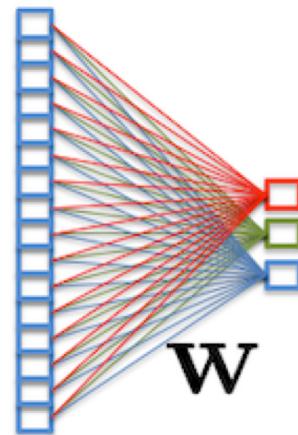
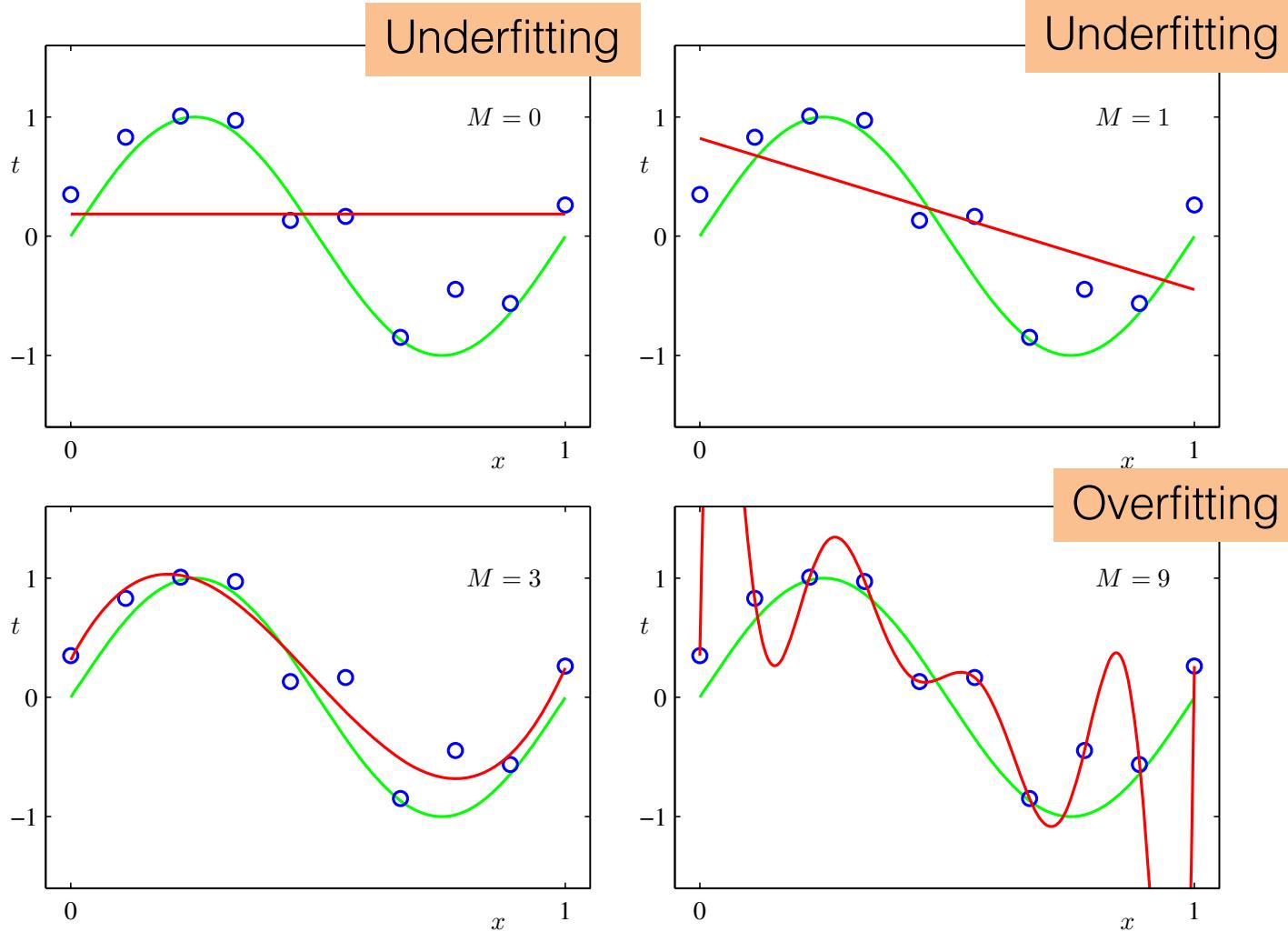


5IF - Deep Learning and Differentiable Programming

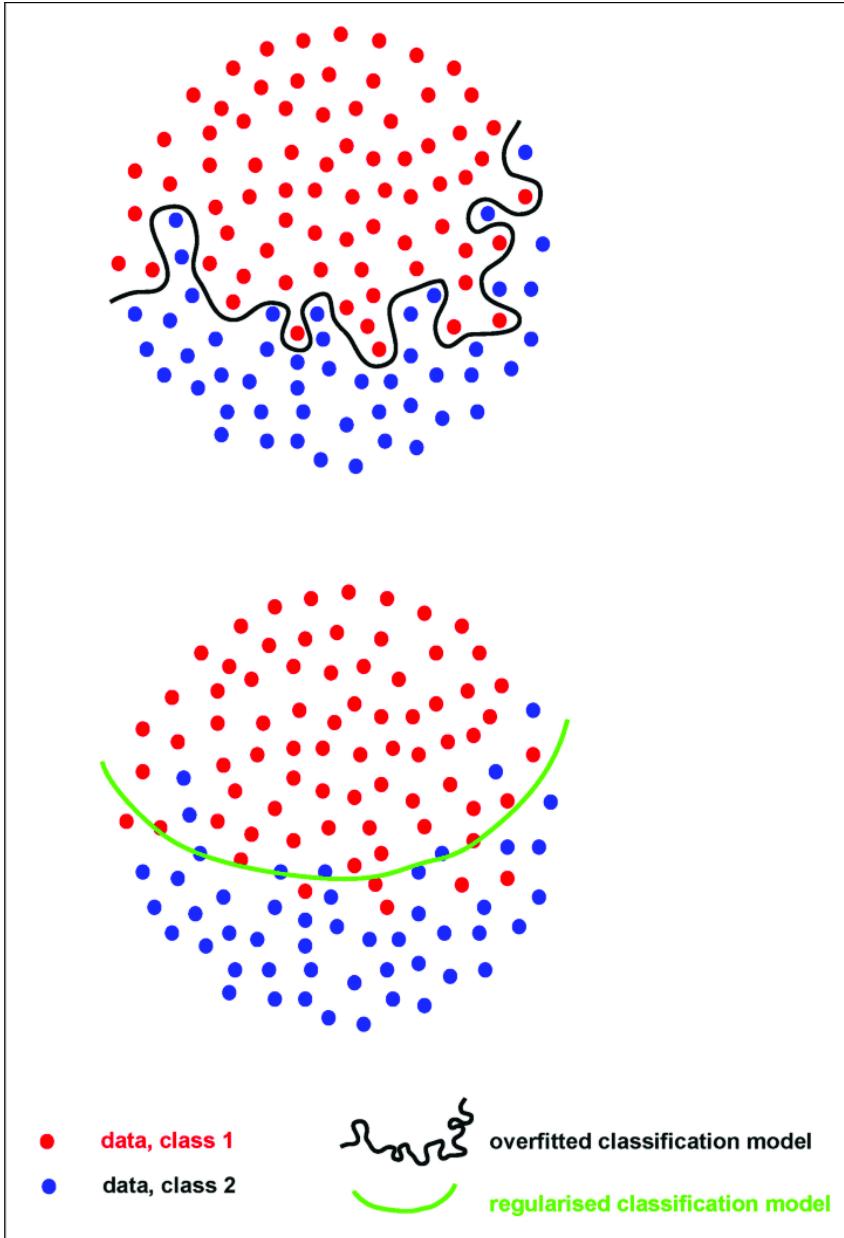
2.4 Train/Validation/Test split - Tensorboard



Under- and Overfitting in Regression



Overfitting in classification

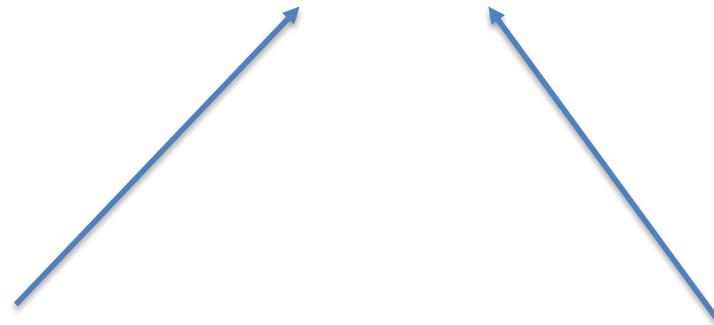


[Figure: Tim Hulsen]

We can overfit on what exactly?

We would like to learn to predict a value y from observed input x

$$y = h(x, \theta)$$



Handcrafted from domain knowledge (e.g. type of model, number of layers, number of units:
architecture)

Learned from training data

Optimization over the training data

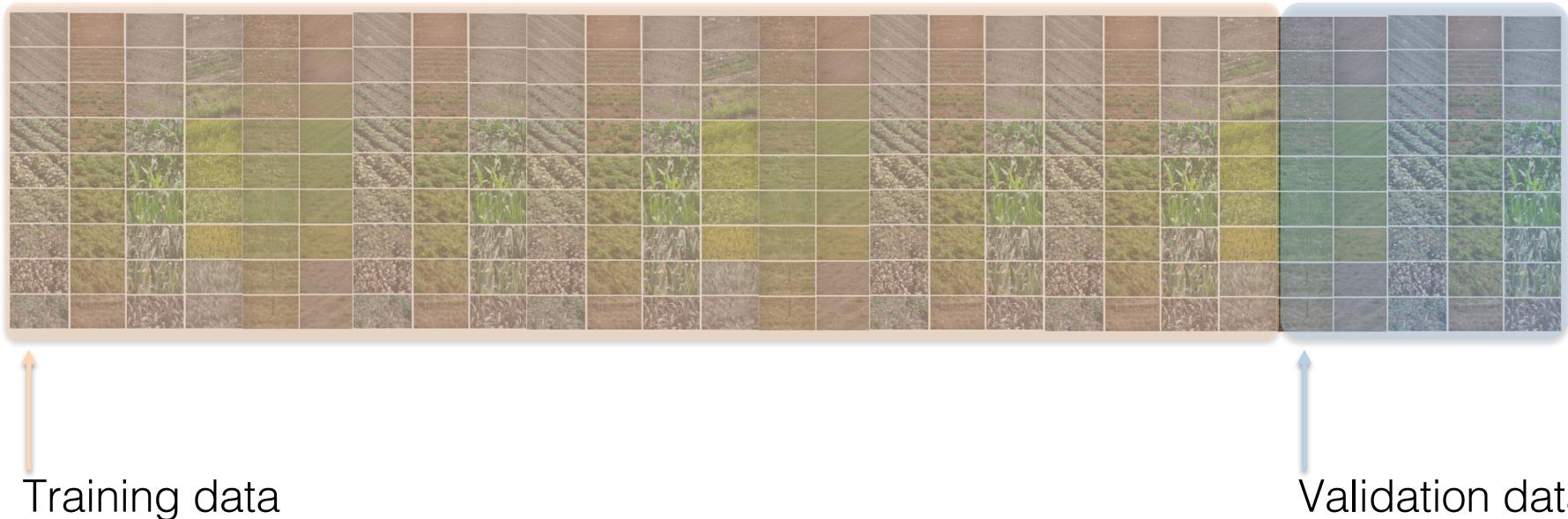
We optimize a loss function \mathcal{L} taking ground-truth targets t and predictions y of a model h over its parameters θ :

$$\theta = \arg \min_{\theta} \mathcal{L}(h(x, \theta), t)$$

Fitting is thus the result of the optimization process ... and every optimization on data process can lead to potential overfitting.
We overfit on the data on which we optimize (here: training data).
The validation data provides a better estimate of the error of the model.

Validation (hold-out) set

We remove parts of the data from training, and use it for evaluation, to check whether we overfit.



Optimization over the validation data

Problem: while working on the problem, we select and optimize network architectures, so a better notation for the function h would be:

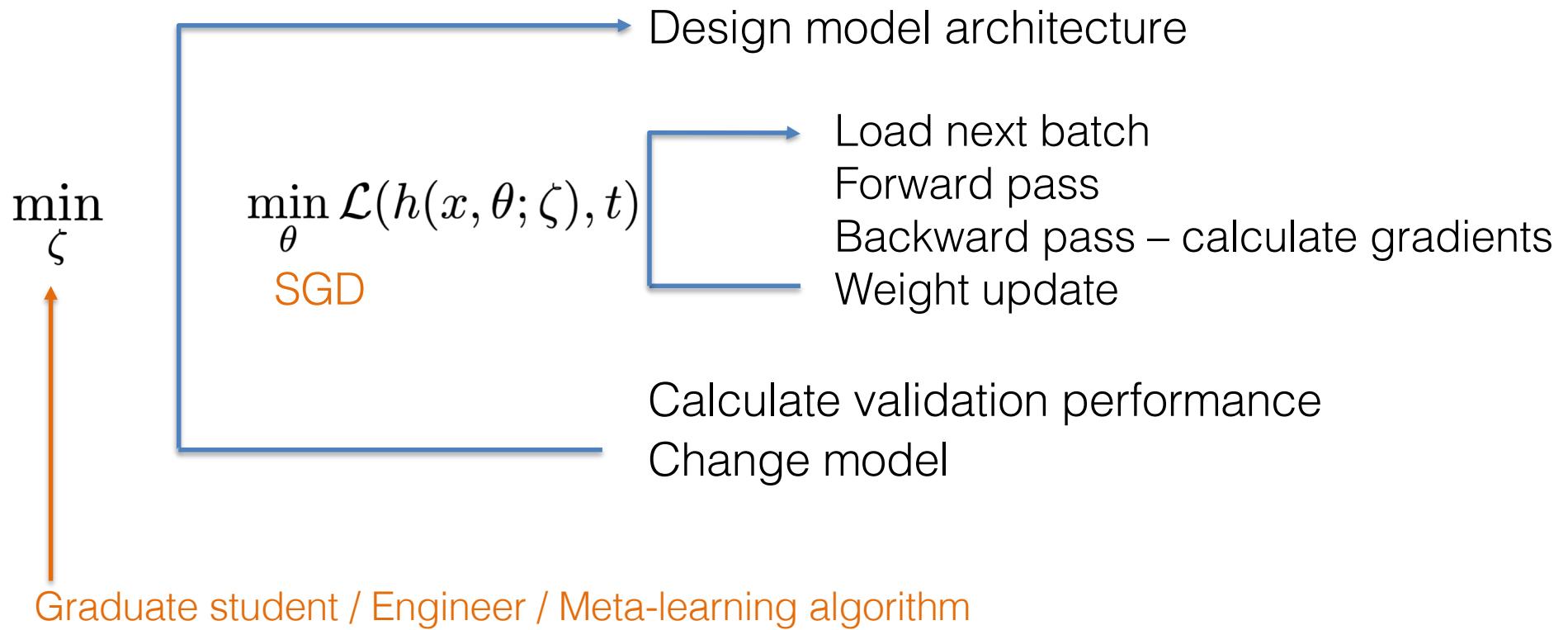
$$h(x, \theta; \zeta)$$

where ζ are the hyper-parameters of the network.

In practice, we really solve the following problem:

$$\min_{\zeta} \min_{\theta} \mathcal{L}(h(x, \theta; \zeta), t)$$

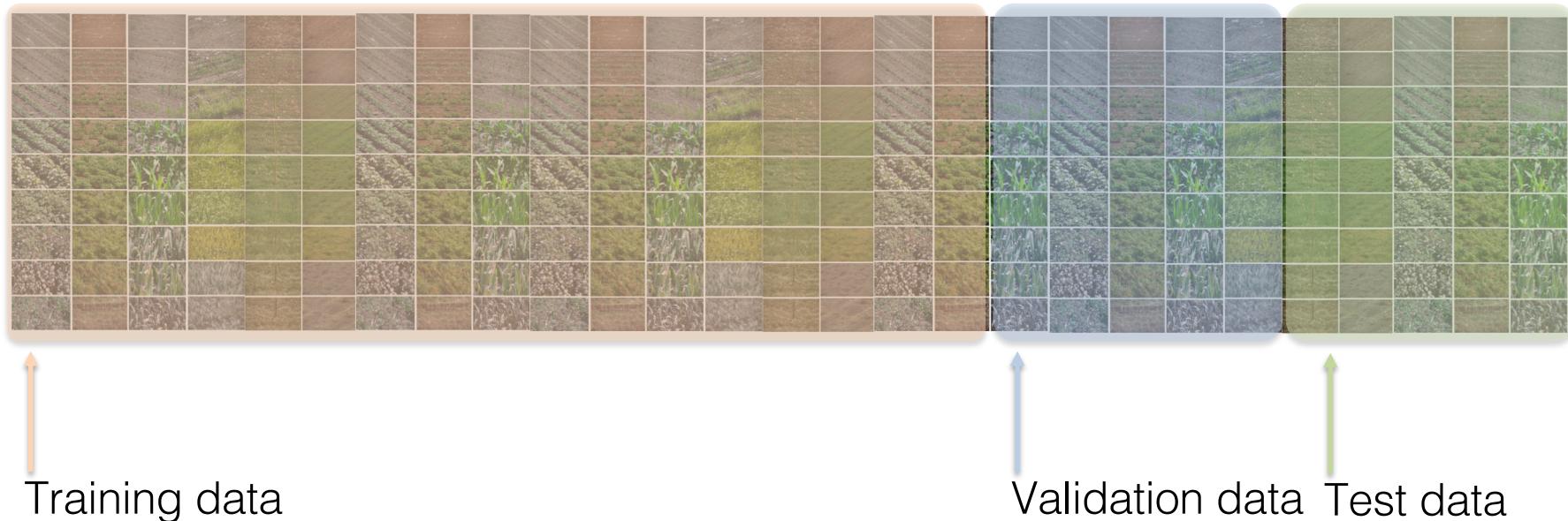
Optimization loops



Validation performance is NOT representative of the performance of the model in the wild! We overfit on the validation data!

Validation (hold-out) set

We remove parts of the data from training, and use it for evaluation, to check whether we overfit.



Visualization of the training process

Running Tensorboard

Install:

```
1 pip3 install tensorboard
```

PyTorch code:

```
1 from torch.utils.tensorboard import SummaryWriter  
2  
3 # Setup a writer and configure a log file directory  
4 writer = SummaryWriter('runs/mnist_mlp_r0.01')
```

Start the tensorboard executable

```
1 tensorboard --logdir=runs
```

The executable sets up a web server on port 6006:

```
1 TensorBoard 2.0.0 at http://localhost:6006/ (Press CTRL+C to quit)
```

Running Tensorboard

We load two different datasets, a training and a validation set:

```
1 # Training set and loader
2 valid_dataset = MNISTDataset ("MNIST-png/testing",
3     transforms.Compose([
4         transforms.ToTensor(),
5         transforms.Normalize((0.1307,), (0.3081,))]))
6 valid_loader = torch.utils.data.DataLoader(
7     valid_dataset,
8     batch_size=BATCHSIZE, shuffle=True)

9
10 # Validation set and loader ("hold out" set)
11 train_dataset = MNISTDataset ("MNIST-png/training",
12     transforms.Compose([
13         transforms.ToTensor(),
14         transforms.Normalize((0.1307,), (0.3081,))]))
15 train_loader = torch.utils.data.DataLoader(
16     train_dataset,
17     batch_size=BATCHSIZE, shuffle=True)
```

Writing into Tensorboard

```
1 for epoch in range(100):
2     for batch_idx, (data, labels) in enumerate(train_loader):
3
4         (...)

5
6     # Print statistics
7     if (batch_idx % STATS_INTERVAL) == 0:
8         train_err=100.0*(1.0-running_correct/count)

9
10    # Call a second loop which iterates over
11    # validation batches, calculating error
12    vloss, verr = calcError(model, valid_loader)

13
14    # Write statistics to the log file
15    writer.add_scalars ('Loss', {
16        'training': running_loss / STATS_INTERVAL,
17        'validation': vloss },
18        epoch * len(train_loader) + batch_idx)

19
20    writer.add_scalars ('Error', {
21        'training': train_err,
22        'validation': verr },
23        epoch * len(train_loader) + batch_idx)
```

Show data download links Ignore outliers in chart scaling

Tooltip sorting method: default

Filter tags (regular expressions supported)

Error

Error

Running Tensorboard

Install:

```
1 pip3 install tensorboard
```

PyTorch code:

```
1 from torch.utils.tensorboard import SummaryWriter  
2  
3 # Setup a writer and configure a log file directory  
4 writer = SummaryWriter('runs/mnist_mlp_r0.01')
```

Start the tensorboard executable

```
1 tensorboard --logdir=runs
```

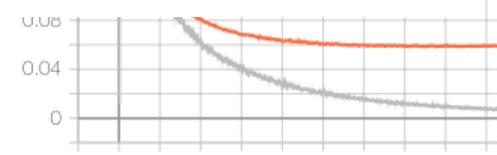
The executable sets up a web server on port 6006:

```
1 TensorBoard 2.0.0 at http://localhost:6006/ (Press CTRL+C to quit)
```

- mnist_mlp_r0.01_h600
- mnist_mlp_r0.01_h600/Loss_training:
- mnist_mlp_r0.01_h600/Loss_validation:
- mnist_mlp_r0.01_h600/Error_training:
- mnist_mlp_r0.01_h600/Error_validation:

TOGGLE ALL RUNS

runs



Logistic Regression underfits

