.0103	0.8005	0.0001	83.9844	0.5118	77.9000	0.6444	0.2195
18	Complete	100.0%	0 hr 15 min 19 sec	1.0000	0.0102	0.8647	0.0014	69.5313	0.8087	68.3800	0.9054	0.3156
19	Complete	100.0%	0 hr 24 min 53 sec	3.0000	0.9588	0.9265	7.9158e-7	86.7188	0.4275	80.5200	0.6099	0.1999
20	Complete	100.0%	0 hr 24 min 58 sec	3.0000	0.0107	0.8950	1.0937e-10	83.2031	0.4905	79.2400	0.6159	0.2088
21	Complete	100.0%	0 hr 15 min 34 sec	1.0000	0.0103	0.9046	3.3038e-8	73.8281	0.7165	72.0400	0.8072	0.2820
22	Complete	100.0%	0 hr 24 min 28 sec	3.0000	0.0109	0.8988	0.0088	82.0313	0.5746	80.9600	0.5791	0.1966
23	Complete	100.0%	0 hr 24 min 54 sec	3.0000	0.0102	0.9536	1.2402e-10	85.5469	0.4035	80.6200	0.5742	0.1970
24	Complete	100.0%	0 hr 25 min 1 sec	3.0000	0.0111	0.8294	0.0081	83.9844	0.5242	80.0200	0.5930	0.1993
25	Complete	100.0%	0 hr 24 min 41 sec	3.0000	0.0110	0.9231	0.0001	85.9375	0.4225	80.6200	0.5865	0.1967
26	Complete	100.0%	0 hr 24 min 37 sec	3.0000	0.9413	0.8803	1.3332e-7	86.7188	0.3673	81.6800	0.5456	0.1872
27	Complete	100.0%	0 hr 24 min 34 sec	3.0000	0.6245	0.9203	0.0080	9.3750	2.3060	10.7200	2.3032	0.9000
28	Complete	100.0%	0 hr 24 min 34 sec	3.0000	0.0100	0.9287	6.4635e-9	83.2031	0.4732	80.2600	0.6060	0.2036
29	Complete	100.0%	0 hr 24 min 51 sec	3.0000	0.1353	0.8426	1.4613e-5	87.8906	0.3850	81.5600	0.5512	0.1864
30	Complete	100.0%	0 hr 25 min 26 sec	3.0000	0.9928	0.8062	2.4694e-9	87.5000	0.3331	82.3200	0.5477	0.1844

Tune Experiment Hyperparameters by Using Bayesian Optimization