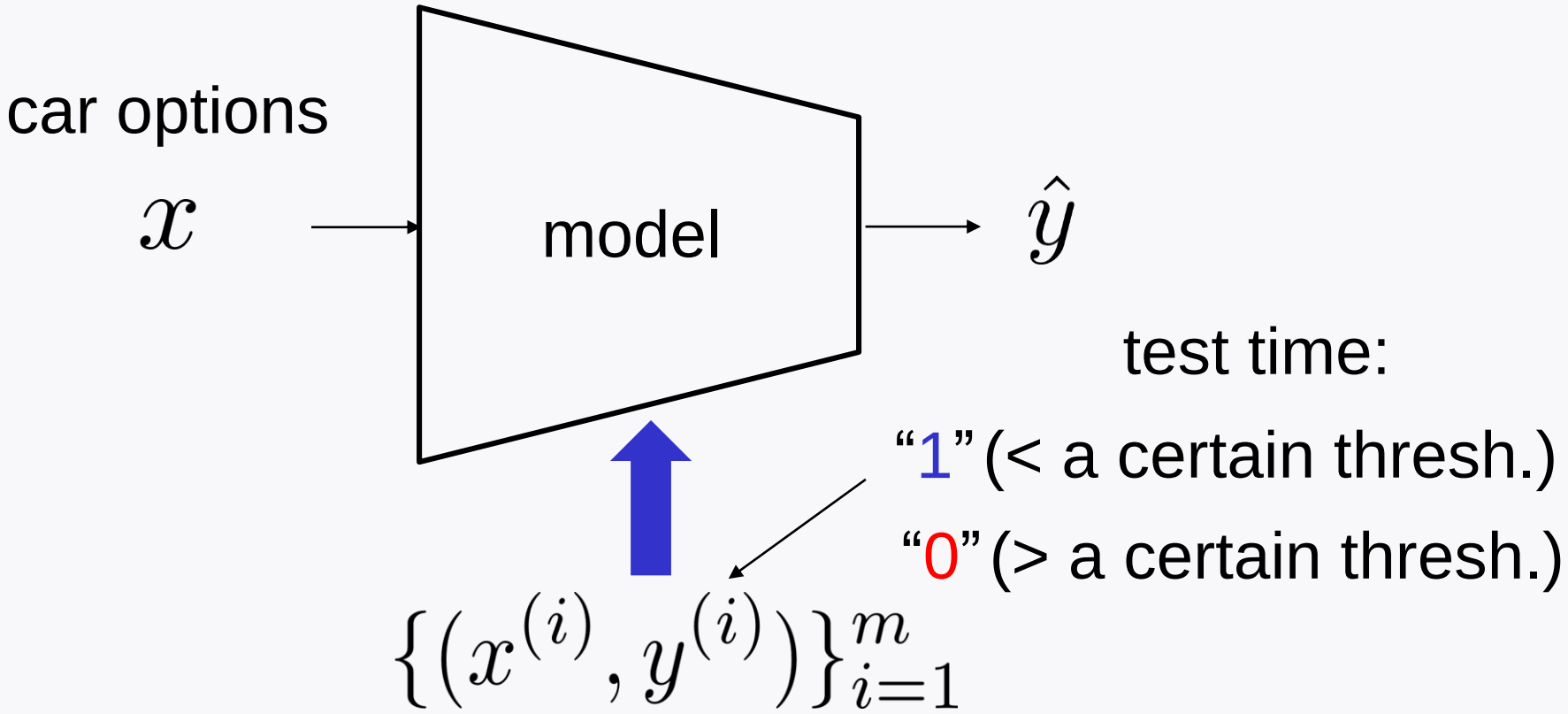
Mini-project #1

Practice Session 18

Changho Suh

January 29, 2024

Recap: Test-time prediction



Recap: Loading MB dataset

```
import pandas as pd
data = pd.read_csv('mercedes_test.csv')
data
```

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
0	0	130.81	k	v	at	a	d	u	j	o	...	0	0	1	0	0	0	0	0	0	0
1	6	88.53	k	t	av	e	d	y	l	o	...	1	0	0	0	0	0	0	0	0	0
2	7	76.26	az	w	n	c	d	x	j	x	...	0	0	0	0	0	0	1	0	0	0
3	9	80.62	az	t	n	f	d	x	l	e	...	0	0	0	0	0	0	0	0	0	0
4	13	78.02	az	v	n	f	d	h	d	n	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4204	8405	107.39	ak	s	as	c	d	aa	d	q	...	1	0	0	0	0	0	0	0	0	0
4205	8406	108.77	j	o	t	d	d	aa	h	h	...	0	1	0	0	0	0	0	0	0	0
4206	8412	109.22	ak	v	r	a	d	aa	g	e	...	0	0	1	0	0	0	0	0	0	0
4207	8415	87.48	al	r	e	f	d	aa	l	u	...	0	0	0	0	0	0	0	0	0	0
4208	8417	110.85	z	r	ae	c	d	aa	g	w	...	1	0	0	0	0	0	0	0	0	0

4209 rows × 378 columns

strings (categorical data)

$$m = 4209 \quad n = 376 (= 378 - 2)$$

Recap: Preprocessing

```
# Choose categorical data columns
cf = data.select_dtypes(include=['object']).columns
# To change it into "categorical" data type
data[cf]=data[cf].astype('category')
# One hot encoding
data = pd.get_dummies(data)
# Obtain X from data (excluding 'ID' and 'y')
X_df = data.drop(['ID','y'],axis=1)
# Obtain y from data
y_df = data['y']
```

```
# Convert y_df into binary labels
import numpy as np
TF_vector= (y_df<np.median(y_df))
y_df=TF_vector.astype(float)
```

```
# Conver data frame into numpy array
X,y = X_df.values, y_df.values
```

Recap: Split into train and test datasets

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,stratify=y)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(3788, 563)
(421, 563)
(3788,)
(421,)
```

Designing an DNN model

Many hyperparameters to search.

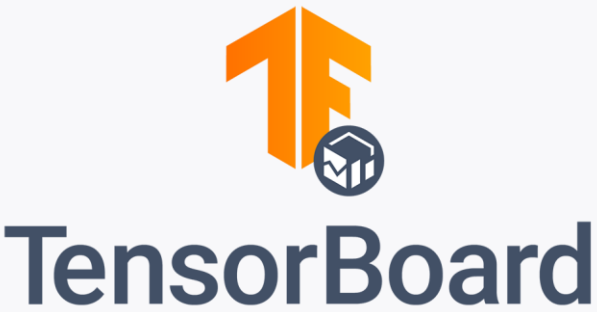
Difficult to develop a well-performing model.

Hence: Important to do very *quick* and *convenient* search for hyperparameters.

There is a tool that may be useful for this:

TensorBoard

TensorBoard



A **visualization** tool for Keras models

Three key features:

1. Draws curves with logs info
2. Provides visualization for models
3. Comes integrated with TensorFlow

How to use TensorBoard?

1. Load TensorBoard notebook extension:

```
%load_ext tensorboard
```

2. Train a Keras model using a command **`TensorBoard`**.

3. Launch a TensorBoard browser:

```
%tensorboard --logdir=logs/
```

directory where logs info will be saved

TensorBoard in action: Load TensorBoard

```
%load_ext tensorboard
```

TensorBoard in action: Train a Keras model

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import LearningRateScheduler


# Construction of an DNN model
def create_model(n_layer=2,lambda_=0):
    model = Sequential()
    for i in range(n_layer-1):
        model.add(Dense(10, activation='relu',
                        kernel_regularizer=l2(lambda_), bias_regularizer=l2(lambda_)))

    model.add(Dense(1, activation='sigmoid',
                    kernel_regularizer=l2(lambda_), bias_regularizer=l2(lambda_)))
    return model
```

TensorBoard in action: Train a Keras model

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import LearningRateScheduler
import os
from tensorflow.keras.callbacks import TensorBoard
model = create_model(n_layer=2,lambda_=0)
opt = Adam(learning_rate=1e-3)
model.compile(optimizer=opt,
              loss='binary_crossentropy',
              metrics=['acc'])
# generate a path directory where logs info will be saved
logdir = os.path.join('logs','[1]no_regularization')
tb_callback = TensorBoard(logdir)
```

logs\[1]no_regularization

To save logs info in the directory “logdir”

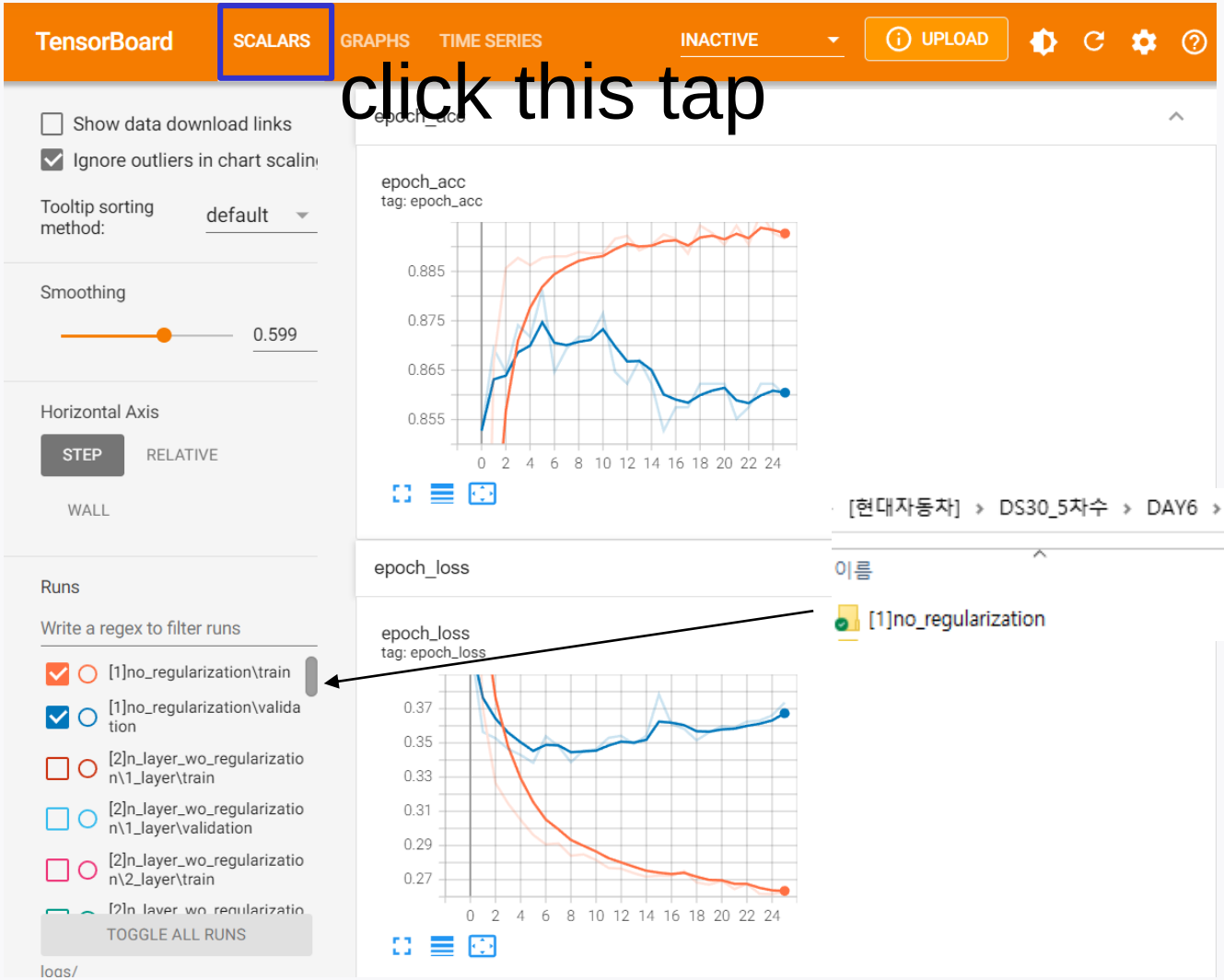
TensorBoard in action: Train a Keras model

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import LearningRateScheduler
import os
from tensorflow.keras.callbacks import TensorBoard
model = create_model(n_layer=2,lambda_=0)
opt = Adam(learning_rate=1e-3)
model.compile(optimizer=opt,
              loss='binary_crossentropy',
              metrics=['acc'])
# generate a path directory where logs info will be saved
logdir = os.path.join('logs','[1]no_regularization')
tb_callback = TensorBoard(logdir)
es_callback = EarlyStopping(monitor='val_acc',patience=20)
hist = model.fit(X_train, y_train,
                 validation_split=1/9, epochs=100,
                 verbose=0,callbacks=[tb_callback, es_callback])
model.evaluate(X_test, y_test)
```

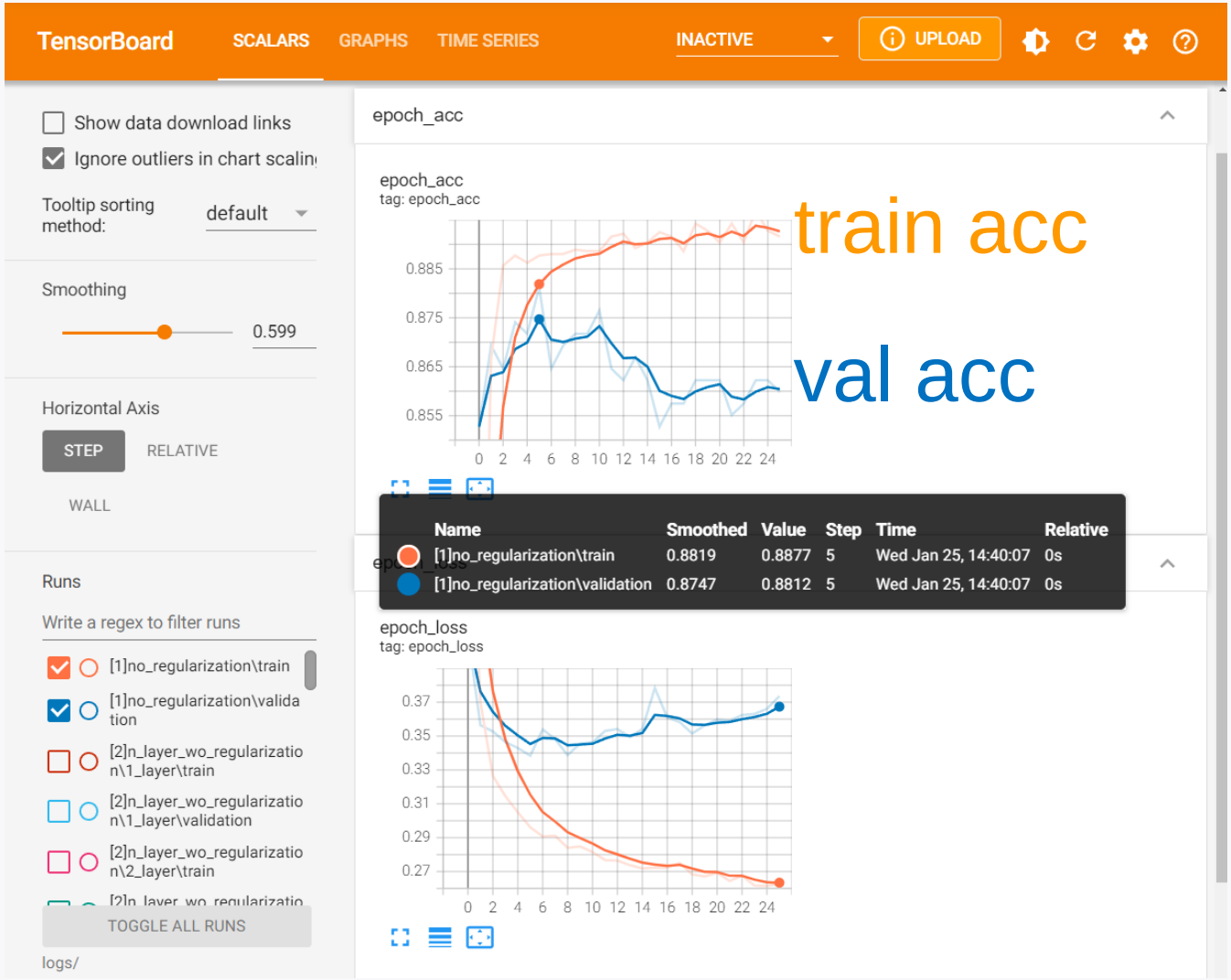
logs\[1]no_regularization

TensorBoard in action: Launch a TB browser

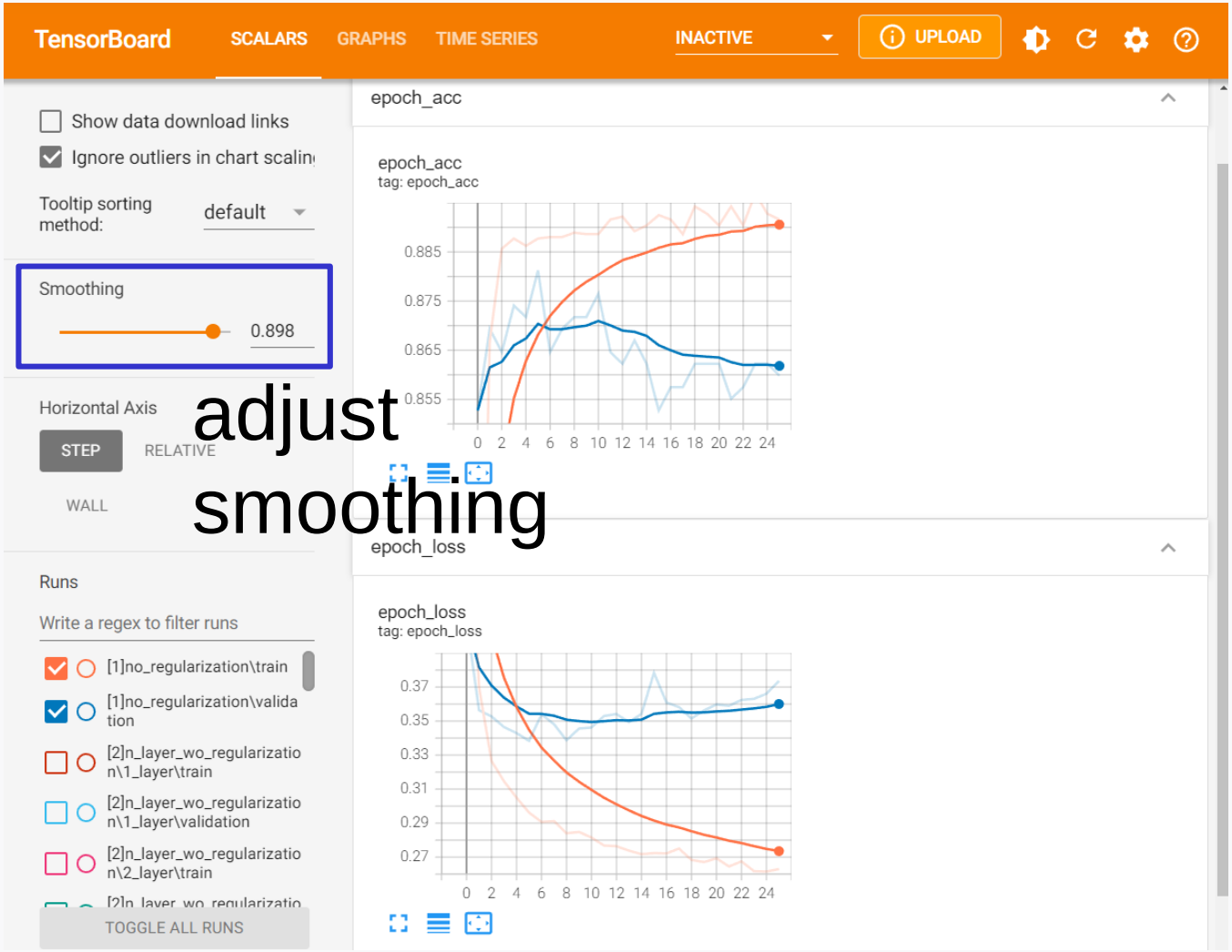
```
%tensorboard --logdir=logs/
```



TensorBoard in action: Launch a TB browser



TensorBoard in action: Launch a TB browser



adjust smoothing

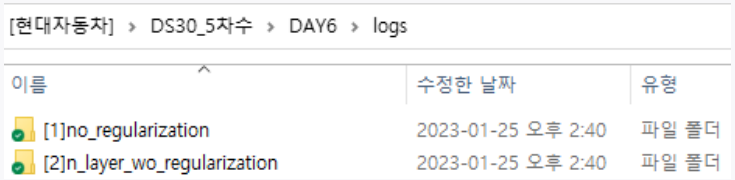
Hyperparameter search: Number of layers

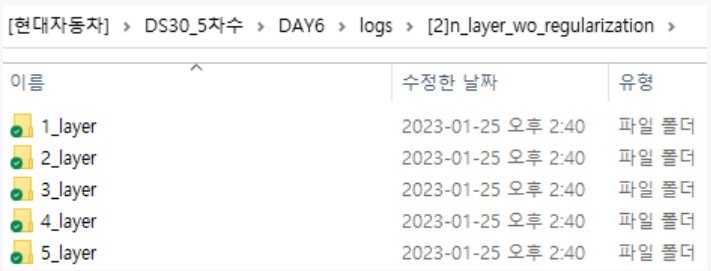
```
n_layer_list = [1,2,3,4,5]

for n_layer in n_layer_list:
    model = create_model(n_layer=n_layer,lambda_=0)
    opt = Adam(learning_rate=1e-3)
    model.compile(optimizer=opt,
                  loss='binary_crossentropy',
                  metrics=['acc'])
    logdir = os.path.join('logs','[2]n_layer_wo_regularization','{}_layer'.format(n_layer))
    tb_callback = TensorBoard(logdir)
    es_callback = EarlyStopping(monitor='val_acc', patience=20)
    hist = model.fit(X_train, y_train,
                  validation_split=1/9, epochs = 100,
                  verbose=0,callbacks=[tb_callback, es_callback])
    model.evaluate(X_test, y_test)
```

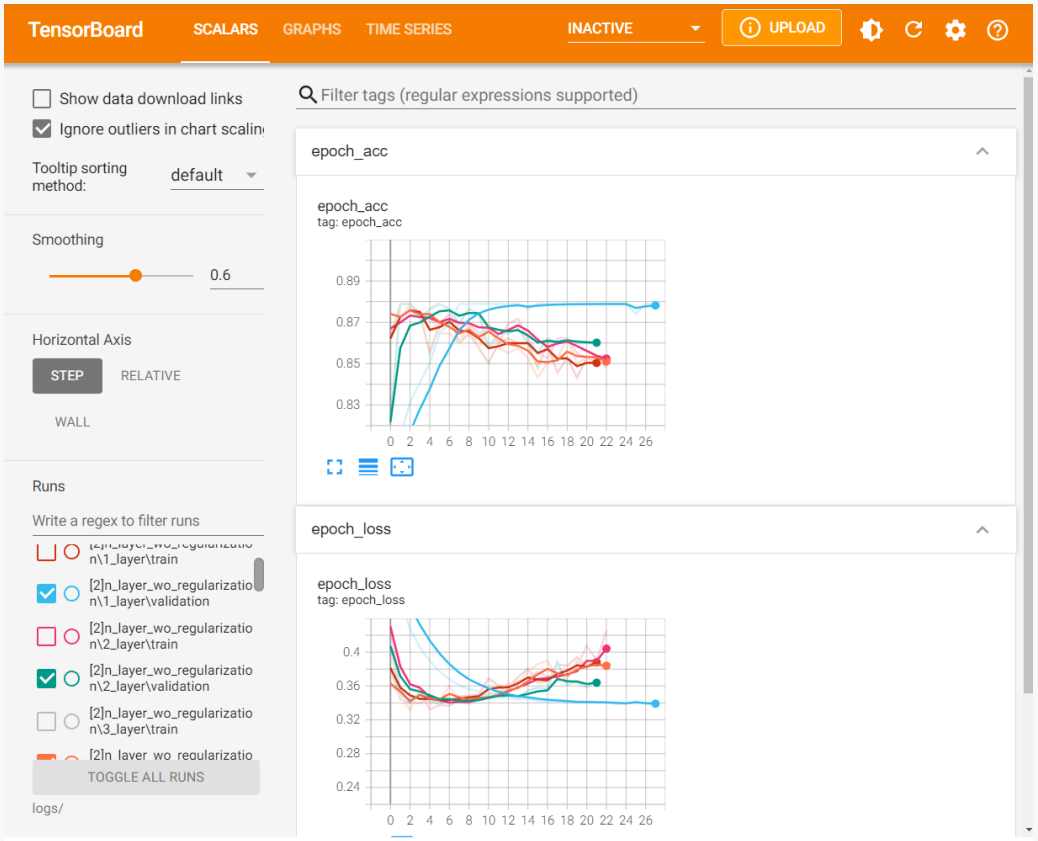
```
14/14 [==============================] - 0s 690us/step - loss: 0.3235 - acc: 0.8789
14/14 [==============================] - 0s 921us/step - loss: 0.3347 - acc: 0.8694
14/14 [==============================] - 0s 691us/step - loss: 0.3521 - acc: 0.8599
14/14 [==============================] - 0s 691us/step - loss: 0.3421 - acc: 0.8765
14/14 [==============================] - 0s 690us/step - loss: 0.3806 - acc: 0.8599
```

Hyperparameter search: Number of layers

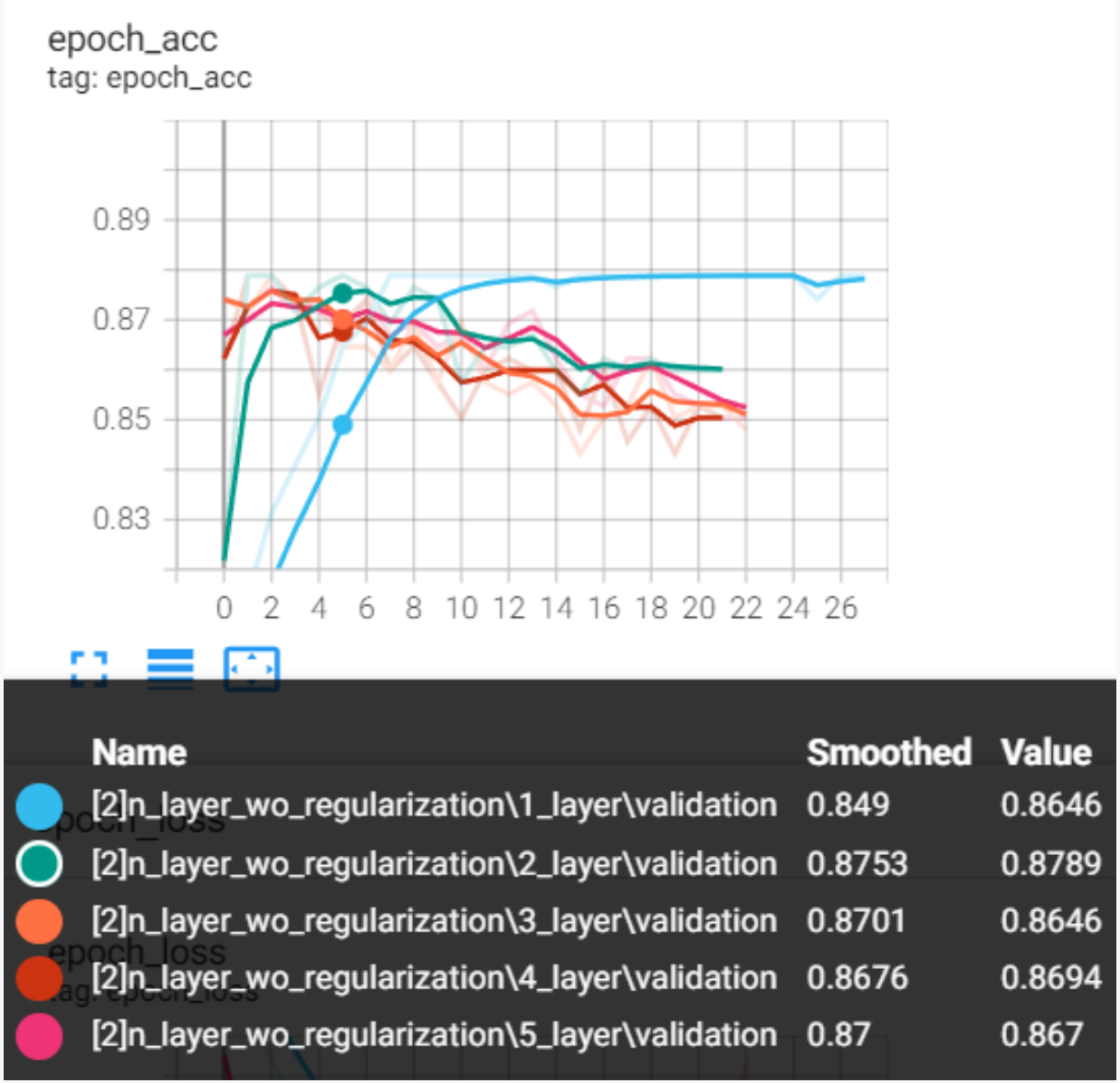





```
%tensorboard --logdir=logs/[2]n_layer_wo_regularization
```



Hyperparameter search: Number of layers



Hyperparameter search: Regularization factor

```
lambda_list = [1e-3, 1e-2, 1e-1, 1, 10]
for lambda_ in lambda_list:
    model = create_model(n_layer=2, lambda_=lambda_)
    opt = Adam(learning_rate=1e-3)
    model.compile(optimizer=opt,
                  loss='binary_crossentropy',
                  metrics=['acc'])
    logdir = os.path.join('logs','[3]2_layer_w_regularization','lambda_{}'.format(lambda_))
    tb_callback = TensorBoard(logdir)
    es_callback = EarlyStopping(monitor='val_acc',patience=20)
    hist = model.fit(X_train, y_train,
                  validation_split=1/9, epochs = 100,
                  verbose=0, callbacks=[tb_callback, es_callback] )
    model.evaluate(X_test, y_test)
```

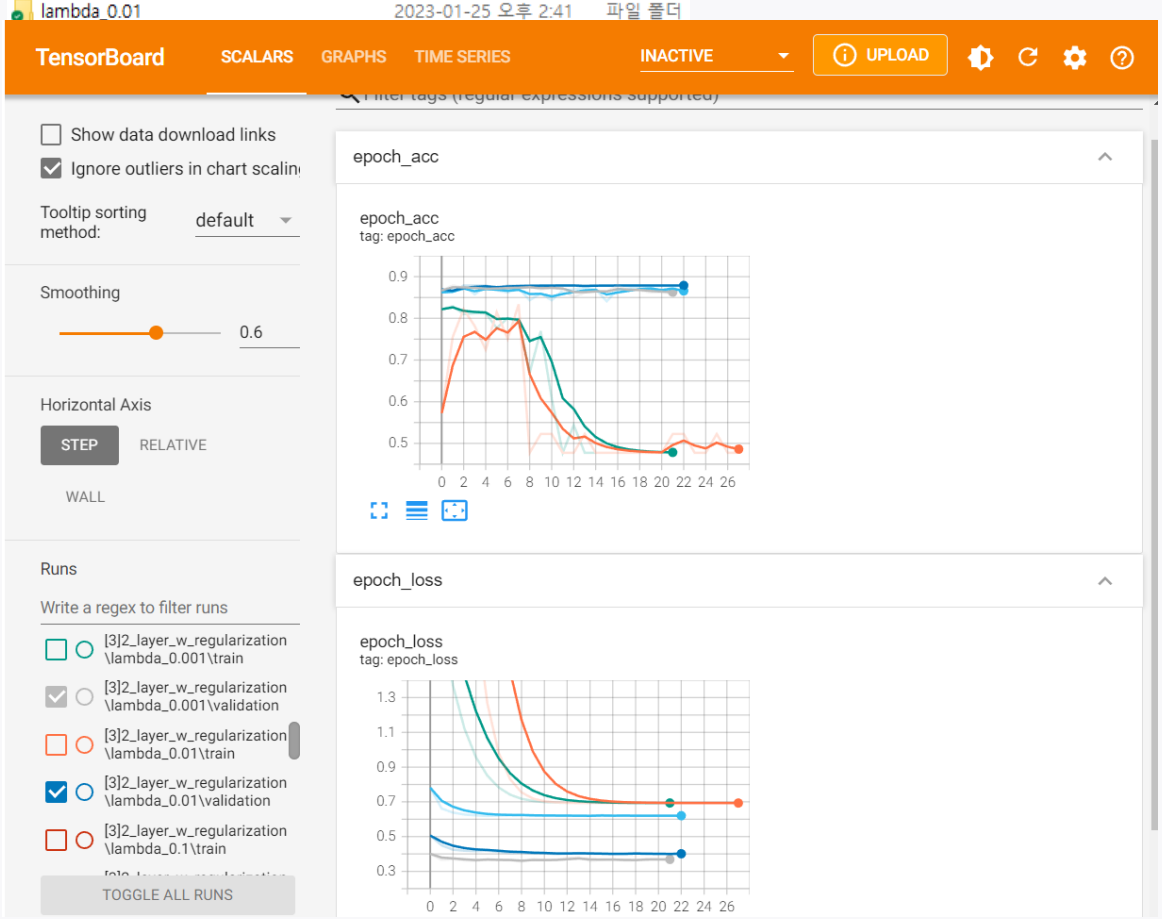
```
14/14 [==============================] - 0s 614us/step - loss: 0.3545 - acc: 0.8622
14/14 [==============================] - 0s 618us/step - loss: 0.3986 - acc: 0.8741
14/14 [==============================] - 0s 691us/step - loss: 0.6349 - acc: 0.8599
14/14 [==============================] - 0s 764us/step - loss: 0.6931 - acc: 0.5012
14/14 [==============================] - 0s 691us/step - loss: 0.6932 - acc: 0.5012
```

Hyperparameter search: Regularization factor

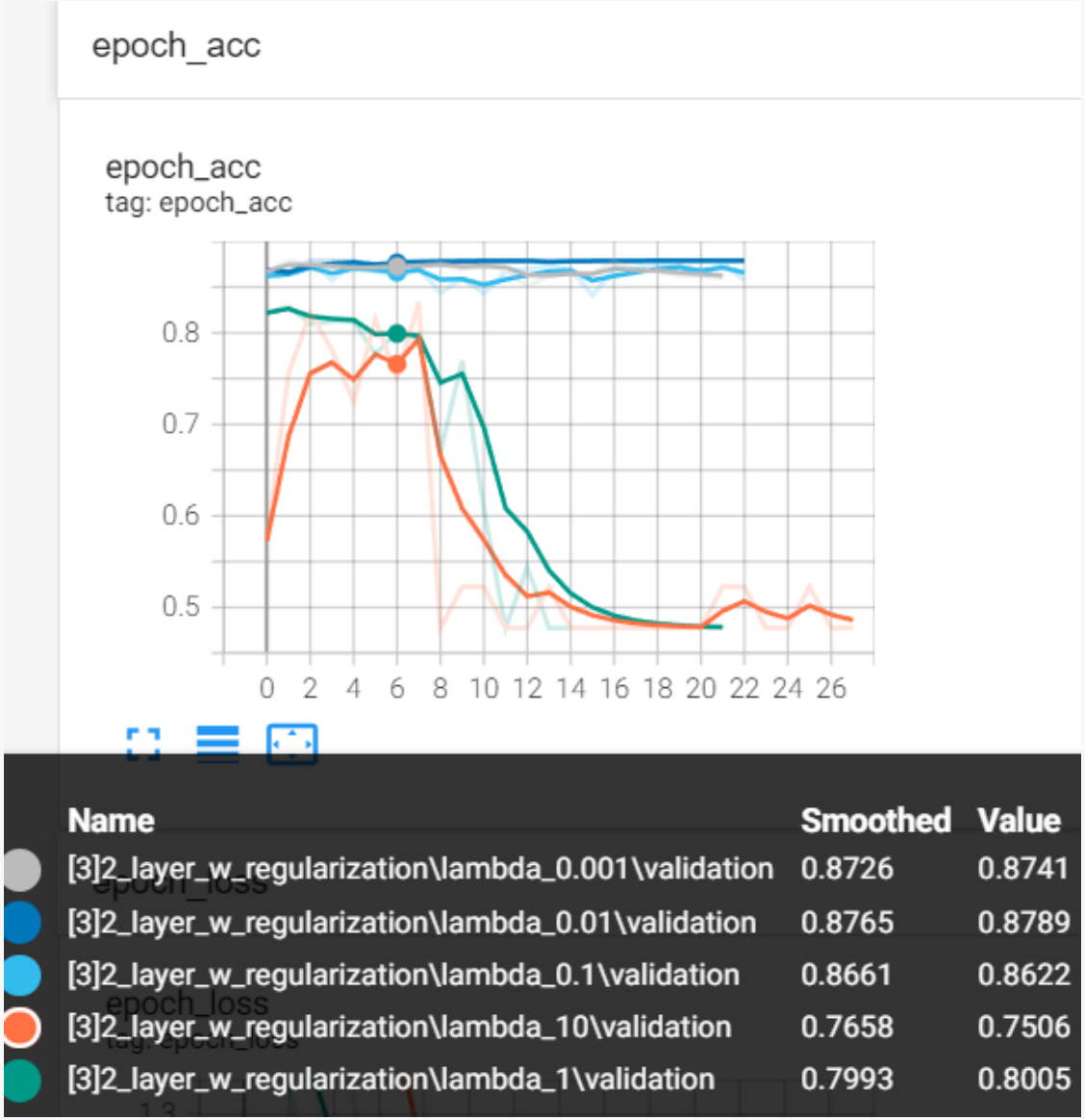
```
%tensorboard --logdir=logs/[3]2_layer_w_regularization
```

[현대자동차] › DS30_5차수 › DAY6 › logs › [3]2_layer_w_regularization ›

이름	수정한 날짜	유형
lambda_0.001	2023-01-25 오후 2:41	파일 폴더
lambda_0.01	2023-01-25 오후 2:41	파일 폴더



Hyperparameter search: Regularization factor



Name	Smoothed	Value
[3]2_layer_w_regularization\lambda_0.001\validation	0.8726	0.8741
[3]2_layer_w_regularization\lambda_0.01\validation	0.8765	0.8789
[3]2_layer_w_regularization\lambda_0.1\validation	0.8661	0.8622
[3]2_layer_w_regularization\lambda_10\validation	0.7658	0.7506
[3]2_layer_w_regularization\lambda_1\validation	0.7993	0.8005

Hyperparameter search: Learning rate

```
lr_list = [1e-1, 1e-2, 1e-3, 1e-4]

for lr in lr_list:
    model = create_model(n_layer=2, lambda_=0)
    opt = Adam(learning_rate=lr)
    model.compile(optimizer=opt,
                  loss='binary_crossentropy',
                  metrics=['acc'])
    logdir = os.path.join('logs','[4]2_layer_lr','lr_{}'.format(lr))
    tb_callback = TensorBoard(logdir)
    es_callback = EarlyStopping(monitor='val_acc',patience=20)
    hist = model.fit(X_train, y_train,
                  validation_split=1/9, epochs = 100,
                  verbose=0,callbacks=[tb_callback,es_callback])
```

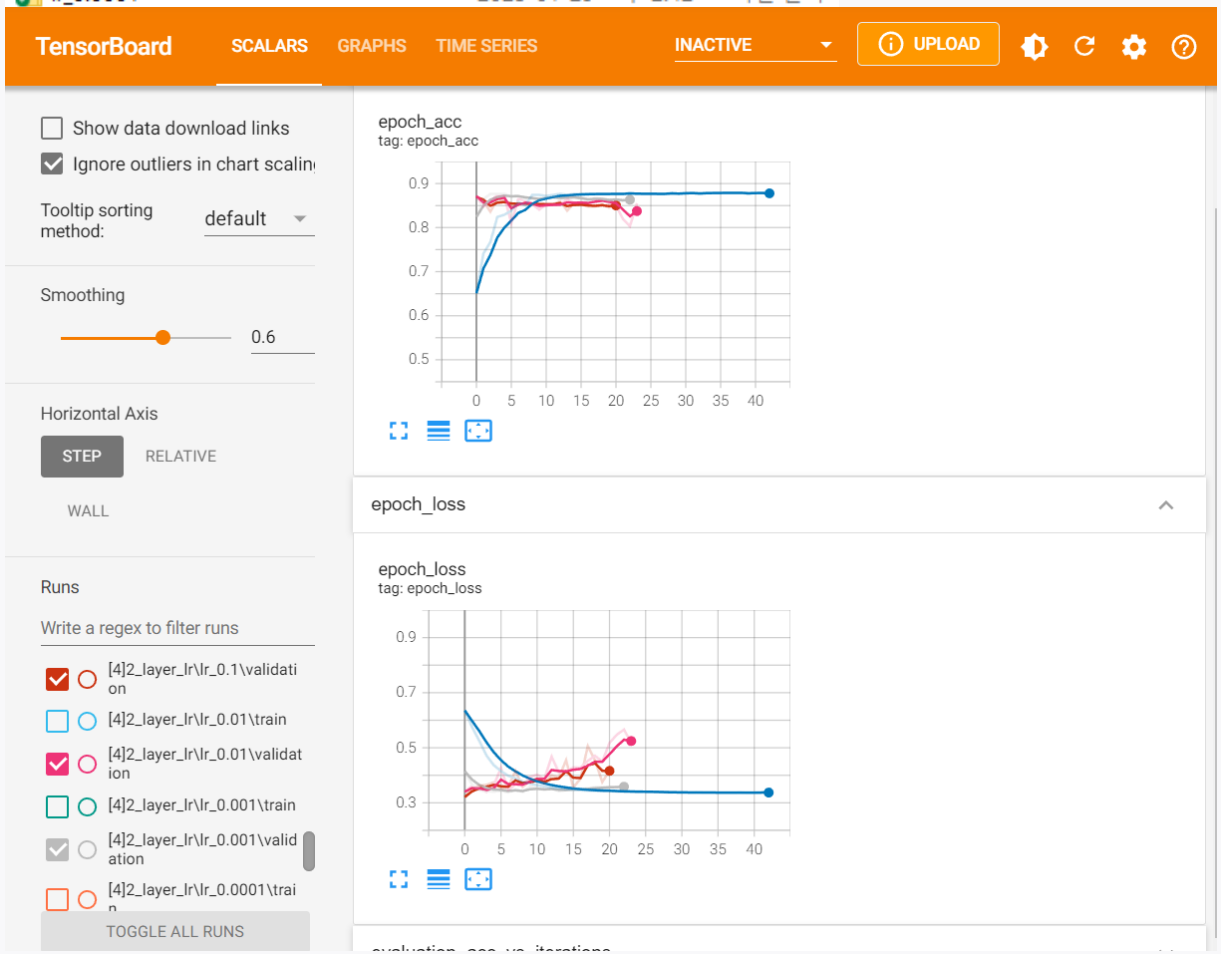
```
14/14 [==============================] - 0s 614us/step - loss: 0.3471 - acc: 0.8694
14/14 [==============================] - 0s 690us/step - loss: 0.4463 - acc: 0.8646
14/14 [==============================] - 0s 616us/step - loss: 0.3337 - acc: 0.8575
14/14 [==============================] - 0s 537us/step - loss: 0.3281 - acc: 0.8741
```

Hyperparameter search: Learning rate

```
%tensorboard --logdir=logs/[4]2_layer_lr
```

[현대자동차] › DS30_5차수 › DAY6 › logs › [4]2_layer_lr ›

이름	수정한 날짜	유형
lr_0.0001	2023-01-25 오후 2:42	파일 폴더



Hyperparameter search: Learning rate

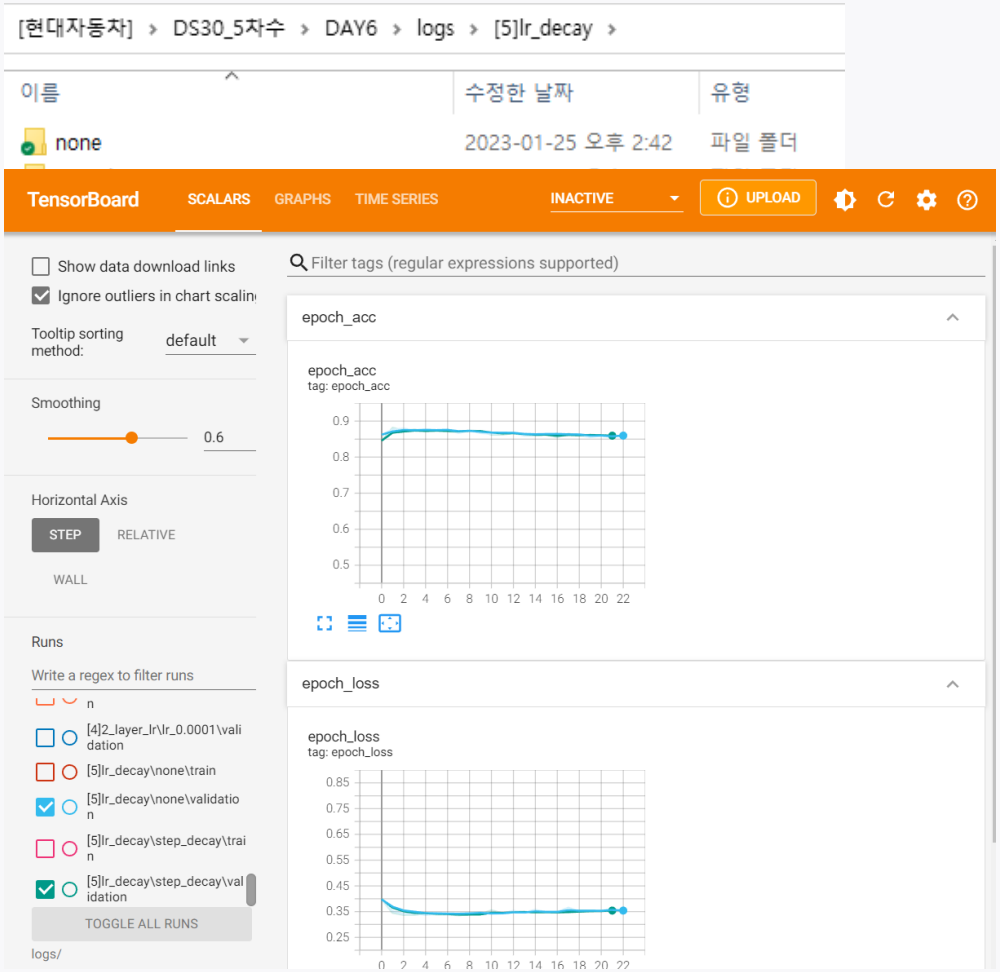
Effect of learning rate decaying

```
# w/o learning rate decay
model = create_model(n_layer=2, lambda_=0)
opt = Adam(learning_rate=1e-3)
model.compile(optimizer=opt,
              loss='binary_crossentropy',
              metrics=['acc'])
logdir = os.path.join('logs','[5]lr_decay','none')
tb_callback = TensorBoard(logdir)
es_callback = EarlyStopping(monitor='val_acc',patience=20)
hist = model.fit(X_train, y_train,
                 validation_split=1/9, epochs = 100,
                 verbose=0, callbacks=[tb_callback, es_callback] )
model.evaluate(X_test,y_test)
```

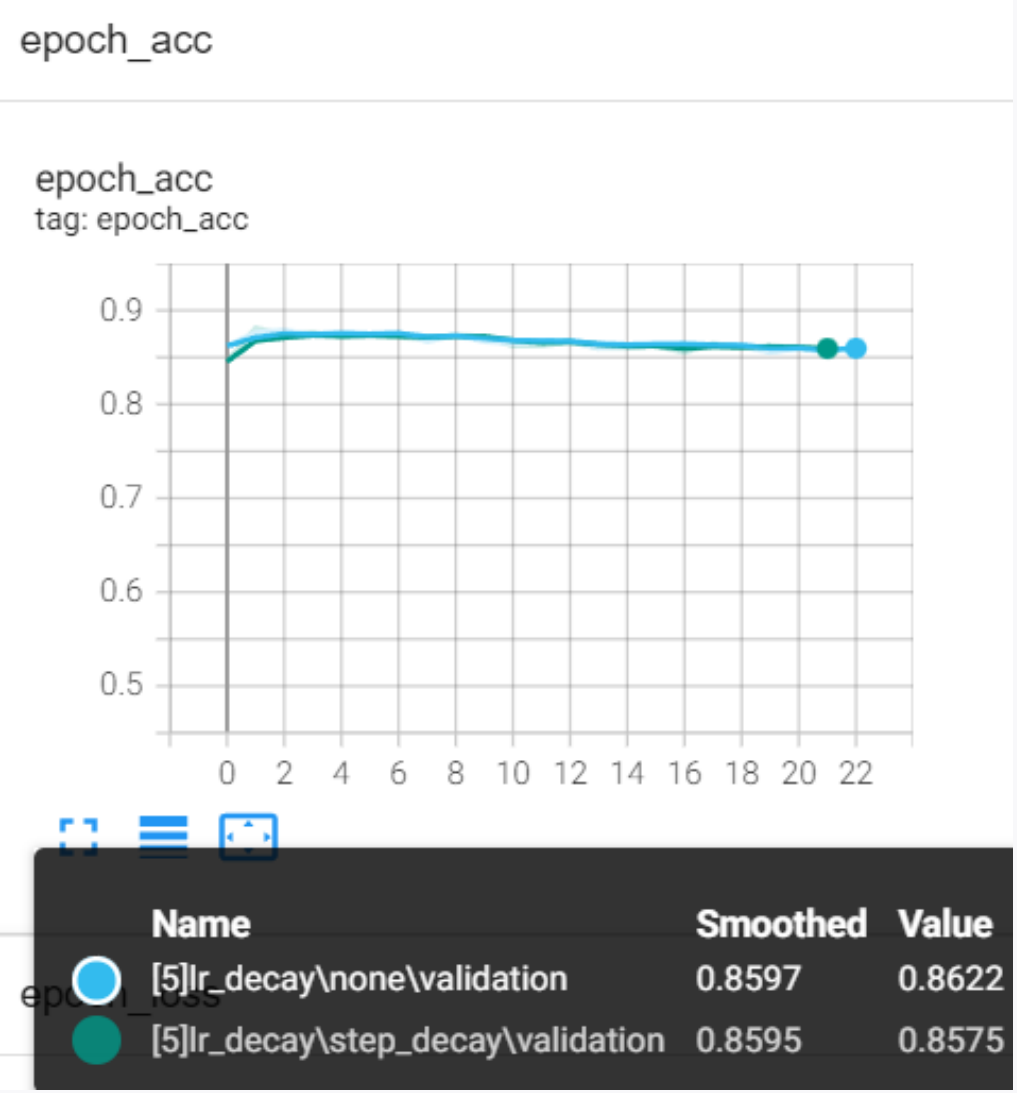
```
# w/ learning rate decay
model = create_model(n_layer=2, lambda_=0)
opt = Adam(learning_rate=1e-3)
model.compile(optimizer=opt,
              loss='binary_crossentropy',
              metrics=['acc'])
logdir = os.path.join('logs','[5]lr_decay','step_decay')
tb_callback = TensorBoard(logdir)
es_callback = EarlyStopping(monitor='val_acc',patience=20)
def scheduler(epoch, lr):
    if epoch in [30, 60, 90]:
        lr = lr*0.1
    return lr
lrs_callback = LearningRateScheduler(scheduler)
hist = model.fit(X_train, y_train,
                 validation_split=1/9, epochs = 100,
                 verbose=0, callbacks=[tb_callback,es_callback,lrs_callback] )
model.evaluate(X_test, y_test)
```

Effect of learning rate decaying

```
%tensorboard --logdir=logs/[5]lr_decay
```



Effect of learning rate decaying



Look ahead

Will study how to implement cross validation for a Keras model.