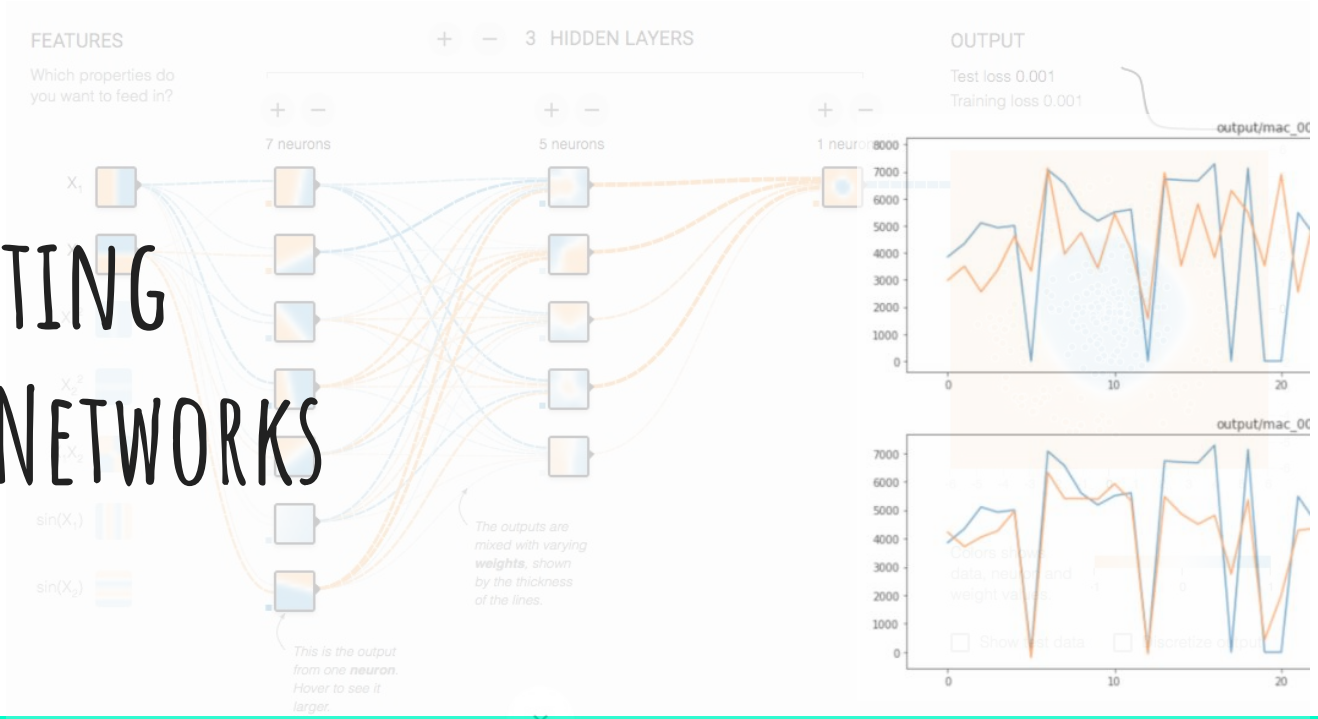
Sales Forecasting with Neural Networks



Grzegorz Gawron, head of data science

VIRTUSLAB

SETUP [OPTIONAL]

- install anaconda
-

```
> git clone https://github.com/gregaw/sales-forecasting-with-nn.git
> cd sales-forecasting-with-nn
> conda create -n mlengine python=2.7 anaconda
> source activate mlengine
> pip install -r requirements.txt
```

- [optional] setup google cloud account (trial available)
  - ml-engine
  - storage
  - gcloud shell

SALES FORECASTING WITH NEURAL NETWORKS

Grzegorz Gawron, head of data science

VIRTUSLAB

ABOUT ME

Grzegorz Gawron
ggawron at virtuslab.com

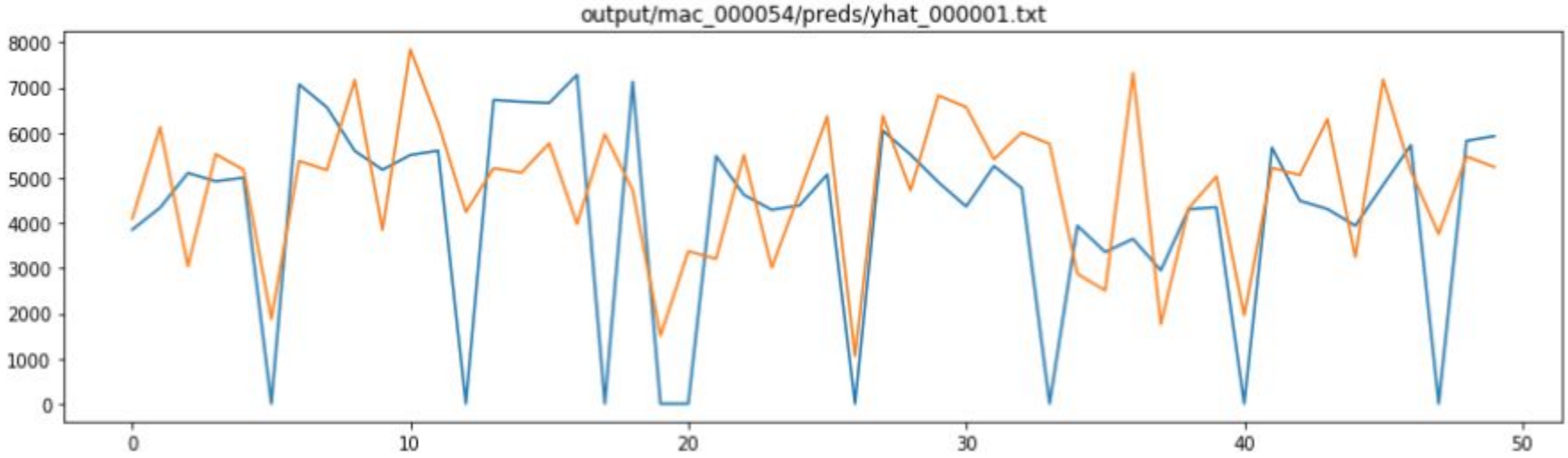
- leading a large data engineering project for a global retailer
- Spark ML pipelines (forecasting, recommendation)
- large hadoop cluster (1000s cores + TBs RAM)
- spark, scala, python

AGENDA

- the problem, data
- the tools
  - jupyter notebook
    (of course!)
  - neural networks
    aka deep learning
  - keras, tensorflow
  - GCP ML-engine
- action

Sales forecasting, data

The Problem - forecasting sales value



The data

https://www.kaggle.com/c/rossmann-store-sales

train.csv

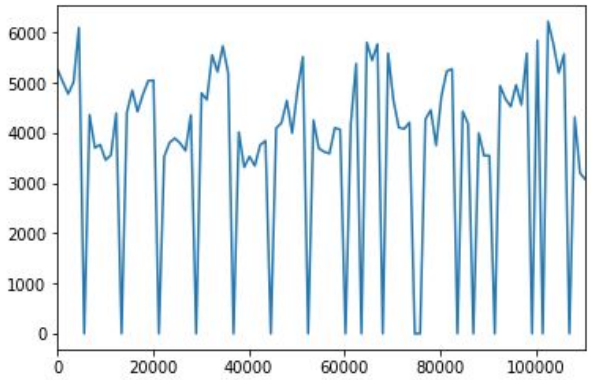
	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
0	1	5	2015-07-31	5263	555	1	1	0	1
1	2	5	2015-07-31	6064	625	1	1	0	1
2	3	5	2015-07-31	8314	821	1	1	0	1
3	4	5	2015-07-31	13995	1498	1	1	0	1

store.csv

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYe
0	1	c	a	1270.0	9.0	2008
1	2	a	a	570.0	11.0	2007
2	3	a	a	14130.0	12.0	2006
3	4	c	c	620.0	9.0	2009

Data Sanity Checking (per store)

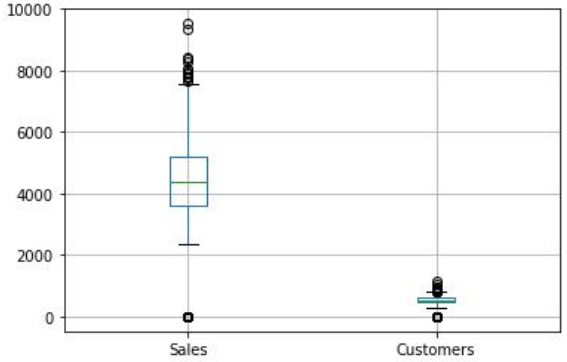
```
df_sales.Sales[:100].plot()
```

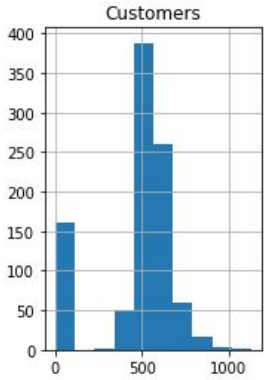


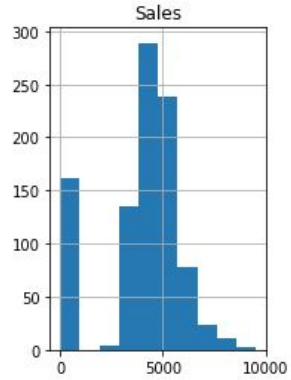

```
df_sales.X.value_counts()
```

```
5     135
4     135
3     135
2     135
7     134
6     134
1     134
Name: DayOfWeek, dtype: int64
1     781
0     161
Name: Open, dtype: int64
0     582
1     360
Name: Promo, dtype: int64
0     915
a      17
b       6
c       4
Name: StateHoliday, dtype: int64
0     749
1     193
Name: SchoolHoliday, dtype: int64
```

```
df_sales[CONTS].boxplot()
df_sales[CONTS].hist()
```



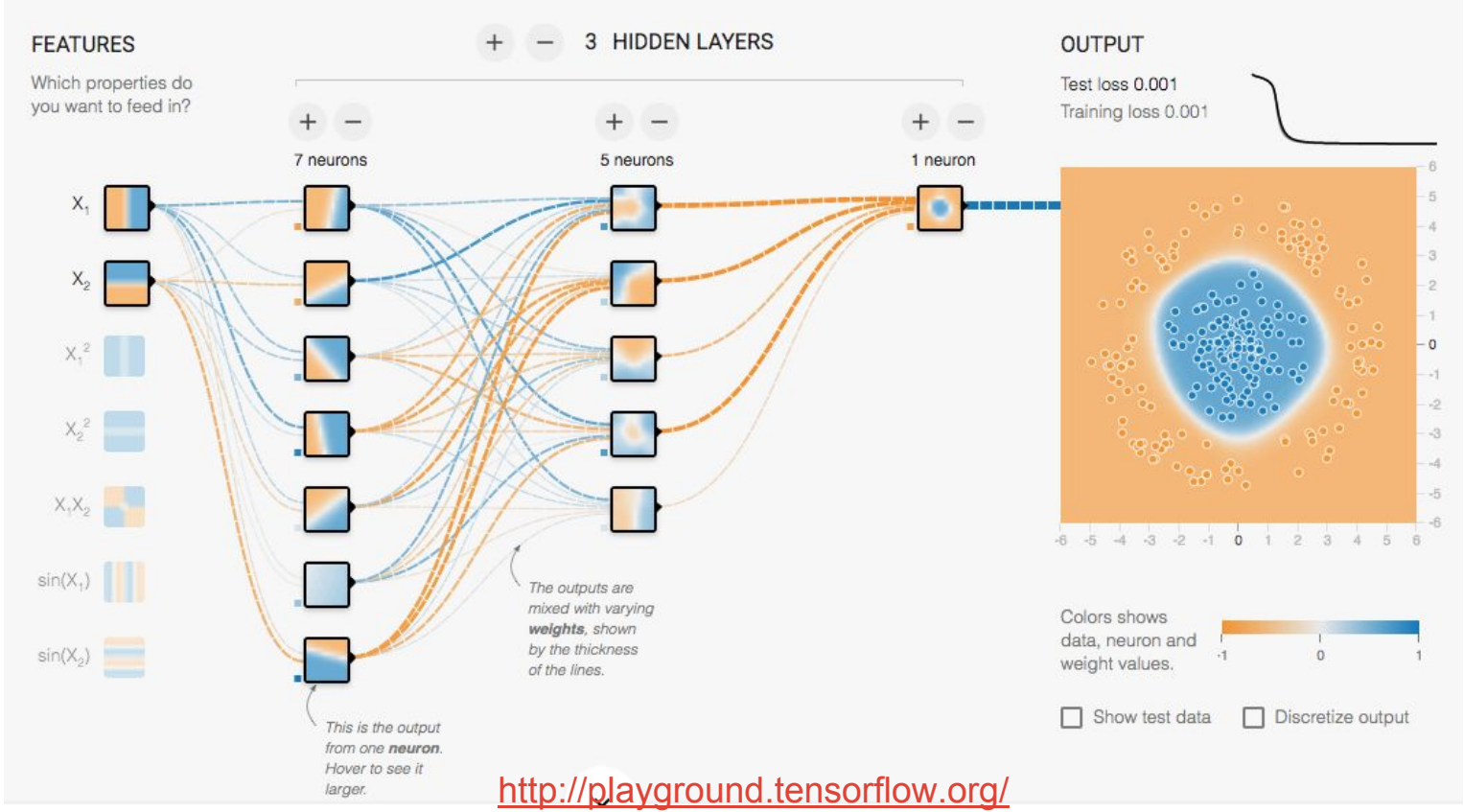




NEURAL NETWORKS PRIMER

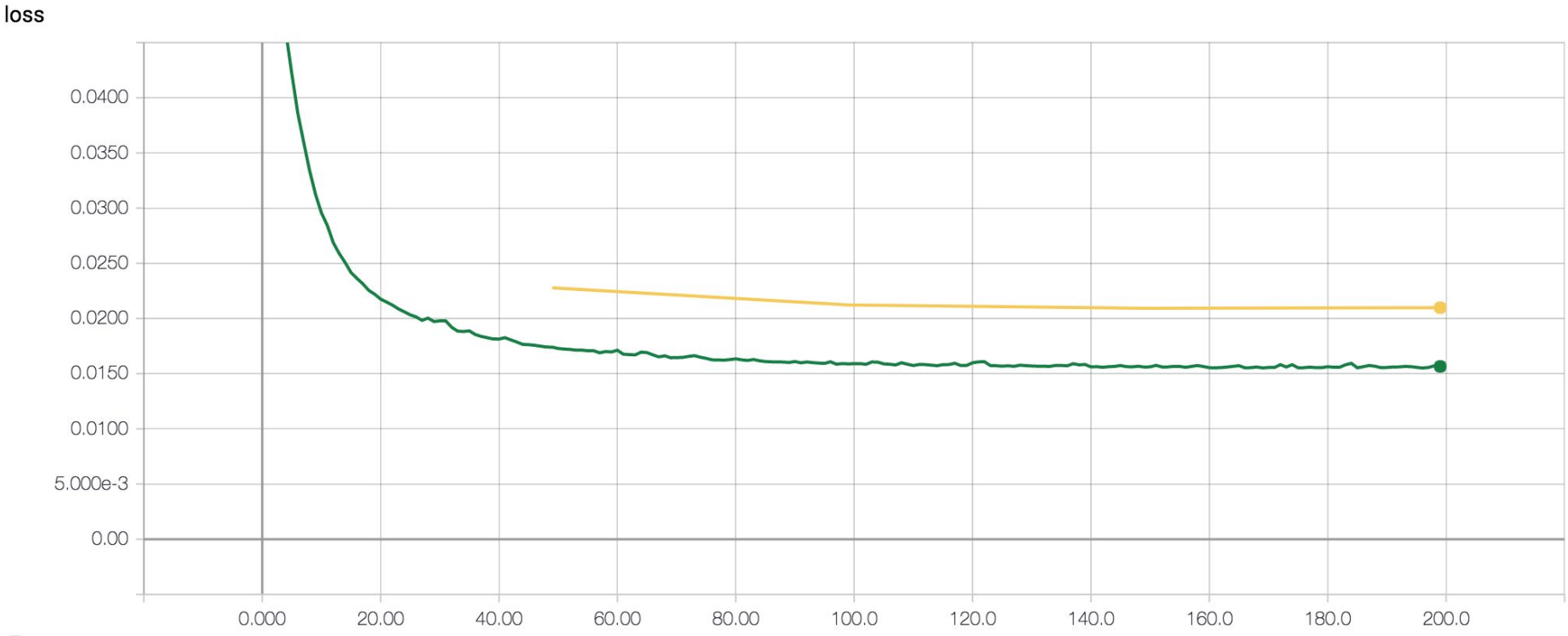
- black box
  - architecture
    - units
    - layers (LSTM, Dense)
    - optimisers
    - activations
  - .fit: (m,n) -> (1,)
  - .predict: (1,n) -> (1,)
- data processing
  - data@#$%^&* -> (m,n)
- NN training strategy
  - bias/variance

BASIC NEURAL NETWORK ARCHITECTURE



http://playground.tensorflow.org/

avoidable Bias/Variance: your guide



Andrew Ng's basic recipe for ML training

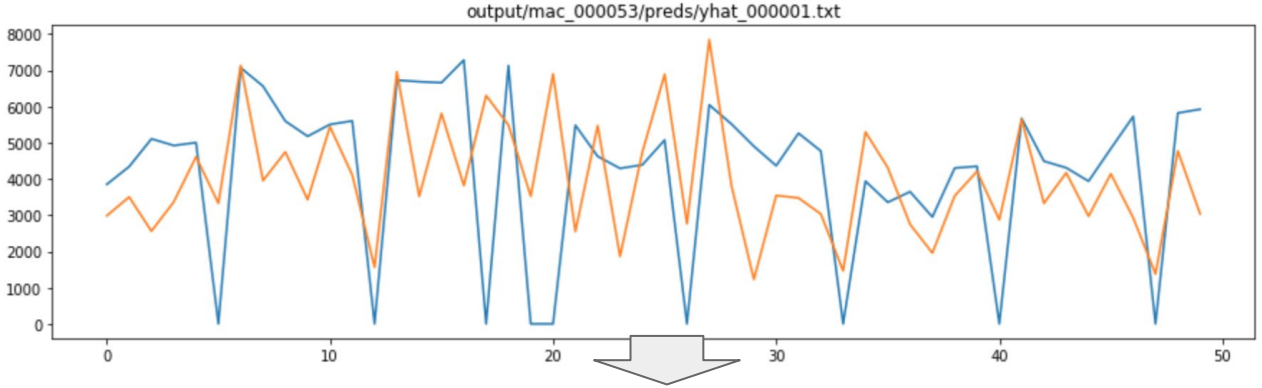
avoidable bias

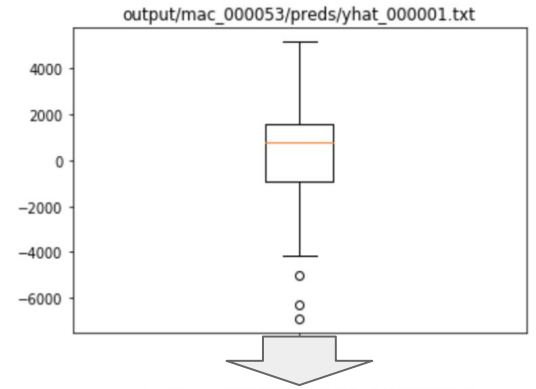
- bigger network
- more epochs
- better optimisation algos
- more features?
- (architecture search)

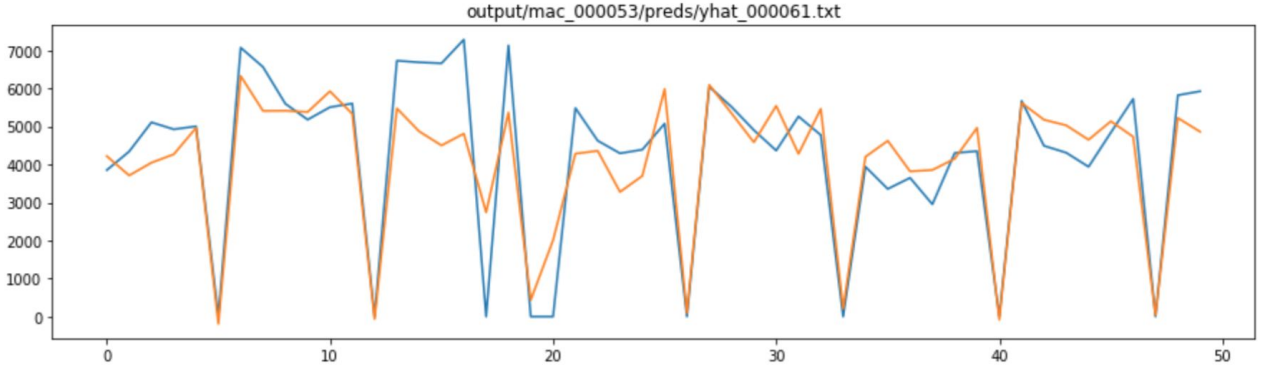
high variance

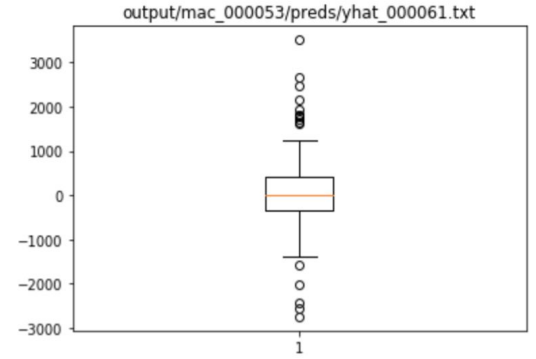
- more data
  - more diverse
  - data augmentation
- dropout, L2 regularisation
- (architecture search)

Learning Process

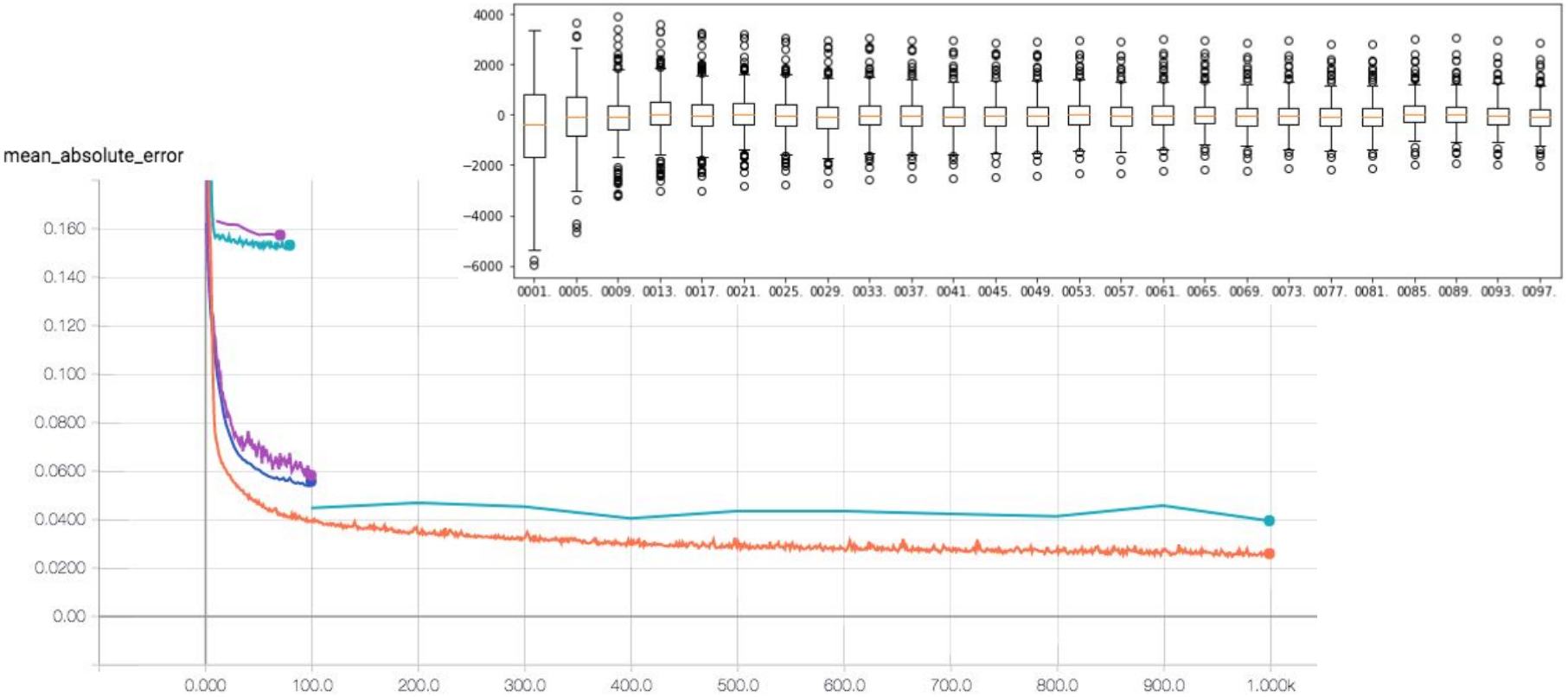








Learning: mae metrics



KERAS, TENSORFLOW, THE CLOUD AND OTHERS

- keras
  - nn wrapper
- tensorflow ecosystem
  - tensorboard for training visualisation
  - gcp ml-engine for scalability

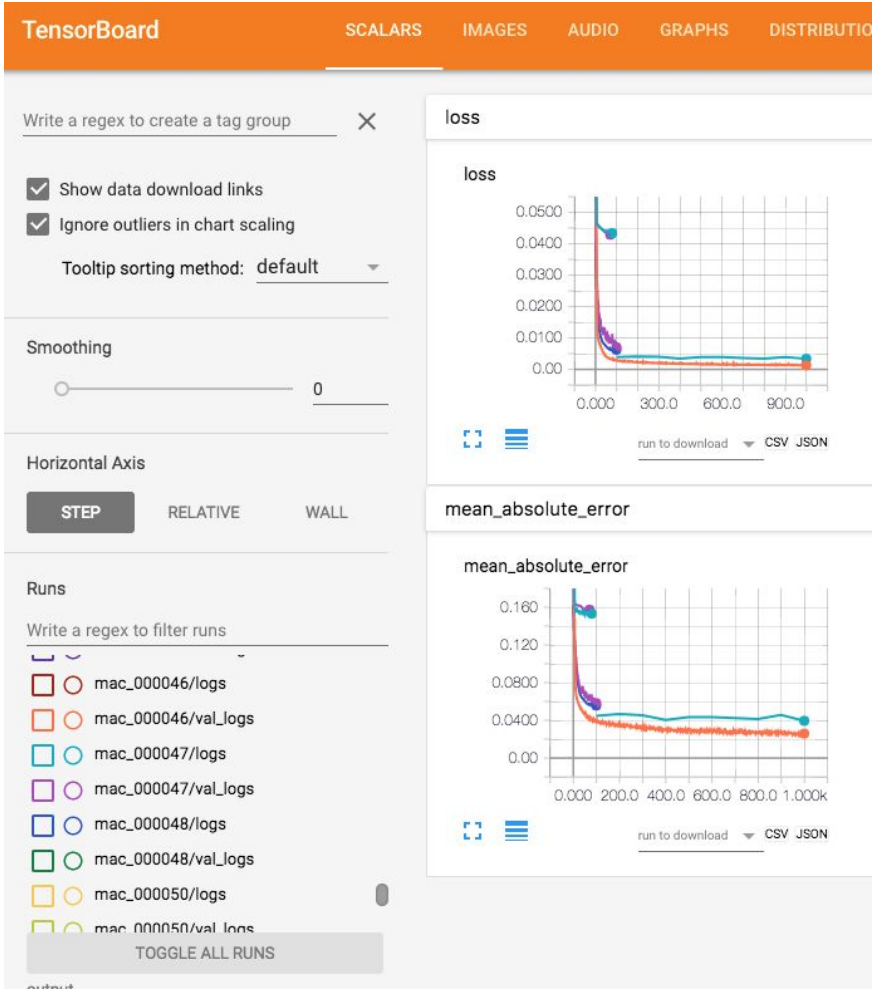
Keras

```
model = models.Sequential()
model.add(layers.Dense(8,
                       input_shape=(FEATURE_SIZE,),
                       activation='relu'
                       )
          )
model.add(layers.Dense(8,
                       activation='relu'
                       )
          )
model.add(layers.Dense(1))

model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mae'])
```

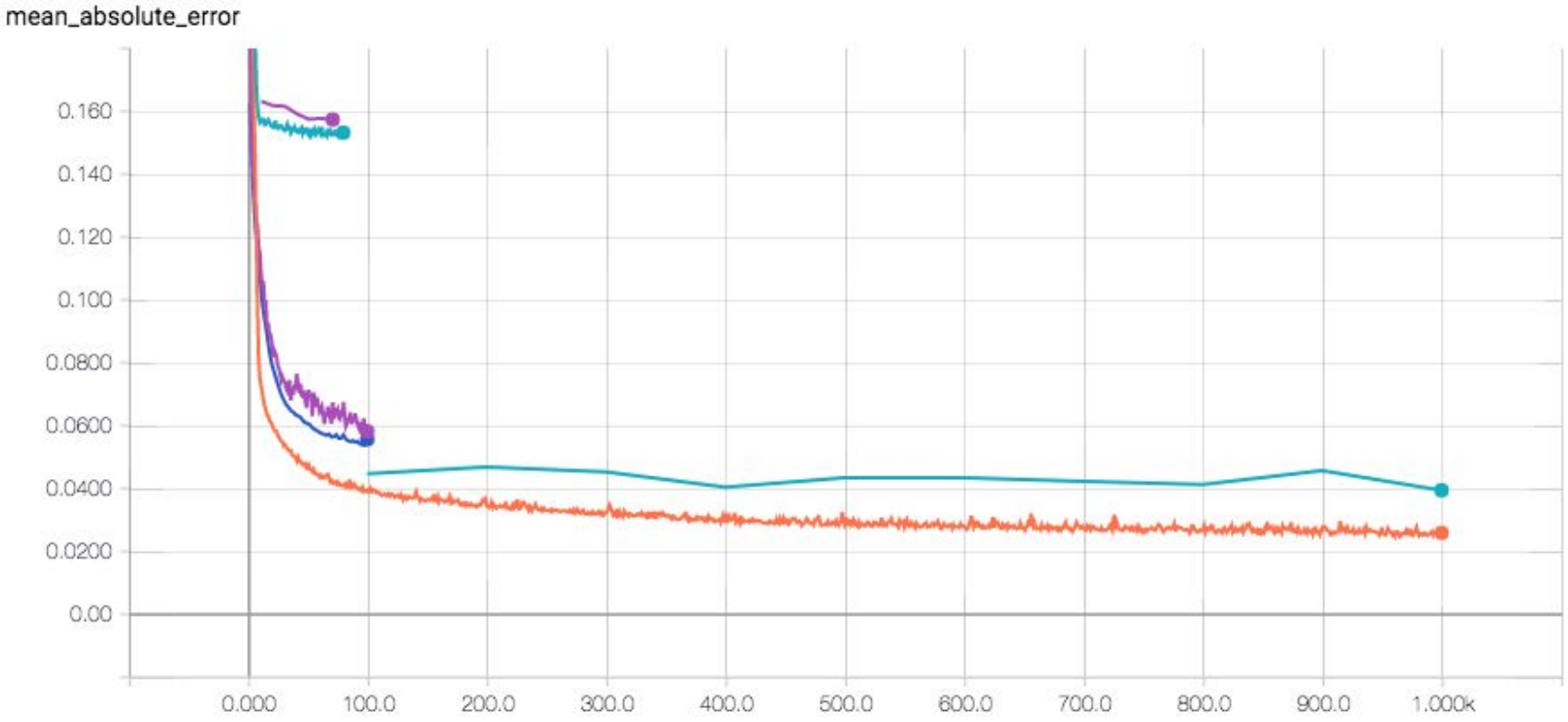
```
forecast_model.fit(
    X, Y,
    epochs=num_epochs,
    callbacks=callbacks)
```

```
yhat = forecast_model.predict(X)
```

TensorBoard



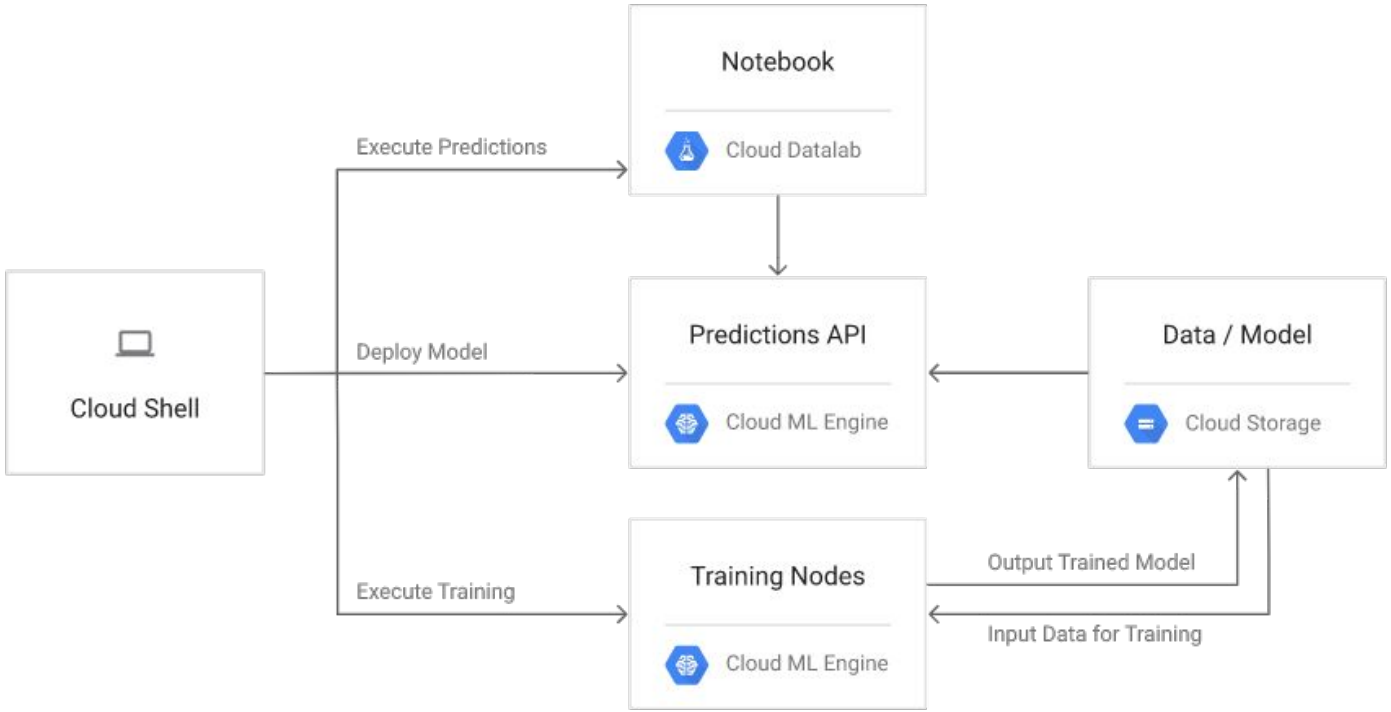
Tensorboard - guide your training



Adding your own TensorBoard metrics

```python
class TensorBoardMetricsLogger:
    """Log your own metrics to be shown by TensorBoard"""

    def __init__(self, log_dir):
        self.log_dir = log_dir
        self.writer = tf.summary.FileWriter(self.log_dir)

    def append(self, metrics_dict, epoch):
        for (name, value) in metrics_dict.iteritems():
            summary = tf.Summary()
            summary_value = summary.value.add()
            summary_value.simple_value = value
            summary_value.tag = name
            self.writer.add_summary(summary, epoch)

        self.writer.flush()

    def close(self):
        self.writer.close()
```

GCP ML Engine for Scalability



GCP SHELL

```
#ML-ENGINE SCALING TIERS
#BASIC  1
#STANDARD_1 10
#PREMIUM_1  75
#BASIC_GPU  3
#
 gcloud ml-engine jobs submit training $JOB_NAME \
                                       --stream-logs \
                                       --runtime-version 1.2 \
                                       --job-dir $GCS_JOB_DIR \
                                       --package-path trainer \
                                       --module-name trainer.task \
                                       --region us-central1 \
                                       --scale-tier BASIC \
                                       -- \
                                       --train-files $GCS_TRAIN_FILE \
                                       --eval-files $GCS_EVAL_FILE \
                                       --num-epochs $NUM_EPOCHS \
                                       --checkpoint-epochs $CHECKPOINT_EPOCHS \
                                       --eval-frequency $EVAL_FREQUENCY
```

ACTION

simple to more complex model

guided by avoidable bias and variance

Tooling

CONDA ENV
> source activate mlengine

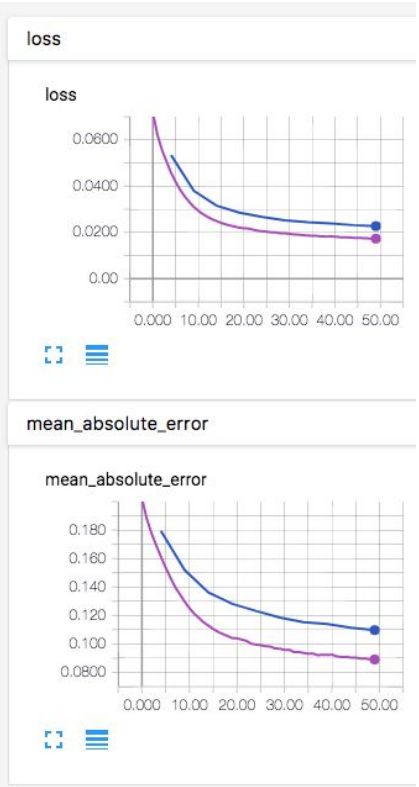
NOTEBOOKS
> jupyter notebook

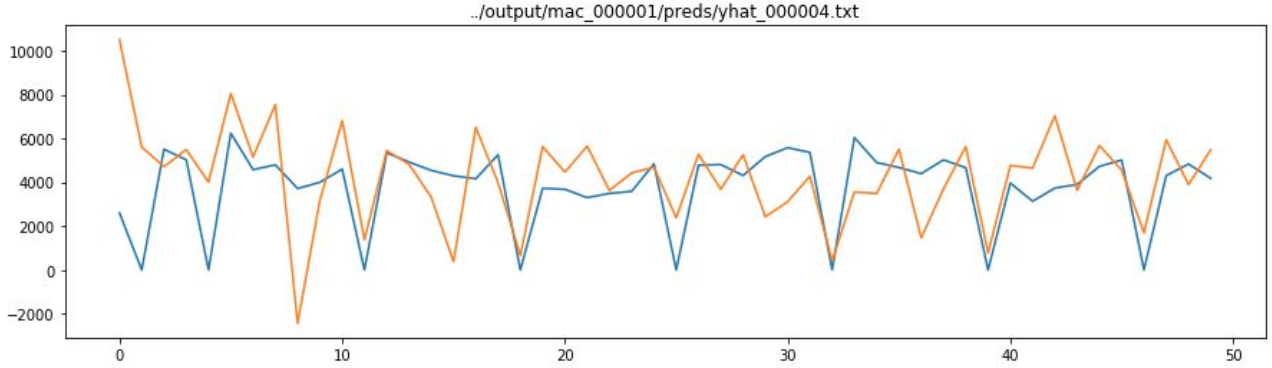
TENSORBOARD (http://...:8080)
> python -m tensorflow.tensorboard --logdir=output --port=8080

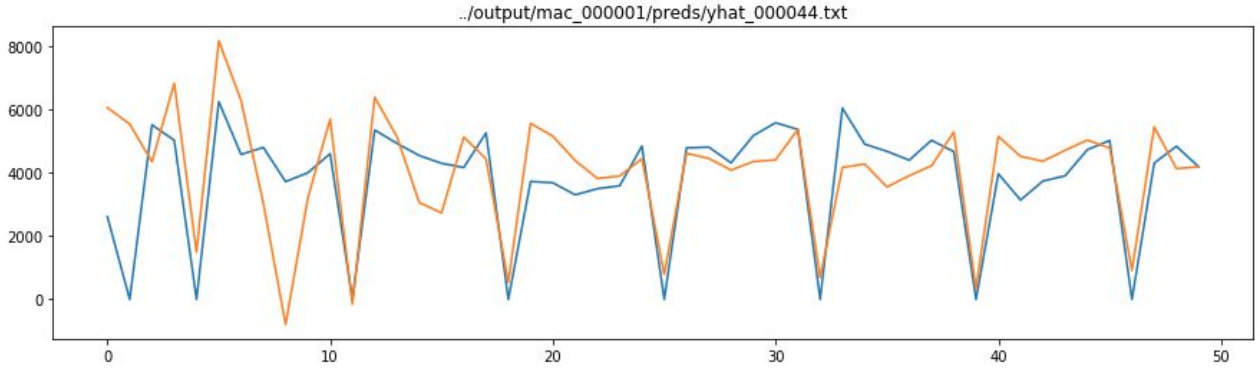
Initial setup

```
def model_fn():
    """Create a Keras Sequential model with layers."""
    model = models.Sequential()
    model.add(layers.Dense(4, input_shape=(FEATURE_SIZE,)))
    model.add(layers.Dense(1))

    compile_model(model)

    return model


def compile_model(model):
    """Compiles the model - either created or loaded"""
    model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mae'])
    return model
```

Simple first



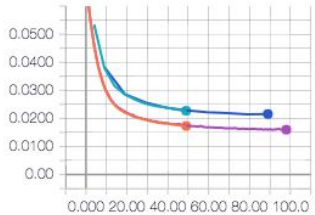




More epochs? 200

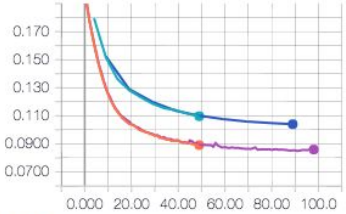
loss

loss

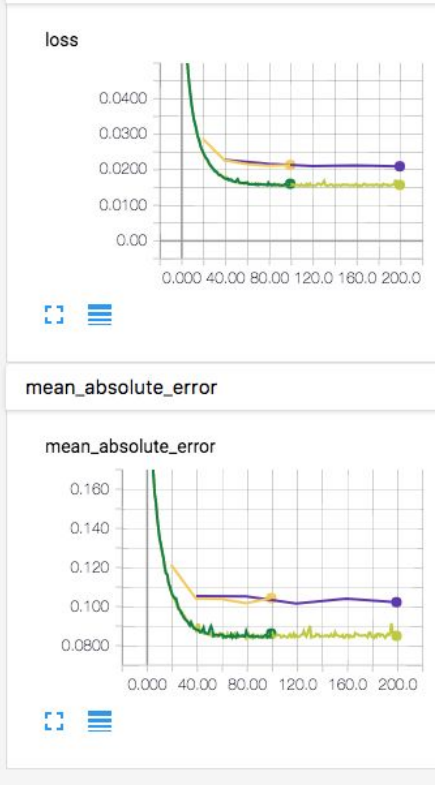


mean_absolute_error

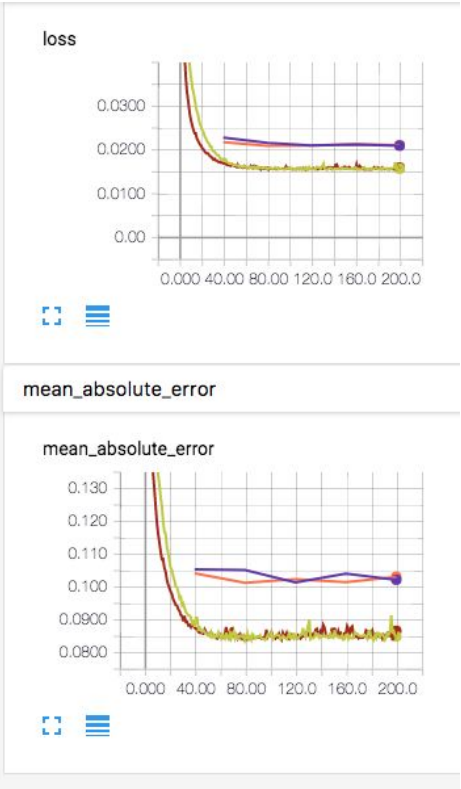
mean_absolute_error



MORE UNITS? 8

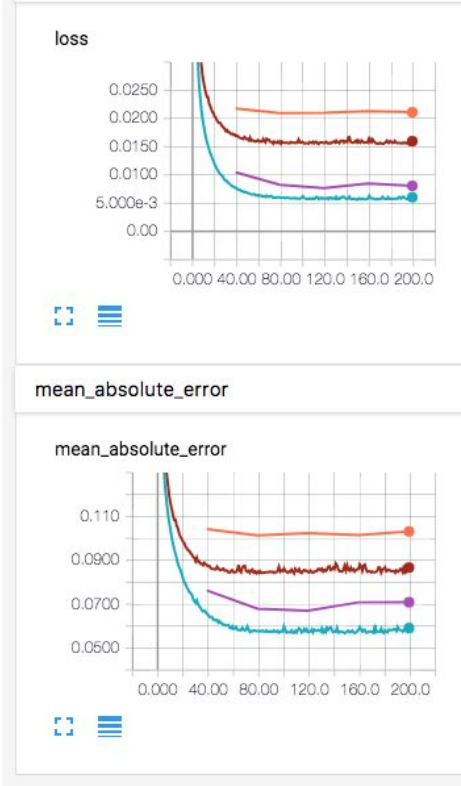

loss
0.0400
0.0300
0.0200
0.0100
0.00
0.000 40.00 80.00 120.0 160.0 200.0
mean_absolute_error
mean_absolute_error
0.160
0.140
0.120
0.100
0.0800
0.000 40.00 80.00 120.0 160.0 200.0

More layers? 2



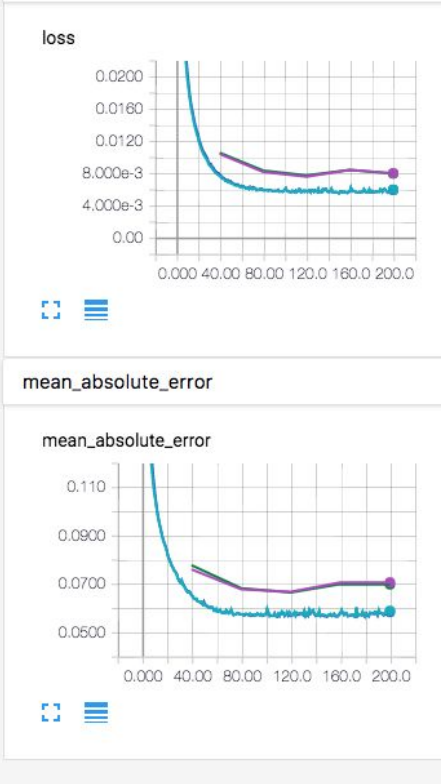

```
model = models.Sequential()
model.add(layers.Dense(8, input_shape=(FEATURE_SIZE,)))
model.add(layers.Dense(8))
model.add(layers.Dense(1))
```

More Data? 'OPEN'

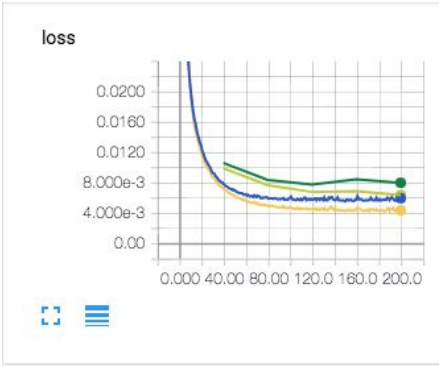



```
FEATURES_CONT = ['Open']
```

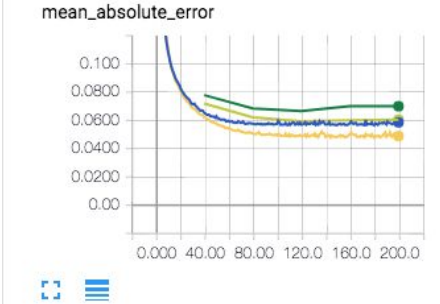
More Data? 'DayOfWeek'



More Data? 'Promo'

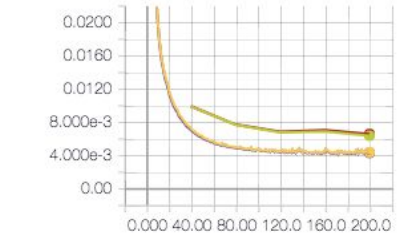


mean_absolute_error



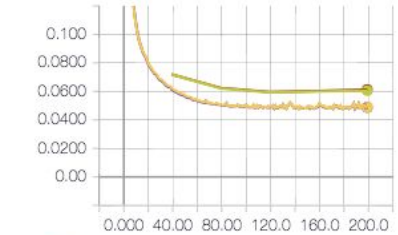
More Data? 'School Holiday'

loss



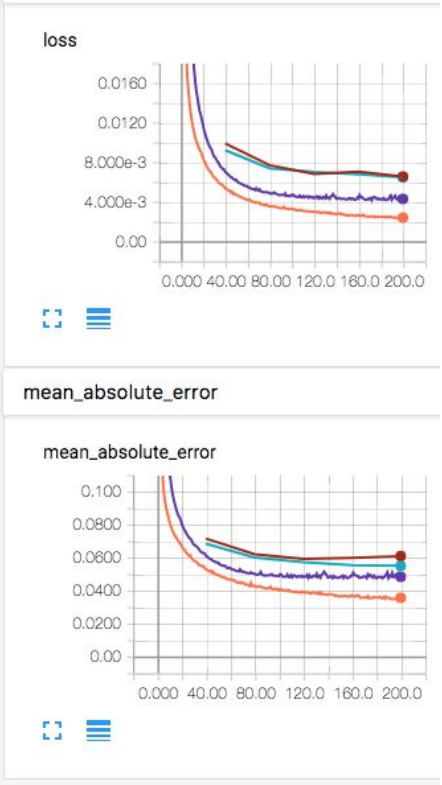
mean_absolute_error

mean_absolute_error

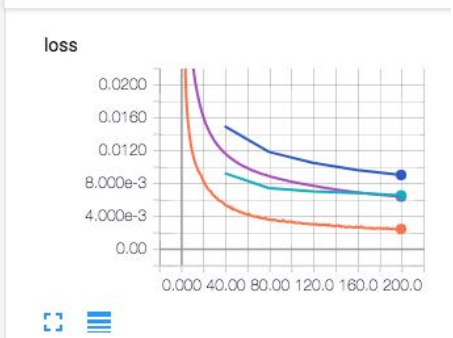


How are we doing so far?

Activation? relu

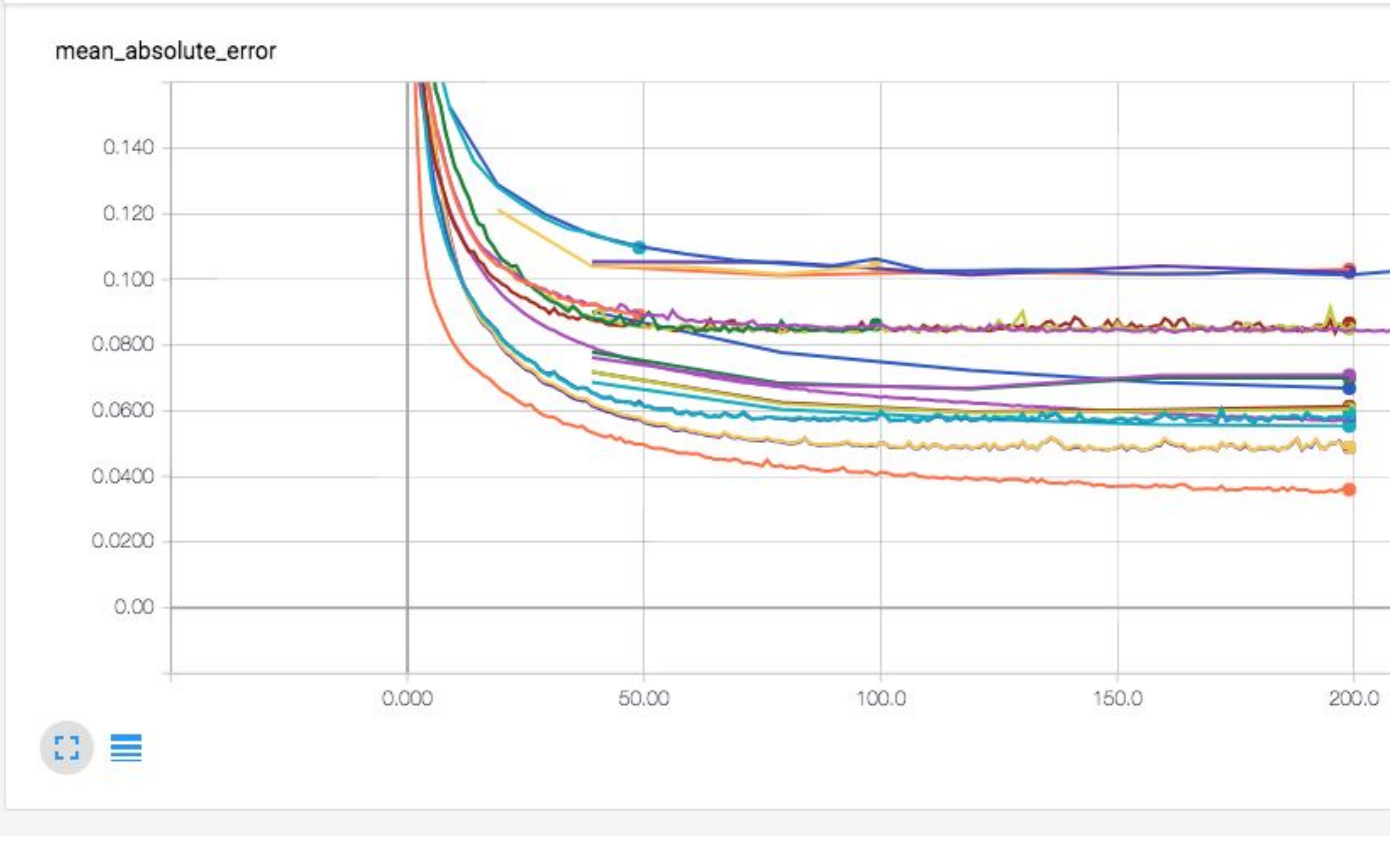


Optimiser? SGD



got worse !

How are we doing?



Extensions

- encode label features
- one-hot categorical features
- train/dev + test set
- learn across the stores (not just one)
- try different architectures (LSTMs?)

References

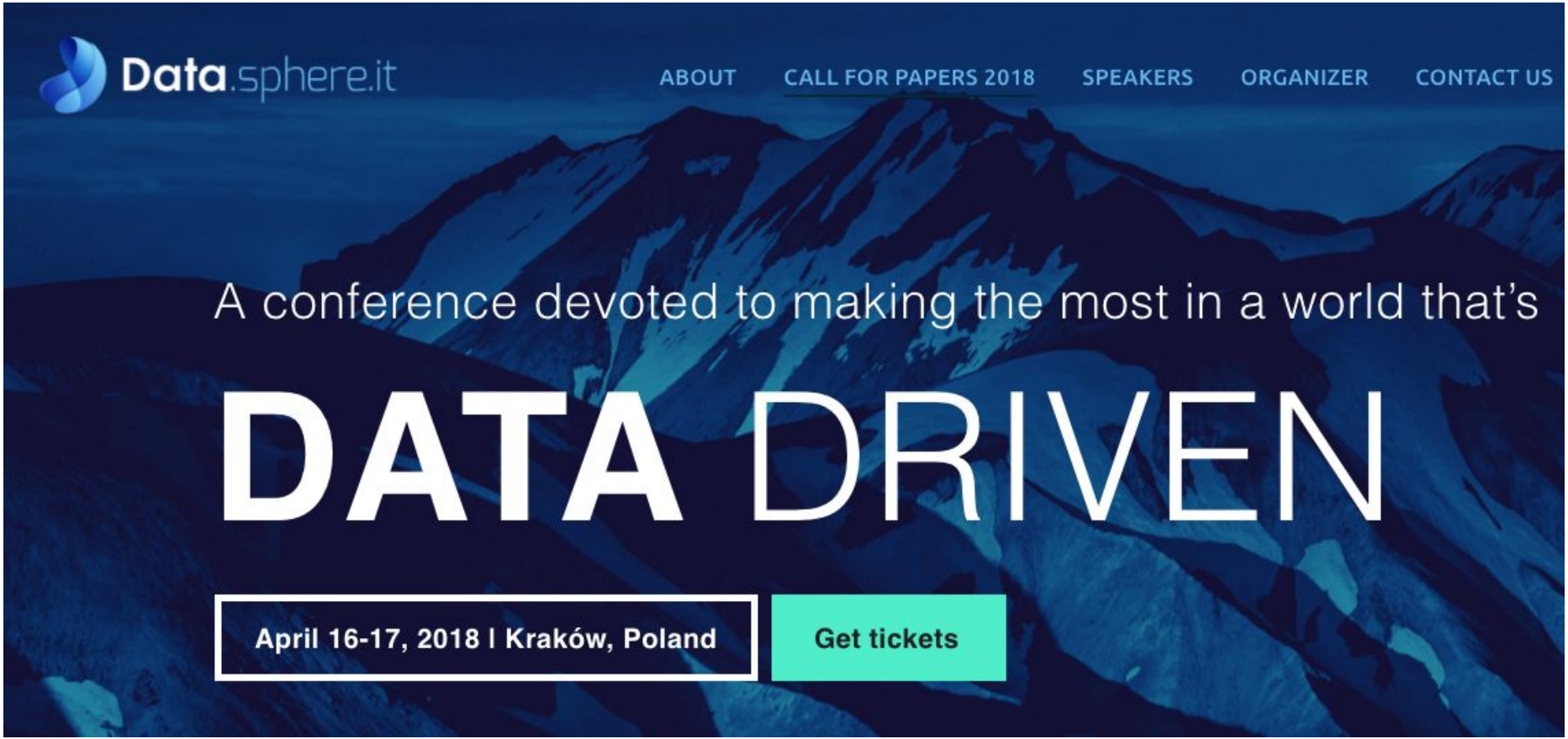
https://www.kaggle.com/c/rossmann-store-sales

https://keras.io/

http://playground.tensorflow.org/

https://cloud.google.com/ml-engine/

https://github.com/GoogleCloudPlatform/cloudml-samples/tree/master/census/keras


Data.sphere.it
ABOUT
CALL FOR PAPERS 2018
SPEAKERS
ORGANIZER
CONTACT US
A conference devoted to making the most in a world that's
DATA DRIVEN
April 16-17, 2018 I Kraków, Poland
Get tickets